3DMARK®06

3DMark06 Whitepaper v1.0.2

Dated: January 18th, 2006
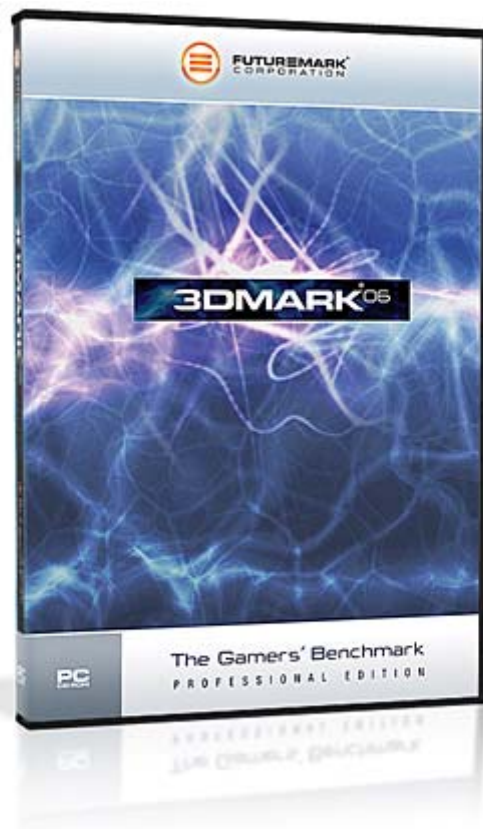
Nicklas Renqvist & Mikko Kallinen

Futuremark® Corporation

Kappelitie 6 D
FIN-02200 Espoo
FINLAND

Index

Introduction

This paper introduces 3DMark®06, the latest in the 3DMark benchmark series built by Futuremark® Corporation. The name 3DMark has become a standard in 3D graphics benchmarking; since the first version released in 1998 it has grown to be used by virtually all on-line and paper publications. It has proven itself as an impartial and highly accurate tool for benchmarking 3D graphics performance of the latest PC hardware. 3DMark has a very large following worldwide among individual PC owners.

More than 30 million benchmark results have been submitted to Futuremark's Online ResultBrowser database. It has become a point of great prestige to be the holder of the highest 3DMark score among PC enthusiasts as well as among the graphics hardware vendors. A compelling, easy download and installation, and easy-to-use interface have made 3DMark very popular among game enthusiasts.

Futuremark's latest benchmark, 3DMark06, continues this tradition by providing a state-of-the-art Microsoft® DirectX® 9 3D performance benchmark.
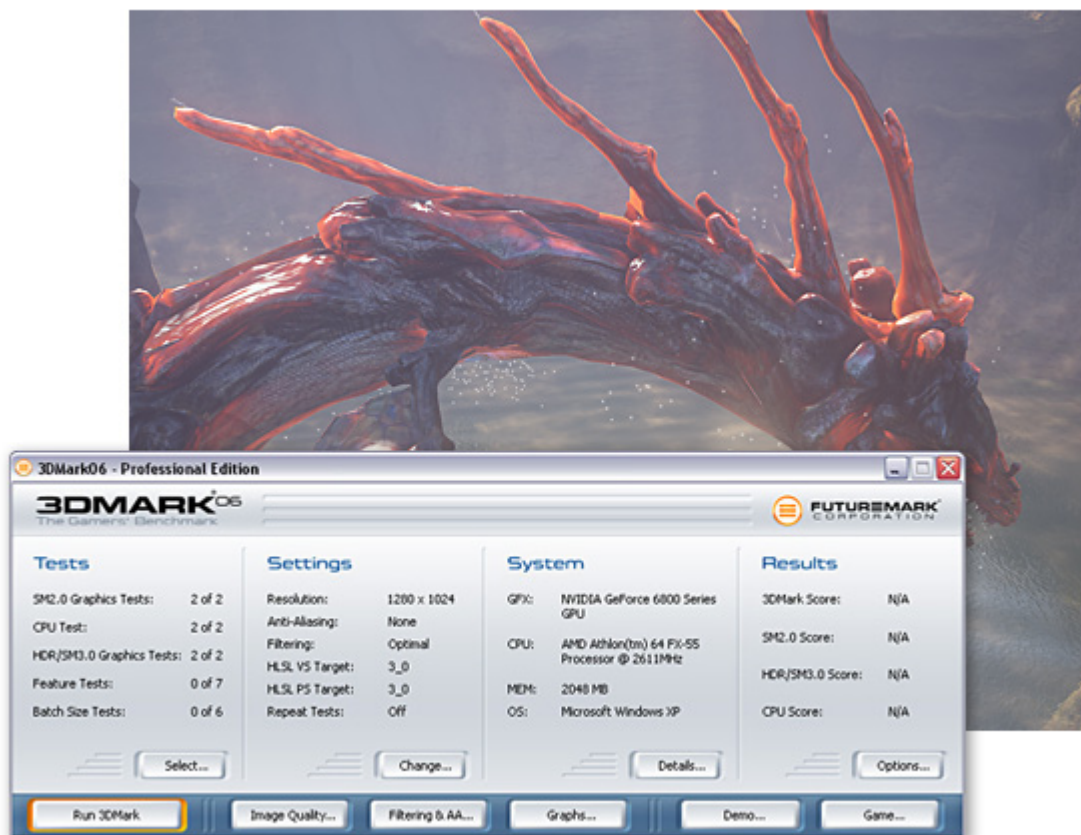
Figure 1: 3DMark06 is Futuremark's latest DirectX 9.0 benchmark with support for ShaderModel 2.0, ShaderModel 3.0 and HDR

3DMark06 is an all new 3DMark version taking all out of Microsoft's DirectX 9.0c. The previous version, 3DMark05, was the first benchmark requiring ShaderModel 2.0. However 3DMark05 used DirectX 9 specific features in a slightly limited manner, because fully ShaderModel 3.0 supporting hardware was rare at the time of its development and launch. In contrast, 3DMark06 requires DirectX 9 hardware with full support for at least ShaderModel 2.0 and for some of the tests ShaderModel 3.0, and once again takes shader usage to levels never seen before.

New Elements, Features and Philosophy

3DMark06 is different in many ways in comparison to its predecessor despite re-using some of the same content. The complexity of the updated graphics tests is raised to new heights by adding more lights, more shadows, more complex geometry etc. The introduction of ShaderModel 3.0 and HDR in two graphics tests are the key elements of 3DMark06. It is also noteworthy that what were called "*Game Tests*" in previous 3DMark's, are now being called "*Graphics Tests*". This change was made since the graphics tests are optimized for pure graphics performance measurement. In light of this, 3DMark06 now takes into account the CPU result based on the CPU tests when calculating the final 3DMark score. The logic behind this change in philosophy of the 3DMark score is the fact that games are increasingly using more advanced AI and complex physics making the CPU an important part of the gaming performance, and since 3DMark is the Gamers' Benchmark, it is logical to follow the same path as games. All these changes make 3DMark06 very different from its predecessor 3DMark05, though they might look alike.

3DMark06 also introduces new sub scores in addition to the 3DMark score:

- SM2.0 Score
- HDR/SM3.0 Score
- CPU Score

The sub scores are extremely useful for anyone who wants to compare different graphics card's or CPU's performance. The sub scores are as comparable as the overall 3DMark score, but are purely graphics / CPU bound results.

3D Graphics Benchmarking

3D graphics benchmarking allows home users to evaluate the performance of their installed 3D hardware. Professional hardware reviewers can compare the performance and features of the competing 3D hardware parts on the market by using those same 3D graphics benchmarking tools. There are different benchmark tools available. There is benchmark functionality included in 3D games, and there are stand-alone benchmark applications. Both of these have their advantages.

Benchmark functionality in a 3D game can give a comparable measurement of how fast the game runs on the tested hardware. Different games may scale very differently on different systems. One game benchmark therefore only indicates how that system runs that specific game. Another game may yield quite different performance results on those same systems. This is both due to the different structures of the games, different feature usage, and different IHV specific optimizations. The user should therefore above all use game benchmarks of games that the user will mostly play. The measurements of a game that will not be played will not offer any relevant performance information for that user. It should also be noted that if the benchmark functionality of a game uses just a recording of the animation being run as the benchmark test, a.k.a. timedemos, many factors influencing the performance during game play may be missing. This in turn gives a different performance measurement than would be observed by actually playing the game.

On the other hand, adding many runtime elements to the benchmark run of the game, including random elements, may change the benchmark result from one run to another. A single measurement of a benchmark like this is therefore not comparable to results of other systems, since the measurement is not repeatable on the same system. Multiple runs of multiple game benchmarks are therefore needed to get reliable performance comparisons between systems.

Benchmark applications are designed to provide reliable and comparable performance measurements. High quality benchmark applications give the very same result (or results within a very small error margin) when run on the same system. Even a single benchmark result thereby gives a truly comparable performance measurement. Professional hardware reviewers may still want to repeat the benchmark run to make sure there is no variance from one run to another. High quality benchmark applications also aim to measure more of an overall performance, than that of a single game. The documentation of the benchmark application should clearly state how the benchmark is designed to scale; does it above all scale with the graphics performance, the CPU performance or does it balance both and in that case how.

A CPU / graphics card workload balance should naturally be balanced to mimic a 3D game load, since that is the most usual 3D load the system will be used for. Then again, what could be a better balance than that of a real game. This makes benchmarks in games ideal for balanced CPU / graphics benchmarking. On the other hand, games are often heavier on the CPU than on the graphics card. This means that the total system throughput in many games is limited by the CPU performance, making such benchmarks less ideal for graphics performance estimations. Also, in the past few years graphics performance has increased faster than CPU performance, which has been possible to observe by the fact that all benchmarks over time have become more and more CPU dependent. Each new graphics hardware generation has pushed the envelope so far that even the most graphics limited benchmark has reached the point where the graphics hardware gets the frame rendered and has to wait for the CPU to catch up before starting to render the next frame.

3DMark06 is balanced to measure gaming performance due to the new ideology of the 3DMark score. For the first time in its history, the CPU result affects the final 3DMark score. This was done due to the fact that there are more and more games using complex AI algorithms and complex physics calculations. The graphics tests have been optimized to be as GPU bound as possible, hence they aren't affected much at all by the CPU. In order to create a very game-like scenario, all graphics tests and the CPU tests are used for the final 3DMark score, and thus correlate very well to what performance users will see in future games.

3DMark06 Compared to Previous Versions

3DMark99 concentrated on measuring fixed function vertex transformation and lighting, and multitexturing. **3DMark2000** added support for graphics hardware supporting transformation and lighting, and the complexity of the game tests was increased. **3DMark2001** increased the complexity of the fixed function game scenes to tens of thousands polygons per frame on average and also introduced shader technology. The scenes mainly used fixed function vertex and pixel processing, while shaders were used for special effects. There was skinning, morphing and massive amount of animated grass and leaves, all using 1.1 vertex shaders. Game test 4 presented the first higher level material using a 1.1 pixel shader.

