



This guide updated March 23, 2017

3DMark – The Gamer's Benchmark	5
3DMark benchmarks at a glance	6
3DMark edition features.....	7
Latest version numbers.....	8
Test compatibility	9
Good testing guide.....	10
Options (Windows).....	11
Android and iOS settings.....	14
Notes on DirectX 11.1	15
Time Spy	17
DirectX 12	17
Direct3D feature levels.....	18
System requirements	19
Graphics tests.....	20
CPU test	21
Scoring.....	22
DirectX 12 features in Time Spy.....	24
Time Spy engine.....	31
Post-processing.....	36
Time Spy version history.....	37
Fire Strike	39
System requirements	40
Default settings	41
Graphics test 1	42
Graphics test 2.....	43
Physics test.....	44
Combined test.....	45
Scoring.....	46
Fire Strike engine.....	48
Post-processing.....	51
Fire Strike version history.....	53
Sky Diver	55
System requirements	56
Default settings	57

Graphics test 1.....	58
Graphics test 2.....	59
Physics test.....	60
Combined test.....	61
Scoring.....	62
Sky Diver engine.....	66
Post-processing.....	68
Sky Diver version history	69
Cloud Gate	71
System requirements	72
Default settings	73
Graphics test 1.....	74
Graphics test 2.....	75
Physics test.....	76
Scoring.....	77
Cloud Gate engine	79
Cloud Gate version history	80
Sling Shot	82
Sling Shot.....	83
Sling Shot Extreme.....	84
Cross-platform benchmarking	85
Device requirements	86
Default settings	87
Graphics test 1.....	88
Graphics test 2.....	89
Physics Test	90
Scoring.....	91
Sling Shot engine	95
Sling Shot version history	97
Ice Storm.....	99
System requirements	100
Ice Storm	102
Ice Storm Extreme.....	103
Graphics test 1.....	104
Graphics test 2.....	105

Physics test.....	106
Scoring.....	107
Ice Storm engine.....	109
Ice Storm version history	110
API Overhead feature test	112
Correct use of the API Overhead feature test	113
System requirements	114
Windows settings	115
Technical details.....	116
DirectX 12 path	118
DirectX 11 path	119
Vulkan path	120
Mantle path.....	121
Metal path.....	122
OpenGL ES 3.0 path.....	123
Scoring.....	124
API Overhead version history	125
Stress Tests	126
Options.....	127
Technical details.....	128
Scoring.....	129
How to report scores.....	130
Release notes	131
Windows edition	131
Windows RT edition.....	142
Android edition	143
iOS edition - 3DMark Sling Shot app	149
iOS edition - 3DMark Ice Storm app	150
iOS edition - 3DMark API Overhead app.....	152
About Futuremark, a UL company	153

3DMark – The Gamer's Benchmark

3DMark is a tool for measuring the performance of computer hardware and mobile devices. It includes many different benchmarks, each designed for a specific class of hardware from smartphones to high-performance gaming PCs.

3DMark works by running intensive graphical and computational tests. The more powerful your hardware, the smoother the tests will run. Don't be surprised if frame rates are low. 3DMark benchmarks are very demanding.

Each benchmark gives a score that you can use to compare similar devices and systems. When testing devices or components, be sure to use the most appropriate test for the hardware's capabilities and report your results using the full name of the benchmark test, for example:

- ✓ "Video card scores 5,800 in 3DMark Fire Strike benchmark."
- × "Video card scores 5,800 in 3DMark benchmark."

3DMark is used by millions of gamers, hundreds of hardware review sites and many of the world's leading manufacturers. We are proud to say that 3DMark is the world's most popular and widely used benchmark.

The right test every time

We've made it easy to find the right test for your hardware and devices. When you open the 3DMark app on any platform, the Home screen will recommend the most suitable benchmark. You can find and run other tests on the Benchmarks screen.

Choose your tests

3DMark grows bigger every year with new tests. When you buy 3DMark from Steam, or download the Android app from Google Play, you can choose to install only the tests you need. In 3DMark Advanced and Professional Editions, tests can be installed and updated independently.

Complete Windows benchmarking toolkit

3DMark includes benchmarks for DirectX 12, DirectX 11, DirectX 10, and DirectX 9 compatible hardware. All tests are powered by modern graphics engines that use Direct3D feature levels to target compatible hardware.

Cross-platform benchmarking

You can measure the performance of Windows, Android, and iOS devices and compare scores across platforms.

3DMark benchmarks at a glance

3DMark includes a number of benchmarks, each designed for specific class of hardware. You will get the most useful and relevant results by choosing the most appropriate test for your system.

Benchmark	Target hardware	Engine	Rendering resolution ¹
Time Spy	High-performance gaming PCs running Windows 10	DirectX 12 feature level 11	2560 × 1440
Fire Strike Ultra	PC systems designed for 4K gaming	DirectX 11 feature level 11	3840 × 2160 (4K UHD)
Fire Strike Extreme	Multi-GPU systems and overclocked PCs	DirectX 11 feature level 11	2560 × 1440
Fire Strike	High-performance gaming PCs	DirectX 11 feature level 11	1920 × 1080
Sky Diver	Gaming laptops and mid-range PCs	DirectX 11 feature level 11	1920 × 1080
Cloud Gate	Notebooks and typical home PCs	DirectX 11 feature level 10	1280 × 720
Sling Shot Extreme	High-end smartphones and tablets	OpenGL ES 3.1 Metal	2560 × 1440
Sling Shot	Mainstream smartphones and tablets	OpenGL ES 3.0	1920 × 1080
Ice Storm Extreme	Low cost smartphones and tablets	DirectX 11 feature level 9 OpenGL ES 2.0	1920 × 1080
Ice Storm Ice Storm Unlimited	Older smartphones and tablets	DirectX 11 feature level 9 OpenGL ES 2.0	1280 × 720

¹ The resolution shown in the table is the resolution used to render the Graphics tests. In most cases, the Physics test or CPU test will use a lower rendering resolution to ensure that GPU performance is not a limiting factor.

3DMark edition features

	Basic Edition	Advanced Edition	Professional Edition
Time Spy	●	●	●
Fire Strike Ultra	×	●	●
Fire Strike Extreme	×	●	●
Fire Strike	●	●	●
Sky Diver	●	●	●
Cloud Gate	●	●	●
Ice Storm Extreme	●	●	●
Ice Storm	●	●	●
API Overhead feature test	×	●	●
Stress Tests	×	●	●
Install tests independently	×	●	●
Hardware monitoring	×	●	●
Custom run settings	×	●	●
Skip demo option	×	●	●
Save results offline	×	●	●
Private, offline results option	×	×	●
Command line automation	×	×	●
Licensed for commercial use	×	×	●
Support	Online	Online	Email & phone

Latest version numbers

	Windows	Android	iOS
3DMark app	2.3.3663	1.6.3439	See table below
Time Spy	1.0	×	×
Fire Strike	1.1	×	×
Sky Diver	1.0	×	×
Cloud Gate	1.1	×	×
Sling Shot	×	2.0	2.0
Ice Storm	1.2	1.2	1.2
API Overhead	1.5	×	1.0

On iOS, 3DMark benchmarks are separate apps due to platform limitations.

iOS app	version
3DMark Sling Shot	1.0.745
3DMark Ice Storm	1.4.978
3DMark API Overhead	1.0.147

Test compatibility

	Windows	Android	iOS
Time Spy	●	×	×
Fire Strike Ultra	●	×	×
Fire Strike Extreme	●	×	×
Fire Strike	●	×	×
Sky Diver	●	×	×
Cloud Gate	●	×	×
Sling Shot Extreme	×	●	●
Sling Shot	×	●	●
Ice Storm Extreme	●	●	●
Ice Storm	●	●	●
API Overhead	●	×	●

Good testing guide

To get accurate and consistent benchmark results you should test clean systems without third party software installed. If this is not possible, you should close as many background tasks as possible, especially automatic updates or tasks that feature pop-up alerts such as email and messaging programs.

If you are testing a mobile device, it is a good idea to close apps that may be running in the background, and disable notifications before running the benchmark. Some high-powered mobile devices use thermal throttling to avoid overheating the CPU, which can lead to lower scores on successive runs. To reduce this effect, we recommended waiting 15 minutes before and after 3DMark runs to allow the device to cool down.

- Running other programs during the benchmark can affect the results.
- Don't touch the mouse, keyboard or touchscreen while running tests.
- Do not change the window focus while the benchmark is running.
- Cancel a test by pressing the ESC key (PC), Back Button (Android), or Home button (iOS).

Recommended process

1. Install all critical updates to ensure your operating system is up to date.
2. Install the latest [approved drivers](#) for your hardware.
3. Close other programs.
4. Run the benchmark.

Expert process

1. Install all critical updates to ensure your operating system is up to date.
2. Install the latest approved drivers for your hardware.
3. Restart the computer or device.
4. Wait 2 minutes for startup to complete.
5. Close other programs, including those running in the background.
6. Wait for 15 minutes.
7. Run the benchmark.
8. Repeat from step 3 at least three times to verify your results.

Options (Windows)

The settings on the Options screen apply to all available benchmark tests.

Register / Unregister

If you have a 3DMark Advanced or Professional Edition upgrade key, copy it into the box and press the Register button. If you wish to unregister your key, so you can move your license to a different machine for example, press the Unregister button.

Validate result online

This option is only available in 3DMark Professional Edition where it is disabled by default. In 3DMark Basic and Advanced Editions, all results are validated online automatically.

Automatically view results online

When this box is checked 3DMark will automatically open a browser window allowing you to view your results on the Futuremark website after you complete a benchmark run.

- 3DMark Basic Edition, selected by default and cannot be disabled.
- 3DMark Advanced Edition, selected by default.
- 3DMark Professional Edition, disabled by default.

Automatically hide results online

Check this box if you wish to keep your 3DMark test scores private. Hidden results are not visible to other users and do not appear in search results. Hidden results are not eligible for competitions or the [Futuremark Overclocking Hall of Fame](#).

- 3DMark Basic Edition, disabled by default and cannot be selected.
- 3DMark Advanced Edition, disabled by default.
- 3DMark Professional Edition, selected by default.

Scan SystemInfo

SystemInfo is a component used by Futuremark benchmarks to identify the hardware in your system or device. It does not collect any personally identifiable information. This option is selected by default and is required in order to get a valid benchmark test score.

SystemInfo hardware monitoring

This option controls whether SystemInfo monitors your CPU temperature, clock speed, power, and other hardware information during the benchmark run. This option is selected by default.

Demo audio

Uncheck this box if you wish to turn off the soundtrack while a demo is running. This option is selected by default.

Language

Use this drop down to change the display language. The choices are:

- English
- German
- Simplified Chinese
- Russian

GPU count

You can use this drop down to tell 3DMark how many GPUs are present in the system you are testing. The default choice, Automatic, is fine in most cases and should only be changed in the rare instances when SystemInfo is unable to correctly identify the system's hardware.

Scaling mode

This option controls how the rendered output of each test, which is at a fixed resolution regardless of hardware, is scaled to fit the system's Windows desktop resolution.

The default option is Centered, which maintains the aspect ratio of the rendered output and, if needed, adds bars around the image to fill the remainder of the screen.

Selecting Stretched will stretch the rendered output to fill the screen without preserving the original aspect ratio. This option does not affect the test score.

Output resolution

3DMark tests are rendered at a fixed resolution regardless of hardware – the *rendering resolution*. The resulting frames are then scaled to fit the system's Windows desktop resolution – *the output resolution*. The default option is automatic, which sets the output resolution to the Windows desktop resolution. Change this option if you wish to display the benchmark at some other resolution. This option does not affect the test score.

Windows custom settings

Each benchmark test has its own settings, found on the Custom Run tab on the Test Details screen. Use custom settings to explore the limits of your PC's performance.

Custom settings are only available in the Advanced and Professional Editions.

You will only get an official 3DMark test score when you run a test with the default settings. When using custom settings you will still get the results from individual sub-tests as well as hardware performance monitoring information.

Android and iOS settings

The settings found on the Settings screen apply to all available 3DMark benchmark tests.

Show demo

Select NO if you wish to skip the demo. This option is set to YES by default.

Language

Use this drop down to change the display language. The choices are:

- English
- Simplified Chinese
- Russian

Notes on DirectX 11.1

3DMark does use DirectX 11.1, but only in a minor way and with a fall-back for DirectX 11 to ensure compatibility with the widest range of hardware and to ensure that all tests work with Windows 7 and Windows 8.

DirectX 11.1 API features were evaluated and those that could be utilized to accelerate the rendering techniques in the tests designed to run on DirectX 11.0 were used.

Discard resources and resource views

In cases where subsequent Direct3D draw calls will overwrite the entire resource or resource view and the application knows this, but it is not possible for the display driver to deduce it, a discard call is made to help the driver in optimizing resource usage. If DirectX 11.1 is not supported, a clear call or no call at all is made instead, depending on the exact situation. This DX11.1 optimization may have a performance effect with multi-GPU setups or with hardware featuring tile based rendering, which is found in some tablets and entry-level notebooks.

16 bpp texture formats

The 16 bpp texture formats supported by DirectX 11.1 are used on Ice Storm game tests to store intermediate rendering results during post processing steps. If support for those formats is not found, 32 bpp formats are used instead. This optimization gives a noticeable performance effect on hardware such as tablets, entry-level notebooks for which the Ice Storm tests provide a suitable benchmark.

There are no visual differences between the tests when using DirectX 11 or DirectX 11.1 in 3DMark and the practical performance difference from these optimizations is limited to Ice Storm on very low-end Windows hardware, and on Windows RT.



Time Spy

Time Spy is a DirectX 12 benchmark test for high-performance gaming PCs running Windows 10. Time Spy includes two Graphics tests, a CPU test, and a demo. The demo is for entertainment only and does not influence the score.

With its pure DirectX 12 engine, which supports features like asynchronous compute, explicit multi-adapter, and multi-threading, Time Spy is the ideal benchmark for testing the DirectX 12 performance of the latest graphics cards.

DirectX 12

DirectX 12, introduced with Windows 10, is a low-level graphics API that reduces processor overhead. With less overhead and better utilization of modern GPU hardware, a DirectX 12 game engine can draw more objects, textures and effects to the screen. How much more? Take a look at the table below that compares Time Spy with Fire Strike, a high-end DirectX 11 test.

Average amount of processing per frame

	Vertices	Triangles	Tessellation patches	Compute shader invocations
3DMark Fire Strike Graphics test 1	3,900,000	5,100,000	500,000	1,500,000
3DMark Fire Strike Graphics test 2	2,600,000	5,800,000	240,000	8,100,000
3DMark Time Spy Graphics test 1	30,000,000	13,500,000	800,000	70,000,000
3DMark Time Spy Graphics test 2	40,000,000	14,000,000	2,400,000	70,000,000

With DirectX 12, developers can significantly improve the multi-thread scaling and hardware utilization of their titles. But it requires a considerable amount of graphics expertise and memory-level programming skill. The programming investment is significant and must be considered from the start of a project.

3DMark Time Spy was developed with expert input from AMD, Intel, Microsoft, NVIDIA, and the other members of the [Futuremark Benchmark Development Program](#). It is one of the first DirectX 12 apps to be built "the right way" from the ground up to fully realize the performance gains that DirectX 12 offers.

Direct3D feature levels

DirectX 11 introduced a paradigm called [Direct3D feature levels](#). A feature level is a well-defined set of GPU functionality. For instance, the 9_1 feature level implements the functionality in DirectX 9.

With feature levels, 3DMark tests can use modern DirectX 12 and DirectX 11 engines and yet still target older DirectX 10 and DirectX 9 level hardware. For example, 3DMark Cloud Gate uses a DirectX 11 feature level 10 engine to target DirectX 10 compatible hardware.

Time Spy uses DirectX 12 feature level 11_0. This lets Time Spy leverage the most significant performance benefits of the DirectX 12 API while ensuring wide compatibility with DirectX 11 hardware through DirectX 12 drivers.

Game developers creating DirectX 12 titles are also likely to use this approach since it offers the best combination of performance and compatibility.

System requirements

OS	Windows 10, 64-bit
Processor	1.8 GHz dual-core Intel or AMD CPU
Storage	2 GB free disk space
GPU	DirectX 12
Video memory	1.7 GB (2 GB or more recommended)

Known compatibility issues

- Time Spy fails to run on multi-GPU systems with Windows 10 build 10240, but this is not the fault of the benchmark. You must upgrade Windows 10 to build 10586 (“November Update”) or later to enable multi-GPU configurations to work.
- AMD Radeon HD 7970/7870 and R9 280 series – the latest available drivers can cause crashes in Time Spy Graphics test 2. This appears to be a video driver issue. Scores from runs that do complete are fine.
- Intel Haswell with eDRAM (Core i7-4770R, for example) – Time Spy crashes with the latest available Intel drivers. This is a driver issue.
- Surface 4 Pro – The latest video drivers available from Microsoft are not compatible with Time Spy.

Graphics tests

Time Spy includes two graphics tests. These tests are designed to stress the GPU while minimizing the CPU workload to ensure that CPU performance is not a limiting factor.

The two tests have a different mix of rendering workload elements. The differences come from the amount of tessellated geometries, the average tessellation factor, the amount of particle effects, the amount of geometries, the amount of transparent geometries, the post processing effects that are used, the amount of lights, and the amount of ray-marched volumetric illumination.

Graphics test 1

Graphics test 1 focuses more on rendering of transparent elements. It utilizes the A-buffer heavily to render transparent geometries and big particles in an order-independent manner. Graphics test 1 draws particle shadows for selected light sources. Ray-marched volumetric illumination is enabled only for the directional light. All post-processing effects are enabled.

Graphics test 2

Graphics test 2 focuses more on ray-marched volume illumination with hundreds of shadowed and unshadowed spot lights. The A-buffer is used to render glass sheets in an order-independent manner. Also, lots of small particles are simulated and drawn into the A-buffer. All post-processing effects are enabled.

Processing performed in an average frame

	Vertices	Tessellation patches	Triangles	Pixels shader invocations ²	Compute shader invocations
Graphics test 1	30 million	0.8 million	13.5 million	80 million	70 million
Graphics test 2	40 million	2.4 million	14 million	50 million	70 million

² This figure is the average number of pixels processed per frame before the image is scaled to fit the native resolution of the device being tested. If the device's display resolution is greater than the test's rendering resolution, the actual number of pixels processed per frame will be even greater.

CPU test

The CPU test is designed to stress the CPU while minimizing the GPU workload to ensure that GPU performance is not a limiting factor.

The CPU test measures performance using a demanding combination of physics simulation, occlusion culling, and procedural generation.

Physics computations are performed with the x86 path of the Bullet Open Source Physics library (v2.83) using rigid bodies and a Featherstone solver.

The CPU test includes two custom simulation features: procedural generation of crystal clusters and simulation of packs of small items based on the combination of a traditional boid-system and a simple mass-spring system.

Crystal generation is performed from a predefined set of rules and parameters combined with sampling of artist-controlled target animation. However, procedural generation can be done fairly effectively and doesn't result in much workload for the CPU. The bulk of the workload associated with crystals comes from the continuous update of the transformation matrices as a result of their multi-part animation.

The simulation of the item flocks is implemented as a traditional boid algorithm: it searches for a nearby object and thus results in $O(n^2)$ complexity. Furthermore, they are simulated to follow an artist-defined target using a simple mass-spring system consisting of two springs while scanning for nearby repulsor objects. The interaction with repulsors consist of a single spring assuming it being within the range of repulsor's influence.

Scoring

Time Spy produces an overall Time Spy score, a Graphics test sub-score, and a CPU test sub-score. The scores are rounded to the nearest integer. The better a system's performance, the higher the score.

