3DMARK®11

Technical Guide

Updated April 19, 2018

3DMARK 11 OVERVIEW

A PC's performance is determined by the interactions between different hardware components, the operating system and the type and amount of software running. A benchmark provides a set of performance tests that can be repeated with a high degree of accuracy on a wide range of computer hardware.

In 3DMark 11 the tests focus on updating and rendering complex game worlds in real-time using DirectX 11. The benchmark workload consists of six tests produce results from which the 3DMark score is calculated. Please see section 5 for more details about each of the tests in 3DMark 11.

Running 3DMark 11 gives your system a score with larger numbers indicating better performance. Comparing scores between systems is easier than comparing specs of individual components.

BENCHMARK PRINCIPLES

The following principles were used to determine the benchmark test suite, architecture, content and scoring in 3DMark 11, ensuring that the benchmark serves its primary purpose of providing accurate, consistent and reliable performance measurement.

Produce consistent results that are repeatable and verifiable.

Running 3DMark is easier and faster than running a group of games and trying to measure and calculate some kind of average performance figure. While running a single game will give you the best possible indication of how well that particular game runs on your PC, running 3DMark will give a much better estimation of how all games will perform on your system.

Represent technology and workloads fairly and accurately.

Many games are too lightweight to provide meaningful benchmark results. It's not clear which games are good benchmarks for general game performance and which are not. 3DMark 11 has been developed with input from the designers, engineers and product managers at AMD, Dell, HP, Intel, Microsoft, NVIDIA, Imagination Technology and many other well known companies. This input ensures that 3DMark is an unbiased, high quality benchmark that delivers relevant and reliable results that scale as technology advances.

Remain relevant over a long period of time.

Today's games do not necessarily indicate the likely loads from future games. 3DMark is designed to be forward-looking providing the load expected from future games, not just those that have been recently released. This ensures that the benchmark results remain relevant for many years after the benchmark is first released.

Be widely used so that results can be compared across a wide variety of systems.

3DMark is the world's most popular graphics benchmark and has been used by millions of people around the world. It is supported by an online database containing over 35 million results with new results submitted every few seconds on average. This huge volume of data makes it easy to compare your system with others to identify problems or the performance gain from overclocking.

3DMark is also used by the majority of the press when reviewing new pieces of gaming hardware such as graphics cards. This makes it easy to compare your 3DMark score with those published in reviews and make better decisions as to whether an upgrade will be worth your money.

3DMARK 11 PRESETS

Benchmark presets ensure relevant results for all levels of hardware. An official 3DMark score can only be obtained when the benchmark is run with one of the presets. The 3DMark score is calculated differently for each preset which means that scores are not comparable across different presets. The available presets and their corresponding code letters are explained below.

PRESET	CODE LETTER	EXAMPLE SCORE
ENTRY	E	E7053
PERFORMANCE	P	P5420
EXTREME	X	X2641

Entry preset (E)

This preset is designed for benchmarking with a low level of load on the graphics card. The benchmark runs in 1024 x 600 resolution making it suitable for most entry level DirectX 11 capable systems including notebooks and netbooks.

Performance preset (P)

This preset is designed for benchmarking with a moderate load on the graphics card. The benchmark runs in 1280 x 720 (720p) making it suitable for most DirectX 11 capable gaming PCs, though at launch high end hardware may be required to achieve a fluid frame rate.

Extreme preset (X)

This preset is designed for benchmarking with a very heavy load on the graphics card. The benchmark runs in 1920 x 1080 (1080p). The Extreme preset extends the lifetime of the benchmark by representing the likely loads used by high end games in years to come. At launch it offers enthusiasts a suitable benchmark load for competing at the extreme end of system performance.