3DMark03 concentrated on testing ShaderModel 1.x and 2.0. Only one game test, meant for legacy systems, offered fixed function multitexturing, while the other three used pixel shaders for all materials. All vertex processing used vertex shaders, mainly of the 1.x model. The last game test presented the first vertex and pixel shaders of ShaderModel 2.0, while the majority of the shaders in that test still were of the 1.x model. The scene complexity was raised to several hundred thousand polygons per frame on average.

3DMark05 raised the technology bar and used exclusively ShaderModel 2 (and 3) for all vertex and pixel processing. DirectX 9 also presented ShaderModel 2a, 2b and 3.0. All compatible shaders could be run using any of the ShaderModel 2.x profiles or 3.0. This is enabled throughout 3DMark05 based on our understanding that this is what shader-heavy games will offer. The scene complexity was raised to over one million rendered polygons per frame on average.

3DMark06 once again breaks the technology boundaries and uses extensively ShaderModel 2.0 and 3.0 for all vertex and pixel processing. The key ShaderModel is 3.0 and in 3DMark06 there are 2 graphics tests using almost all key SM3.0 features, and above all, rendered in HDR. The scene & shader complexity and use of dynamic soft shadows has again been raised incrementally over 3DMark05.

Version Comparison Chart

Feature	3DMark05	3DMark06
ShaderModel	2.0	2.0 and 3.0
HDR Rendering	No	Yes
Dynamic Shadows	2 LiSPSM Shadow Maps	5 Uniform Shadow Maps
Smooth Shadows	Yes (4 sample)	Yes (16 sample & 4 sample)
Graphics Test	3 (All SM2.0)	4 (Two SM2.0 + Two HDR/SM3.0)
CPU Tests	2 (Software vertex shaders)	2 (Complex AI and physics)
SM3.0 Feature tests	No	Yes
Number of tests	16	19
Level of Detail	High	Very High
Texture Size	High	Very High
Materials	Simple	Complex
Shader Complexity	High (Max 96 instructions)	Very High (Max 512 instructions)
Amount of lights	High	Very High
Sub Scores	1 (CPU Score)	3 (SM2.0, HDR/SM3.0 and CPU Score)
VRAM Usage	~128MB	~256MB
Playable Game	No	Yes
Target GPU	2nd Generation SM2.0 or better	2nd Generation SM3.0 or better
Target CPU	Single Core	Dual/Multi Core & Single Core

Version Comparison Screen Shots

Comparison SM2.0 "Return to Proxycon": 3DMark05 on the left, 3DMark06 on the right

Comparison SM2.0 "Firefly Forest": 3DMark05 on the left, 3DMark06 on the right

Comparison HDR/SM3.0 "Canyon Flight": 3DMark05 on the left, 3DMark06 on the right

ShaderModels

DirectX shader technology was introduced with DirectX 8, which brought ShaderModel 1.0 (SM1.0). Shaders are short programs for manipulating vertices and pixels in a Direct3D scene, and these can be efficiently executed on dedicated graphics hardware. There are two types of shaders: vertex shaders and pixel shaders. SM1.0 brought quite a huge leap forwards from fixed function vertex transformation and lighting to fully programmable vertex shaders and from fixed function multitexturing to programmable pixel shaders. Just transformation and lighting, and multitexturing, naturally can be done using shaders, but they can do so much more.

With the introduction of SM1.0, it was possible to perform character skinning also on the graphics hardware, while previously the only option was to execute it on the system CPU. On the pixel processing side, the earlier DirectX versions always illuminated everything per vertex and only offered some very limited fixed function per pixel lighting functionality, like environment bump mapping, or DOT3. Pixel shaders made it possible to illuminate all surfaces in the scene per pixel, using custom programmed surface materials. This was a huge step towards more photorealistic rendering.

ShaderModel 1.0 (SM1.0) was still fairly limited. Each shader, especially pixel shader, could only be very short and the operations available were limited. SM1.0 was extended with pixel shader 1.4, offering more flexibility, but shader usage was still limited. The shaders themselves had to be written in the cumbersome shader language, which closely resembles CPU assembly code. It is very low level, hard to learn for a less experienced programmer, and is far from modern elegant and structured programming languages. Most games utilizing SM1.0 therefore do traditional multitexturing on most surfaces, and use pixel shaders only for special effects on some surfaces.

ShaderModel 2.0 (SM2.0) extended shaders greatly, offering both longer and more complex shaders and more operations to choose from. SM2.0 additionally added the ability to perform the calculations using floating point values. This is significant, since the physical lighting calculations of the real world are all floating point dominated. SM1.x fixed point lighting calculations were therefore only crude approximations, offering a lower level of photorealism. DirectX9 and SM2.0 also brought the higher level shader language (HLSL). Instead of writing shaders in the assembly type shader language, developers could now use a higher level language somewhat like C-code.

DirectX provides an efficient compiler that compiles HLSL to the low level shader language - but that's not all. DirectX includes a number of shader profiles dedicated to different hardware architectures, usually also extending SM2.0. The developer can now write one HLSL shader and DirectX compiles it to optimal shader language for the installed hardware. The available profiles in the DirectX 9 Summer Update 2004, also called DirectX 9c are: the basic 2.0, 2.a, 2.b and 3.0 (or also expressed as 2_0, 2_a, 2_b and 3_0).

ShaderModel 3.0 (SM3.0) is the new key element of 3DMark06. Though it has been available in DirectX9 for quite some time it has not been used significantly in any games or applications until recently. It is difficult to pinpoint the reason, but the lack of wide audience with SM3.0 hardware could be one. Today we have more IHV's with SM3.0 compliant hardware, so expect things to change. SM3.0 opens up a lot of new features to be used. vPos register, dynamic flow control, derivative instructions, large number of instruction slots etc. are all key elements of SM3.0 and are features game developers will be using in their upcoming titles.

3DMark06 uses both SM2.0 and SM3.0 extensively, and uses by default the highest compilation profile supported by the hardware, including 3.0. The profile can also be manually selected for interesting performance comparisons between the different profiles.

Our Development Methodology

Futuremark approaches all of the benchmarks it creates with a standard development methodology. We believe that the process we follow is central to the development of a successful and dependable benchmark. The key part of the development process is cooperation with all major manufacturers. For 3DMark, we have found that groups with the most interest in this process are: graphics chipset makers, CPU manufacturers, and PC manufacturers. These companies are willing to cooperate with us because they share the vision that strong, objective benchmarks are in everyone's interest. We have been running a formalized Benchmark Development Program (BDP) that allows these vendors to participate in designing leading benchmarking standards in the PC industry. The cornerstones of our design process are *transparency* and *neutrality*. We make a strong effort to document all processes that make up the benchmark; we continuously strive to make these documents better. Also, we always try to maintain the highest standards of neutrality, neither favoring nor ignoring any party.

Figure 2: Benchmark Development Methodology

Step One. In this first step, we use a variety of resources to enumerate a first set of possible features for the benchmark. We draw upon our own experience with previous benchmarks and the feedback we have subsequently received. Communications with the gamer and game developer communities also provide valuable input. The BDP member group is another source of insights. From these, we develop high-level ideas of features we may include in the benchmark. At this early stage, features and implementation options are intentionally kept open.

Step Two. The document produced in the previous step is a proposal. It is designed to present features and implementation options in a format amenable for getting constructive feedback. This proposal is circulated to our BDP members. The feedback received helps us in choosing features and implementation methods.

Step Three. In the third step we create a written benchmark specification. Each of the workload tests is specified in detail with exact versions of the technologies used. The specification is circulated to the BDP members. The feedback is analyzed and incorporated at our discretion.

Step Four. In the next step we implement prototype code to see if the available technology will support our plans. We may discover the certain tests are not possible, or may be surprised to discover that more can be achieved. The results are incorporated into the specification and again circulated.

Step Five. The fifth step consists of implementing the workloads or tests. Periodic releases are made to the BDP members and their feedback may again be taken into account.

For more information and an up-to-date list of participating BDP members, please refer to www.futuremark.com/bdp.

3DMark06 Overview

Benchmark Structure. 3DMark06 is a collection of 3D graphics tests and CPU tests. These include a set of four *graphics tests* and two *CPU tests*; these are the tests used to calculate the overall 3DMark06 score. The benchmark also includes a set of *feature* and *batch size* tests. Each of these tests measures specific 3D-related functionality, but their result is not included in the overall score. They do not fall into the target usage, but are included to allow the user to evaluate these features. The feature tests isolate the performance of some key 3D features. The batch size tests expose a traditional weakness of 3D drivers; executing rendering batches of different sizes. The software also includes a set of image quality tools. These help studying more closely the different benchmark settings improving the rendering quality, and provide a powerful way to ensure integrity of the graphics hardware and drivers.

Real-Time Rendering. Each 3DMark06 graphics test is a real-time rendering of a 3D scenario. It is important to note that these renderings are not merely animations or a set of recorded events; they are designed to function like 3D games work. As with 3D games, all computations are performed in real time. This is a critical part of our philosophy of 3D graphics benchmarking.

DirectX. All tests have been compiled and linked with **DirectX 9.0c (December 2005)** libraries. Every graphics test and almost all other benchmark tests too require DirectX 9 hardware with support for Pixel Shader 2.0 or higher.

3D Engine. Early versions of 3DMark used the MAX-FX 3D engine. 3DMark03 used a very lightweight DirectX wrapper as more work was transitioned to the API and graphics card with the introduction of the shader technology. The engine only had to load the artwork and the shader language shaders. Game tests 2 and 3 and the "Ragtroll" test used a more complex engine version rendering the dynamic stencil shadows.

For 3DMark05, we had an all new rendering engine. It was more like a game engine than the one we used in 3DMark03. Still, there were only rendering related tasks for the CPU in the Game Tests, no other CPU tasks typical for games.

For 3DMark06, we updated the engine from 3DMark05 and added support for extensive use of ShaderModel 3.0 and other DirectX9 features such as 16 bit floating point textures and 16 bit floating point blending. The main and key element of the new engine is support for High Dynamic Range (HDR) rendering. This enables extremely realistic scenes to be rendered, and visually looks incredible. HDR has been one of the hottest topics amongst game developers for some time, and it seems to be a feature many games will support in the near future.

The engine in 3DMark06 dynamically builds shaders for each material in HLSL format. These shaders are then runtime compiled to best fit the installed hardware, or the user may manually set which compilation profile to use. A professional hardware reviewer can compare the performance difference using different shader compiler profiles for the same hardware. Handling all materials and shaders, and all other rendering related tasks like the shadow system discussed below, adds quite a bit of work to the CPU and memory.

HDR. One of the next big things in real-time 3D is High Dynamic Range, or also known as HDR. 3DMark06 features two HDR/SM3.0 graphics tests which are prime examples of what HDR rendering looks like. There are several approaches to achieve some level of HDR, but in 3DMark06 the HDR/SM3.0 graphics tests wouldn't have been possible by cutting corners. The approach used in 3DMark06 is technically very taxing on the hardware, and very complex. The two HDR/SM3.0 graphics tests require a lot of precision in the lower end of the brightness range, while still having extremely bright values in specific places, such as the sun. To achieve the level of vividness you see in 3DMark06 wouldn't be possible with any other approach, unless giving up on the quality and accuracy.