Overall Time Spy score

The 3DMark Time Spy score formula uses a weighted harmonic mean to calculate the overall score from the Graphics and CPU test scores.

$$\text{Time Spy score} = \frac{W_{graphics} + W_{cpu}}{\frac{W_{graphics}}{S_{graphics}} + \frac{W_{cpu}}{S_{cpu}}}$$

Where:

$W_{graphics}$	=	The Graphics score weight, equal to 0.85
W_{cpu}	=	The CPU score weight, equal to 0.15
$S_{graphics}$	=	Graphics test score
S_{cpu}	=	CPU test score

For a balanced system, the weights reflect the ratio of the effects of GPU and CPU performance on the overall score. Balanced in this sense means the Graphics and CPU test scores are roughly the same magnitude.

For a system where either the Graphics or CPU score is substantially higher than the other, the harmonic mean rewards boosting the lower score. This reflects the reality of the user experience. For example, doubling the CPU speed in a system with an entry-level graphics card doesn't help much in games since the system is already limited by the GPU. Likewise for a system with a high-end graphics card paired with an underpowered CPU.

Graphics test scoring

Each Graphics test produces a raw performance result in frames per second (FPS). We take a harmonic mean of these raw results and multiply it by a scaling constant to reach a Graphics score ($S_{graphics}$) as follows:

$$S_{graphics} = 164 \times \frac{2}{\frac{1}{Fgt1} + \frac{1}{Fgt2}}$$

Where:

$Fgt1$ = The average FPS result from Graphics test 1
 $Fgt2$ = The average FPS result from Graphics test 1

The scaling constant is used to bring the score in line with traditional 3DMark score levels.

CPU test scoring

The CPU test consists of three increasingly heavy levels, each of which has a ten second timeline. The third, and thus heaviest, level produces a raw performance result in frames per second (FPS) which is multiplied by a scaling constant to give a CPU score ($Scpu$) as follows:

$$Scpu = 298 \times Fcpu_3$$

Where:

$Fcpu_3$ = The average FPS from the CPU test's third level

The scaling constant is used to bring the score in line with traditional 3DMark score levels.

DirectX 12 features in Time Spy

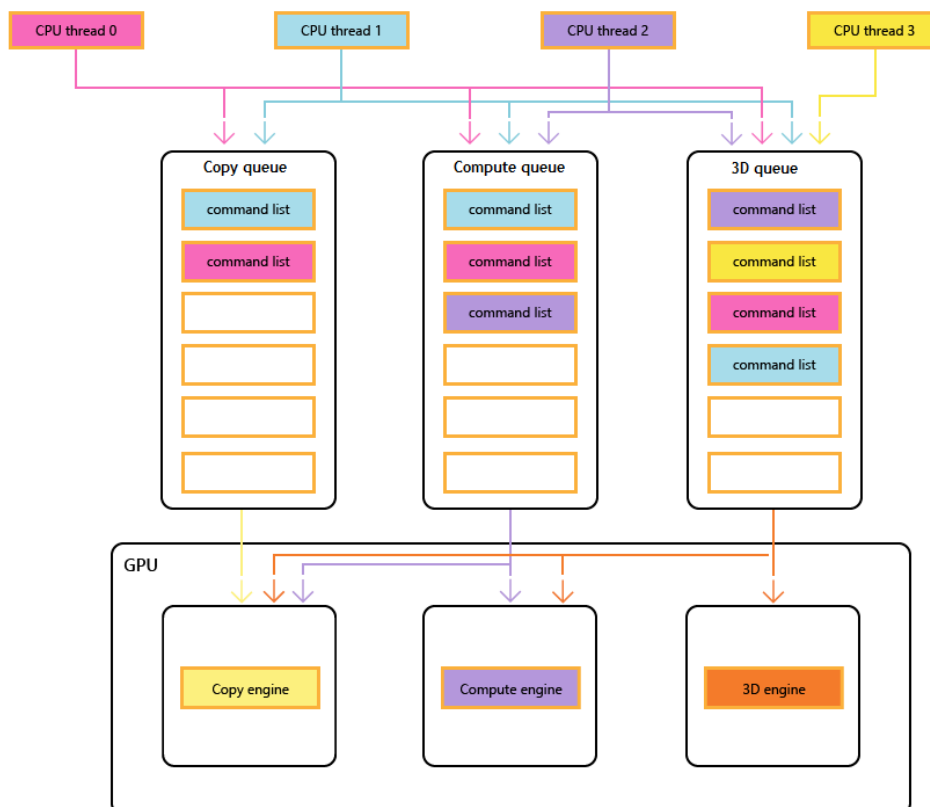
Command lists and asynchronous compute

Unlike the Draw/Dispatch calls in DirectX 11 (with immediate context), In DirectX 12, the recording and execution of command lists are decoupled operations. There is no thread limitation on recording command lists. Recording can happen as soon as the required information is available.

Quoting from [MSDN](#):

"Most modern GPUs contain multiple independent engines that provide specialized functionality. Many have one or more dedicated copy engines, and a compute engine, usually distinct from the 3D engine. Each of these engines can execute commands in parallel with each other. Direct3D 12 provides granular access to the 3D, compute and copy engines, using queues and command lists.

"The following diagram shows a title's CPU threads, each populating one or more of the copy, compute and 3D queues. The 3D queue can drive all three GPU engines, the compute queue can drive the compute and copy engines, and the copy queue simply the copy engine.



Command list execution

For GPU work to happen, command lists are executed on queues, which come in variants called DIRECT (commonly known as graphics or 3D as in the diagram above), COMPUTE and COPY. Submission of a command list to a queue can happen on any thread. The D3D runtime serializes and orders the lists within a queue.

DIRECT command list	This command list type supports all types of commands including Draw calls, compute Dispatches and Copies.
COMPUTE command list	This command list type supports compute Dispatch and Copy commands.
DIRECT queue	This queue can be used for executing all types of command lists supported by DirectX 12.
COMPUTE queue	This queue accepts compute and copy command lists.
COPY command list and queues	This command list and queue type accepts only copy commands and lists respectively.

Once initiated, multiple queues can execute in parallel. This parallelism is commonly known as 'asynchronous compute' when COMPUTE queue work is performed at the same time as DIRECT queue work.

It is up to the driver and the hardware to decide how to execute the command lists. The application cannot affect this decision through the DirectX 12 API.

Please see MSDN for an introduction to the [Design Philosophy of Command Queues and Command Lists](#), and for more information on [Executing and Synchronizing Command Lists](#).

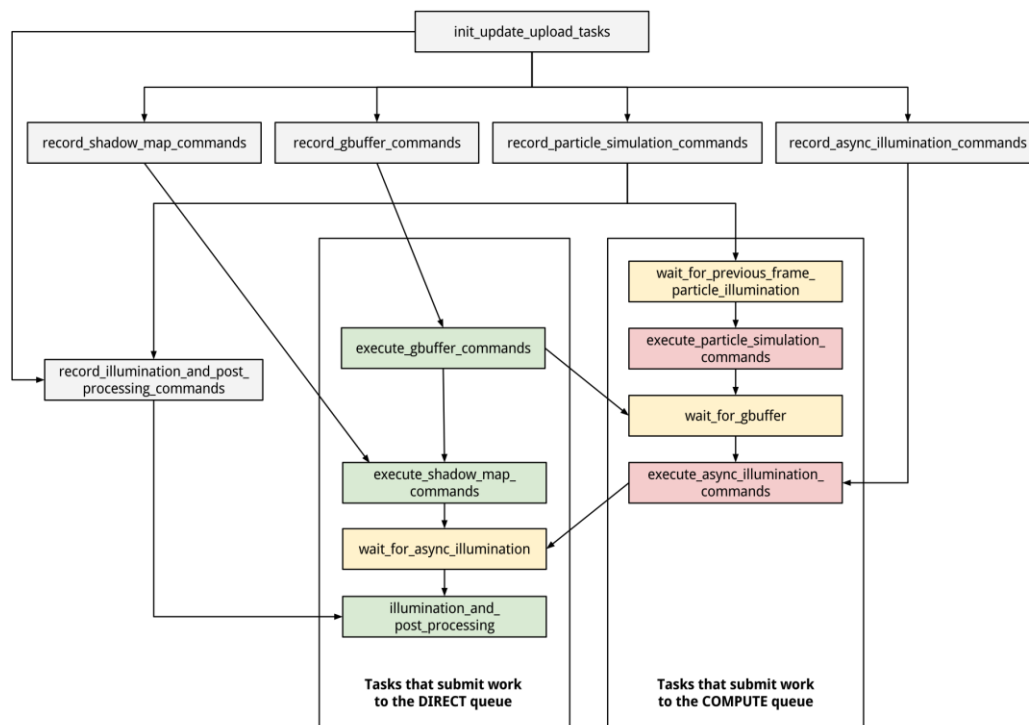
In Time Spy, the engine uses two command queues: a DIRECT queue for graphics and compute and a COMPUTE queue for asynchronous compute.³

The implementation is the same regardless of the capabilities of the hardware being tested. It is ultimately the decision of the underlying driver whether the work in the COMPUTE queue is executed in parallel or in serial.

There is a large amount of command lists as many tasks have their own command lists, (several copies so that frames can be pre-recorded).

³ The COPY queue is generally used for streaming assets. It is not needed in Time Spy as we load all assets before the benchmark run begins to ensure the test does not gain a dependency on storage or main memory.

Simplified DAG⁴ of 3DMark Time Spy queue usage



Each task encapsulates a complex task substructure that is omitted in this simplified graph for clarity. If there are no dependencies, tasks are executed on the CPU in parallel.

Grey tasks are CPU tasks. The `async_illumination_commands` task contains light culling and tiling, environment reflections, HBAO, and unshadowed surface illumination.

Green tasks are submissions to the DIRECT (graphics) queue. G-buffer draws, shadow map draws, shadowed illumination resolve, and post-processing are executed on the DIRECT queue. G-buffer draws, shadow maps and some parts of the post-processing are done with graphics shaders, while illumination resolve and the rest of the post processing is done in compute shaders.

Red tasks are submissions to the COMPUTE queue. Particle simulation, light culling and tiling, environment reflections, HBAO and unshadowed surface illumination resolve are executed on the COMPUTE queue. All tasks in the compute queue must be done in compute shaders.

Yellow tasks are submissions of synchronization points. The significance of these can be seen by noting that `execute_async_illumination_commands` cannot be executed on the GPU before `execute_gbuffer_commands` is

⁴ Directed Acyclic Graph (DAG), see https://en.wikipedia.org/wiki/Directed_acyclic_graph.

completed, but the submission happens ahead of the execution, (unless we are CPU bound). The GPU needs to know that it should wait for a task to complete execution before a dependent task can begin executing. When the execution is split between queues then this operation should be done by the engine otherwise a [RAW](#) hazard occurs. There is another dependency between particle simulation and completion of particle illumination in the previous frame. The simulation happens on the compute queue, which will cause a [WAR](#) hazard if it is not synchronized with the Present occurring on the graphics queue.

The order of submission can be obtained from the dependency graph. However, it is entirely up to the driver and the hardware to decide when to actually execute the given list as long as it is executed in order in its queue.

Compute queue work items (in order of submission)

1. Particle simulation

This pass is recorded and executed at the beginning of a frame because it doesn't depend on the G-buffer. Thus its recording and submission is done in parallel with recording and submission of geometry draws (G-Buffer construction).

2. Light culling and tiling
3. Environment reflections
4. Horizon based ambient occlusion
5. Unshadowed surface illumination

These passes are recorded and submitted in parallel with G-Buffer recording and submission, but executed only after the G-Buffer is finished executing and in parallel with shadow maps execution. This is because they depend on the G-Buffer, but not on the shadow maps.

Disabling asynchronous compute in benchmark settings

The asynchronous compute workload per frame in Time Spy varies between 10% and 20%. To observe the benefit on your own hardware, you can optionally choose to disable asynchronous compute using the Custom run settings in 3DMark Advanced and Professional Editions.

Running with asynchronous compute disabled in the benchmark forces all work items usually associated with the COMPUTE queue to instead be put in the DIRECT queue.

Explicit multi-adapter

In DirectX 11, control of GPU adapters is implicit - the drivers use multiple GPUs on behalf of an application.

In DirectX 12, control of multiple GPUs is explicit. The developer can control what work is done on each GPU and when. With explicit multi-adapter control,

one can implement more complex multi-GPU models, for example choosing to execute partial workloads for a frame across different GPUs.

A GPU adapter can be any graphics adapter, from any manufacturer, that supports D3D12. Each adapter is referred to as a node. There are two multi-adapter modes called linked-node adapter and multi-node adapter.

With linked-node (LDA) the programmer has access to and control over an SLI/Crossfire configuration of similar GPUs through one device interface. LDA enables some extra features over multi-node, such as faster transfers between GPUs, cross-node resource sharing and shared swap-chain (back-buffer).

With multi-node (MDA) each GPU appears as a separate device, even if they are similar and linked. With MDA, the programmer can control any and all GPUs available in the system. But the programmer must explicitly declare which GPU should execute the recorded work. MDA allows much more fine-grained control over rendering and work submission, allowing you to divide work between a discrete graphics card and an integrated GPU for example.

Time Spy uses explicit alternate frame rendering on linked-node configurations to improve performance on the most common multi-GPU setups used by gamers today. MDA configurations of heterogeneous adapters are not supported.

Multi-threaded GPU work recording and submission

DirectX 11 offers multi-threaded (deferred) context support, but not all vendors implement it in hardware, so it is slow. And overall, it is quite limited.

DirectX 12 really takes multi-threaded rendering to the next level. With DirectX 12, the programmer is in the control of everything. There are a few operations that cannot be executed at the same time on multiple threads, but otherwise, there are not many rules.

Resources must be manually transitioned to the correct states, progress within a frame must be tracked explicitly, and any potential hazards must be handled explicitly. All synchronization of CPU and GPU workloads must be done using fences and barriers, as there is no validation or checks in the driver.

In Time Spy, the rendering is heavily multithreaded. Command lists are recorded on all logical cores.

Improved resource allocation, explicit state tracking, and persistent mapping

In DirectX 11, there are no heaps. The driver manages everything, including all states. Transfers to GPU memory must go through the API layer.

In DirectX 12, there are multiple ways to allocate resources. Programmers can create heaps, big piles of data that can later be filled with textures and buffers.

Heaps also save memory by allowing resources to be placed on top of each other, for example render target surfaces.

All resource states must be explicitly declared. Resources have an initial state, and they must be transitioned to the correct state before the rendering commands are executed. For example, if a resource is going to be written to, it must be transitioned to a write state. The same applies for all other operations.

Since all state is explicit, the driver no longer has 'guess' the intent of the programmer, which allows faster execution. State can be changed across different work packets (command lists).

Some buffers can be persistently mapped to CPU memory to mirror the same buffer in GPU memory. This allows transfers to GPU memory with less stalls and also removes the need to invalidate buffers. But on the other hand, it puts the responsibility of managing the buffer on the programmer.

In Time Spy, all features are used, including heaps with overlapping resources to save memory. States are explicitly handled as they should be. Persistently mapped (streaming) buffers are used for all dynamic data with custom resource hazard prevention using fences.

Pre-built GPU state objects

In DirectX 11, individual states (like bound shaders) can be changed at any time. There are no limitations. But the driver must optimize during runtime if necessary, which can lead to stalled rendering.

In DirectX 12, the GPU pipeline state is managed by separate pipeline state objects that encapsulate the whole state of the graphics/compute engine. In the graphics case, this encompasses things like the rasterizer state, different shaders (e.g. vertex and pixel shader), and the blending mode. State switching is done in one step by replacing the whole pipeline at once.

Since pipelines are pre-built before they are bound, the driver can optimize them beforehand. During runtime, only the GPU state reconfiguration is required based on the already optimized state. This allows very fast state switching. It removes the need for 'warm-up' before rendering, since the drivers don't cache state as often as with DirectX 11.

Pipelines can also be compiled during runtime, of course. Games can compile only the necessary pipelines during startup. If a new pipeline object is required later, it can be created easily in a separate thread without halting any of the application logic threads.

In Time Spy, all pipelines are built during startup. State changes are minimized by sorting by pipeline state object during rendering.

Resource binding

As mentioned in the previous section on pipelines, when a new state is bound to the GPU everything about it is already known. This also applies for resource bindings. Pipeline state objects also contain information about the resources that will be bound to the shader and how they will reside in the GPU memory.

DirectX 12 uses descriptors and descriptor tables to bind resources.

Descriptors are very lightweight objects that contain information about the resource that is to be bound. Descriptors can be arranged in tables for easy binding of multiple resources at once. This operation is also very fast, as the table can be described by binding only one pointer.

In Time Spy, resource binding is used as it should be to optimize performance.

Explicit synchronization between CPU, GPU, multiple GPUs, and multiple GPU queues

In DirectX 12, synchronization won't happen without programmer intervention. All possible resource hazards must be handled by the programmer by using various synchronization objects.

And since multiple GPU queues are supported, fences must also be used on the GPU side to make sure queues execute work when they should. It's programmer's responsibility to handle all synchronization.

In Time Spy, synchronization is used as it should be to optimize performance.

Time Spy engine

To fully take advantage of the performance improvements that DirectX 12 offers, Time Spy uses a custom game engine developed in-house from the ground up. The engine was created with the input and expertise of AMD, Intel, Microsoft, NVIDIA, and the other members of the [Futuremark Benchmark Development Program](#).

Multi-threading

The rendering, including scene update, visibility evaluation, and command list building, is done with multiple CPU threads using one thread per available logical CPU core. This reduces CPU load by utilizing multiple cores.

Multi-GPU support

The engine supports the most common type of multi-GPU configuration, i.e. two identical GPU adapters in Crossfire/SLI, by using explicit multi-adapter with a linked-node configuration to implement explicit alternate frame rendering. Heterogeneous adapters are not supported.

Visibility solution

The Umbra occlusion library (version 3.3.17 or newer) is used to accelerate and optimize object visibility evaluation for all cameras, including the main camera and light views used for shadow map rendering. The culling runs on the CPU and does not consume GPU resources.

Descriptor heaps

One descriptor heap is created for each descriptor type when the scene is loaded. Hardware Tier 1 is sufficient for containing all the required descriptors in the heaps. Root signature constants and descriptors are used when suitable.

Resource heaps

Implicit resource heaps created by `ID3D12Device::CreateCommittedResource()` are used for most resources. Explicitly created heaps are used for some target resources to reduce memory consumption by placing resources that not needed at the same time on top of each other.

Asynchronous compute

Asynchronous compute is utilized heavily to overlap multiple rendering passes for maximum utilization of the GPU. Async compute workload per frame varies between 10-20%.

Tessellation

The engine supports Phong tessellation and displacement-map-based detail tessellation.

Tessellation factors are adjusted to achieve the desired edge length for the output geometry on the render target (G-buffer, shadow map or other). Additionally, patches that are back-facing and patches that are outside of the view frustum are culled by setting the tessellation factor to zero.

Tessellation is turned entirely off by disabling hull and domain shaders when the size of an object's bounding box on the render target drops below a given threshold.

If an object has several geometry LODs, tessellation is used on the most detailed LOD.

Geometry rendering

Objects are rendered in two steps. First, all opaque objects are drawn into the G-buffer. In the second step, transparent objects are rendered to an A-buffer, which is then resolved on top of surface illumination later on.

Geometry rendering uses a LOD system to reduce the number of vertices and triangles for objects that are far away. This also results in bigger on-screen triangle size.

The material system uses physically based materials. The following textures can be used as input to materials. Not all textures are used on all materials.

Material Texture	Format
Albedo (RGB) + metalness (A)	BC3 or BC7
Roughness (R) + Cavity (G)	BC5
Normal (RG)	BC5
Ambient Occlusion (R)	BC4
Displacement	BC4
Luminance	BC1 or BC7
Blend	BC4, BC5 or BC3
Opacity	BC4

Opaque objects

Opaque objects are rendered directly to the G-buffer. The G-buffer is composed of textures shown in the table below. A material might not use all target textures. For example, a luminance texture is only written into when drawing geometries with luminous materials.