PRESET CONFIGURATIONS

	ENTRY	PERFORMANCE	EXTREME
SCREEN RESOLUTION	1024x600	1280x720	1920x1080
MEMORY BUDGET	256MB	768MB	1024MB
MSAA SAMPLE COUNT	off	off	4
TEXTURE FILTERING MODE	Trilinear	Trilinear	Anisotropic
MAX AF ANISOTROPY			16
MAX TESSELLATION FACTOR	6	10	15
SHADOW MAP SIZE	Low (largest 1024x1024)	Medium (largest 2048x2048	High (largest 4096x4096)
SHADOW CASCADE COUNT	Low (3 on directional light)	Medium (4 on directional light)	High (5 on directional light)
SURFACE SHADOW SAMPLE COUNT	8	16	16
VOLUMETRIC ILLUMINATION QUALITY	Low (ray marching step count -25%)	Medium	Medium
AMBIENT OCCLUSION QUALITY	Low (sample pattern 3x4)	Medium (sample pattern 4x5)	High (sample pattern 5x6)

	ENTRY	PERFORMANCE	EXTREME
DEPTH OF FIELD QUALITY	Low (bokeh texture size 16x16, bokeh radius threshold for splatting at reduced resolution -33%)	Medium (bokeh texture size 24x24)	High bokeh texture size 32x32, bokeh radius threshold for splatting at reduced resolution +33%)

CUSTOM SETTINGS

The benchmark can be run with customized settings offering a fine degree of control over resolution, anti-aliasing, tessellation, shadow quality, texture quality and many other graphical settings. When using a resolution with an aspect ratio other than 16:9, the benchmark runs in a 16:9 letterbox. Note that benchmarking with custom settings will not produce a 3DMark 11 score. Nevertheless, there are many times when using custom settings can be useful for measuring your system's performance.

RENDERING ENGINE

Multithreading

The multithreading model is based on DirectX 11 device contexts and command lists. The engine utilizes one thread per available physical CPU core in graphics tests and one thread per available logical CPU core in physics and combined tests. 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 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, visibility culling, 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

3DMark 11 uses two types of tessellation: Displacement map based detail tessellation and Phong tessellation with optional displacement map. In detail tessellation vertices produced by the tessellator are positioned based on a displacement map. Displacement happens in the direction of smoothed geometry normal. In Phong tessellation, vertices are placed on an approximated curved surface before displacement. The curved surface approximation is computed from the positions and normals of the triangle vertices.

The tessellation factors are computed based on length of each input triangle edge projected to screen space. This means that tessellation factors adapt to distance from camera keeping output triangle size roughly same as input triangle's distance varies. Desired output triangle edge length in screen pixels is controlled by artist. Tessellation factor for the inside of the triangle is determined as minimum of the factors computed for edges. This helps to avoid generating subpixel triangles. Finally, tessellation factors are clamped to a given maximum value.

Input triangles with vertex normals facing away from the viewer are culled in the hull shader. Culling happens when angle of all vertex normals against view vector exceeds a given threshold. Input triangles are also culled against left, right, bottom and top planes of the view frustum.

Tessellation factors of input triangles facing directly towards viewer are not reduced despite the detail generated cannot be directly seen. The parallax effect in surfaces of moving objects and screen space ambient occlusion effect reveal the detail generated to such triangles.

When generating shadow maps, the same tessellation factors are used as in rendering the screen image. This provides good self shadow quality and works well from low to high levels of tessellation detail allowing easy scaling of the content.

Tessellation is turned off from objects that are at a distance where tessellation would not produce any visible detail.

Lighting

Rendering 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 buffers. Finally illumination is rendered based on those attributes. Textures and their formats in the G-buffer are as follows:

TEXTURE	FORMAT
DIFFUSE	DXGI_FORMAT_R8G8B8A8_UNORM_SRGB
SPECULAR	DXGI_FORMAT_R8G8B8A8_UNORM_SRGB
NORMAL	DXGI_FORMAT_R10G10B10A2_UNORM
AMBIENT OCCLUSION	DXGI_FORMAT_R8_UNORM
DEPTH	DXGI_FORMAT_R32_TYPELESS

Illumination from all unshadowed point lights in the scene is rendered in two draw calls. One call for lights whose volume of influence does not intersect with camera near plane and one for lights whose volume of influence does. Light parameters are sent to the GPU in a dynamic vertex buffer.

Illumination from spot lights is rendered with one draw call per light. Draw call building is done in the worker threads. Shadow texture size for shadow mapped spot lights is selected based on size of the light volume on screen.

Illumination from directional lights is rendered with one draw call per light. Shadow mapped directional lights are rendered with one draw call per shadow cascade. When rendering illumination, there are separate render targets for surface illumination and volume illumination with DXGI_FORMAT_R11G11B10_FLOAT format.