The HDR/SM3.0 graphics tests require both 16 bit floating point texture, and 16 bit floating point blending support. The HDR/SM3.0 graphics tests also use 16 bit floating point filtering, but use a highly efficient shader emulation fallback for hardware with no support for 16 bit floating point filtering.

The HDR/SM3.0 graphics tests use a more advanced post-processing bloom effect to really get the most out of HDR rendering. The post-process also include a star shaped glow effect which simulates the six edge shutter of traditional cameras with exposure control. In addition to that a ghost/lens reflection effect is being done, and finally the whole image is being processed by tone-mapping in order to get the correct light values for normal displays.

ShaderModel 3.0. Key features of ShaderModel 3.0 used in 3DMark06:

- vPos Register
- Derivative Instructions
- Dynamic Flow Control
- Large number of interpolators
- Large number of constants
- Large number of instruction slots
- Texture instructions with explicit LODVertex
- Vertex Texture Fetch (Required in the *Shader Particles* feature test)

3DMark06 uses all key SM3.0 features, except for vFace Register. In addition to the key SM3.0 features listed above, 3DMark06 utilizes some other essential DirectX9 features for HDR rendering:

- 16 bit Floating Point Textures (Required in HDR/SM3.0 Graphics Tests 1 and 2)
- 16 bit Floating Point Blending (Required in HDR/SM3.0 Graphics Tests 1 and 2)
- 16 bit Floating Point Filtering (Supported in HDR/SM3.0 Graphics Tests 1 and 2)

Dynamic Shadow Rendering. Ever since 3DMark2001, an essential part of the 3DMark game tests has been the rendering of dynamic shadows. 3DMark2001 used projection shadow maps for dynamic shadow rendering. This is a fairly light method to render shadows and games released lately have used this method a lot, but it has limitations. For example, self shadowing does not work; an object cannot cast a shadow on itself. Also, the shadow is projected on all surfaces behind the object, even on a floor of a room three floors below the light source and the object! 3DMark03 used stencil shadows, which is a very different approach. Here, edges of the object seen from the light source are extruded as polygons away from the light. Objects in that shadow volume are then in the shadow. This technique works well and correctly as a global lighting solution and includes self shadowing. Then again, it is better suited for certain types of scenes and low polygon objects.

The selection of the edges to be extruded to become a shadow volume is quite expensive CPU work, the shadow volumes adds polygon load and the shadow volume polygons take up a lot of fill rate, even though they are invisible. 3DMark05 introduced our first real attempt in using perspective shadow maps (more specifically, Light Space Perspective Shadow Maps (LiSPSM)). The shadow technique is very good, but it wasn't really at the level as we wanted it to be. Since the introduction of LiSPSM in 3DMark05, there have been a lot of new ideas and approaches to real-time dynamic shadows.

3DMark06 uses a type of depth shadow maps called *Cascaded Shadow Maps* (CSM). The implementation is a new approach to achieve angle independent quality on dynamic shadows for all objects on screen. This method works by dividing the view frustum in to 5 sections along the z-axis. Each section gets shadowed by a standard uniform 2048x2048 shadowmap. If the hardware supports depth textures, a D24X8 or DF24 depth map is used. If Depth Textures are not supported, an R32F single component 32 bit floating point texture will be used as a depth map.

Hardware shadow mapping can be disabled, if a more exact rendering performance comparison is desired, but the hardware shadow mapping is on by default (with the exception of D24X8 in the HDR/SM3.0 graphics tests). A single buffer will be reused for all 5 sections.

Shadows from point light sources use a 1024x1024x6 cube map of the format R32F as depth map, unless the hardware supports D24X8 or DF24 depth textures. These depth maps sound enormous, and one would think they take up quite a bit of fill rate. They most certainly do, and 3DMark06 is therefore less sensitive to changes of the screen resolution. As with LiSPSM, in some cases even five 2K maps are not enough. Canyon Flight (HDR/SM3.0 Graphics test 1) is a scene which is one of the most difficult environments for any kind of real time dynamic shadows.

The large and rounded rock face far away from the camera is likely to have some surfaces' normal almost always nearly perpendicular to the light direction. In these places there may be some minor flickering at the edge of the shadow. This is caused by a phenomenon called *projection aliasing,* which is a yet unsolved problem in most shadow mapping approaches feasible for real time use.

In order to smoothen out the edges of the shadows, we have come up with a very neat technique. The two HDR/SM3.0 graphics tests in 3DMark06 use a 16 sample kernel that is randomly rotated for each pixel. By using the 16 sample rotated kernel, we are able to produce very high quality smooth edges for the shadows. Point sampling is currently used both for hardware shadow mapping and R32F depth maps. One could argue that 16 samples is a bit on the high side, but then again, 3DMark06 is an ahead–looking benchmark and we strongly believe that in a year we will see games using up to 16 samples to get smooth shadow edges. To get super-smooth shadows we would need to up the sample count to 128 or possibly even higher, but no current hardware would get realistic performance numbers with such high sample counts and no game developer would do it, at least yet.

The SM2.0 graphics tests use a 4 point sample rotated grid kernel to produce relatively smooth edges for the shadows, unless the hardware supports D24X8 Percentage Closer Filtering (PCF) or DF24 & FETCH4. For hardware with support for PCF or FETCH4, only one bilinear sample is fetched.

The depth map D24X8 is sampled using PCF in the SM2.0 Graphics Tests only. DF24 is sampled using FETCH4 with bilinear filtering in the shaders. If the hardware supports D24X8 or DF24 and hardware accelerated PCF or FETCH4, a single bilinearly filtered sample is taken. The non-hardware shadow mapped rendering path uses four rotated point samples without bilinear filtering. These implementations produce a bit different rendering, which can be seen in close inspection, by magnifying parts of frames with some shadow artifacts, and comparing these side by side.

One could argue that the hardware shadow mapping and hardware PCF or FETCH4 implementation vs. the un-accelerated point sampling code paths do not produce comparable performance measurements, since the resulting rendering shows slight differences. 3DMark06 was designed with the firm belief that those two are indeed comparable, and in the fact that it is the right way to reflect future 3D game performance. Our study has proved that over a dozen of the biggest game developers are using hardware shadow mapping for dynamic shadow rending in their latest or upcoming titles. So if hardware shadow mapping is supported, they should be used in depth shadow map implementations, because that is what is done also in the latest and future games. However, if the benchmark user wishes to compare exactly identical rendering performance across different architectures, hardware shadow mapping can be disabled in the benchmark settings, and the dynamic shadows are then always rendered using R32F depth maps and four point samples.

Texture compression. All color maps are DXT1 compressed, the alpha maps are DXT3 compressed, and all normal maps are DXT5 compressed.

Heterogeneous fog. Since this is a new effect to most people, it is in order to explain how it has been achieved. We simply ray-march through a volume whose density is defined by a volume texture as well as simple height based function. Since the density distribution has a very low frequency we are able to get away with using only 5 samples along each ray. The arithmetic to texture ratio for this shader is around 20:1, meaning that this effect puts a lot of pressure on the pixel shader units. This effect can be used in many different ways, and we use it both as a humid fog effect in *Canyon Flight* as well as a snow storm effect in *Deep Freeze*.

Subsurface scattering effect. To create real subsurface scattering is still not really feasible on any current hardware since realistic subsurface scattering would require extremely heavy ray-tracing computations. There are several tricks to achieve a subsurface scattering effect, and in 3DMark06 we use such an effect in the HDR/SM3.0 tests. The basic idea is to estimate the thickness through an object with the angle of incidence between the viewing ray and the surface normal. Quite a bit of additional hand tuning is required to make the effect look more convincing. This approach is very simple compared to a full subsurface scattering implementation, but the effect works and is relatively lightweight. This effect is used in both HDR/SM3.0 graphics tests. In *Canyon Flight* it has been applied to the sea-monster and the flying creatures. In *Deep Freeze* it has been used for the snow. Without the effect, the snow wouldn't look as realistic as it does with the effect enabled.

3DMark06 Graphics Tests

SM2.0 Graphics Test 1: Return to Proxycon

Game Test 1 from 3DMark05 is being re-used in 3DMark06, but will be using the updated engine, improved artwork, more shadow casting objects, more lights and using the new shadow-technique. This test will still only require SM2.0. In this test a crew of space pirates attack and board a transport vessel with valuable cargo. There will be a 'full story' in the demo and a shorter version as the graphics test. Please watch the 3DMark06 demo for the whole story.

It should be obvious that this test reflects the 3D performance of shooter games, which many times take place indoors. In this test, the indoor areas are a bit larger, as opposed to the narrow corridors that are typical for first and third person shooters. The larger area allows a larger number of characters fighting in the same room, which is desirable especially in multiplayer games.
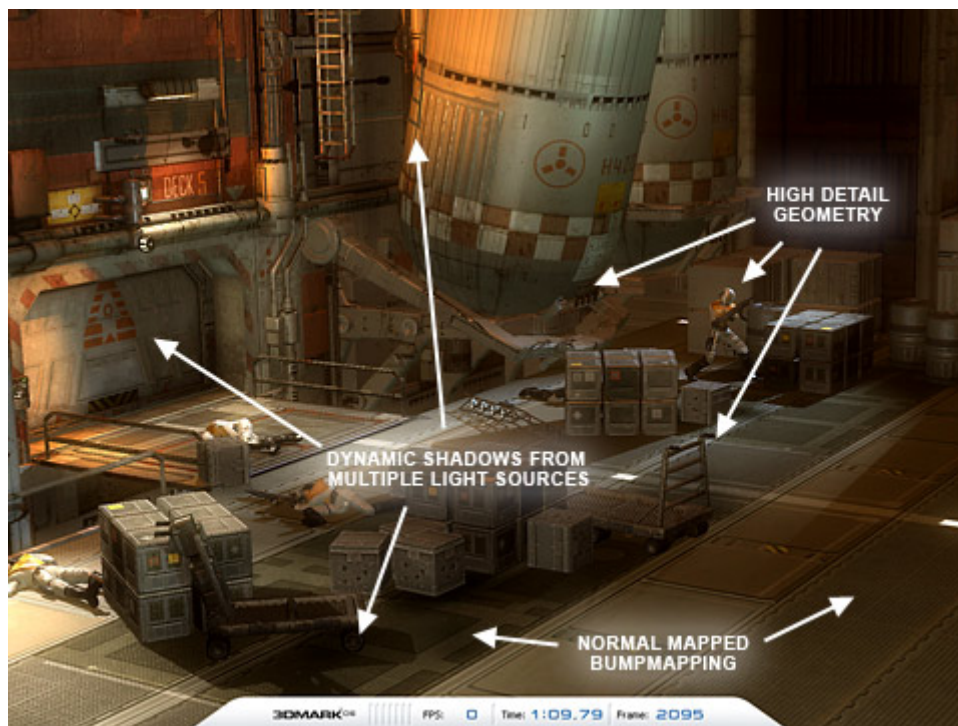
Figure 3: SM2.0 Graphics Test 1 "Return to Proxycon"

Materials. Most surfaces of the space ship interior are of a material doing a Blinn-Phong reflection. The exponent calculations are implemented to use lookups rather than calculating them mathematically.