G-buffer Texture	Format
Depth	D24_UNORM_S8_UINT
Normal	R10G10B10A2_UNORM
Albedo	R8G8B8A8_UNORM_SRGB
Material Attributes	R10G10B10A2_UNORM
Luminance	R11G11B10_FLOAT

Transparent objects

For rendering transparent geometries, the engine uses a variant of an order-independent transparency technique called Adaptive Transparency (Salvi et al. 2011). Simply put, a per-pixel list of fragments is created for which a visibility function (accumulated transparency) is approximated. The fragments are blended according to the visibility function and illuminated in the lighting pass to allow them to be rendered in any order. The A-buffer is drawn after the G-buffer to fully take advantage of early depth tests.

In addition to the per-pixel lists of fragments, per 2x2 quad lists of fragments are created. The per-quad lists can be used for selected renderables instead of the per pixel lists. This saves memory when per pixel information is not required for a visually satisfying result. When rendering to per quad lists, a half resolution viewport and depth texture is used to ignore fragments behind opaque surfaces. When resolving the A-buffer fragments for each pixel, both per pixel list and per quad list are read and blended in the correct order. Each per quad list is read for four pixels in the resolve pass.

Lighting

Lighting is evaluated using a tiled method in multiple separate passes.

Before the main illumination passes, asynchronous compute shaders are used to cull lights, evaluate illumination from prebaked environment reflections, compute screen-space ambient occlusion, and calculate unshadowed surface illumination. These tasks are started right after G-buffer rendering has finished and are executed alongside shadow rendering. All frustum lights, omni-lights

and reflection capture probes are culled to small tiles (16x16 pixels) and written to an intermediate buffer. Reflection illumination is evaluated for the opaque surfaces by sampling the precomputed reflection cubes. The results are written out to a separate texture. Ambient occlusion and unshadowed illumination results are written out to their respective targets.

Second, illumination from all lights and GI data is evaluated for the surface. The A-buffer is also resolved in a separate pass and then composed on top of surface illumination. This produces the final illumination that is sampled in the screen space reflection step, which also blends in previously computed environment illumination based on SSR quality. Reflections are applied on top of surface illumination. Surface illumination is also masked with SSAO results.

Third, volume illumination is computed. This includes two passes. The first one evaluates volume illumination from global illumination data and the second one calculates illumination from direct lights. The evaluation is done by raymarching the light ranges.

Finally, surface illumination, GI volume illumination, and direct volume illumination are composed into one final texture with some blurring, which is then fed to post-processing stages.

Shadows are sampled in both surface and volume illumination shaders. For shadow casting lights, the textures in the table below can be rendered.

Shadow Texture	Format
Shadow Depth	D16_UNORM
Particle Transmittance	R8G8B8A8_UNORM

Particles

Particles are simulated on the GPU using asynchronous compute queue. Simulation work is submitted to the asynchronous queue while G-buffer and shadow map rendering commands are submitted to the main command queue.

Particle illumination

Particles are rendered by inserting particle fragments into an A-buffer. The engine utilizes a separate half-resolution A-buffer for low-frequency particles to allow more of them to be visible in the scene at once. They are blended together with the main A-buffer in the combination step. Particles can be illuminated with scene lights or they can be self-illuminated. The output buffers of the GPU light-culling pass and the global illumination probes are used as inputs for illuminated particles. The illuminated particles are drawn without tessellation and they are illuminated in the pixel shader.

Particle shadows

Particles can cast shadows. Shadow casting particles are rendered into transmittance 3D textures for lights that have particle shadows enabled. Before being used as an input to illumination shaders, an accumulated version of the transmittance texture is created. If typed UAV loads are supported, the transmittance texture is accumulated in-place. Otherwise the accumulated result is written to an additional texture. The accumulated transmittance texture is sampled when rendering surface, particle and volume illumination by taking one sample with bilinear filtering per pixel or per ray marching step. Resolution of the transmittance texture for each spotlight is evaluated on each frame based on screen coverage of the light. For directional light, fixed resolution textures are used.

Post-processing

Depth of field

The effect is computed by scattering the illumination in the out-of-focus parts of the input image using the following procedure.

1. Using CS, circle of confusion radius is computed for all screen pixels based on depth texture. The information is additionally reduced to half and quarter resolutions. In the same CS pass, a splatting primitive (position, radius and color) for out-of-focus pixels whose circle of confusion radius exceeds a predefined threshold is appended to a buffer. For pixel quads and 4x4 tiles that are strongly out of focus, a splatting primitive per quad or tile is appended to the buffer instead of per pixel primitives.
2. The buffer with splatting primitives for the out-of-focus pixels is used as point primitive vertex data and, using Geometry Shader, an image of a bokeh is splatted to the positions of these primitives. Splatting is done to a texture that is divided into regions with different resolutions using multiple viewports. First region is screen resolution and the rest are a series of halved regions down to 1x1 texel resolution. The screen space radius of the splatted bokeh determines the used resolution. The larger the radius the smaller the used splatting resolution.
3. The different regions of the splatting texture are combined by up-scaling the data in the smaller resolution regions step by step to the screen resolution region.
4. Finally, the out-of-focus illumination is combined with the original illumination.







Bloom

Bloom is based on a compute shader FFT that evaluates several effects with one filter kernel. The effects are blur, streaks, anamorphic flare and lenticular halo.

Lens Reflections

The effect is computed by first applying a filter to the computed illumination in frequency domain like in the bloom effect. The filtered result is then splatted in several scales and intensities on top of the input image using additive blending. The effect is computed in the same resolution as the bloom effect and therefore the forward FFT needs to be performed only once for both effects. The filtering and inverse FFT are performed using the CS and floating point textures.

Time Spy version history

Version				Notes
1.0				Launch version




Fire Strike

Fire Strike is a DirectX 11 benchmark for high-performance gaming PCs. Fire Strike includes two graphics tests, a physics test and a combined test that stresses both the CPU and GPU.

3DMark Advanced and Professional Editions include Fire Strike Extreme and Fire Strike Ultra, two benchmarks designed for high-end systems with multiple GPUs (SLI / Crossfire).

Scores from 3DMark Fire Strike, Fire Strike Extreme and Fire Strike Ultra should not be compared to each other - they are separate tests with their own scores, even though they share the same content. Fire Strike Ultra is the highest workload 3DMark can currently offer.

Fire Strike benchmarks are only available in the Windows editions of 3DMark.

 Fire Strike tests are demanding benchmarks designed for high-end hardware. If your system scores less than 2800 in Fire Strike you should run Sky Diver instead.

Fire Strike

Fire Strike is a DirectX 11 benchmark for high-performance gaming PCs and overclocked systems. Fire Strike is very demanding, even for the latest graphics cards. If your frame rate is low, use Sky Diver instead.

Fire Strike Extreme

Fire Strike Extreme is designed for testing PCs with multiple GPUs (minimum 1.5 GB graphics card memory required). It raises the rendering resolution from 1920x1080 to 2560x1440 and improves the visual quality.

Fire Strike Ultra

Fire Strike Ultra is a dedicated test for 4K gaming. It raises the rendering resolution to 3840 × 2160 (4K UHD), four times larger than 1080p. A 4K monitor is not required, but your graphics card must have at least 3GB of memory.

System requirements

	Fire Strike	Fire Strike Extreme	Fire Strike Ultra
OS ⁵	Windows 7 or later	Windows 7 or later	Windows 7 or later
Processor	1.8 GHz dual-core Intel or AMD CPU	1.8 GHz dual-core Intel or AMD CPU	1.8 GHz dual-core Intel or AMD CPU
Storage	6 GB free space	6 GB free space	6 GB free space
GPU	DirectX 11	DirectX 11	DirectX 11
For systems with integrated graphics	3 GB RAM	5.5 GB RAM	7 GB RAM
For systems with a discrete graphics card	2 GB RAM 1 GB video card memory	4 GB RAM 1.5 GB video card memory	4 GB RAM 3 GB video card memory

⁵ Windows 7 users must install Service Pack 1.

Default settings

	Fire Strike	Extreme	Ultra
Resolution	1920 × 1080	2560 × 1440	3840 × 2160
GPU Memory Budget	1 GB	1.5 GB	3 GB
Tessellation Detail	Medium	High	High
Surface Shadow Sample Count	8	16	16
Shadow Map Resolution	1024	2048	2048
Volume Illumination Quality	Medium	High	High
Particle Illumination Quality	Medium	High	High
Ambient Occlusion Quality	Medium	High	High
Depth of Field Quality	Medium	High	High
Bloom Resolution	1/4	1/4	1/4

Graphics test 1

3DMark Fire Strike Graphics test 1 focuses on geometry and illumination. Particles are drawn at half resolution and dynamic particle illumination is disabled. There are 100 shadow casting spot lights and 140 non-shadow casting point lights in the scene. Compute shaders are used for particle simulations and post processing. Pixel processing is lower than in Graphics test 2 as there is no depth of field effect.

Processing performed in an average frame

	Vertices	Tessellation patches	Triangles	Pixels ⁶	Compute shader invocations
Fire Strike	3.9 million	500,000	5.1 million	80 million	1.5 million
Fire Strike Extreme	3.9 million	560,000	9.9 million	150 million	3.4 million
Fire Strike Ultra	3.7 million	650,000	12.4 million	330 million	3.4 million

⁶ This figure is the average number of pixels processed per frame before the image is scaled to fit the native resolution of the device being tested. If the device's display resolution is greater than the test's rendering resolution, the actual number of pixels processed per frame will be even greater.

Graphics test 2

3DMark Fire Strike Graphics test 2 focuses on particles and GPU simulations. Particles are drawn at full resolution and dynamic particle illumination is enabled. There are two smoke fields simulated on GPU. Six shadow casting spot lights and 65 non-shadow casting point lights are present. Compute shaders are used for particle and fluid simulations and for post processing steps. Post processing includes a depth of field effect.

Processing performed in an average frame

	Vertices	Tessellation patches	Triangles	Pixels ⁷	Compute shader invocations
Fire Strike	2.6 million	240,000	5.8 million	170 million	8.1 million
Fire Strike Extreme	3.9 million	260,000	12.9 million	400 million	10.4 million
Fire Strike Ultra	6.0 million	260,000	17.6 million	1100 million	10.4 million

⁷ This figure is the average number of pixels processed per frame before the image is scaled to fit the native resolution of the device being tested. If the device's display resolution is greater than the test's rendering resolution, the actual number of pixels processed per frame will be even greater.

Physics test

3DMark Fire Strike Physics test benchmarks the hardware's ability to run gameplay physics simulations on the CPU. The GPU load is kept as low as possible to ensure that only the CPU is stressed. The Bullet Open Source Physics Library is used as the physics library for the test.

The test has 32 simulated worlds. One thread per available CPU core is used to run simulations. All physics are computed on CPU with soft body vertex data updated to GPU each frame.

Combined test

3DMark Fire Strike Combined test stresses both the GPU and CPU simultaneously. The GPU load combines elements from Graphics test 1 and 2 using tessellation, volumetric illumination, fluid simulation, particle simulation, FFT based bloom and depth of field.

The CPU load comes from the rigid body physics of the breaking statues in the background. There are 32 simulation worlds running in separate threads each containing one statue decomposing into 113 parts. Additionally there are 16 invisible rigid bodies in each world except the one closest to camera to push the decomposed elements apart. The simulations run on one thread per available CPU core.

The 3DMark Fire Strike Combined test uses the Bullet Open Source Physics Library.

Processing performed in an average frame

	Vertices	Tessellation patches	Triangles	Pixels ⁸	Compute shader invocations
Fire Strike	7.5 million	530,000	7.9 million	150 million	110 million
Fire Strike Extreme	9.2 million	540,000	14.8 million	390 million	110 million
Fire Strike Ultra	10.8 million	540,000	19.6 million	960 million	120 million

⁸ This figure is the average number of pixels processed per frame before the image is scaled to fit the native resolution of the device being tested. If the device's display resolution is greater than the test's rendering resolution, the actual number of pixels processed per frame will be even greater.

Scoring

Scores from different benchmarks should not be compared to each other. Fire Strike, Fire Strike Extreme, and Fire Strike Ultra are separate tests with their own scores, even though they share the same content.

Overall Fire Strike score

The 3DMark Fire Strike score formula uses a weighted harmonic mean to calculate the overall score from the Graphics, Physics, and Combined scores.

$$\text{Fire Strike score} = \frac{W_{graphics} + W_{physics} + W_{combined}}{\frac{W_{graphics}}{S_{graphics}} + \frac{W_{physics}}{S_{physics}} + \frac{W_{combined}}{S_{combined}}}$$

Where:

$W_{graphics}$	=	The Graphics score weight, equal to 0.75
$W_{physics}$	=	The Physics score weight, equal to 0.15
$W_{combined}$	=	The Combined score weight, equal to 0.10
$S_{graphics}$	=	Graphics score
$S_{physics}$	=	Physics score
$S_{combined}$	=	Combined score

For a balanced system, the weights reflect the ratio of the effects of GPU and CPU performance on the overall score. Balanced in this sense means the Graphics, Physics and Combined scores are roughly the same magnitude.

For a system where either the Graphics or Physics score is substantially higher than the other, the harmonic mean rewards boosting the lower score. This reflects the reality of the user experience. For example, doubling the CPU speed in a system with an entry-level graphics card doesn't help much in games since the system is already limited by the GPU. Likewise for a system with a high-end graphics card paired with an underpowered CPU.

Graphics score

Each Graphics test produces a raw performance result in frames per second (FPS). We take a harmonic mean of these raw results and multiply it by a scaling constant to reach a Graphics score ($S_{graphics}$) as follows:

$$S_{graphics} = 230 \times \frac{2}{\frac{1}{F_{gt1}} + \frac{1}{F_{gt2}}}$$

Where:

F_{gt1} = The average FPS result from Graphics test 1

F_{gt2} = The average FPS result from Graphics test 2

The scaling constant is used to bring the score in line with traditional 3DMark score levels.

Physics score

$$S_{physics} = 315 \times F_{physics}$$

Where:

$F_{physics}$ = The average FPS result from the Physics Test

The scaling constant is used to bring the score in line with traditional 3DMark score levels.

Combined score

$$S_{combined} = 215 \times F_{combined}$$

Where:

$F_{combined}$ = The average FPS result from the Combined Test

The scaling constant is used to bring the score in line with traditional 3DMark score levels.

Fire Strike engine

Fire Strike benchmarks require graphics hardware with full DirectX 11 feature level 11 support.

Multithreading

The multithreading model is based on DX11 deferred device contexts and command lists. The engine utilizes one thread per available CPU core. One of the threads is considered as the main thread, which uses both immediate device context and deferred device context. The other threads are worker threads, which use only deferred device contexts.

Rendering workload is distributed between the threads by distributing items (e.g. geometries and lights) in the rendered scene to the threads. Each thread is assigned roughly equal amount of scene items.

When rendering a frame, each thread does the work associated to items assigned to the thread. That includes, for example, computation of transformation matrix hierarchies, computation of shader parameters (constants buffer contents and dynamic vertex data) and recording of DX API calls to a command list. When the main thread is finished with the tasks associated to its own items, it executes the command lists recorded by worker threads.

Tessellation

The engine supports rendering with and without tessellation. The supported tessellation techniques are PN Triangles, Phong, and displacement map based detail tessellation. Both triangle and quad based tessellation is supported.

Tessellation factors are adjusted to achieve desired edge length for output geometry on the render target. Additionally, patches that are back facing and patches that are outside of the view frustum are culled by setting the tessellation factor to zero.

Tessellation is turned entirely off by disabling hull and domain shaders when size of object's bounding box on render target drops below a given threshold. This applies both to g-buffer and shadow map drawing.

Lighting

Lighting is done in deferred style. Geometry attributes are first rendered to a set of render targets. Ambient occlusion is then computed from depth and normal data. Finally illumination is rendered based on those attributes.

Surface illumination

Two different surface shading models and g-buffer compositions are supported. The more complex model uses four textures and depth texture as the g-buffer. The simpler model uses two textures and depth texture.

Surface illumination model is either combination of Oren-Nayar diffuse reflectance and Cook-Torrance specular reflectance or basic Blinn Phong reflectance model. Simple surface shading model is used on Feature Level 10 demo and tests while the complex model is used on Feature Level 11 demo and tests. Optionally atmospheric attenuation is also computed.

Horizon based screen space ambient occlusion can be applied to the surface illumination.

Point, spot and directional lights are supported. Spot and directional lights can be shadowed. For spot lights, shadow texture size is selected based on size of the light volume in screen space. Shadow maps are sampled using best candidate sample distribution. Sample pattern is dithered with 4×4 pixel pattern.

Volumetric illumination

The renderer supports volume illumination. It is computed by approximating the light scattered towards the viewer by the medium between eye and the visible surface on each lit pixel. The approximation is based on volume ray casting and the Rayleigh-Mie scattering and attenuation model.

One ray is cast on each lit pixel for each light. The cast ray is sampled at several depth levels. Sampling quality is improved by dithering sampling depths with a 4×4 pixel pattern. The achieved result is blurred to combine the different sampling depths on neighboring pixels before combining the volume illumination with the surface illumination.

When rendering illumination, there are two high dynamic range render targets. One is for surface illumination and the other for volume illumination.

Particle illumination

Particle effects are rendered on top of opaque surface illumination with additive or alpha blending. Particles are simulated on the GPU. Particles can be either simply self-illuminated or receive illumination from scene lights.

Lights that participate in particle illumination can be individually selected. To illuminate particles, the selected lights are rendered to three volume textures that are fitted into view frustum. The textures contain incident radiance in each texel stored as spherical harmonics. Each of the three textures holds data for one color channel storing four coefficients. Incident radiance from each light is rendered to these volume textures as part of light rendering.

When rendering illuminated particles, hull and domain shaders are enabled. Incident radiance volume texture sampling is done in the domain shader. Tessellation factors are set to produce fixed size triangles in screen pixels. Tessellation is used to avoid sampling incident radiance textures in the pixel shader.

Particles can cast shadows on opaque surface and on other particles. For generating particle shadows, particle transmittance is first rendered to a 3D texture. The transmittance texture is rendered from the shadow casting light like a shadow map. After particles have been rendered to the texture, an accumulated transmittance 3D texture is generated by accumulating values of each depth slice in the transmittance texture. The accumulated transmittance texture can then be sampled when rendering illumination or incident radiance that is used to illuminate particles.

Post-processing

Particle based distortion

Particles can be used to generate a distortion effect. For particles that generate the effect, a distortion field is rendered to a texture using a 3D noise texture as input. This field is then used to distort the input image in post processing phase.

Depth of field

The effect is computed using the following procedure:

6. Circle of confusion radius is computed for all screen pixels and stored in a full resolution texture.
7. Half and quarter resolution versions are made from the radius texture and the original illumination texture.
8. Positions of out-of-focus pixels whose circle of confusion radius exceeds a predefined threshold are appended to a buffer.
9. The position buffer is used as point primitive vertex data and, using Geometry Shaders, the image of a hexagon-shaped bokeh is splatted to the positions of these vertices. Splatting is done to a texture that is divided into regions with different resolutions using multiple viewports. First region is screen resolution and the rest are a series of halved regions down to 1x1 texel resolution. The screen space radius of the splatted bokeh determines the used resolution. The larger the radius the smaller the used splatting resolution.
10. Steps 3 and 4 are performed separately for half and quarter resolution image data with different radius thresholds. Larger bokeh are generated from lower resolution image data.
11. The different regions of the splatting texture are combined by up-scaling the data in the smaller resolution regions step by step to the screen resolution region.
12. The out-of-focus illumination is combined with the original illumination.

Lens reflections

The effect is computed by first applying a filter to the computed illumination in frequency domain like in the bloom effect. The filtered result is then splatted in several scales and intensities on top of the input image using additive blending. The effect is computed in the same resolution as the bloom effect and therefore the forward FFT needs to be performed only once for both effects. As in the bloom effect, the forward and inverse FFTs are performed using the CS and 32bit floating point textures.