Surface Illumination

Surface illumination model is combination of Oren-Nayar diffuse reflectance and Cook-Torrance specular reflectance. Rayleigh-Mie atmospheric attenuation is also computed. Horizon based screen space ambient occlusion affects the surface illumination. Shadow maps are sampled using best candidate sample distribution. Sample pattern is dithered with 4x4 pixel pattern.

Volumetric Illumination

The renderer supports optional volume illumination. It is computed by approximating the light scattered towards the viewer by the medium between the 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 4x4 pixel pattern. The achieved result is blurred to combine the different sampling depths on neighboring pixels before adding the volume illumination to the surface illumination.

Optical density of the illuminated volume can be adjusted with a configurable noise function. On each frame, the optical density is pre-computed to an array of 2D textures fitted to the view frustum. Each texture in the array contains density for one depth slice of the view frustum. In addition, accumulated transmittance from camera based on the pre-computed density is stored to another array of 2D textures. These textures are then sampled on each volume ray sample. The used texture formats are DXGI_FORMAT_R16_FLOAT for the pre-computed density and DXGI_FORMAT_R10G10B10A2_UNORM for the accumulated transmittance.

POST PROCESSING

The renderer supports depth of field and bloom effects. An optional film grain effect is integrated into the tone mapping step.

Depth of Field

The effect is computed using the following procedure:

1. Circle of confusion radius is computed for all screen pixels and stored in a full resolution DXGI_FORMAT_R16_FLOAT texture.
2. Half and quarter resolution versions are made from the radius texture and the original illumination texture.
3. Positions of out-of-focus pixels whose circle of confusion radius exceeds a predefined threshold are appended to a buffer.
4. The position buffer is used as point primitive vertex data and, utilizing Geometry Shader (GS), image of hexagon-shaped bokeh is splatted to positions of these vertices. Splatting is done to a DXGI_FORMAT_R16G16B16A16_FLOAT texture. Multiple viewports are used to partition the texture to regions with different sizes. First region is screen size and the rest are a series of halved regions down to size 1x1 texels. The radius of the splatted bokeh determines the used viewport. The larger the radius the smaller the used viewport.
5. Steps 3 and 4 are done separately for full, half, and quarter resolution image data with different radius thresholds. Larger bokehs are generated from lower resolution image data.
6. The different regions of the splatting texture are combined by up-scaling the data in the smaller regions to the screen size region.
7. The combined splatted 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 (CS). The effect is computed in reduced resolution. The input image resolution is halved twice and then rounded up to nearest power of two. The FFTs are computed using DXGI_FORMAT_R32G32B32A32_FLOAT textures. A procedurally precomputed texture is used as the bloom filter. The filter combines blur, streak, lenticular halo and anamorphic flare effects.

Lens Reflections

The effect is computed by first applying a filter to the computed illumination in frequency domain similar to 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 DXGI_FORMAT_R32G32B32A32_FLOAT textures.

TESTS

Graphics Test 1

A scene with large number of both shadow casting and non shadow casting spot lights and non shadow casting point lights is rendered. The scene uses volumetric illumination with noise adjusted optical density. The scene contains no tessellated geometry.

Graphics Test 2

A scene with moderate number of both shadow casting and non shadow casting spot lights and non shadow casting point lights is rendered. The scene uses volumetric illumination with noise adjusted optical density. The scene contains tessellated geometry.

Graphics Test 3

A scene with one shadow casting directional light and a moderate number of non shadow casting point lights is rendered. Volumetric illumination is only enabled for the shadow casting directional light. The shadow casting directional light uses volumetric illumination with optical density varied along world space height. The scene contains tessellated geometry.

Graphics Test 4

A scene with one shadow casting directional light and a few shadow casting spot lights is rendered. The scene uses volumetric illumination with optical density varied along world space height. Majority of the rendering workload comes from drawing tessellated geometry to shadow maps and G-buffer.

Physics Test

A scene with a large number of rigid bodies is simulated and rendered. The rendering is done using light-weight techniques. The rigid bodies collide with each other and some of them are connected with joints. The simulation is divided to multiple threads by partitioning the simulated world to several isolated regions. The Bullet Open Source Physics Library C++ path is used as the physics SDK for this test, and compiled and linked statically into the test binary at development time.