Lighting. The numerous lights in the ceiling of the hangar are approximated with a directional lights from behind and above. Only one of the directional lights cast CSM. Additionally there are a number of point lights filling the total lighting nicely, and most of them cast shadows. The corridor has point lights throwing shadows, using a 1024x1024x6 cube depth map/hardware shadow map each, and some are masked and animated such as the rotating warning lights at the doorway, and at the end of the corridor. There are all in all 26 light sources in the graphics test level, two directional, 12 small non-overlapping shadowmapped spot lights and the rest are point lights.

SM2.0 Graphics Test 2: Firefly Forest

Again, 3DMark06 re-uses the test from 3DMark05, but using the updated engine, improved artwork and new shadow technique. This test will still only require SM2.0. The test has also another "firefly" flying in the forest, in order to create more graphics load. In this test a forest is inhabited by magic fireflies that fly around at night. The moon is nearly full, illuminating the forest with a bluish faint light. The magic fireflies have flickering bright green lights that playfully move around the forest. The graphics test only shows a part of this scene. Please watch the demo for the whole story.

This scene is a nice example of a smaller scale outdoor scene with rich vegetation. Immediate visibility is limited, and there is a skybox surrounding the whole scene. There are a large number of trees, all swaying in the light breeze, the branches swinging separately, and there is dense vegetation on the ground. The vegetation on the ground is actually one of the key interests in this test. It is dynamically distributed where needed, according to the camera movements. Its level of detail is also dynamically altered depending on the distance to the camera. The other key interest in this scene is the amazing lighting and dynamic shadow system. This scene really is ideal for showing the benefits of CSM and high resolution shadow mapped point lights.

Figure 4: SM2.0 Graphics Test 2 "Firefly Forest"

Materials. The ground material is similar to the metals in graphics test one, but with added diffuse, diffuse detail, normal and normal detail maps. The rock surfaces also have a specular map. The tree branches are also a modified metal material without a specular map and with a diffuse cube map and no bump mapping. The sky is created using a procedural light scattering shader.

Lighting. The moonlight is directional, generating CSM. The illuminating fireflies are shadow mapped point lights with a cubemap mask. The illuminating fireflies are masked point lights, throwing shadows using a 1024x1024x6 cube depth map/hardware shadow maps.

HDR/SM3.0 Graphics Test 1: Canyon Flight

Canyon Flight from 3DMark05 has been updated with the improved engine, HDR rendering, use of SM3.0, a new shadow-technique and a completely new shadow filtering technique. A lot of the artwork has also been improved. The graphics test only shows a part of this adventure. Please watch the demo for the whole story.

This test gives an example of a large scale outdoor scene with HDR rendering, smooth shadows and complex SM3.0 shaders. The scene is very complex with large areas of water reflecting the high canyon walls. The HDR rendering is one of the key points of interest in this scene, proving the increasing importance of floating point rendering to achieve realistic scenes. The water in this scene not only features realistic looking HDR reflections and HDR refractions, but it also has a depth fog, making the sea monster swimming under the airship actually look deep down in the water. The surface of the water is distorted using 2 scrolling normal maps and four Gerstner wave functions. This scene also uses a complex heterogeneous fog to make the whole canyon appear humid. The air in this scene also uses the same atmospheric light scattering algorithm as in 3DMark05 making distant cliffs of the canyon really look far away. The sky uses a more complex atmospheric light scattering algorithm than in 3DMark05, with cloud blending.
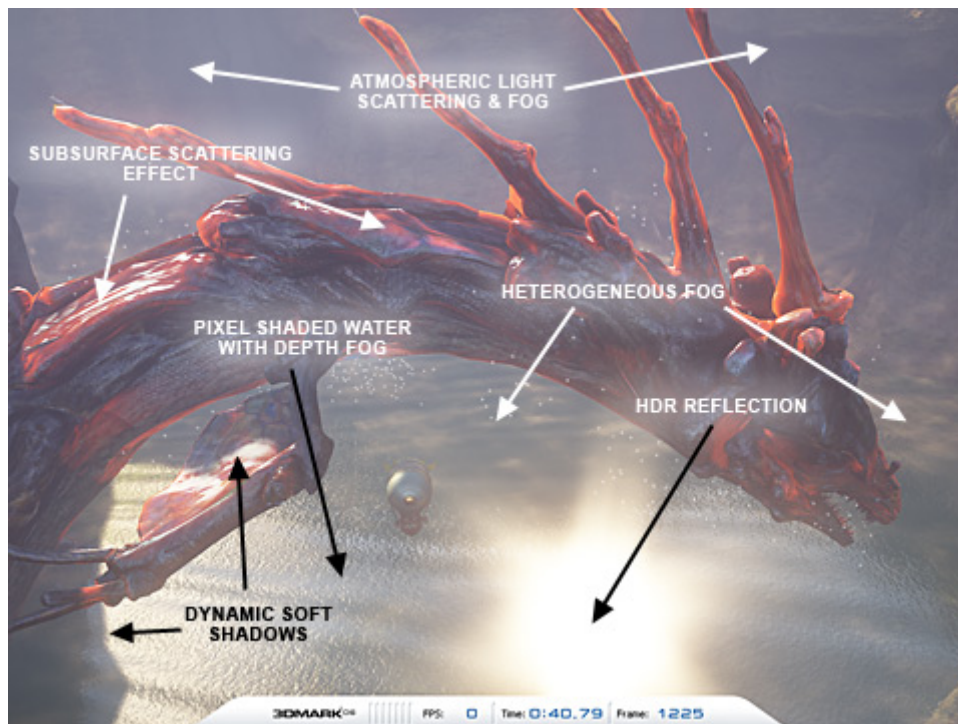
Figure 5: HDR/SM3.0 Graphics Test 1 "Canyon Flight"

Materials. The ship and the crewmen are shaded using Strauss shading model, making them look more realistic than in 3DMark05. The canyon material has three color maps, three normal maps and Lambertian diffuse shading. The water is a further developed version of the water shader in 3DMark05 doing 2 scrolling normal maps and four Gerstner wave functions and full HDR reflection and refraction. There is also a depth fog for making objects deeper look more blurred and darker. A R32F depth map is used for the depth fog.

Lighting. Since it is a sunny day, there is only one single directional light source - the sun. This scene is very challenging for dynamic shadows, because of the large area and the round shapes of the canyon walls but thanks to the CSM technique most of the artefacts are practically completely eliminated.
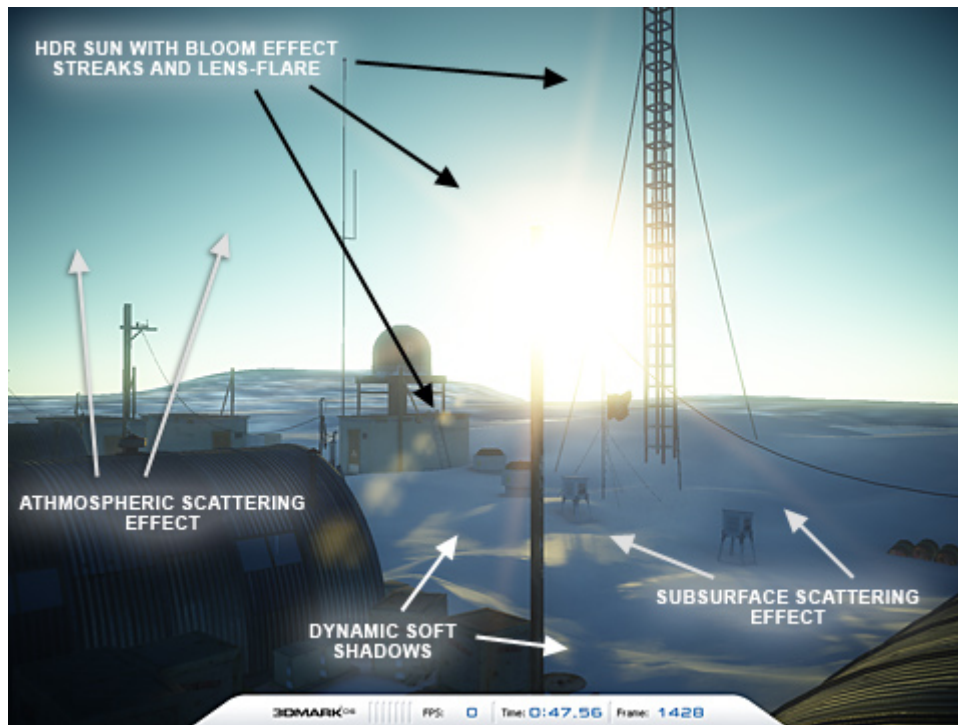
HDR/SM3.0 Graphics Test 2: Deep Freeze

Deep Freeze is a completely new graphics test, introducing an Antarctic research base. The mood in this test is very movie-like, and has a pinch of horror in it. The graphics test only shows a part of this wonderful scene, so please run the demo for the whole story.

This test is a good showcase for using HDR effects in vast landscapes, and stunning dynamic long soft shadows for daytime -> night-time scenarios. As the sun goes down, the shadows increase in length and really show off the robustness of CSM. The snow uses Blinn-Phong shading model, 2 normal maps and 1 color map. The metallic and other surfaces use Strauss shading model. This test also uses the heterogeneous fog, along with heavy use of particles, to create a nice snow-storm effect. The snow also uses the subsurface scattering effect.

Figure 6: HDR/SM3.0 Graphics Test 2 "Deep Freeze"

Materials. The snow surface in the graphics test uses the Blinn-Phong shading model, 2 normal maps and 1 color map as well as a custom approximation of sub-surface light scattering. The sky uses a procedural atmospheric light scattering. All other materials use the Strauss shading model, which makes them even more realistic.

Lighting. Since this test is also an outdoor scene, there is only one single directional light source - the sun. This scene is again very challenging for dynamic shadows, but thanks to the CSM technique this is now possible. The lighting and shadows all are dynamic, and they really show off as the sun goes down and the length of the shadows increases. The change in ambient lighting when the sun is setting, is achieved by blending two pairs of cubemaps - one for diffuse, and another for specular light. In this scene, floating point rendering is absolutely necessary due to the need for extra precision in the lower brightness range, plus the average HDR value is around 11.000 - neither of which would be feasible without using floating point rendering.

CPU Tests

CPU Test 1 & 2: Red Valley

The 3DMark06 CPU tests consist of a game scene with a maze of canyons, and 87 fast-moving game units ("bots", speeder bikes and hovering tanks). The speeders attempt to navigate to a goal position at a castle at the other end of the canyon system, all the while avoiding the defending tanks, and collisions with other speeders. The tanks attempt to hunt down the speeders, and shoot them.

The game scene yields three types of load in the CPU tests: game logic, physics and path finding AI. The game logic, including the graphics engine operation, runs in a single main thread that also drives the other two tasks. The physics simulation runs in a single separate thread, and is synched with the main thread at each physics step. The path finding AI runs in a number of worker threads (the number of threads is scaled with available processors), and is synched with the main thread at set intervals, generally some multiple of the physics step interval.