Bloom

The effect is computed by transforming the computed illumination to frequency domain using Fast Fourier Transform (FFT) and applying bloom filter to the input in that domain. An inverse FFT is then applied to the filtered image. The forward FFT, applying the bloom filter and inverse FFT are done with the CS. The effect is computed in reduced resolution. The input image resolution is halved two or three times depending on settings and then rounded up to nearest power of two. The FFTs are computed using 32bit floating point textures. A procedurally pre-computed texture is used as the bloom filter. The filter combines blur, streak, lenticular halo and anamorphic flare effects.

Anti-aliasing

MSAA and FXAA anti-aliasing methods are supported.

In MSAA method G-buffer textures are multisampled with the chosen sample count. Edge mask is generated based on differences in G-buffer sample values. The mask is used in illumination phase to select for which pixels illumination is evaluated for all G-buffer samples. For pixels that are not considered edge pixels, illumination is evaluated only for the first G-buffer sample. Volume illumination is always evaluated only for the first G-buffer sample due to its low frequency nature.




FXAA is applied after tone mapping making it the final step in post processing.

Smoke simulation

The implementation of the smoke simulation is based on Ronald Fedkiw's paper "Visual Simulation of Smoke" with the addition of viscous term as in Jos Stam's "Stable Fluids" but without a temperature simulation. Thus the smoke is simulated in a uniform grid where velocity is modeled with incompressible Euler equations. Advection is solved with a semi-Lagrangian method.

Vorticity confinement method is then applied to the velocity field to reinforce vortices. Diffusion and projection is then computed by the Jacobi iteration method. The simulation is done entirely with Compute Shaders. Cylinders that interact with the smoke are implicit objects which are voxelized into the velocity and density field in Compute Shaders.

Fire Strike version history

Version				Notes
1.1	●	×	×	Fixed issues when benchmarking systems with multiple GPUs. Scores improve significantly on systems with multiple GPUs.
1.0	●	×	×	Launch version

Fire Strike Ultra, added in 3DMark v1.4.775, uses the Fire Strike v1.1.0 workload.


3DMark v2.1.2852, released July 14, 2016, used an incorrect setting for Fire Strike Custom runs that resulted in slightly lower than expected scores. Results from Fire Strike Custom runs using that version should not be compared with any other version of 3DMark. The issue was fixed in 3DMark v2.1.2969 released August 18, 2016. The standard Fire Strike benchmark was not affected, nor were Fire Strike Extreme and Fire Strike Ultra.



Sky Diver

Sky Diver is a DirectX 11 benchmark for mid-range gaming PCs and laptops.

Use 3DMark Sky Diver to benchmark gaming PCs and laptops with mid-range graphics cards, mobile GPUs, or integrated graphics. It is especially suitable for DirectX 11 compatible systems that struggle to run the more demanding Fire Strike test.

 If your system scores more than 12000 in Sky Diver, you should run Fire Strike.

Use 3DMark Sky Diver to benchmark:

- Integrated GPUs like AMD A10-7850K and Intel i5-4570R
- Mobile integrated GPUs like AMD A10-5757M and Intel i7-4750HQ
- Mobile discrete GPUs like AMD R7 M265 and NVIDIA GT 840M
- Entry level discrete GPUs like AMD R7 240

Sky Diver includes two Graphics tests, a Physics test and a Combined test designed to stress the CPU and GPU at the same time.

Sky Diver is compatible with Windows 8 and Windows 7. A DirectX 11 compatible GPU is required. 3DMark Sky Diver runs on all DirectX 11 feature level 11_0 compatible hardware and uses optimized code paths on feature level 11_1 devices.

Sky Diver is only available in the Windows editions of 3DMark.

How is Sky Diver different from Fire Strike?

Sky Diver and Fire Strike are complementary benchmarks designed to cover the full performance range of DirectX 11 graphics hardware. Fire Strike is equivalent to a modern DirectX 11 game running on ultra-high settings. Sky Diver is equivalent to running a game on normal settings.

Scores from Sky Diver and Fire Strike are not directly comparable.

System requirements

OS ⁹	Windows 7 or later
Processor	1.8 GHz dual-core Intel or AMD CPU
Storage	6 GB free disk space
GPU	DirectX 11
For systems with integrated graphics	2.5 GB RAM ¹⁰
For systems with a discrete graphics card	2 GB RAM + 512 MB video card memory ¹¹

⁹ Windows 7 users must install Service Pack 1.

¹⁰ The benchmark tests require 2.5 GB of RAM. The demo requires 3 GB of RAM.

¹¹ The benchmark tests require 512 MB of video card memory. The demo requires 1 GB of video card memory.

Default settings

Resolution	1920 × 1080
GPU Memory Budget	1 GB
Tessellation Detail	Medium ¹²

¹² The tessellation detail setting is relative. Sky Diver's medium value is roughly the same as Fire Strike's low value.

Graphics test 1

3DMark Sky Diver Graphics test 1 focuses on tessellation. The test uses a forward lighting method with one shadow casting directional light. The test utilizes a depth of field post processing effect, which is not used in the other tests.

Processing performed in an average frame

	Vertices	Tessellation patches	Triangles	Pixels ¹³	Compute shader invocations
Sky Diver	1.6 million	150,000	3.9 million	30.3 million	0.78 million

¹³ This figure is the average number of pixels processed per frame before the image is scaled to fit the native resolution of the device being tested. If the device's display resolution is greater than the test's rendering resolution, the actual number of pixels processed per frame will be even greater.

Graphics test 2

This test focuses on pixel processing and compute shader utilization. The test uses a compute shader-based deferred tiled lighting method with screen space ambient occlusion. Post processing creates a lens reflection effect, which is not used in Graphics test 1.

Processing performed in an average frame

	Vertices	Tessellation patches	Triangles	Pixels ¹⁴	Compute shader invocations
Sky Diver	0.9 million	90,000	1.5 million	13.9 million	2.7 million

¹⁴ This figure is the average number of pixels processed per frame before the image is scaled to fit the native resolution of the device being tested. If the device's display resolution is greater than the test's rendering resolution, the actual number of pixels processed per frame will be even greater.

Physics test

3DMark Sky Diver Physics test benchmarks the hardware's ability to run gameplay physics simulations on the CPU. The GPU load is kept as low as possible to ensure that only the CPU is stressed. The test uses the Bullet Open Source Physics Library.

Sky Diver Physics test introduces a new approach to CPU testing in 3DMark designed to extend the performance range for which the test is relevant. With this new approach, the test has four levels of work. The first level is the lightest and the last is the heaviest.

The test starts with the first level and continues to the fourth level unless the frame rate drops below a minimum threshold. The score is calculated from the last two completed levels.

There are 96 simulation worlds with identical structure in total. In the first level, 8 worlds are triggered. On the second level, 16 more. On the third level, a further 24, and on the fourth and final level, another 48 so that all 96 worlds are being simulated at once.

Each world contains a statue that collapses when struck by a hammer swinging from a chain. Each statue contains 49 fragments. Each fragment is a mesh collision shape and, together, the 49 fragments have 6590 triangles. The hammer piece hangs on a chain with 39 links simulated using the Featherstone articulated body algorithm.

Combined test

This test contains both graphics workloads and physics simulations to stress the CPU and GPU.

The test uses the compute shader based deferred tiled lighting method from Graphics test 2. The CPU workload is similar to the third level of the Physics test where 48 worlds are being simulated at once.

The workloads are designed to be of equal weight so that on balanced systems both the GPU and CPU are well utilized.

The 3DMark Sky Diver Combined test uses the Bullet Open Source Physics Library.

Processing performed in an average frame

	Vertices	Tessellation patches	Triangles	Pixels ¹⁵	Compute shader invocations
Sky Diver	1.3 million	100,000	1.6 million	29.6 million	2.5 million

¹⁵ This figure is the average number of pixels processed per frame before the image is scaled to fit the native resolution of the device being tested. If the device's display resolution is greater than the test's rendering resolution, the actual number of pixels processed per frame will be even greater.

Scoring

Sky Diver produces an overall Sky Diver score, a Graphics test sub-score, a Physics test sub-score, and a Combined test sub-score. The scores are rounded to the nearest integer. The higher the score, the better the performance.

Overall Sky Diver score

The 3DMark Sky Diver score formula uses a weighted harmonic mean to calculate the overall score from the Graphics, Physics, and Combined scores.

$$\text{Sky Diver score} = \frac{W_{graphics} + W_{physics} + W_{combined}}{\frac{W_{graphics}}{S_{graphics}} + \frac{W_{physics}}{S_{physics}} + \frac{W_{combined}}{S_{combined}}}$$

Where:

$W_{graphics}$	=	The Graphics score weight, equal to 0.75
$W_{physics}$	=	The Physics score weight, equal to 0.15
$W_{combined}$	=	The Physics score weight, equal to 0.10
$S_{graphics}$	=	Graphics score
$S_{physics}$	=	Physics score
$S_{combined}$	=	Combined score

For a balanced system, the weights reflect the ratio of the effects of graphics and physics performance on the overall score. Balanced in this sense means the Graphics, Physics and Combined scores are roughly the same magnitude.

For a system where either the Graphics or Physics score is substantially higher than the other, the harmonic mean rewards boosting the lower score. This reflects the reality of the user experience. For example, doubling the CPU speed in a system with an entry-level graphics card doesn't help much in games since the system is already limited by the GPU. Likewise for a system with a high-end graphics card paired with an underpowered CPU.

Graphics score

Each Graphics test produces a raw performance result in frames per second (FPS). We take a harmonic mean of these raw results and multiply it with a scaling constant to reach a Graphics score ($S_{graphics}$) as follows:

$$S_{graphics} = 219 \times \frac{2}{\frac{1}{F_{gt1}} + \frac{1}{F_{gt2}}}$$

Where:

F_{gt1} = The average FPS result from Graphics Test 1
 F_{gt2} = The average FPS result from Graphics Test 2

The scaling constant is used to bring the score in line with traditional 3DMark score levels.

Physics score

3DMark Sky Diver Physics test uses a different approach to testing than that used in Fire Strike, Cloud Gate and Ice Storm. The aim of the new approach is to extend the performance range for which the test is relevant.

The test has four levels of work. The first level is the lightest and the last is the heaviest. The test begins with the first level and continues until the frame rate drops below a minimum threshold L_{low} , or until the last available level is run.

Each level produces a raw performance result in frames per second (FPS). The score is defined as a weighted average of the two highest successfully completed levels.

$$S_{physics} = 56 \times ((1 - W_i)N_{i-1}F_{i-1} + W_iN_iF_i)$$

Where:

W = The weighting factor for a level
 i = The index of the last level to run
 N = The frame rate normalization factor for a level
 F = The frame rate of a level

The weight W for a level is defined as:

$$W_i = \min\left(1, \frac{F_{i-1} - L_{low}}{L_{high} - L_{low}}\right)$$

Where:

L_{low} = The minimum frame rate threshold, set to 30 FPS
 L_{high} = Upper frame rate threshold used for weighting, set to 40 FPS

When the first level is the last level to run above L_{low} then the score is defined as follows:

$$S_{physics} = 56 \times F_1$$

Frame rate normalization factors are used to normalize the frame rates of different levels before using them in the score calculation. A set of reference CPUs was used to define the factors.

Reference CPUs for N_2	Level 1 frame rate	Level 2 frame rate	Relative difference
AMD A4-5150M	30.53	17.14	1.78
Intel Core i5-4200U	56.49	33.43	1.69
Reference CPUs for N_3	Level 2 frame rate	Level 3 frame rate	Relative difference
AMD A10-7850K	51.48	28.84	1.79
Intel Core i5-4430	68.58	39.62	1.73
Reference CPUs for N_4	Level 3 frame rate	Level 4 frame rate	Relative difference
AMD A10-7850K	28.84	16.57	1.74
Intel Core i7-4770K	55.30	31.87	1.74

The following table defines values for the frame rate normalization factors. N_1 is always set to 1. N_{i+1} is the average relative frame rate difference of levels i and $i + 1$ on the reference CPUs multiplied by N_i .

N_1	N_2	N_3	N_4
1.000	1.735	3.054	5.314

Combined score

$$S_{combined} = 243 \times F_{combined}$$

Where:

$F_{combined}$ = The average FPS result from the Combined Test

The scaling constant is used to bring the score in line with traditional 3DMark score levels.

Sky Diver engine

Multithreading

The engine utilizes one thread per available CPU core, less one physical core that is left free for the display driver. Draw calls are issued through immediate device context only.

Tessellation

The engine supports rendering with and without tessellation. The supported tessellation techniques are Phong tessellation and displacement map based detail tessellation.

Tessellation factors are adjusted to achieve desired edge length for output geometry on the render target. Additionally, patches that are back facing and patches that are outside of the view frustum are culled by setting the tessellation factor to zero.

Tessellation is turned entirely off by disabling hull and domain shaders when size of object's bounding box on render target drops below a given threshold.

Lighting

The engine supports two alternative methods of lighting the scene.

Forward lighting

The forward lighting method is used for the first part of the Demo and Graphics test 1.

It supports one shadow casting directional light and a limited number of additional un-shadowed point lights as well as cube map-based ambient illumination. All lights are rendered in one pass to DXGI_FORMAT_R11G11B10_FLOAT texture.

Compute shader-based tiled deferred lighting

The compute shader based tiled deferred lighting method is used the second part of the Demo, Graphics test 2 and the Combined test.

It supports point lights, spot lights and cube map-based ambient illumination. The geometry is first rendered to gbuffer that contains depth, normal and surface illumination parameters stored in three textures with DXGI_FORMAT_D24_UNORM_S8_UINT, DXGI_FORMAT_R10G10B10A2_UNORM and DXGI_FORMAT_R8G8B8A8_UNORM_SRGB formats. Screen space ambient occlusion is computed to a DXGI_FORMAT_R8_UNORM texture.

Lighting is evaluated in one compute shader pass that splits the screen to tiles and culls scene lights for each tile evaluating the illumination for visible lights on each tile. Lighting is rendered to a DXGI_FORMAT_R11G11B10_FLOAT texture.

Particles

Particle effects are rendered on top of opaque surface illumination with additive or alpha blending. Particles are simulated on the GPU. Particles are simply self-illuminated.

Post-processing

Depth of field

The effect is computed using the following procedure:

1. Using Compute Shader, the circle of confusion radius is computed for all screen pixels based on depth texture and the information is reduced to half and quarter resolutions. In the same CS pass, data about out-of-focus pixels whose circle of confusion radius exceeds a predefined threshold is appended to a buffer.
2. The buffer with information about the out-of-focus pixels is used as point primitive vertex data and, using Geometry Shader, the image of a hexagon-shaped bokeh is splatted to the positions of these vertices. Splatting is done to a texture that is divided into regions with different resolutions using multiple viewports. First region is screen resolution and the rest are a series of halved regions down to 1x1 texel resolution. The screen space radius of the splatted bokeh determines the used resolution. The larger the radius the smaller the used splatting resolution.
3. The different regions of the splatting texture are combined by up-scaling the data in the smaller resolution regions step by step to the screen resolution region.
4. The out-of-focus illumination is combined with the original illumination.




Bloom

The effect is computed by transforming the computed illumination to frequency domain using Fast Fourier Transform (FFT) and applying bloom filter to the input in that domain. An inverse FFT is then applied to the filtered image. The forward FFT, applying the bloom filter and inverse FFT are done with the Compute Shader. The effect is computed in reduced resolution. The input image resolution is halved three times and rounded up to nearest power of two. With the 1920×1080 screen resolution, 256×256 resolution is used to perform the FFT. `DXGI_FORMAT_R16G16B16A16_FLOAT` textures are used to store the frequency domain data. A procedurally pre-computed texture is used as the bloom filter. The filter combines blur, streak, lenticular halo and anamorphic flare effects.

Windows

3DMark Sky Diver runs on all DirectX 11 feature level 11_0 compatible hardware and uses optimized code paths on feature level 11_1 devices.

Sky Diver version history

Version				Notes
1.0.0	●	×	×	Launch version



Cloud Gate

Cloud Gate is a new test designed for Windows notebooks and typical home PCs. It is a particularly good benchmark for systems with integrated graphics. Cloud Gate includes two graphics tests and a physics test. The benchmark uses a DirectX 11 engine limited to Direct3D feature level 10 making it suitable for testing DirectX 10 compatible hardware. Cloud Gate is only available in the Windows edition of 3DMark.

- Designed for typical home PCs and notebooks.
- DirectX 11 engine supporting DirectX 10 hardware.
- Includes two graphics tests and a physics test.

3DMark Cloud Gate and 3DMark Vantage compared

3DMark Vantage and 3DMark Cloud Gate are both benchmarks for DirectX 10 compatible hardware. The difference is in the engine powering each benchmark.

3DMark Vantage, released in April 2008, uses a DirectX 10 engine. 3DMark Cloud Gate uses a DirectX 11 engine limited to Direct3D feature level 10. Using Direct3D feature levels is the modern approach to game engine design as it allows developers to use a DirectX 11 engine and still support older generation hardware all the way down to DirectX 9 level models.

We recommend using 3DMark Cloud Gate for testing DirectX 10 based systems. Scores from 3DMark Vantage and 3DMark Cloud Gate cannot be directly compared.

System requirements

OS ¹⁶	Windows 7 or later
Processor	1.8 GHz dual-core Intel or AMD CPU
Memory	2 GB
Storage	6 GB free disk space
GPU	DirectX 10
Video card memory	256 MB

¹⁶ Windows 7 users must install Service Pack 1.

Default settings

Rendering Resolution	1280 × 720
GPU memory Budget	256 MB
Shadow Sample Count	4
Shadow Map Resolution	1024
Depth of Field Quality	Low
Bloom Resolution	1/8

Graphics test 1

Cloud Gate Graphics test 1 has an emphasis on geometry processing while having simple shaders. Volumetric illumination is disabled, but the scene contains particle effects. FFT based bloom effects and a depth of field effect are added as post processing steps.

Processing performed in an average frame

	Vertices	Triangles	Pixels ¹⁷
Cloud Gate	3.0 million	1.1 million	15.6 million

¹⁷ This figure is the average number of pixels processed per frame before the image is scaled to fit the native resolution of the device being tested. If the device's display resolution is greater than the test's rendering resolution, the actual number of pixels processed per frame will be even greater.

Graphics test 2

Cloud Gate Graphics test 2 has shaders that are more mathematically complex than Graphics test 1, but has less geometry to process. Simple volumetric illumination is used, but the scene has no particle effects. Post processing steps are similar to Graphics test 1.

Processing performed in an average frame

	Vertices	Triangles	Pixels ¹⁸
Cloud Gate	1.8 million	690,000	16.3 million

¹⁸ This figure is the average number of pixels processed per frame before the image is scaled to fit the native resolution of the device being tested. If the device's display resolution is greater than the test's rendering resolution, the actual number of pixels processed per frame will be even greater.

Physics test

The Cloud Gate Physics test benchmarks the hardware's ability to run gameplay physics simulations on CPU. The GPU load is kept as low to ensure that only the CPU is stressed.

The test has 32 simulated worlds. Each world has 4 soft bodies, 4 joints and 20 rigid bodies colliding with each other. The rigid bodies are invisible and are there to cause the blast effect on the soft bodies.

The simulations run on one thread per available CPU core. All physics are computed on the CPU with soft body vertex data updated to the GPU each frame. Each world also has one CPU simulated particle system. The Physics test uses a forward renderer for minimum GPU load.

The test duration is 20 seconds but the score calculation begins after 8 seconds. The first 8 seconds skipped to allow all simulated objects to actively participate in simulation.

The Cloud Gate Physics test uses the Bullet Open Source Physics Library.

Scoring

Overall Cloud Gate score

The 3DMark Cloud Gate score formula uses a weighted harmonic mean to calculate the overall score from the Graphics and Physics scores.

$$\text{Cloud Gate score} = \frac{W_{\text{graphics}} + W_{\text{physics}}}{\frac{W_{\text{graphics}}}{S_{\text{graphics}}} + \frac{W_{\text{physics}}}{S_{\text{physics}}}}$$

Where:

W_{graphics}	=	The Graphics score weight, equal to 7/9
W_{physics}	=	The Physics score weight, equal to 2/9
S_{graphics}	=	Graphics score
S_{physics}	=	Physics score

For a balanced system, the weights reflect the ratio of the effects of graphics and physics performance on the overall score. Balanced in this sense means the Graphics and Physics sub-scores are roughly the same magnitude.