Combined Test

A scene with a large number of rigid and soft bodies is simulated and rendered. The rigid bodies collide with each other and some of them are connected with joints. The simulation is divided to multiple threads by partitioning the simulated world to several isolated regions. The soft body simulation is computed using DirectX Compute Shaders. The Bullet Open Source Physics Library is used as the physics SDK for this test, and compiled and linked statically into the test binary at development time. The test contains also considerable graphics workload in addition to the physics workload. It contains some tessellation, volumetric illumination and post processing effects.

SCORING

Graphics Test scoring

Each of the Graphics Tests produces a result in frames per second (fps). The Graphics Score $S_{graphics}$ is calculated by multiplying the harmonic mean of test results with a scaling constant:

$$S_{graphics} = C_{graphics}\frac{4}{\frac{1}{F_{gt1}}+\frac{1}{F_{gt2}}+\frac{1}{F_{gt3}}+\frac{1}{F_{gt4}}}$$

$C_{graphics}$ is the scaling constant for the graphics score and $Fgt1-4$ are the fps results for Graphics Tests 1-4. The scaling brings the score in line with traditional 3DMark score levels.

Physics Test scoring

The Physics Test produces a performance result in frames per second (fps). The Physics score $S_{physics}$ is calculated by multiplying the result with a scaling constant:

$$S_{physics} = C_{physics}F_{physics}$$

$C_{physics}$ is the scaling constant and $F_{physics}$ is the fps result for the Physics Test. As with the graphics tests, scaling is used to bring the score into the traditional range.

Combined Test scoring

The Combined Test produces a performance result in frames per second (fps). The Combined score $S_{combined}$ is calculated by multiplying the result with a scaling constant:

$$S_{combined} = C_{combined}F_{combined}$$

$C_{combined}$ is the scaling constant and $F_{combined}$ is the fps result for the Combined Test. As with other tests, scaling is used to bring the score into the traditional range.

3DMark score

The 3DMark score S_{3DMark} for each preset in 3DMark 11 is formed from the Graphics score, Physics score and the Combined score using a weighted harmonic mean, as follows:

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

$W_{Graphics}$ is the graphics score weight for the preset, $W_{physics}$ is the physics score weight for the preset and $W_{combined}$ is the combined score weight for the preset.

The constants and weights in the score formulas for each preset are set as follows:

	ENTRY	PERFORMANCE	EXTREME
$W_{graphics}$	0.70	0.75	0.80
$W_{physics}$	0.20	0.15	0.10
$W_{combined}$	0.10	0.10	0.10
$C_{graphics}$		230	
$C_{physics}$		315	
$C_{combined}$		215	

ABOUT UL

UL is an independent, global company that offers a wide range of testing, inspection, auditing, and certification services. With 10,000 people in 40 countries, UL helps customers, purchasers, and policymakers navigate market risk and complexity. UL builds trust in the safety, security, and sustainability of products, organizations and supply chains – enabling smarter choices and better lives. Visit https://www.ul.com/ to find out more.

UL benchmarking software is developed by the Product Supply Chain Intelligence division. We enable global product compliance, innovation and promotion throughout the supply chain with our intelligent software and services backed by world-class scientific and technical expertise. Please visit https://psi.ul.com/ to find out more.

UL benchmarks help people measure, understand and manage computer hardware performance. Our talented team creates the industry's most trusted and widely used performance tests for desktop computers, notebooks, tablets, smartphones, and VR systems.

We work in cooperation with leading technology companies to develop industry-standard benchmarks that are relevant, accurate, and impartial. As a result, our benchmarks are widely used by the press. UL maintains the world's largest and most comprehensive hardware performance database, using the results submitted by millions of users to drive innovative online solutions designed to help people make informed purchasing decisions.

Our benchmarks are developed in Finland just outside the capital Helsinki. We also have a performance lab and sales office in Silicon Valley and sales representatives in Taiwan.

Press UL.BenchmarkPress@ul.com
Sales UL.BenchmarkSales@ul.com
Support UL.BenchmarkSupport@ul.com