Figure 7: CPU Test

The CPU tests are run in fixed frame rate (2fps) to make a more equal CPU load for all systems. The resolution is locked to 640x480 and the tests use no dynamic shadows to decrease the graphics performance influence on the result.

The physics load consists of simulating the game world with its 87 units and their rigid bodies, at 20 ms physics step. Some physics operations like traces into collision meshes, and some overhead, is also included in the main thread load. The path finding AI load consists of unit path requests in a dynamic path finding grid, where each moving unit also represents a moving obstacle of certain radius on the grid. The resulting paths are synched back to the units at intervals ranging from 200 ms to 600 ms. The complexity of individual path request fulfilments varies greatly, as we use a dynamic re-planning algorithm that is often able to re-use state from previous searches. The length of the requested paths and the grid obstacle configuration also contribute to the path finding variance.

The path finding algorithm used is D* Lite (http://www.cc.gatech.edu/fac/Sven.Koenig/) and the physics are being computed using the AGEIA PhysX library (http://www.ageia.com).

There are two CPU tests (1 and 2) in 3DMark06. The tests differ in certain parameters affecting the type of game-like load that is run on the CPU. The main settings and differences between the tests are as follows:

Test 1:
Higher path finding task complexity
Tighter AI synchronization intervals
Duration 40 frames
Fixed Frame Rate 2 frames per second
Shader Profile 2_0
Resolution 640x480

Test 2:
Lower path finding task complexity
Laxer AI synchronization intervals
Duration 60 frames
Fixed Frame Rate 2 frames per second
Shader Profile 2_0
Resolution 640x480

The differences result in Test 2 having a larger relative physics load (around 13%) than the relative physics load exhibited in Test 1 (around 8%), and conversely a lower relative AI load.

Both CPU tests are forward looking, since CPU technology is clearly moving towards either virtual or physical dual core technology. The CPU tests are mainly designed for dual systems, either with an on-chip solution or with two separate CPUs, but are an excellent CPU test for single core processors as well. Professional reviewers can disable the other CPU or the other core of a virtual or physical dual core system, and compare the results.

Feature Tests

3DMark06 includes a very large set of feature tests. These do not affect the overall score. However, their results provide valuable information for the benchmarking professional, as they often cannot be obtained from any other source. 3DMark06 feature tests measure the performance of a few important 3D graphics features primarily relating to vertex and pixel shader technologies. It should be noted that even though the feature tests have been designed to measure a certain characteristic, other features of the hardware may affect the measurement. Please read this chapter carefully for more info on what may affect the measurements of the feature tests.

Fill Rate

3DMarks have included fill rate tests ever since 3DMark2000. These have been pretty much the same, and why not, since fill rate is one of the central features in graphics hardware regarding performance.
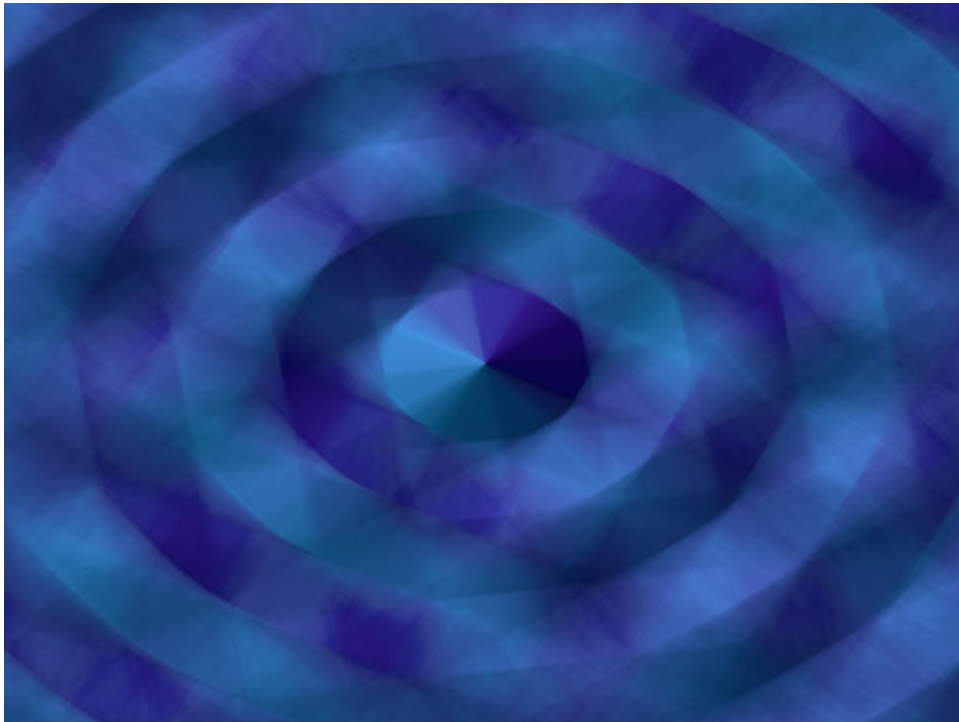
Figure 8: Fill rate Test

The fill rate tests of 3DMark06 the same as in 3DMark05; more theoretical, minimizing the influence of bandwidth and attempting to concentrate on measuring fill rate. The tests do not look as nice as the previous versions, but any nicer effect would require larger texture maps and thereby again move the bottleneck towards graphics memory bandwidth.

Single-Texturing. This test works just like in 3DMark05. The size of the texture used is 2x2 in order to decrease bandwidth limitation of the performance. 64 quads cover the screen and are single textured and additively blended.

Multi-Texturing. This test works just like in 3DMark05. The size of the texture used is 2x2 in order to decrease bandwidth limitation of the performance. 8 quads cover the screen and each quad has 8 textures additively blended.

Pixel Shader

One of the more complex materials in the graphics tests is the rock face shader of HDR/SM3.0 graphics test 1. This is separated to a feature test, showing the light moving on the rough surface. There are no real time shadows, only vertex lighting. There is also no water surface, only the rock face.
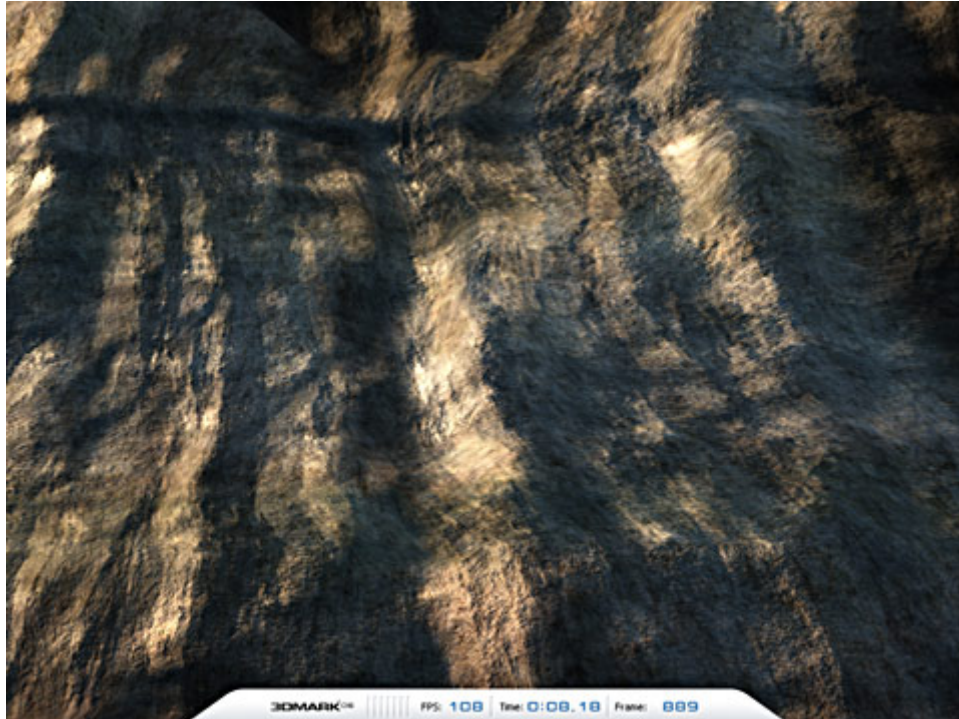
Figure 9: Pixel Shader Test

Filling the screen with a rock face is naturally fairly fast, compared to the graphics test showing huge amounts of that rock face in addition to water, the air ship and the sea monster. This test will be somewhat bandwidth dependent, since any game like material with a complex shader like this will also have a number of lookups to large textures. It seems like most PC games will mainly utilize normal color maps that have been made during development, instead of burdening the pixel shader with creating procedural textures.

Vertex Shader

Vertex shader feature tests have been included in 3DMark since 3DMark2001, which presented DirectX8, where vertex shaders were initially made available. The vertex shader tests replaced the earlier polygon throughput tests called high polygon tests in 3DMark2000. Polygon throughput together with fill rate are the most important single performance characteristics in 3D rendering. Since 3DMark03, 3DMark has used vertex shaders for all vertex processing in the graphics tests. Vertex shader tests have therefore been a logical substitute for the earlier fixed function vertex throughput tests. There are two different vertex shader tests in 3DMark06: One very simple doing only a very simple transformation and one light lighting. The other does a more complex vertex shader waving a large number of grass straws.

Vertex Shader – Simple. This test does simple transformation and single light lighting on four high polygon sea monster models. Each sea monster has over one million vertices to transform and illuminate, so the total workload is quite substantial. The vertex shader used here could quite well fit into a ShaderModel 1 vertex shader, but since 3DMark06 concentrates on SM2.0 and SM3.0 (and offers different SM profiles to choose from), the shader is declared in HLSL as SM2.0.

Figure 10: Vertex Shader Test - Simple

Vertex Shader – Complex. This illuminates, and most importantly, transforms a large number of grass straws. Each straw is skinned and bent separately, more towards the tip of the straw, like real grass straws waving in the wind. The straws are waved according to a fractal noise calculated on the CPU, but it is highly optimized to decrease the influence of the CPU performance on the measurement. The grass is kept at a distance from the camera, offering a less interesting visual effect, but this is necessary to decrease the influence of fill rate to the measurement.

Figure 11: Vertex Shader Test – Complex

Shader Particles Test (SM3.0)

This test runs simple particle physics using a pixel shader and displays the results through vertex texture fetches. The use of graphics hardware for physics computations in games is increasing. Simple physics computations are inherently parallelizable, which allows them to be implemented on graphics hardware fairly effortlessly. As a result a game developer is able to use far more particles than would be feasible on a CPU while the CPU is freed to perform other computations.


Figure 12: Shader Particles Test

The physics simulation in this test consists of Euler integration for the trajectory of 409600 particles under a simple gravity and air resistance model as well as checking for collisions against a height field. This test requires SM3.0 with hardware Vertex Texture Fetch (VTF) support.