For a system where either the Graphics or Physics score is substantially higher than the other, the harmonic mean rewards boosting the lower score. This reflects the reality of the user experience. For example, doubling the CPU speed in a system with an entry-level graphics card doesn't help much in games since the system is already limited by the GPU. Likewise for a system with a high-end graphics card paired with an underpowered CPU.

Graphics score

Each Graphics test produces a raw performance result in frames per second (FPS). We take a harmonic mean of these raw results and multiply it with a scaling constant to reach a Graphics score (S_{graphics}) as follows:

$$S_{\text{graphics}} = 230 \times \frac{2}{\frac{1}{Fgt1} + \frac{1}{Fgt2}}$$

Where:

$Fgt1$	=	The average FPS result from Graphics test 1
$Fgt2$	=	The average FPS result from Graphics test 2

The scaling constant is used to bring the score in line with traditional 3DMark score levels.

Physics score

The Physics score is calculated from the raw performance result in frames per second (FPS) of the Physics test.

$$S_{physics} = 315 \times F_{physics}$$

Where:

$$F_{physics} = \text{The average FPS result from the Physics Test}$$




The scaling constant is used to bring the score in line with traditional 3DMark score levels.

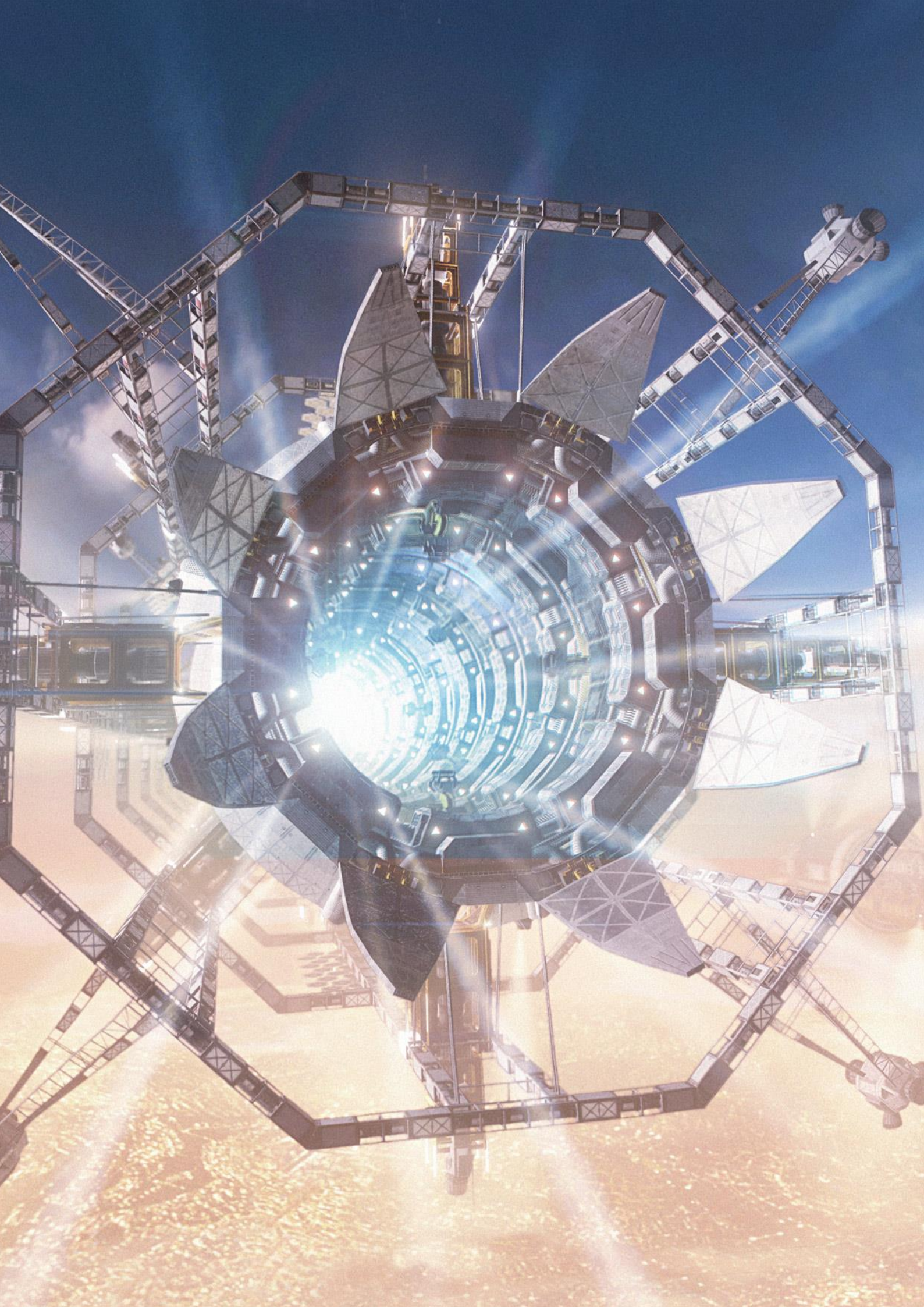
Cloud Gate engine

Cloud Gate tests use same engine as Fire Strike, but with a reduced set of features including a simplified lighting model and some fall-backs implemented for Direct3D feature level 10.

Cloud Gate requires graphics hardware with support for Direct3D feature level 10 or greater.

Cloud Gate version history

Version				Notes
1.1.0	●	×	×	Fixed issues when benchmarking systems with multiple GPUs. Scores improve significantly on systems with multiple GPUs.
1.0.0	●	×	×	Launch version



Sling Shot

Sling Shot is a cross-platform benchmark for modern mobile devices. Use it to compare mainstream Android devices with popular iPhone and iPad models.

Use Sling Shot Extreme to compare high-end Android phones and tablets with the latest Apple devices.

- Designed for the latest high-end smartphones and tablets.
- Mobile-optimized rendering engine using OpenGL ES 3.x and Metal.
- Benchmark the GPU with Graphics tests and the CPU with the Physics test.

Android devices must have Android 5.0 or later and support OpenGL ES 3.0 to run Sling Shot and Open GL ES 3.1 to run Sling Shot Extreme.

On iOS, Sling Shot and Sling Shot Extreme compatibility starts with iPhone 5s, iPad Air, iPad mini 2, and runs to the latest models.

Sling Shot

Use 3DMark Sling Shot to compare mainstream Android smartphones and tablets with popular iPhone and iPad models.

Sling Shot is a demanding OpenGL ES 3.0 benchmark that tests the full range of API features including multiple render targets, instanced rendering, uniform buffers and transform feedback. It includes volumetric lighting and particle illumination, as well as depth of field and bloom post-processing effects.

The Graphics tests are rendered at 1920×1080 before being scaled to the device's display resolution. The Physics test is rendered at 1280×720 to ensure that GPU performance is not a limiting factor.

Use **Sling Shot Unlimited** to make chip-to-chip comparisons without vertical sync, display resolution scaling and other operating system factors affecting the result. Sling Shot Unlimited uses the same content and settings as Sling Shot but runs offscreen using a fixed time step between frames. It renders exactly the same frames in every run on every device. The display is updated with frame thumbnails every 100 frames to show progress.

Sling Shot Extreme

Run Sling Shot Extreme to compare high-end Android phones and tablets with the latest Apple devices.

Sling Shot Extreme uses OpenGL ES 3.1 on Android and the Metal API on Apple devices. It tests the full range of API features including multiple render targets, instanced rendering, and uniform buffers. It also includes volumetric lighting, depth of field and bloom post-processing effects using Compute Shaders.

The Graphics tests are rendered at 2560×1440 then scaled to the device's native display resolution. The Physics test is rendered at 1280×720 to ensure that GPU performance is not a limiting factor.

Use **Sling Shot Extreme Unlimited** to make chip-to-chip comparisons without vertical sync, display resolution scaling and other operating system factors affecting the result. Sling Shot Extreme Unlimited uses the same content and settings as Sling Shot Extreme but runs offscreen using a fixed time step between frames. It renders exactly the same frames in every run on every device. The display is updated with frame thumbnails every 100 frames to show progress.

Cross-platform benchmarking


The rows in the table below show valid cross-platform comparisons, i.e. you can compare Sling Slot scores across platforms, and you can compare Sling Shot Extreme scores across platforms. But you cannot compare Sling Shot scores with Sling Shot Extreme scores. Though they appear to be similar, they use different rendering resolutions and post-processing techniques.

When comparing scores across platforms, note that the test results reflect both hardware and software. APIs with low overhead, such as Metal and Vulkan, can deliver more performance than OpenGL ES even on devices with similar hardware.

Benchmark	Rendering resolution	Android API	iOS API
Sling Shot	1920 × 1080	Open GL ES 3.0	Open GL ES 3.0
Sling Shot Extreme	2560 × 1440	Open GL ES 3.1	Metal

Sling Shot results are not comparable with Cloud Gate


Even though the Sling Shot and Cloud Gate benchmarks share content they are separate tests. To create Sling Shot, the assets and rendering techniques used in Cloud Gate were modified to make them suitable for mobile hardware. In broad terms, Sling Shot is a lighter test than Cloud Gate.

 Sling Shot scores from mobile devices should not be compared with Cloud Gate scores from Windows PCs.

Device requirements

Android

	Sling Shot	Sling Shot Extreme
OS	Android 5.0	Android 5.0
Memory	1 GB	1.5 GB
Graphics	OpenGL ES 3.0	OpenGL ES 3.1
Storage ¹⁹	203 MB	203 MB

 Sling Shot is temporarily disabled for ASUS Fonepad 8 FE380CG due to problems with the driver included in the Android 5.0 OS image. Intel has been notified.

Apple iOS

	Sling Shot	Sling Shot Extreme
OS	iOS 9	iOS 9
Memory	1 GB	1.5 GB
Graphics	OpenGL ES 3.0	Metal
Storage	135 MB	135 MB

¹⁹ With the Android app you can choose to install only the tests you need. This figure is the storage space required for installing the 3DMark app and the Sling Shot test. The total install size for 3DMark with all tests is 339 MB.

Default settings

	Sling Shot	Sling Shot Extreme
Resolution	1920 × 1080	2560 × 1440
GPU Memory Budget	1 GB	1.5 GB
Bloom/FFT	Using Pixel Shaders	Using Compute Shaders

Graphics test 1

Sling Shot Graphics test 1 has an emphasis on geometry processing while having simple shaders. Volumetric illumination is disabled, but the scene contains particle effects. FFT-based bloom effects and a depth of field effect are added as post processing steps. In Sling Shot Extreme, the bloom effects use Compute Shaders. In Sling Shot, they use Pixel shaders.

- No volumetric illumination
- Particle effects
- Post-processing

Processing performed in an average frame

	Resolution ²⁰	Vertices	Triangles	Pixels ²¹
Sling Shot Extreme	2560 × 1440	620,000	320,000	33.6 million
Sling Shot	1920 × 1080	620,000	320,000	20.4 million

For contrast

	Resolution	Vertices	Triangles	Pixels
Cloud Gate	1280 × 720	3.0 million	1.1 million	15.6 million

²⁰ This is the resolution used to render the Graphics tests. The Physics test is rendered at 1280 x 720 to ensure that GPU performance is not a limiting factor.

²¹ This figure is the average number of pixels processed per frame before the image is scaled to fit the native resolution of the device being tested. If the device's display resolution is greater than the test's rendering resolution, the actual number of pixels processed per frame will be even greater.

Graphics test 2

Sling Shot Graphics test 2 has shaders that are more mathematically complex than Graphics test 1, but has less geometry to process. Simple volumetric illumination is used, but the scene has no particle effects. Post processing steps are similar to Graphics test 1.

- Volumetric illumination
- No particle effects
- Post-processing

Processing performed in an average frame

	Resolution ²²	Vertices	Triangles	Pixels ²³
Sling Shot Extreme	2560 × 1440	427,000	220,000	35.1 million
Sling Shot	1920 × 1080	427,000	220,000	20.8 million

For contrast

	Resolution	Vertices	Triangles	Pixels
Cloud Gate	1280 × 720	1.8 million	690,000	16.3 million

²² This is the resolution used to render the Graphics tests. The Physics test is rendered at 1280 x 720 to ensure that GPU performance is not a limiting factor.

²³ This figure is the average number of pixels processed per frame before the image is scaled to fit the native resolution of the device being tested. If the device's display resolution is greater than the test's rendering resolution, the actual number of pixels processed per frame will be even greater.

Physics Test

The test has three levels with different workloads. The first level is the lightest and the last is the heaviest. The purpose of the three levels is to extend the performance range for which the test is relevant.

The physics test is run with a fixed timestep at 30 FPS. The physics test always begins with the first level and continues to the next level until either the test is finished or 90 seconds have passed. If a level did not finish completely, then it will contribute proportionally less to the final score. The final score is a weighted sum of all levels.

The first level of the test has 8 simulation worlds running in separate threads. Each world has one soft body with 107 vertices and 64 rigid bodies. The rigid bodies are invisible and are there to cause the blast effect to soft bodies. Additionally, there are 32 CPU simulated particle systems with about 500 particles in each.

The second level of the test adds 8 simulation worlds running in separate threads. Each new world has one soft body with 499 vertices and additionally 64 rigid bodies.

The third level of the test adds another 16 simulation worlds running in separate threads. Each new world has one soft body with 499 vertices and additionally 64 rigid bodies.

Each of the 32 worlds is simulated at 30 FPS and the whole test takes 16 seconds at 30 FPS. The first level starts at time 0, the second level starts at time 5 and the third level at time 10.

All physics are computed on CPU. Soft body vertex data is updated to GPU on each frame.

The Bullet Open Source Physics C++ Library version 2.83 alpha is used for physics computation.

Scoring

Sling Shot scores are rounded to the nearest integer. Higher is better.

Graphics Test scoring

Each of the Graphics Tests produces a raw performance result in frames per second (FPS). A harmonic mean of the raw results is evaluated and multiplied with a scaling constant to reach a Graphics score as follows:

$$S_{graphics} = 230 \frac{2}{\frac{1}{F_{g1}} + \frac{1}{F_{g2}}}$$

Where:

$S_{graphics}$	= Graphics score
F_{g1}	= Average frames per second in Graphics test 1
F_{g2}	= Average frames per second in Graphics test 2

The scaling constant is used to bring the score in line with traditional 3DMark score levels.

Physics Test scoring

The levels of the physics test produce a raw performance result in frames per second (FPS). The score is defined as a sum of raw results from levels that can be completed before a given time limit of 90 seconds. A scaling constant is used to reach the final physics score.

The score is defined as follows:

$$S_{physics} = 9 (N_{p1}W_{p1} + N_{p2}W_{p2} + N_{p3}W_{p3})$$

Where:

$S_{physics}$	= Physics score
$C_{physics}$	= A scaling constant, set to 9
N_{pn}	= a frame rate normalization factor for level N

The scaling constant is used to bring the score in line with traditional 3DMark score levels.

W_{pn} is defined as follows for the default version of the test:

$$W_{pn} = \max(0, \min(Lhigh, F_n))$$

Where:

$Lhigh$ = A maximum frame rate limit set to 60
 F_n = Average frames per second for the current level

And as follows for the Unlimited version of the test:

$$W_{pn} = \max(0, F_n)$$

Where:

F_n = Average frames per second for the current level

The frame rate normalization factors N_{p1} , N_{p2} , and N_{p3} are used to normalize the frame rates of the different levels before using them in score calculation. A set of reference CPUs is used to define the factors.

- N_{p1} is always set to 1.
- N_{p2} is the average relative frame rate difference of levels 1 and 2 on the reference CPUs.
- N_{p3} is the average relative frame rate difference of levels 2 and 3 on the reference CPUs multiplied by N_{p2} .

The following table lists the CPUs that are used as reference.

Reference CPUs for Np2	Level 1 frame rate	Level 2 frame rate	Relative difference
Apple A7 (iPhone 5s)	36.69	6.35	5.78
Qualcomm Snapdragon 800 (LG Nexus 5)	40.19	8.59	4.68
Qualcomm Snapdragon 805 (Motorola Nexus 6)	46.67	9.89	4.71

Reference CPUs for Np2	Level 1 frame rate	Level 2 frame rate	Relative difference
Tegra K1 (NVIDIA SHIELD)	59.05	20.59	2.87
AMD A4-5150M	112.36	13.57	8.28

Reference CPUs for Np3	Level 2 frame rate	Level 3 frame rate	Relative difference
AMD A10-4600M	70.79	6.15	11.51
Intel Core i5-3317U	79.50	18.17	4.38
Intel Core i7 920	226.89	145.83	1.56

The following table defines values for the frame rate normalization factors.

N_{p1}	1.00
N_{p2}	3.26
N_{p3}	10.60

3DMark Sling Shot score

The 3DMark Sling Shot and Sling Shot Extreme score is formed from the Graphics score and Physics score using a weighted harmonic mean as follows:

$$S_{3DMark} = \frac{W_{graphics} + W_{physics}}{\frac{W_{graphics}}{S_{graphics}} + \frac{W_{physics}}{S_{physics}}}$$

Where:

$W_{graphics}$ = the weight for the Graphics score, set to 7/9
 $W_{physics}$ = the weight for the Physics score, set to 2/9

⚠ Even though the tests appear to be similar, Sling Shot scores should not be compared with scores from Sling Shot Extreme. The two tests use different rendering resolutions and post-processing techniques making their scores incomparable.

Sling Shot engine

Rendering

Multithreading

The engine utilizes one thread per available CPU core. One of the threads is considered as the main thread, which makes the graphics API calls. The other threads are worker threads, which do not make API calls.

The rendering workload is distributed between the threads by distributing items (e.g. geometries and lights) in the rendered scene to the threads. Each thread is assigned roughly equal amount of scene items. When rendering a frame, each thread does the work associated to items assigned to the thread. That includes, for example, computation of transformation matrix hierarchies and computation of shader parameters (constants buffer contents and dynamic vertex data). When the main thread is finished with the tasks associated to its own items, it executes API calls for items assigned to the worker threads.

Lighting

Lighting is done in deferred style. Geometry attributes are first rendered to a set of render targets. Finally illumination is rendered based on those attributes.

Surface illumination

The g-buffer is composed from two 32 bits per pixel textures and a depth texture. Surface illumination model the basic Blinn Phong reflectance model.

Point, spot and directional lights are supported. Spot and directional lights can be shadowed. For spot lights, shadow texture size is selected based on size of the light volume in screen space. Shadow maps are sampled using best candidate sample distribution. Sample pattern is dithered with 4x4 pixel pattern.

Volumetric illumination

The renderer supports volume illumination. It is computed by approximating the light scattered towards the viewer by the medium between eye and the visible surface on each lit pixel. The approximation is based on volume ray casting and simple scattering and attenuation model.

One ray is cast on each lit pixel for each light. The cast ray is sampled at several depth levels. Sampling quality is improved by dithering sampling depths with a 4x4 pixel pattern. The achieved result is blurred to combine the different sampling depths on neighboring pixels before combining the volume illumination with the surface illumination.

When rendering illumination, there are two high dynamic range render targets. One is for surface illumination and the other for volume illumination.

Particle illumination

Particle effects are rendered on top of opaque surface illumination with additive or alpha blending. Particles are simulated on the GPU utilizing transform feedback. Particles are simply self-illuminated.

Post-processing

Depth of field

The effect is computed by filtering rendered illumination in half resolution with three separable skewed box filters that form hexagonal bokeh pattern when combined.

The filtering is performed in two passes that exploit similarities in the three filters to avoid duplicate work.

The first pass renders to two render targets and the second pass the one target combining results of the three filters. Before filtering, a circle of confusion radius is evaluated for each pixel and the illumination is premultiplied with the radius.

After filtering, illumination is reconstructed by dividing the result with the radius. This makes the filter gather out of focus illumination and prevents it from bleeding in focus illumination to neighbor pixels.

Bloom




The effect is computed by transforming the computed illumination to frequency domain using Fast Fourier Transform (FFT) and applying a bloom filter to the input in that domain. An inverse FFT is then applied to the filtered image.

The forward FFT, applying the bloom filter and inverse FFT are done using the fragment shader. The FFT is performed with Cooley-Tukey algorithm as a series of render passes.

The effect is computed in 256×256 resolution in both Sling Shot and Sling Shot Extreme. In Sling Shot, the FFTs are computed using 16-bit floating point textures. In Sling Shot Extreme, the FFTs are computed using 32-bit floating point textures. A procedurally pre-computed texture is used as the bloom filter. The filter combines blur, streak, lenticular halo and anamorphic flare effects.

With Sling Shot Extreme, Compute Shaders are used for the FFT and bloom. A total of 256 invocations within a work group is required.

Sling Shot version history

Version				Notes
2.2	×	×	●	Fixed a Physics test bug introduced in 3DMark Sling Shot version 1.0.745. Scores from 2.2 workloads are again comparable across platforms and devices.
2.0	×	●	●	Improved compatibility with ARM Mali GPUs. Scores from v2.0 are not comparable with v1.0 results.
1.0	×	●	×	Launch version



Ice Storm


Ice Storm is a cross-platform benchmark for low cost, basic smartphones and tablets and older mobile devices.

Ice Storm includes two Graphics tests focusing on GPU performance and a Physics test targeting CPU performance.

On Android and iOS, Ice Storm uses OpenGL ES 2.0. On Windows, Ice Storm uses a DirectX 11 engine limited to Direct3D feature level 9.

Ice Storm's test content, settings and rendering resolution are the same on all platforms and scores can be compared across Windows, Android and iOS.

- Cross-platform benchmark for older mobile devices.
- Includes two Graphics tests and a Physics test.
- Compare scores across Windows, Android and iOS.

 With most modern mobile devices you will get more useful results by benchmarking with one of the Sling Shot tests.

System requirements

Windows

	Ice Storm Ice Storm Unlimited	Ice Storm Extreme
OS ²⁴	Windows 7 or later	Windows 7 or later
Processor	1.8 GHz dual-core Intel or AMD CPU	1.8 GHz dual-core Intel or AMD CPU
Memory	2 GB	4 GB
Storage	6 GB free disk space	6 GB free disk space
GPU ²⁵	DirectX 9	DirectX 9
Video card memory	128 MB	256 GB

Windows RT

OS	Windows RT
Memory	1 GB
Device	All Windows RT devices
Storage	151 MB

²⁴ Windows 7 users must install Service Pack 1.

²⁵ DirectX 9 hardware needs Shader Model 3.0 support, 128 MB and WDDM 1.1 drivers. Note that ATI Radeon X1x00 series cards do not have WDDM 1.1 drivers available and cannot run 3DMark. The oldest cards confirmed to work with 3DMark are Radeon HD 2x00 series (Ice Storm, Cloud Gate), NVIDIA GeForce 7x00 series (Ice Storm) and Intel GMA X4500 (Ice Storm).

Android

OS	Android 4.0 ²⁶
Memory	1 GB
Graphics	OpenGL ES 2.0
Storage	235 MB ²⁷

Apple iOS

OS	iOS 6.0
Memory	512 MB
Device	iPhone 4, iPad 2, iPod touch (5 th Gen)
Storage	174 MB

²⁶ Minimum Android requirement raised from 3.1 to 4.0 with 3DMark Android version [1.3.1309](#).

²⁷ With 3DMark on Android you can choose to install only the tests you need. This figure is the storage space required for installing the 3DMark app and the Ice Storm test. The total install size for 3DMark with all tests is 339 MB.

Ice Storm

Rendering resolution	1280 × 720
GPU memory budget	128 MB
Texture quality	Low
Bloom resolution	1/8

Use Ice Storm for device-to-device comparisons of older mobile devices. Ice Storm is rendered at a fixed 1280 × 720 resolution and then scaled to the native resolution of the display. This is the best approach for ensuring that devices can be compared fairly.

Many mobile devices lock their display refresh rate to 60 Hz and force the use of vertical sync. If your device is able to run this test at more than 60 frames per second you will be prompted to run a more demanding test instead.

Ice Storm Unlimited

Use Ice Storm Unlimited to make chip-to-chip comparisons. Ice Storm Unlimited uses the same content and settings as Ice Storm but runs offscreen using a fixed time step between frames. Unlimited mode renders exactly the same frames in every run on every device. The display is updated with frame thumbnails every 100 frames to show progress.

Ice Storm Unlimited measures the performance of the device hardware without vertical sync, display resolution scaling and other operating system factors affecting the result.

Ice Storm Extreme

Graphics tests rendering resolution	1920 × 1080
Physics test rendering resolution	1280 × 720
GPU memory budget	256 MB
Texture Quality	High
Bloom resolution	1/4

Use Ice Storm Extreme for device-to-device comparisons of low cost, basic model mobile devices.

Ice Storm Extreme raises the Graphics tests rendering resolution from 1280 × 720 to 1920 × 1080 and uses higher quality textures and post-processing effects to create a more demanding load. The Physics test renders at 1280 × 720 to ensure performance is not limited by the GPU.

Many mobile devices lock their display refresh rate to 60 Hz and force the use of vertical sync. If your device is able to run this test at more than 60 frames per second you will be prompted to run a more demanding test instead.

Graphics test 1

Ice Storm Graphics test 1 stresses the hardware's ability to process lots of vertices while keeping the pixel load relatively light. Hardware on this level may have dedicated capacity for separate vertex and pixel processing. Stressing both capacities individually reveals the hardware's limitations in both aspects. Pixel load is kept low by excluding expensive post processing steps, and by not rendering particle effects.

Processing performed in an average frame

	Vertices	Triangles	Pixels ²⁸
Ice Storm	530,000	180,000	1.9 million
Ice Storm Extreme	580,000	190,000	4.4 million

²⁸ This figure is the average number of pixels processed per frame before the image is scaled to fit the native resolution of the device being tested. If the device's display resolution is greater than the test's rendering resolution, the actual number of pixels processed per frame will be even greater.

Graphics test 2

Graphics test 2 stresses the hardware's ability to process lots of pixels. It tests the ability to read textures, do per pixel computations and write to render targets. The additional pixel processing compared to Graphics test 1 comes from including particles and post processing effects such as bloom, streaks and motion blur. The numbers of vertices and triangles are considerably lower than in Graphics test 1 because shadows are not drawn and the processed geometry has a lower number of polygons.

Processing performed in an average frame

	Vertices	Triangles	Pixels ²⁹
Ice Storm	79,000	26,000	7.8 million
Ice Storm Extreme	89,000	28,000	18.6 million

²⁹ This figure is the average number of pixels processed per frame before the image is scaled to fit the native resolution of the device being tested. If the device's display resolution is greater than the test's rendering resolution, the actual number of pixels processed per frame will be even greater.

Physics test

The purpose of the Physics test is to benchmark the hardware's ability to do gameplay physics simulations on CPU. The GPU load is kept as low as possible to ensure that only the CPU's capabilities are stressed.

The test has four simulated worlds. Each world has two soft bodies and two rigid bodies colliding with each other. One thread per available CPU core is used to run simulations. All physics are computed on the CPU with soft body vertex data updated to the GPU each frame. The background is drawn as a static image for the least possible GPU load.

The Ice Storm Physics test uses the Bullet Open Source Physics Library.

Scoring

Scores from individual Ice Storm benchmarks can be compared across platforms, for example you can compare 3DMark Ice Storm Extreme scores from Android and iOS devices.

Scores from different benchmarks should not be compared to each other. Ice Storm, Ice Storm Unlimited and Ice Storm Extreme are separate tests with their own scores, even though they share the same content.

Overall Ice Storm score

The 3DMark Ice Storm score formula uses a weighted harmonic mean to calculate the overall score from the Graphics and Physics scores.

$$\text{Ice Storm score} = \frac{W_{graphics} + W_{physics}}{\frac{W_{graphics}}{S_{graphics}} + \frac{W_{physics}}{S_{physics}}}$$

Where:

$W_{graphics}$	=	The Graphics score weight, equal to 7/9
$W_{physics}$	=	The Physics score weight, equal to 2/9
$S_{graphics}$	=	Graphics score
$S_{physics}$	=	Physics score

For a balanced system, the weights reflect the ratio of the effects of graphics and physics performance on the overall score. Balanced in this sense means the Graphics and Physics sub-scores are roughly the same magnitude.

For a system where either the Graphics or Physics score is substantially higher than the other, the harmonic mean rewards boosting the lower score. This reflects the reality of the user experience. For example, doubling the CPU speed in a system with an entry-level graphics card doesn't help much in games since the system is already limited by the GPU. Likewise for a system with a high-end graphics card paired with an underpowered CPU.

Graphics score

Each Graphics test produces a raw performance result in frames per second (FPS). We take a harmonic mean of these raw results and multiply it with a scaling constant to reach a Graphics score ($S_{graphics}$) as follows:

$$S_{graphics} = 230 \times \frac{2}{\frac{1}{Fgt1} + \frac{1}{Fgt2}}$$

Where:

$Fgt1$ = The average FPS result from Graphics test 1

$Fgt2$ = The average FPS result from Graphics test 2

The scaling constant is used to bring the score in line with traditional 3DMark score levels.

Physics score

The Physics score is calculated from the raw performance result in frames per second (FPS) of the Physics test.

$$S_{physics} = 315 \times F_{physics}$$

Where:

$F_{physics}$ = The average FPS result from the Physics Test

The scaling constant is used to bring the score in line with traditional 3DMark score levels.

Ice Storm engine

Ice Storm uses the same engine on all platforms. The engine supports the following features.

- Traditional forward rendering using one pass per light.
- Scene updating and visibility computations are multithreaded.
- Draw calls are issued from a single thread.
- Support for skinned and static geometries.
- Surface lighting model is basic Blinn Phong.
- Supported light types include unshadowed point light & optionally shadow mapped directional light as well as pre-computed environmental cube.
- Support for transparent geometries and particle effects.
- 16-bit color formats are used in illumination buffers if supported by the hardware.

Windows

On Windows and Windows RT, Ice Storm requires support for Direct3D feature level 9_3 or 9_1 with the optional shadow filtering support.




Android

Ice Storm does not use any vendor specific OpenGL ES 2.0 extensions. Textures are compressed using ETC. Textures that require an alpha channel are loaded uncompressed.

iOS

Textures, including those with an alpha channel, are compressed using PVRTC.

Ice Storm version history

Version		RT			Notes
1.2.0	●	●	●	●	Added Ice Storm Unlimited
1.1.1	●	×	●	×	Ice Storm Extreme Physics test now runs at 1080 × 720 to ensure performance is not limited by the GPU. Scores may improve slightly on devices with low-end GPUs.
1.1.0	●	×	●	×	Added Ice Storm Extreme
1.0.0	●	×	×	×	Launch version



API Overhead feature test

Feature tests are special tests designed to highlight specific techniques, functions or capabilities. A 3DMark feature test differs from a 3DMark benchmark in that the nature of the test may be necessarily artificial rather than based on real-world uses and applications.

Even so, feature tests are designed such that performance improvements in the test should benefit other applications as well, i.e. any driver optimization that results in improved performance in the API Overhead feature test will also benefit other games and applications.

The 3DMark API Overhead feature test is an independent test for measuring differences in Vulkan, DirectX 12 and DirectX 11 API performance on Windows PCs. On iOS devices, the test measures the difference between OpenGL ES 3.0 and Metal APIs.

New low-overhead APIs like Vulkan, DirectX 12 and Metal make better use of multi-core CPUs to streamline code execution and eliminate software bottlenecks, particularly for draw calls.

A draw call happens when the CPU tells the GPU to draw an object on the screen. Games typically make thousands of draw calls per frame, but each one creates performance-limiting overhead for the CPU.

As the number of draw calls rises, graphics engines become limited by API overhead. APIs like Vulkan, DirectX 12 and Metal reduce that overhead allowing more draw calls. With more draw calls, the graphics engine can draw more objects, textures and effects to the screen.

The 3DMark API Overhead feature test measures API performance by making a steadily increasing number of draw calls. The result of the test is the number of draw calls per second achieved by each API before the frame rate drops below 30 FPS.

For Windows, the API Overhead feature test is only available in 3DMark Advanced Edition and 3DMark Professional Edition.

Correct use of the API Overhead feature test

The API Overhead feature test is not a general-purpose GPU benchmark, and it should not be used to compare graphics cards from different vendors.

The test is designed to make API overhead the performance bottleneck. It does this by maximizing the number of draw calls in a scene, (by drawing a huge number of individual 'buildings'), while minimizing the GPU load, (by using simple shaders and no lighting effects). This is an artificial scenario that is unlikely to be found in games, which typically aim to achieve high levels of detail and exceptional visual quality.

The benefit of reducing API overhead is greatest when the CPU is the limiting factor. With modern APIs and fast CPUs, the test can become GPU bound, but not always in a way that is meaningful from a general GPU performance perspective. The point at which the test moves from being CPU-bound to GPU-bound changes from system to system. It is not easy to tell from the test results whether the run was CPU or GPU limited. And what's more, it is difficult to isolate the relative impact of GPU performance and driver performance.

As a result, you should be careful making conclusions about GPU performance when comparing API Overhead test results from different systems. For instance, we would advise against comparing the Vulkan score from an AMD GPU with the DirectX 12 score from an NVIDIA GPU. Likewise, it could be misleading to credit the GPU for any difference in DirectX 12 performance between an AMD GPU and an NVIDIA GPU.

Another scenario, for example, would be to test DirectX 12 performance with a range of CPUs in a system with a fixed GPU. Or, you could test a vendor's range of GPUs, from budget to high-end, and keep the CPU fixed. But in both cases, the nature of the test means it will not show you the extent to which the performance differences are due to the hardware and how much is down to the driver.

The proper use of the test is to compare the relative performance of each API on a single system, rather than the absolute performance of different systems.

The focus on single-system testing is one reason why the API Overhead test is called a feature test rather than a benchmark.

System requirements

Windows

	DirectX 11	DirectX 12 ³⁰	Vulkan
OS ³¹	Windows 7 or later, 64-bit	Windows 10, 64-bit	Windows 7 or later, 64-bit
Processor	1.8 GHz dual-core Intel or AMD CPU	1.8 GHz dual-core Intel or AMD CPU	1.8 GHz dual-core Intel or AMD CPU
Memory	6 GB	6 GB	6 GB
GPU	DirectX 11 compatible	DirectX 12 compatible	Vulkan compatible
Video card memory	1 GB	1 GB	1 GB

Apple iOS

The 3DMark API Overhead app requires iOS 8.0 or later. Compatibility starts with iPhone 5s, iPad Air, iPad mini 2, and runs to the latest models. The app requires 33 MB of free storage space.

³⁰ The DirectX 12 part of the API Overhead feature test requires Windows 10, graphics hardware that supports DirectX 12, and the appropriate drivers. DirectX 12 is not yet supported on all DirectX 11 level hardware. Check with your video hardware vendor for the latest drivers. The API Overhead feature test does not yet support multi-GPU systems, and you may need to disable Crossfire/SLI. Close other apps that tie into the DirectX stack, for example applications like FRAPS that draw overlays. Apps that are not compatible with DirectX 12 will prevent the test from running.

³¹ Windows 7 users must install Service Pack 1.

Windows settings

Windowed mode

Check this box to run the test in a window. The default is unchecked, meaning the test runs full screen.

Rendering resolution

Use this drop-down menu to set the rendering resolution for the test. This is the resolution used for the internal render target, before the output is scaled to the back buffer. This option is cosmetic, since changing the rendering resolution will not affect the test results on the majority of systems. The default is 1280 × 720.

Technical details

The test is designed to make API overhead the performance bottleneck. The test scene contains a large number of geometries. Each geometry is a unique, procedurally-generated, indexed mesh containing 112 -127 triangles.

The geometries are drawn with a simple shader, without post processing. The draw call count is increased further by drawing a mirror image of the geometry to the sky and using a shadow map for directional light.

The scene is drawn to an internal render target before being scaled to the back buffer. There is no frustum or occlusion culling to ensure that the API draw call overhead is always greater than the application side overhead generated by the rendering engine.

Starting from a small number of draw calls per frame, the test increases the number of draw calls in steps every 20 frames, following the figures in the table below.

To reduce memory usage and loading time, the test is divided into two parts. On Windows, the first part runs until 98,304 draw calls per frame. On iOS, the first part runs until 12,288 draw calls per frame. The second part starts from the beginning on both Windows and iOS.

Draw calls per frame	Draw calls per frame increment per step	Accumulated duration in frames
192 - 384	12	320
384 - 768	24	640
768 - 1536	48	960
1536 - 3072	96	1280
3072 - 6144	192	1600
6144 - 12288	384	1920
12288 - 24576	768	2240
24576 - 49152	1536	2560
49152 - 98304	3072	2880

Draw calls per frame	Draw calls per frame increment per step	Accumulated duration in frames
98304 – 196608	6144	3200
196608 – 393216	12288	3520

Geometry batching

To improve content streaming performance, reduce API overhead and shorten loading times, games often batch geometries together by storing the vertex data for a group of geometries in a single, large buffer.

Allocating one large buffer is faster than allocating several small buffers. And uploading the contents of one large buffer from the CPU to the GPU is faster than uploading the contents of several small buffers.

In games and other real-world applications, the extent to which batching is possible depends on many factors. API overhead is reduced if consecutive draw calls can use the same buffer and there is no buffer changing operation required between draw calls.

The 3DMark API Overhead feature test makes a vertex buffer change operation on every tenth draw call. This represents neither the worst case nor the optimal scenario and was chosen to best reflect the nature of real-world workloads.

For fairness, we use the same batching and buffer management code on all platforms. Some platforms restrict the minimum size of buffer allocations, which in practice requires applications to store the data for smaller geometries together in large buffers. Therefore, the test uses large buffers to hold the data for several geometries.

DirectX 12 path

All lighting draw calls use the same primitive topology and pipeline state object. The following DirectX 12 API calls are made, at least once, for each lighting draw call:

```
SetIndexBuffer()  
SetGraphicsRootDescriptorTable()  
SetGraphicsRootConstantBufferView()  
DrawIndexedInstanced() with a single instance
```

All shadow map draw calls use the same primitive topology and pipeline state object. The following DirectX 12 API calls are made, at least once, for each shadow map draw call:

```
SetIndexBuffer()  
SetGraphicsRootConstantBufferView()  
DrawIndexedInstanced() with a single instance
```

Neither lighting nor shadow map passes use tessellator or geometry shader.

The test uses one thread for each logical CPU core. Draw call recording work is divided evenly between all threads for both the shadow map and lighting passes. Each thread records draw calls for a fixed set of geometries for both passes.

DirectX 11 path

All lighting draw calls use the same primitive topology, shaders and rasterizer, depth stencil and blend states. The following DirectX 11 API calls are made for each lighting draw call:

```
IASetIndexBuffer()  
IASetVertexBuffers()  
VSSetConstantBuffers()  
PSSetConstantBuffers()  
PSSetSamplers()  
PSSetShaderResources()  
DrawIndexed()
```

All shadow map draw calls use the same primitive topology, shaders and rasterizer, depth stencil and blend states. The following API calls are made for each shadow map draw call:

```
IASetIndexBuffer()  
IASetVertexBuffers()  
VSSetConstantBuffers()  
DrawIndexed()
```

Neither lighting nor shadow map passes use tessellator or geometry shader.


Single-threaded

When single threaded mode is selected, draw calls for all geometries are made through `ImmediateDeviceContext` using a single thread, first for the shadow map pass and then for the lighting pass.

Multi-threaded

When multi-threaded mode is selected, (and there are more than two logical CPU cores available), one core is intentionally left unused to ensure it is available for the display driver. The other threads, (one less than the number of available cores), are used to record draw calls to command lists through `DeferredDeviceContexts`.