Perlin Noise (SM3.0)

This test computes six octaves of 3-dimensional Perlin simplex noise using a combination of arithmetic instructions and texture lookups. Perlin noise is a basic building block in many procedural texturing and modelling techniques, which are expected to increase in popularity in future games due to both reduced memory and bandwidth requirements as well as the increasing computation power in graphics hardware.
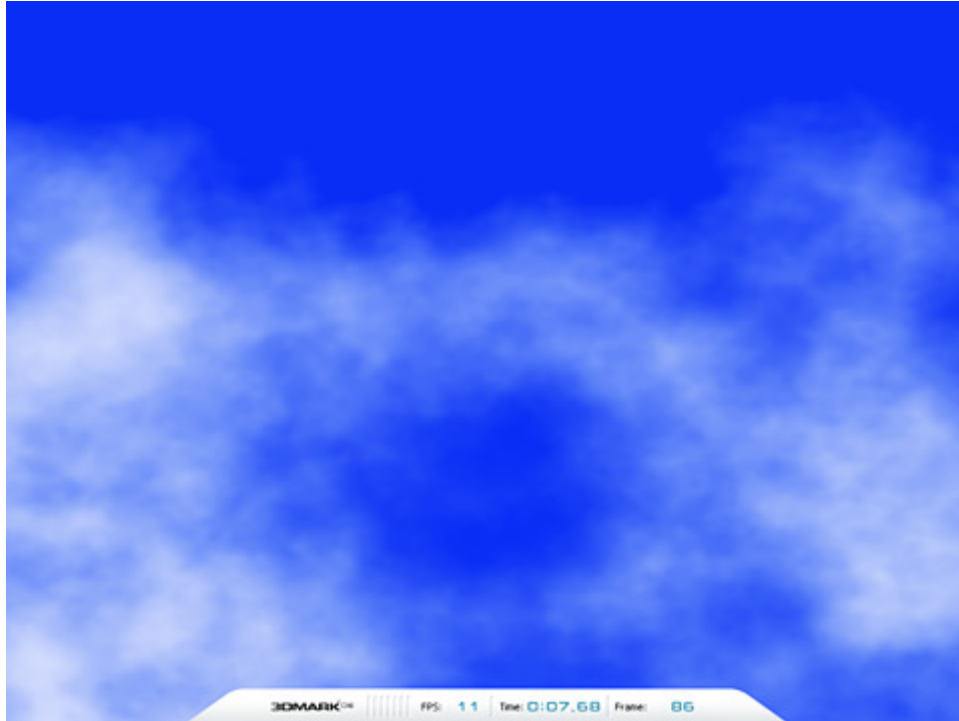

Figure 13: Perlin Noise Test

The pixel shader used in this test consists of a total of 48 texture lookups and 447 arithmetic instructions, resulting in a total of 495 instructions and around 9:1 ratio of arithmetic instructions to texture instructions. The minimum specification for PS3.0 allows shaders with up to 512 instructions, so this test rides pretty close to the limit. All texture lookups are made into a single 32 bit 256x256 texture (64 kilobytes), which keeps the memory bandwidth requirement reasonably low despite the large number of lookups. This test requires SM3.0.

Batch Size Tests

The test renders a very simple scene very much un-optimized, targeting a weak spot in most graphics drivers available today. Graphics IHVs have for years educated game developers to render as large batches as possible. However, it would be beneficial if the rendering of smaller batches would be optimized too.
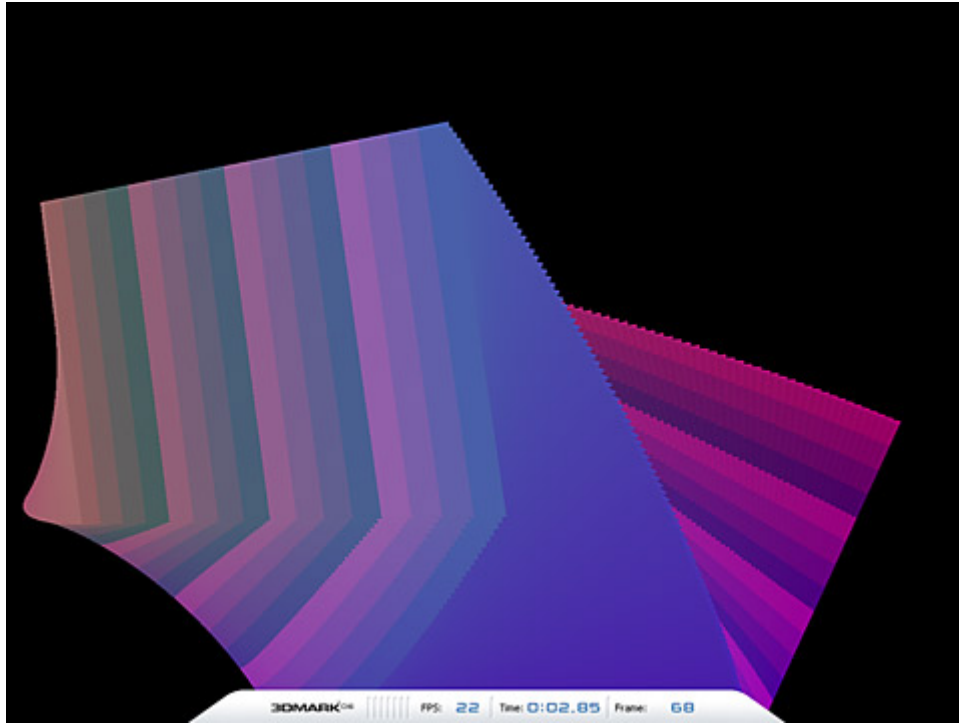
Figure 14: Batch Size Test

There are six runs of this test, where 128 meshes of 128x128 quads are draw with 8, 32, 128, 512, 2048 and 32768 triangles per batch.

The last two batch sizes should be considered an optimized one for most drivers today, but the smaller the batch sizes get, the slower the rendering will be. Color change state changes were added between the rendering batches to make sure DirectX doesn't collapse the whole rendering into a single or very few batches. Early versions of the test without the state changes were hit by this, and gave quite obscure results. The test therefore also is somewhat dependent on how fast the driver does rendering state changes.

3DMark Settings

The **Benchmark Settings** in 3DMark06 Advanced and Professional Editions introduces a multitude of options even for the most demanding benchmarkers.
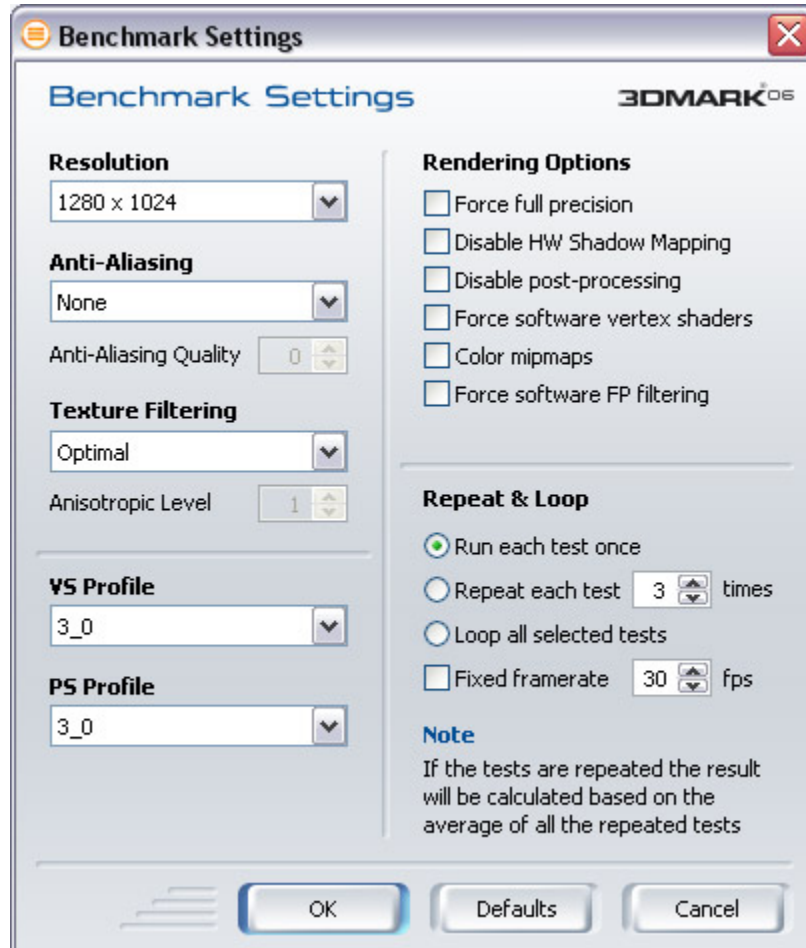

Figure 15: Benchmark Settings

Resolution. This drop-down menu allows you to select the frame resolution that all the tests are rendered at. Choosing a setting higher than the default value of 1280 x 1024 may result in test scores lower than with the default values. The range of resolutions available depends on what the graphics adapter and monitor both support. 3DMark uses 32 bit colors, with an 8-8-8-8 bit RGBA color palette in all tests. This setting does not affect the CPU tests.

Anti-Aliasing. This drop-down menu selects which level of Full Scene Anti-Aliasing (FSAA) to use, provided the graphics adapter has support for FSAA via DirectX. Choosing higher levels of anti-aliasing will produce a finer looking rendering, but the test results will be lower. You can choose between non-maskable anti-aliasing (IHV and driver dependent), or anti-aliasing of certain sample counts, depending on what your hardware supports. If you choose non-maskable FSAA, 3DMark automatically uses the lowest quality level. There is also an option to choose FSAA quality, which is offered by some hardware and drivers. The meaning of these quality levels varies a lot from one graphics card to another.

Please read the documentation of your graphics hardware to find out what these quality levels offer on your hardware. Hardware with no or limited support for FSAA in the HDR/SM3.0 graphics tests will skip the tests if FSAA is enabled (or an unsupported level of FSAA is selected), and no 3DMark score will be calculated. This setting does not affect the CPU tests.

Texture Filtering Mode. This drop-down menu allows you to select what form of texture filtering is applied in all tests. There are various options:

- **Optimal** - a mixture of bilinear and trilinear filtering is used, depending on what objects will benefit most from the higher quality textures.
- **Bilinear** - all textures are filtered using the standard bilinear interpolate. Depending on the test and graphics adapter, this setting may produce higher test results than the default setting.
- **Trilinear** - all textures are filtered using a trilinear interpolate. Depending on the test and graphics adapter, this setting may produce lower test results than the default setting.
- **Anisotropic** - all textures are filtered using anisotropic filtering. Depending on the test and graphics adapter, this setting may produce lower test results than the default setting.

If you select anisotropic filtering, you can set the anisotropy amount below the Texture Filtering drop-menu. The default anisotropy level is 1. Some value lookup textures are not affected by filtering mode changes, since this would not change the rendering in any way. This setting does not affect the CPU tests.

HLSL Profile. The HLSL Profile drop-menu selects HLSL profile to be used during the benchmark run. DirectX offers different compilation targets, each optimized for different hardware architectures. The HLSL shader code is the same for all hardware, but the optimal compiler target is selected by default for each graphics hardware type. The available compiler targets are:

- PS 2_0
- PS 2_a
- PS 2_b
- PS 3_0
- VS 2_0
- VS 2_a
- VS 3_0

By default, the best optimized compiler target is selected for all different graphics hardware. If all new graphics hardware is used, which is not identified by 3DMark06, the compiler target will be selected according to a 'best guess' based on the DirectX Caps bits. This setting does not affect the CPU tests.

Note: PS2.x and VS3.0 and PS3.0 and VS2.x may not be mixed. When either profile is manually changed, the other will be automatically changed to match.

Force Full Precision The *Force full precision* check box enables you to choose between partial (at least 16 bit) and full (at least 24 bit for SM2.0 tests and at least 32 bit for HDR/SM3.0 tests) floating point precision in shader calculations. By default, partial precision is used in a pixel shader but only in places where the rendering quality is at the required level; otherwise full precision is used. This setting forces full precision to be used in all shader calculations. Cards that only support a single shader calculation precision are not affected by this setting. An image quality analysis was made during benchmark development and it was pre-set which shaders may use partial precision. This setting does affect the CPU tests.

Disable HW Shadow Mapping. The *Disable Hardware Shadow Mapping* check box allows you to disable hardware shadow mapping usage in dynamic shadow rendering. HWSM is enabled by default on hardware supporting it. HWSM is an additional hardware feature that can be used to optimize dynamic shadow rendering. The resulting shadows are of the same quality as those produced by hardware without this support. This setting is implemented for professional reviewers that may want a more "apples to apples" comparison for some hardware reviews. The supported hardware shadow mapping formats are D24X8 (SM2.0 GT1 and SM2.0 GT2) and DF24 (SM2.0 GT1, SM2.0 GT2, HDR/SM3.0 GT1 and HDR/SM3.0 GT2). This setting does not affect the CPU tests.

Disable Post-Processing. The *Disable post-processing* checkbox disables full frame post-process effects in all graphics tests. The post-processing in the SM2.0 tests does bloom filtering that is done by rendering every frame to a texture and modifying the frame using a pixel shader. The bloom filter is a cinematic effect that makes bright surfaces appears over bright. The bright surface not only gets brighter discarding any details too small to be visible in the intense reflection, the light also partly spills over darker areas in the frame surrounding the bright surface.

The HDR/SM3.0 graphics tests use a much more advanced post-processing bloom effect, where the scene is rendered to a texture and progressively halved downscaled versions of that is generated (a bit like mip-maps). Each of these sub-frames is blurred using a Gaussian filter of the same pixel size. The final frame is then produced of a weighted sum of those downscaled and blurred frames. We are also using a star shaped glow effect that again uses the progressively halved frame copies for six separate line convolutions, one for each of the 6 streaks in the star. Each line is rendered in 3 passes. The first pass takes 5 samples of the image at the beginning of a line. The second pass takes again 5 samples from the output of the first pass, spaced five pixels apart. Finally the third pass takes yet again 5 samples from the output of the second pass, spaced 5*5=25 pixels apart. This simulates the six edge shutter of traditional cameras with exposure control. Also a ghost/lens reflection effect is being done. We paste copies of this new buffer on the screen, each scaled around the centre of the screen using suitable scaling factors.

In addition the above, the whole image is being processed by tone-mapping in order to get the correct light values for normal displays. Disabling the post-processing from the options disables all the effects mentioned above, and thus does not represent the way the scenes were designed.

In 3DMark06 the post-processing is always on by default. This setting does not affect the CPU tests.

Force Software Vertex Shaders. The *Force software vertex shaders* check box controls whether vertex shading routines in the tests are processed by the graphics adapter or the CPU. Note that if the graphics adapter does not support hardware processing, all vertex shaders will be processed by the CPU anyway. Selecting *Force software vertex shaders*, if hardware accelerated is available, may result in some tests producing significantly lower results. On the other hand, running the default benchmark with forced software vertex shaders is an excellent way to test CPU performance. By default all vertex shaders are run in the graphics hardware if supported. This setting does affect the CPU tests.

Color Mipmaps. The *Color mipmaps* check box colors mipmaps red, green and blue. Mipmaps allow textures to be decreased in resolution if they are further away from the camera, thus saving GPU processing power and reducing annoying flickering of sub-pixel texels. Coloring mipmaps produces a somewhat distorted rendering, but it is a useful feature for inspecting texture filtering quality. The full size textures are not colored. This setting does affect the CPU tests.

Force Software FP Filtering. The *Force software FP filtering* check box controls whether the post-processing filtering is done in hardware (FP16 filtering) or in the shader (software FP filtering) in the HDR/SM3.0 graphics tests. This setting is implemented for professional reviewers that may want a more "apples to apples" comparison for some hardware reviews. By default all filtering is being done in the graphics hardware if supported. This setting does not affect the CPU tests.

Repeat & Loop. Selecting Repeat will cause each test to run the number of times chosen. The final result is given as an average of the repeated runs. Loop will make 3DMark06 run all of the selected tests in a continuous loop until the ESC key is pressed during a test. This is feature is useful for testing the long-term stability of a system. Please note that other issues than system stability may cause the loop to be interrupted.

Fixed Framerate. 3DMark06 provides two different real-time rendering mechanisms: *time-based* rendering and *frame-based* rendering.

Each graphics test has a timeline or natural pace of action. Time-based rendering adjusts the frame-rate to maintain this timeline. For simple scenes, the hardware may be able keep up a high frame-rate; for complex ones, it may need to lower the frame-rate to maintain the timeline. For very low-end hardware, the user may even see a "slide-show" effect, as the hardware struggles to keep up the natural pace of action.

In addition to time-based rendering, 3DMark06 provides an option for a frame-based rendering mechanism that renders a fixed number of frames for each second of the timeline. The number of frames is user configurable. Frame-based rendering forces each run to generate exactly the same number of total frames regardless of the PC used. Of course, the hardware would render these frames at the fastest pace that it can handle, so a fluctuating frame-rate is still observed.

The default settings in 3DMark06 are as follows:

- 1280 x 1024 resolution (except the CPU tests which are locked at 640 x 480)
- 32-bit colour depth (8-bit RGBA channels).
- 24-bit z-buffer with an 8-bit stencil.
- DXT1 and 3 compressed texture formats and DXT5 for normal map compression.
- No anti-aliasing.
- Optimal texture filtering.
- Highest supported shader model available.
- Partial precision is used where allowed in the shaders.
- Hardware shadow mapping enabled if supported by the hardware.
- Post-processing enabled.
- Hardware accelerated vertex shaders (dependent on hardware support).
- Mipmaps not colored.
- Hardware FP16 filtering (dependent on hardware support).
- Single test run benchmarking.
- Time-based benchmarking mode (except the CPU tests which are locked as frame-based).

Tools

3DMark06 includes an improved set of image quality tools for the professional tester. There are three tools included: *Image Quality* tool, *Texture Filtering* and *Antialiasing* tool and *Graphs* tool.

Image Quality Tool. This powerful tool allows the user to choose any set of frames from any of the tests for quality inspection. The selected frames are rendered using the graphics hardware as well as using the DirectX reference rasterizer. Alternatively the same set of frames can be rendered on two different PC systems.

Figure 16: Image Quality Tool

The user's favourite imaging software can then be used to visually compare the frames for rendering quality. Various options are available to allow fine-grain control of this tool. All settings are the same as found in the *Benchmark Settings*.

The user selects the first and last frame to be rendered. When generating the frames, the tool uses frame-based rendering at 30 frames per second. For frames to be rendered using the DirectX reference rasterizer, the DirectX®9 SDK (December 2005) must be installed on the PC.

This tool and the frame-based rendering option mentioned earlier combine to form an excellent mechanism to ensure integrity of the graphics hardware and drivers.

Texture Filtering and Anti-Aliasing Tool. This is a synthetic image quality tool for visually verifying texture filtering quality. The tool highlights for the user any imperfections or artifacts that might occur on particular hardware.
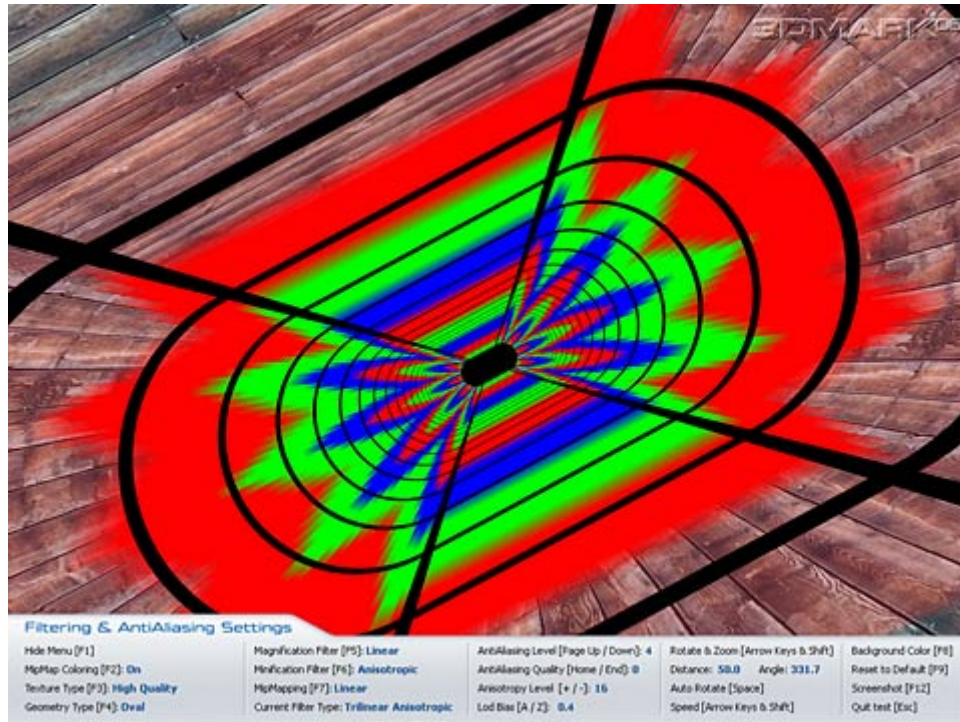

Figure 17: Texture Filtering & Anti-Aliasing Tool

The camera can be controlled using the keyboard for better image inspection; it can be moved back and forth as well as rotated either manually or automatically. Keyboard options also allow the user to switch between different types of textures, geometry, level of anisotropic filtering, level of anti-aliasing, LOD Bias and more. Filters to handle Magnification and Minification artifacts can also be chosen; nearest point, linear, and anisotropic are available. The Mipmapping approach used can also be chosen from: none, point or linear. The filtering modes typical for 3D applications are:

- Bilinear filtering - linear magnification, linear minification, point mipmapping.
- Trilinear filtering - linear magnification, linear minification, linear mipmapping.
- Bilinear Anisotropic filtering - anisotropic magnification, anisotropic minification, point mipmapping.
- Trilinear Anisotropic filtering - anisotropic magnification, anisotropic minification, linear mipmapping.