Draw call recording work is divided evenly across all used threads for both shadow map and lighting passes. Each thread records draw calls for a fixed set of geometries for both passes. First all command lists are recorded without synchronization points. After being recorded, the command lists are executed by the main thread in the appropriate order.

 Since one thread is reserved for the display driver, running multi-threaded on a dual-core CPU will return the same result as running the single-threaded test.

Vulkan path

All lighting draw calls use the same primitive topology and pipeline state object. The following Vulkan API calls are made for each lighting draw call:

```
vkCmdBindDescriptorSets()  
vkCmdDrawIndexed()
```


All shadow map draw calls use the same primitive topology and pipeline state object. The following Vulkan API calls are made for each shadow map draw call:

```
vkCmdBindDescriptorSets()  
vkCmdDrawIndexed()
```

Neither lighting nor shadow map passes use tessellator or geometry shader.

The test uses one thread for each logical CPU core. Draw call recording work is divided evenly between all threads for shadow map and lighting passes. Each thread records draw calls for a fixed set of geometries for both passes.

Mantle path

 Please note that the Mantle test was replaced with a Vulkan test in 3DMark v2.3.3663 released on March 23, 2017.

All lighting draw calls use the same primitive topology, shaders and rasterizer, depth stencil and blend states. The following Mantle API calls are made for each lighting draw call:

```
grCmdBindDescriptorSet()  
grCmdBindIndexData()  
grCmdDrawIndexed()
```

All shadow map draw calls use the same primitive topology, shaders and rasterizer, depth stencil and blend states. For each shadow map draw call, the following Mantle API calls are made:

```
grCmdBindDescriptorSet()  
grCmdBindIndexData()  
grCmdDrawIndexed()
```

Neither lighting nor shadow map passes use tessellator or geometry shader.

All shader constants are stored in one large constant buffer that is updated with a single `grMapMemory()` call. The memory states for the constant buffer are set with `grCmdPrepareMemoryRegions()`.

The test uses one thread for each logical CPU core. Draw call recording work is divided evenly between all threads for shadow map and lighting passes. Each thread records draw calls for a fixed set of geometries for both passes.

Metal path

All lighting draw calls use the same primitive topology, render pipeline state, and other graphics rendering state. The following Metal API calls are made for each lighting draw call:

```
setVertexBuffer:  
setVertexTexture:  
setFragmentBuffer:  
setFragmentTexture:  
drawIndexedPrimitives:
```

All shadow map draw calls use the same primitive topology, render pipeline state, and other graphics rendering state. On each shadow map draw call, the following Metal API calls are made:

```
setVertexBuffer:  
drawIndexedPrimitives:
```

OpenGL ES 3.0 is only used in single-threaded mode, where the draw calls for all geometries are made first for the shadow map pass and then for the lighting pass.

OpenGL ES 3.0 path

All lighting draw calls use the same primitive topology, shader program, and other graphics rendering state. The following OpenGL ES 3.0 API calls are made for each lighting draw call:

```
glActiveTexture()  
glBindTexture()  
glBindSampler()  
glBindBufferRange()  
glBindVertexArray()  
glDrawElements()
```

All shadow map draw calls use the same primitive topology, shader program, and other graphics rendering state. On each shadow map draw call, the following OpenGL ES 3.0 API calls are made:

```
glBindBufferRange()  
glBindVertexArray()  
glDrawElements()
```

OpenGL ES 3.0 is only used in single-threaded mode, where the draw calls for all geometries are made first for the shadow map pass and then for the lighting pass.


Scoring

The test increases the number of draw calls per frame in steps, until the frame rate drops below 30 frames per second.

Note that if a single frame takes more than 3 times as long to render than the average time for the 20 previous frames, it is treated as an outlier and ignored. This is necessary because the first frame after raising the draw call count sometimes has a longer frame time, which would cause the test to end earlier than it should.

Once the frame rate drops below 30 frames per second, the number of draw calls per frame is kept constant and the average frame rate is measured over 3 seconds.

This frame rate value is then multiplied by the number of draw calls per frame to give the result of the test: the number of draw calls per second achieved by each API.

 The API Overhead feature test is not a general-purpose GPU benchmark, and it should not be used to compare graphics cards from different vendors. The proper use of the test is to compare the relative performance of each API on a single system, rather than the absolute performance of different systems.

API Overhead version history

Windows

Version	Notes
---------	-------

1.5	Vulkan test replaces Mantle.
-----	------------------------------

1.3	Minor bug fixes. Scores are not affected.
-----	---

1.2	Minor bug fixes. Scores are not affected.
-----	---

1.1	Updated for Windows 10 RTM. Scores are not affected.
-----	--

1.0	Launch version
-----	----------------

iOS

Version	Notes
---------	-------

1.0	Launch version
-----	----------------

Stress Tests

Stress testing is a useful way to check the reliability and stability of your system. It can also identify faulty hardware or a need for better cooling. The best time to run the stress test is after buying or building a new PC, upgrading your graphics card, or overclocking your GPU.


If your GPU crashes, hangs, or produces visual artifacts during the test, it may indicate a reliability or stability problem. If it overheats and shuts down, you may need more cooling in your computer.

Stress Tests are not available in 3DMark Basic Edition or the Steam demo.

Options

Test Selection

Use this drop down menu to choose which Stress Test to run. 3DMark offers many tests, each designed for a specific class of hardware. You should use the test most suited to the system you are testing.

 Note that Fire Strike Ultra requires at least 3 GB of dedicated video card memory. A crash on a system that does not meet this requirement is not a sign of a hardware stability problem.

Number of loops

In 3DMark Professional Edition, you can use this option to set the number of loops for the test. The minimum number of loops is 2. The maximum is 5000. You can stop the test at any time by pressing the ESC key.

Enable window mode

In 3DMark Professional Edition, use this option to run the test in a window.

Technical details

The aim of stress testing is to place a high load on the system for an extended period of time to expose any problems with stability or cooling capability.

3DMark Stress Tests work by looping a benchmark graphics test continuously without pausing for loading screens or other breaks. A Stress Test takes around 10 minutes to run when set to the default 20 loops, which is usually enough to find any significant stability or cooling issues.

Stress Test	Target hardware	Engine	Rendering resolution
Time Spy	High-performance gaming PC running Windows 10	DirectX 12 feature level 11	2560 × 1440
Fire Strike Ultra	PC systems designed for 4K gaming	DirectX 11 feature level 11	3840 × 2160 (4K UHD)
Fire Strike Extreme	Multi-GPU systems and overclocked PCs	DirectX 11 feature level 11	2560 × 1440
Fire Strike	High-performance gaming PCs	DirectX 11 feature level 11	1920 × 1080
Sky Diver	Gaming laptops and mid-range PCs	DirectX 11 feature level 11	1920 × 1080

Scoring

The main result from the Stress Test is the system's Frame Rate Stability expressed as a percentage.

$$\text{Frame Rate Stability} = \frac{\text{fpsLow}}{\text{fpsHigh}} * 100$$

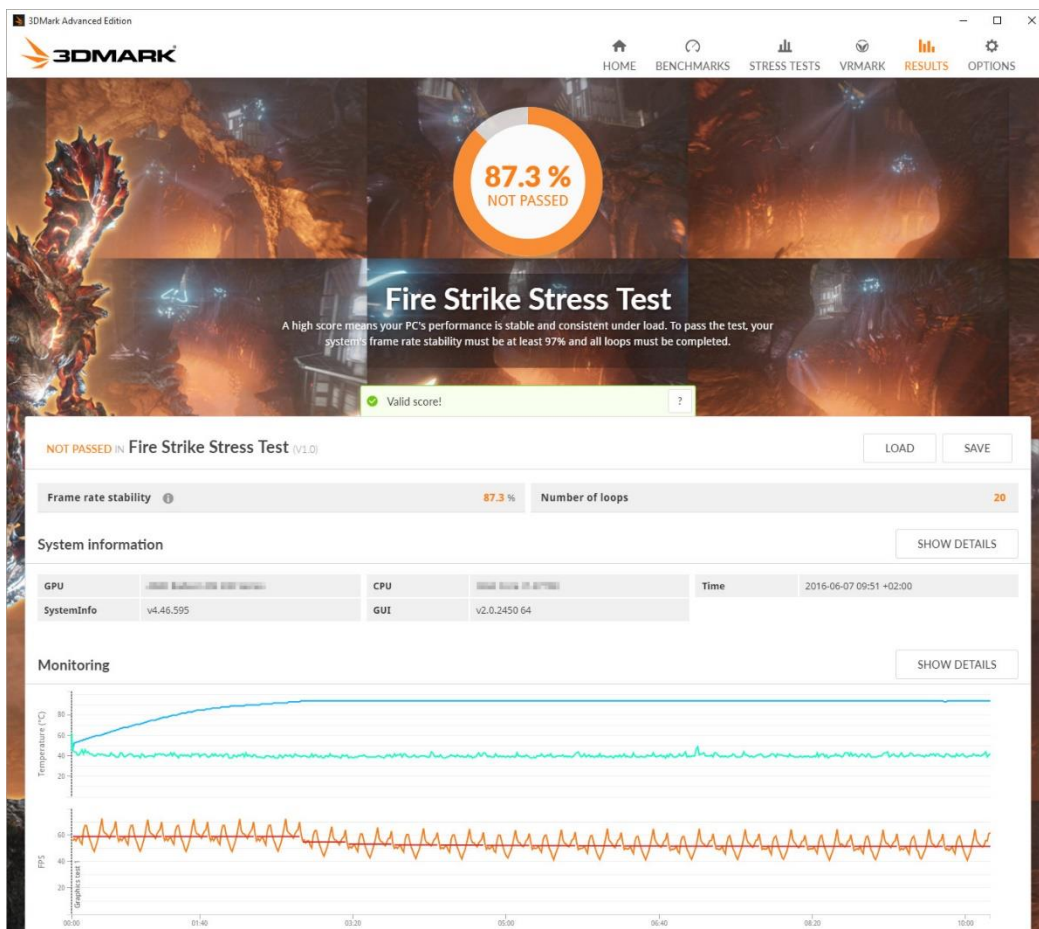
Where:

fpsHigh = The average frame rate from the best performing loop of the test.

fpsLow = The average frame rate from the worst performing loop of the test.

A high score means your PC's performance under load is stable and consistent. To pass the test, your system's frame rate stability must be at least 97% and all loops must be completed.

In the example below, the system failed the test because its average frame rate dropped noticeably after the GPU reaches its peak temperature.



How to report scores

3DMark includes many tests, each designed for a specific type of hardware ranging from smartphones to high-performance gaming PCs. When testing devices or components, make sure you use the most appropriate test for the hardware's capabilities.

Each test gives its own score, which you can use to compare similar devices or systems. There is no overall 3DMark score. Scores from different tests are not comparable. Do not use 3DMark as a unit of measurement.

- ✓ "Video card scores 10,000 in 3DMark Fire Strike benchmark."
- × "Video card scores 10,000 3DMarks."

Always include details of the hardware setup you used to obtain the score. Be sure to include the operating system, system hardware and version numbers for relevant drivers.

World record scores

Futuremark's [Hall of Fame](#) is the only source of official 3DMark world record scores. You should not present scores from any other website or leaderboard as world records. In those cases we suggest using alternative wording such as:

"Video card takes the number one spot on [website] leaderboard."

Delisted Devices

We have [rules](#) for manufacturers and developers that specify how a platform can interact with our benchmark software. When a device is suspected of breaking those rules it is delisted. Scores from delisted devices are not shown in the [Hardware Channel](#) and should not be used to compare devices.

Using 3DMark scores in marketing material

You must have a commercial license to use 3DMark scores in marketing material. A commercial license is granted with the purchase of 3DMark Professional Edition or through our site licensing program. Please [contact us](#) if you wish to use 3DMark scores from mobile platforms.

On the first mention of 3DMark in marketing text, such as an advertisement or product brochure, please write "3DMark benchmark" to protect our trademark.

"We recommend 3DMark® benchmarks from Futuremark®."

Please include our legal text in your small print.

3DMark® is a registered trademark of Futuremark Corporation.

Release notes

Windows edition

3DMark Windows v2.3.3663 – March 23, 2017

This is a major update that adds Vulkan support to the API Overhead feature test. Benchmark scores are not affected with the exception of API Overhead feature test, which now produces scores for Vulkan instead of Mantle.

New

- Added Vulkan support to the API Overhead feature test. Use the API Overhead feature test to compare Vulkan, DirectX 12, and DirectX 11 API performance on your PC. The Vulkan test requires compatible video drivers with Vulkan support. Check with your GPU vendor for Vulkan driver support if your hardware is unable to run the test. Note that the Vulkan test replaces the Mantle test found in previous versions of 3DMark.

Improved

- SystemInfo scan time greatly improved on X99 systems.

Fixed

- Fixed an issue that could cause the API Overhead feature test to fail to show a score at the end of an otherwise normal run on some systems.
- Fixed Time Spy test to properly recover from a corrupted shader cache - if runtime compiled shaders are found to be corrupted, they are deleted and recompiled. Uninstallation also now completely removes the shader cache folder.
- Fixed a scaling issue that could cause parts of the UI to end up outside the display area on 1080p monitors with 150% DPI scaling. UI will now scale appropriately even on high DPI scaling settings.

Professional Edition

- Fixed an issue that could cause the Command Line interface to refuse to work after registering a Time Spy Professional Edition key with an expiration date.

3DMark Windows v2.2.3509 – December 15, 2016

This update fixes a GUI issue that resulted in marginally lower than expected scores when starting a test from the Benchmark Details screen in 3DMark versions 2.1.2852 and later. Benchmark runs started from the Home screen or the Command Line were not affected.

It is normal for 3DMark scores to vary by up to 3% between runs since there are factors in a modern, multitasking operating system that cannot be completely controlled.

With this update, overall scores are expected to increase by up to 0.3%. Scores from the Physics and CPU parts of benchmark tests may improve by up to 2.5%. Scores from this version of 3DMark are consistent with results from previous versions that did not have the GUI issue.

Compatibility

- Added a two-minute timeout to the SystemInfo scan to prevent it from stalling for long periods on some specific systems.

3DMark Windows v2.2.3491 – December 10, 2016

This is a minor update. Benchmark scores are not affected.

Fixed

- Fixed an issue with the output resolution setting on the Option screen.

3DMark Windows v2.2.3488 – December 9, 2016

This is a minor update. Benchmark scores are not affected.

Improved

- 3DMark now warns you when vsync, FreeSync, or G-SYNC is enabled. For accurate results, you should disable these features in your video driver settings before benchmarking.

Fixed

- Fixed a SystemInfo timing issue that most commonly affected systems with the X99 chipset. 3DMark now waits for the SystemInfo scan to finish before starting the test.
- Fixed a rare issue that could cause the UI to open on an empty white window.

VRMark Preview

VRMark is now available from futuremark.com and Steam. You can still install and run the VRMark Preview in 3DMark, but it is no longer recommended or supported.

- Moved the Preview from the main navigation bar to the Benchmarks screen.
- Added an uninstall button to the VRMark Preview screen.

3DMark Windows v2.1.2973 – August 19, 2016

This is a minor update. Benchmark scores are not affected.

Fixed

- Updating 3DMark from within the app will now properly close previous versions before applying the update.

3DMark Windows v2.1.2969 – August 18, 2016

This is a minor update to fix problems reported by some users. Benchmark scores are not affected with one exception - see the section about Fire Strike Custom runs below for details.

Improved

- SystemInfo module updated to 4.48 for improved compatibility with the latest hardware.
- The video RAM check that warns if your system may not be able to run a test now accepts extra main RAM beyond the minimum requirement as VRAM for integrated graphics.
- We've added a DETAILS button to the panel for the Recommended test on the Benchmarks screen to make it easier to find more information and the settings for the test. This is also where you find the option to enable or disable the demo for each test.

Fixed Fire Strike Custom run settings

Unfortunately, the previous version (3DMark v2.1.2852) used an incorrect setting for Fire Strike Custom runs that resulted in slightly lower than expected scores. Fire Strike Custom run results from the previous version should not be compared with this latest version nor with any other version of 3DMark. The standard Fire Strike benchmark run was not affected, nor were Fire Strike Extreme and Fire Strike Ultra.

- Restored the control for volumetric illumination sample count setting on the Fire Strike Custom run screen, which was missing in the previous version.
- Fixed the default value for volumetric illumination sample count for Fire Strike Custom runs. In 3DMark v2.1.2852, Fire Strike Custom run used an incorrect default setting of 1.5. This has been reverted to 1.0, which is the correct value for the test.

Standalone version fixes

- Fixed an issue that caused installation to fail if the unzipped installer content resided in a path that included a folder name with a space.

- Fixed an issue that could prevent the in-app update from working properly. If you are affected by this issue and cannot update 3DMark from within the app, you should download the full installer.

Steam version fixes

- Fixed a problem that could cause 3DMark to appear to be still running in the Steam client after exiting, which then blocked Steam from closing.
- Fixed an issue that prevented DLCs from installing into a custom Steam library folder when the folder name included a space.

Other fixes

- Fixed an issue that prevented Sky Diver from starting on 32-bit Windows.
- Fixed an issue that caused Time Spy to crash when scaling mode was set to Stretched.
- Fixed an issue that could cause result parsing to fail on complex systems with lots of devices due to the unusually large data set generated by the SystemInfo scan.

Known Issues

- Time Spy fails to run on multi-GPU systems with Windows 10 build 10240, but this is not the fault of the benchmark. You must upgrade Windows 10 to build 10586 ("November Update") or later to enable multi-GPU configurations to work.
- Installing the standalone version and the DLC test data to the same folder is not a supported configuration. The latest version will prevent you from installing both to the same folder. If you currently have 3DMark and the DLC test data installed to the same custom folder you will need to uninstall 3DMark then reinstall the latest version using the full installer.

3DMark Windows v2.1.2852 – July 14, 2016

This major update adds Time Spy, a new DirectX 12 benchmark test. With its pure DirectX 12 engine, which supports new API features like asynchronous compute, explicit linked multi-adapter, and multi-threading, 3DMark Time Spy is the ideal benchmark for testing the DirectX 12 performance of the latest graphics cards.

New

- Added Time Spy Stress Test - a new dedicated Stress Test for high-performing PCs running on Windows 10. Time Spy Stress Test is not available in 3DMark Basic Edition or 3DMark Time Spy upgrade in Steam.

Fixed

- Fixed an issue that could cause all Stress Test runs to end with 0% score.

- Fixed an issue that could prevent self-update from working (standalone version only). If you are running 3DMark 2.0.2724 or 2.0.2809 Advanced Edition, you need to download and install the full 2.1 installer to update.

3DMark Windows v2.0.2809 - July 12, 2016

This is a minor update. Benchmark scores are not affected.

Fixed

- Fixed further compatibility issues with the Steam launcher and some specific operating system configurations that could cause the 64-bit version to refuse to start.
- Fixed an issue with result file processing that could cause the benchmark to hang with a black screen at the end of a demo or test on some systems.

3DMark Windows v2.0.2724 - July 4, 2016

This is a minor update that fixes several compatibility issues. Benchmark scores are not affected.

Improved

- SystemInfo module updated to 4.47 for improved compatibility with the latest hardware.

Fixed

- Fixed an issue that could cause the Sky Diver Stress Test to hang on a white screen on very fast systems.
- Fixed an issue that prevented 3DMark from installing on Windows 7 if UAC was disabled. You can now click 'Ignore' on the warning to continue the installation.
- Fixed compatibility issues with the Steam launcher and some specific operating system configurations that could cause the 64-bit version to refuse to start.
- Fixed an issue that could cause the benchmark to fail if your Windows user folder name contained UTF-8 characters.

3DMark Windows 2.0.2530– June 13, 2016

This major update adds new Stress Tests for checking the stability of your PC.

New

- Use the new Stress Tests to check the stability of your system after buying or building a new PC, upgrading your graphics card, or overclocking your GPU. Stress testing can help you identify faulty hardware or the need for better cooling. Stress Tests are not available in 3DMark Basic Edition or the Steam demo.

Improved

- SystemInfo module updated to 4.46 for improved hardware detection.
- Reintroduced the option to set up a Custom run using only the Demo.

Fixed

- Fixed an issue that could cause 3DMark to fail to install test DLC files.

3DMark Windows v2.0.2067 - April 15, 2016

This minor update fixes a few issues that came to light after the v2.0.1979 release on April 6. Benchmark scores are unaffected.

Fixed

- SystemInfo module updated to 4.45 to fix a compatibility issue with Russian and Chinese language versions of Windows.
- Fixed the Unicode compatibility issue with Russian and Chinese language versions of Windows.
- Fixed the white screen issue when installing 3DMark under a NTFS Junction or Mount Point.
- Fixed the missing button text issue affecting a small number of users.

Improved

- Updated Russian localization.

3DMark Windows v2.0.1979 – April 6, 2016

This is a major update that adds a redesigned UI for all editions and a preview of VRMark for Advanced and Professional Edition users.

New

- 3DMark UI has been redesigned and rebuilt to be faster and more flexible.
- Home screen recommends the best test based on your system details.
- Run other benchmarks and feature tests from the Benchmarks screen.
- Russian localization.

Improved

- Each benchmark test can now be updated independently.
- Ice Storm Extreme and Ice Storm Unlimited are unlocked in 3DMark Basic Edition.
- SystemInfo module updated to 4.43 for improved hardware detection.

VRMark preview

- Explore two test scenes in a preview of VRMark, our new benchmark for VR systems. The preview does not produce a score.

- The preview is not available in 3DMark Basic Edition or the Steam demo.

Fixed

- Workaround for the AMD driver issue where the preview videos in the UI caused some AMD graphics cards to use low power mode and run at lower clock speeds.

3DMark Windows v1.5.915 – June 5, 2015

This is a minor update. Benchmark scores are unaffected. Note that while Windows 10 is in development there may be unforeseeable compatibility problems with some hardware configurations.

Improved

- SystemInfo module updated to 4.39 for improved detection of upcoming hardware from AMD and Intel.

Compatibility

- API Overhead feature test updated to work with Windows 10 Technical Preview build 10130.

Known issues

- AMD Catalyst Driver 15.200.1023.5 for Windows 10 has an issue that prevents the DirectX 12 API Overhead test from working on Radeon R9 280, Radeon HD 79xx series, and Radeon HD 78xx series graphics cards. We expect AMD to fix the issue with its next driver update.
- Intel HD Graphics Driver 10.18.15.4204 for Windows 10 does not appear to have working full screen DirectX 12 support. We are investigating this issue for a future update.

3DMark Windows v1.5.893 – April 24, 2015

This is a minor update. Benchmark scores are unaffected.

Compatibility

- Fixed a bug that could cause the API Overhead feature test to hang on Windows 10 Technical Preview build 10061.

Steam version only

- Fixed an issue that prevented Steam Achievements from being unlocked.

3DMark Windows v1.5.884 - March 26, 2015

This major update adds the API Overhead feature test, the world's first independent test for comparing the performance of DirectX 12, Mantle, and

DirectX 11. See how many draw calls your PC can handle with each API before the frame rate drops below 30 FPS.

New

- Compare DirectX 12, DirectX 11 and Mantle with the new API Overhead Feature Test, available in 3DMark Advanced Edition and 3DMark Professional Edition.
- Added Feature Test selection screen.

Improved

- Improved formatting of larger scores to make them more readable.
- Result screen automatically shows FPS after running a single test.

Fixed

- Fixed a bug that could cause the Sky Diver demo to hang at the cave entrance scene.

3DMark Windows v1.4.828 - December 1, 2014

This is a minor update. Benchmark scores are unaffected.

Improved

- SystemInfo module updated to 4.32 for improved hardware detection.
- Reduced hardware monitoring overhead (was already negligible).
- Product key is no longer visible on the Help tab unless you choose to reveal it.

Fixed

- Fixed a memory access violation issue with Ice Storm and Cloud Gate that could occasionally cause crashes in stress testing scenarios.
- Letterboxed mode now retains 16:9 aspect ratio even when selecting a non-default Output Resolution on the Help tab.

Professional Edition only

- Fixed the "No outputs found on DXGI adapter" issue in the Command Line application affecting laptops with NVIDIA Optimus graphics switching technology.
- Fixed custom_x.3dmdef files to use the centered scaling mode by default.
- You can now change the scaling mode from a .3dmdef file and via command line.

3DMark Windows v1.4.780 - October 23, 2014

This is a minor update. Benchmark scores are unaffected.

Fixed

- Fixed the "No outputs found on DXGI adapter" issue affecting laptops with NVIDIA Optimus graphics switching technology.

3DMark Windows v1.4.778 - October 14, 2014

This is a minor update. Benchmark scores are unaffected.

Fixed

- Fixed the "Workload Single init returned error message: bad lexical cast" issue affecting some systems.

3DMark Windows v1.4.775 - October 13, 2014

This is a major update that adds Fire Strike Ultra, the world's first 4K Ultra HD benchmark. Fire Strike Ultra is available in 3DMark Advanced Edition and 3DMark Professional Edition.

New

- Added Fire Strike Ultra, a new 4K Ultra HD benchmark test. You don't need a 4K monitor to run Fire Strike Ultra, though you will need a GPU with at least 3 GB of dedicated memory.

Improved

- New design for main benchmark selection screen.
- Improved benchmark logging to assist customer support.

Fixed

- 3DMark is now more robust when there is a problem identifying or monitoring the hardware in the system.

Professional Edition only

- You can now set command line options within .3dmdef files.
- Minor syntax changes to the .3dmdef definition files. You may need to update your existing scripts if using automation. See Command Line Guide for details.
- Added command line logging options.
- Command line progress logging now includes workload names and loop numbers.
- Removed empty log lines from command line output.

3DMark Windows v1.3.708 – June 11, 2014

This update adds Sky Diver, a new DirectX 11 benchmark for gaming laptops and mid-range PCs. Sky Diver is ideal for testing systems with mainstream

graphics cards, mobile GPUs, integrated graphics and other DirectX 11 hardware that cannot achieve double-digit frame rates in Fire Strike.

Improved

- You can now run benchmarks individually in 3DMark Basic Edition.
- SystemInfo module updated to 4.29 for improved hardware detection.

Compatibility

- On Windows 7, Service Pack 1 is required for 3DMark version 1.3.708 onwards.

Professional Edition only

- The filenames of the .3dmdf definition files used for running 3DMark from the command line have changed with this release. You may need to update your existing scripts if using automation.

3DMark Windows v1.2.362 - March 12, 2014

Improved

- Improved reliability when submitting results over an internet connection with very high latency.
- SystemInfo module updated to 4.26 for improved hardware detection.

Fixed

- DirectX 10 level video cards no longer attempt to run the Fire Strike benchmark.
- Fixed a rare issue that could corrupt the saved product key.

Steam version only

- Fixed a bug that prevented Steam Achievements from being unlocked.
- Fixed a rare issue with results not always being associated with a Steam ID.

Professional Edition only

- Fixed an issue with command line XML export of Ice Storm scores.

3DMark Windows v1.2.250 – December 10, 2013

New

- Added Ice Storm Unlimited test enabling comparison of Windows 8 tablets with the latest Android and iOS devices.

Improved

- 3DMark now uses technology provided by TechPowerUp for improved GPU hardware detection.

Fixed

- Hardware monitoring performance graphs show clock speeds and temperatures for the CPU and GPU again (with compatible hardware).

Tests

- Ice Storm updated to version 1.2

3DMark Windows Edition v1.1.0 – May 6, 2013

This update fixes issues when testing systems with multiple GPUs. Fire Strike and Fire Strike Extreme scores will increase slightly on systems with two GPUs and significantly on systems with three or four GPUs.

New

- Added Ice Storm Extreme benchmark to 3DMark Advanced and Professional Editions.

Fixed

- 3DMark now works correctly on systems with up to four GPUs.
- Fixed the issue caused by Windows update KB2670838, which added partial DX11.1 support to Windows 7.
- Fixed a problem with the bloom post-processing effect when using very high rendering resolutions in custom settings.

Tests

- Ice Storm updated to version 1.1.0
- Cloud Gate updated to version 1.1.0
- Fire Strike updated to version 1.1.0

3DMark Windows Editions v1.0.0 – February 4, 2013

- Launch version.

Tests

- Ice Storm version 1.0.0
- Cloud Gate version 1.0.0
- Fire Strike version 1.0.0

Windows RT edition

3DMark Windows RT v1.2.42 – November 26, 2013

New

- Device models that do not comply with our benchmark rules have been delisted from the Device Channel.
- You should not use scores from delisted models to compare devices.

Improved

- Compare your score with Windows 8 tablets in the Device Channel.

Compatibility

- Improved compatibility with the latest NVIDIA hardware.

3DMark Windows RT v1.1.11.4749 – October 30, 2013

- Added a UI prompt directing PC and notebooks users to download the complete desktop version of 3DMark.

3DMark Windows RT v1.1.11 - October 11, 2013

- Launch version.

Tests

- Ice Storm version 1.2.0

Android edition

3DMark Android 1.6.3439 – January 13, 2017

Compatibility

- This update fixes an issue in the previous version (1.6.3428) that could affect performance on some Qualcomm-powered devices. Benchmark scores on those devices will return to the correct level with this update.

3DMark Android 1.6.3428 – December 7, 2016

This minor update renames some tests to make it easier to compare scores across platforms. The test names in the Android app now match the test names used in the iOS app. Benchmark scores are not affected.

Improved

- "Sling Shot using ES 3.0" test is now simply called "Sling Shot."
- "Sling Shot using ES 3.1" test is now called "Sling Shot Extreme."
- Improved benchmark compatibility information.

3DMark Android 1.5.3285 – August 31, 2015

Compatibility

- Sling Shot fix for devices that support OpenGL ES 3.2.
- Sling Shot compatibility warning is now specific about requiring Android 5.0 or later.

3DMark Android 1.5.3263 – July 27, 2015

This update completely replaces the standalone Sling Shot app released for limited testing in April 2015. That app is no longer available from Google Play and should not be used for further testing since its scores are not comparable with this update.

New

- New Sling Shot benchmark for testing OpenGL ES 3.1 and OpenGL ES 3.0 performance on compatible devices running Android 5.0 or later.
- New hardware monitoring chart shows how CPU clock frequency, CPU usage, temperature and battery charge changed while the benchmark was running.

Improved

- 3DMark now recommends the best test for your device.
- New layout makes choosing benchmarks easier.
- Save storage space by installing only the benchmark tests you need.

- Added Russian localization.

Compatibility

- Added support for Android TV.
- Sling Shot now runs on ARM Mali GPUs.

3DMark Android 1.4.2925 - December 15, 2014

Fixed

- Fixed a bug that caused the UI to hang with a black screen if the benchmark run ended unexpectedly.
- Minor UI improvements and bug fixes.

3DMark Android 1.4.2717 - December 5, 2014

New

- 3DMark now recommends the best benchmark test for your device.
- New Benchmarks page shows which tests are currently installed.
- New Test Details pages explain what each benchmark measures.

Improved

- Downloading and installing tests is now faster and uses less memory.
- Optimized app startup time and update check.
- Device Channel is now called Best Devices.

Compatibility

- 3DMark is compatible with Android 5.0 (Lollipop).

Fixed

- Fixed a problem where 3DMark could not install tests even though there was enough space on the storage or SD card.
- Fixed sorting by screen size on Best Devices list.

3DMark Android 1.3.1439 - March 24, 2014

Improved

- Added the ability to selectively delist devices based on Android version.

Compatibility

- Delisted Samsung devices that comply with Futuremark benchmark rules after being updated* have been relisted in the 3DMark Device Channel.
- Samsung Galaxy S IV: scores are valid when using Android 4.2.2 and 4.4.x or later.

- Samsung Galaxy Note III: scores are valid when using Android 4.4.2 or later.

*Provided that the device is running the official update provided by Samsung.

Fixed

- Fixed a bug that prevented downloads on devices set to use Turkish language.

3DMark Android 1.3.1375 - March 7, 2014

Improved

- Optimizations to improve the speed at which tests are downloaded and installed.

Fixed

- Fixed a bug that prevented the Device Channel from updating with the latest devices.

3DMark Android 1.3.1309 – February 21, 2014

New

- Do you love your phone? Let other 3DMark users know with the new Recommend My Device feature.
- 3DMark Android benchmark is now available in Simplified Chinese.

Improved

- 3DMark now supports the ability to install and uninstall benchmark tests from within the app. Look out for new tests coming soon.
- Improved UI rendering performance.

Compatibility

- Now requires Android 4.0.0 or higher. Android 3.2 support has been retired.
- Fixed a compatibility issue affecting Sony Xperia SP and Samsung Galaxy S III models with the Qualcomm Snapdragon MSM8960 chip.
- Improved reliability on devices with less than the recommended 1 GB of memory.
- Support for installing 3DMark on AOSP devices that cannot access Google Play store.
- Sideloading support for networkless installation.

3DMark Android v1.2.0.1232 – November 25, 2013

New

- Device models that do not comply with our benchmark rules have been delisted from the Device Channel.
- You should not use scores from delisted devices to compare devices.

Improved

- Compare your score with Windows 8 tablets in the Device Channel.
- Devices are colour-coded by OS in the Device Channel.
- You can filter scores by OS in the Device Channel.

Compatibility

- Improved compatibility with the latest NVIDIA hardware.
- Improved compatibility with some Samsung Galaxy S3 models.

3DMark Android Edition v1.1.0.1179 – Sept 9, 2013

New

- New Ice Storm Unlimited test.
- Compare 3DMark scores with Apple iOS devices in the Device Channel.

Improved

- Forced vertical sync on Android devices limits apps to displaying a maximum of 60 frames per second. Your score will be shown as "Maxed out" if your device hits the vertical sync limit during a test.
- 3DMark will recommend the best test for your device to avoid vertical sync limits.

Compatibility

- Nexus 7 (2013) is correctly identified.

Tests

- Ice Storm updated to version 1.2.0

3DMark Android Edition v1.0.3-1138 - August 20, 2013

Compatibility

- Added a workaround for a driver bug on Nexus 7 devices running Android 4.3.

3DMark Android Edition v1.0.2-1109 - May 2, 2013

New

- Added chipset model to device detail pages.

Improved

- Ice Storm Extreme Physics test for measuring CPU performance now runs at 720p to ensure the result is not influenced by the GPU. Ice Storm Extreme scores may improve slightly on devices with low-end GPUs.

Optimized

- Faster image loading in the Device Channel.
- Better score bar scaling in search result lists.
- Reduced app size to 149 MB.

Compatibility

- Automatically skip demo on devices with TI OMAP 44xx chipset to avoid a memory-related crash. There will be some visual corruption during the Physics test however this does not affect the score and you will now be able to complete all the tests on these devices.
- Automatically skip demo if the device runs out of memory during the demo, typically on devices with 512 MB of memory. The recommended minimum device memory for 3DMark is 1 GB (1024 MB).

Fixed

- Fixed a bug that prevented your best score being shown as your Highest Score.
- My Device now shows the Android OS version installed on the current device. Device Channel detail pages show the Android OS version shipped with the model.

Tests

- Ice Storm version 1.1.1

3DMark Android Edition v1.0.1-949 – April 11, 2013

New

- You can now search the Device Channel.
- You can now report unknown and incorrectly identified devices.

Fixed

- Fixed a bug that caused crashes for some users.
- Fixed the "no score" bug that could cause the test to exit after the demo.

- The Device Channel list loads faster and uses less memory.

3DMark Android Edition v1.0.0 – April 2, 2013

- Launch version.
- Added Ice Storm Extreme test for the latest smartphones and tablets.

Tests

- Ice Storm version 1.1.0

iOS edition - 3DMark Sling Shot app

3DMark Sling Shot iOS Edition 1.0.745 – April 18, 2016

Fixed

This update fixes a bug from the previous release (1.0.734) that inflated Physics test scores. Scores from this version are again comparable across all compatible platforms and devices.

Added Sling Shot

3DMark Sling Shot iOS Edition 1.0.734 – March 29, 2016

New

Added Sling Shot Extreme, a more demanding test for the latest high-end smartphones and tablets. Sling Shot Extreme uses Apple's Metal API and a 2560 × 1440 rendering resolution. You can compare Sling Shot Extreme scores across platforms.

Improved

- UI improvements.

3DMark Sling Shot iOS Edition v1.0.484 – October 16, 2015

- Launch version.

iOS edition - 3DMark Ice Storm app

3DMark Ice Storm iOS Edition 1.4.978 – March 29, 2015

New

- New hardware monitoring chart.

Improved

- 3DMark now recommends the best test for your device.
- New layout makes choosing benchmarks easier.
- Added Russian localization.

Fixed

- Fixed UI stuns when updating device data.

Compatibility

- Requires iOS 7.0.

3DMark iOS Edition v1.3.116 – March 27, 2014

Compatibility

- Fixes for iOS 7.1.

3DMark iOS Edition v1.3.0 – March 13, 2014

New

- Do you love your phone? Let other 3DMark users know with the new Recommend My Device feature.
- 3DMark is now available in Simplified Chinese. Change languages on the Settings screen.

3DMark iOS Edition v1.2.0 – December 2, 2013

Compatibility

- Now includes a 64-bit version for iPhone 5s, iPad Air and iPad mini 2.
- Scores from those devices will improve slightly with this version.

New

- Device models that do not comply with our benchmark rules have been delisted from the Device Channel.
- You should not use scores from delisted models to compare devices.
- For more details, please visit: <http://bit.ly/3dmark-rules>

Improved

- Compare your score with Windows 8 tablets in the Device Channel.

3DMark iOS Edition v1.1.1 – September 19, 2013

Improved

- Devices are color coded by OS in the Device Channel.

Compatibility

- Compatible with iOS 7.

3DMark iOS Edition v1.1.0 – September 9, 2013

- Launch version.

Tests

- Ice Storm version 1.2.0

iOS edition - 3DMark API Overhead app

3DMark API Overhead iOS 1.0.147 – March 29, 2016

Improved

- Interface improvements.

3DMark API Overhead iOS v1.0.53 – October 16, 2015

- Launch version.

About Futuremark, a UL company

Futuremark creates benchmarks that enable people to measure, understand and manage computer hardware performance. Our talented team creates the industry's most authoritative and widely used performance tests for desktop computers, notebooks, tablets, smartphones and VR systems.

We work in cooperation with many of the world's leading technology companies to develop industry standard benchmarks that are relevant, accurate, and impartial. As a result, our benchmarks are widely used by the world's leading press publications and review sites.

Futuremark maintains the world's most comprehensive hardware performance database, using results submitted by millions of users to help consumers make better purchasing decisions.

Our headquarters are in Finland just outside the capital Helsinki. We also have sales and field application engineering support in Silicon Valley and Taiwan.

Futuremark became a part of UL in 2014. UL is a global safety science company with more than a century of expertise and innovation in the fields of product safety testing, inspection and verification services. With more than 10,000 professionals in 40 countries, UL is dedicated to creating safe working and living environments for all.

UL partners with businesses, manufacturers, trade associations, regulators, and governments to play a key role in the development and harmonization of national and international standards. For more information about certification, testing, inspection, advisory and education services, visit <http://www.UL.com>.

Please don't hesitate to contact us if you have a question about 3DMark.

Press	press@futuremark.com
Business	sales@futuremark.com
Support	http://www.futuremark.com/support

© 2017 Futuremark® Corporation. 3DMark® and Futuremark® trademarks and logos, Futuremark® character names and distinctive likenesses, are the exclusive property of Futuremark Corporation. Microsoft, Windows 10, Windows 8, Windows 7, DirectX, and Direct3D are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Vulkan™ is a trademarks of the Khronos Group Inc. The names of other companies and products mentioned herein may be the trademarks of their respective owners.