Graph Benchmark Mode. The Graphs tool records per frame runtime statistics of a selected graphics test with selected settings, and draws a MS Excel graph of the recorded data.
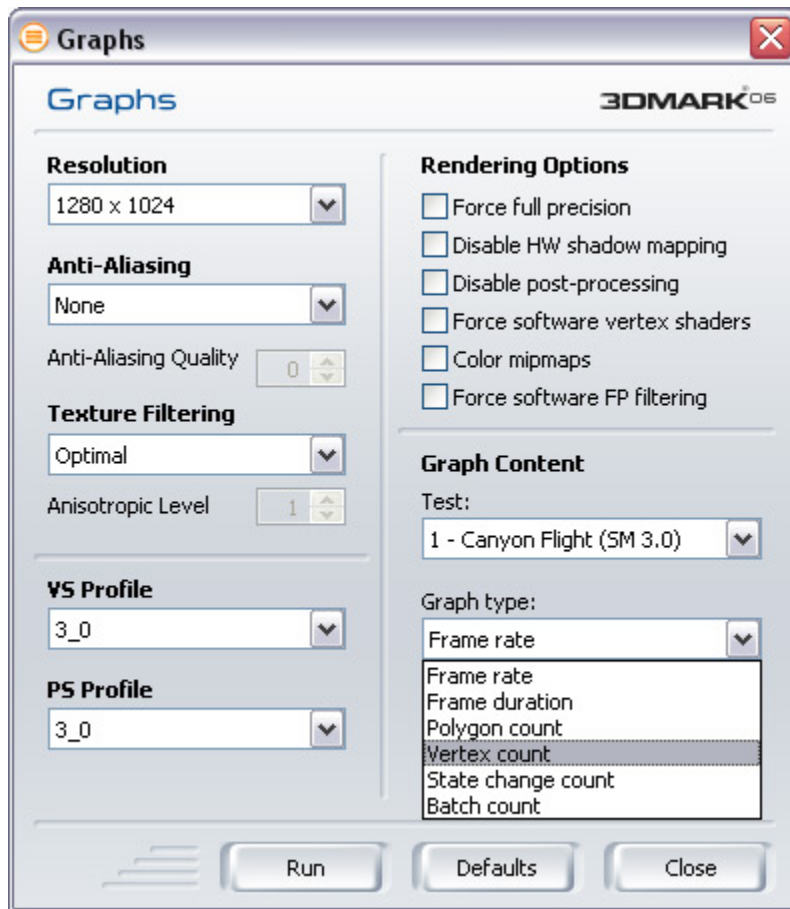
Figure 18: Graph Tool

The available settings are the same as those available in Benchmark Settings, but you can additionally select which test to run and what statistics to record. All four graphics tests and the CPU tests are available for collecting per frame runtime statistics. The available statistics to record for each frame in the graphics & CPU tests are:

- Frame rate - the resulting average frame rate if all frames would be drawn as fast as this frame.
- Frame duration - the time it took to draw this frame.
- Polygon count - how many polygons were drawn in this frame.
- Vertex count - how many vertices were drawn in this frame.
- State change count - how many state changes were made when drawing this frame.
- Batch count - how many drawing batches were executed when drawing this frame.

Score Calculation

There are 4 graphics tests (two SM2.0 graphics tests (SM2.0) and two HDR/SM3.0 graphics tests (SM3.0, FP16 Textures, FP16 Blending)) - each one is designed to represent a certain type of 3D game, and thereby offering a variety of 3D game graphics workloads. 3DMark06 records the total number of frames rendered. Using the length of time for the test, an average frame rate is calculated. A higher average framerate is better.

Figure 19: Results Dialogue

The 3DMark Score is calculated using the following formula:

SM2.0 Score = 120 x 0.5 x (SM2 GT1 fps + SM2 GT2 fps)
HDR/SM3.0 Score = 100 x 0.5 x (SM3 GT1 fps + SM3 GT2 fps)
CPU Score = 2500 x Sqrt (CPU1 fps x CPU2 fps)

We'll define for clarity:
GS for hardware capable of running all graphics tests = 0.5 x (SM2S + HDRSM3S)
GS for hardware capable of running only SM2.0 graphics tests = 0.75 x SM2S

3DMark Score = 2.5 x 1.0/ ((1.7/GS + 0.3/CPU Score)/2)
i.e. 2.5 x weighted harmonic mean of GS and CPU Score.

GT1 fps means the average frame rate measured in SM2.0 graphics test 1. CPU fps means the frame rate measured in the CPU tests.

Command Line Options

3DMark06 Professional Edition has a large number of command line options. These can be used for benchmarking automation and automatic 3DMark start-up after system boot. Command line options are options that are entered when 3DMark is started. The traditional way to enter command line options is to open a command prompt, and type manually the name of the executable that is to be run and the command line options for it. The examples of using the *-help* options shows two other ways to use command line options.

All available command line options are presented in a window, when you start 3DMark06 with the switch *-help*. This can be done in at least two ways:

1. Drag the 3DMark06.exe file or the 3DMark06 desktop shortcut to the Run dialog of your Start menu. Press the Start button on your desktop and then select Run to open the Run dialog. When the 3DMark06 file or shortcut is in the text field of the Run dialog, move the cursor to the end of the text, add a space, type -help and click OK.
2. Edit the 3DMark06 shortcut on your desktop by clicking it with the right mouse button. Select properties and edit the Target: text field. Move the cursor to the end of the text, add a space, type -help and click OK. Next time you click the 3DMark06 desktop shortcut the command line switches will be displayed. Remember to change the shortcut back to the way it was, so that you can run 3DMark06 again.

Use all command line switches the same way you used the *-help* switch. Many command line options can be used concurrently. The command line switches are the following:

All Editions:

-help or **-?** – show all available command line switches.
-unregister – to unregister your 3DMark06 Professional or Advanced Edition.
-nosysteminfo – disables the system info component. Note that results without system info can **not** be submitted to the Online ResultBrowser.

Professional Edition:

-gtall – runs all available graphics tests
-sm2all – runs both SM2.0 graphics tests
-gt1 – run SM2.0 graphics test 1 (SM2.0)
-gt2 – run SM2.0 graphics test 2 (SM2.0)
-sm3all – runs both HDR/SM3.0 graphics tests
-hdr1 – run HDR/SM3.0 graphics test 1 (HDR/SM3.0)
-hdr2 – run HDR/SM3.0 graphics test 2 (HDR/SM3.0)
-cpuall – runs all CPU tests
-cpu1 – run CPU test 1
-cpu2 – run CPU test 2
-featureall – runs all feature tests
-fillrate1 – run fill rate test 1 - single texturing
-fillrate2 – run fill rate test 2 - multi-texturing
-ps – run the pixel shader test.
-vssimple – run the vertex shader test - simple
-vscomplex – run the vertex shader test - complex
-shaderparticle – run the shader particle test (SM3.0)
-perlinnoise – run the perlin noise test (SM3.0)
-batchall – run all batch size tests
-batch8 – run the batch size test with 8 triangles / batch
-batch32 – run the batch size test with 32 triangles / batch

-batch128 – run the batch size test with 128 triangles / batch
-batch512 – run the batch size test with 512 triangles / batch
-batch2048 – run the batch size test with 2048 triangles / batch
-batch32768 – run the batch size test with 32768 triangles / batch

-res=<w>x<h> – set benchmark resolution <w>x<h>. Example: *-res=1280x1024* which is the default.
-filter=<x> – set texture filtering mode, where <x> can be 'optimal', 'bilinear', 'trilinear' or 'anisotropic'. Example: *-filter=optimal* which is the default.
-aniso=<x> – set maximum anisotropy to be used with anisotropic filtering.
-aa=<x> – set anti-aliasing mode, where <x> 'nonmaskable' for non-maskable anti-aliasing, or the sample count for multisample anti-aliasing. Example: *-aa=4* for four sample anti-aliasing.
-aaq=<x> – set anti-aliasing quality
-vsprofile=<x> – set vertex shader profile, possible values for <x> are 2_0, 2_a, 3_0
-psprofile=<x> – set pixel shader profile, possible values for <x> are 2_0, 2_a, 2_b, 3_0
-fullprec – forces full precision for shader calculations.
-nohwsm – disables hardware shadow mapping optimizations.
-nopp – disables full frame post-processing.
-vshader=<x> – enable or disable forced software vertex shaders, where <x> can be 'hw' or 'sw'. Example: *-vshader=sw* for software forced vertex shaders. The default value is 'hw'.
-colmipmap – colors mipmaps for filtering study
-fpfiltering – to force software floating point filtering on.
-repeat=<x> – set how many times each test should be repeated, where <x> is the amount of repeats. Example: *-repeat=3* The default is to run each test once.
-fixedfps=<x> – run in fixed frame mode, where <x> is the desired frame rate
-verbose – write a verbose file of benchmark transactions during loading and running of the tests.
-nosplash – start 3DMark06 without the splash screen.
-noscreens – disables the test loading screens.
-loopdemo – loops the demo infinitely

You can also put the name of a 3DMark06 result file after the switches. This will run the benchmark specified by the other options and save the results in the result file. Example: *3DMark06 myperf.3dr* . This will run the default benchmark and save the results in the file myperf.3dr.

One good use of command line switches allows you to start the 3DMark06 benchmark automatically after Windows has booted up. Make a 3DMark06 shortcut with the desired command line switches. Copy this shortcut to the Windows Start-up folder. Then every time the PC is booted-up, 3DMark06 is automatically run. The Basic and Advanced Editions of 3DMark06 can not use most command line options. The Professional Edition has all options enabled.

Batch Run. If you want to run a number of results on a system using different settings and test selections, this can easily be done using the batch run functionality. The batch run functionality works like standard Microsoft Windows batch files, which use the file ending '.bat'. Just write a text file with a number of 3DMark06 runs, using various settings as command line options, save that text file in the 3DMark06 installation directory, and give the file the ending '.bat'. The batch run can now be run by double clicking the .bat file. The obtained result files can be viewed with the Online ResultBrowser, with the 'Submit Saved Result' functionality of the right-click menu, or they can be viewed in Microsoft Excel, using the 'Open in Excel' functionality of the right click menu.

Example 1:
Save a text file with the following contents with the file name batchrun1.bat:

3DMark06 results.3dr

This batch run will run the default benchmark and save the results in the file 'results.3dr'

Example 2:
Save a text file with the following contents with the file name batchrun2.bat:

3DMark06 -res=640x480 -aa=4 -gt1 results1.3dr
3DMark06 -res=1024x768 -gt1 -gt2 -vsprofile=2_a -psprofile=2_a results2.3dr

This batch run will run the SM2.0 graphics test 1 in 640x480 resolutions and with 4 sample anti-aliasing and save the result in results1.3dr. Then it will run SM2.0 graphics tests 1 and 2 in 1024x768 resolution, using vertex and pixel shader profile 2_a, and save the results in the file 'results2.3dr'

The Basic and Advanced Editions of 3DMark06 can not do batch runs. The Professional license is required for batch run functionality. Please note that 3DMark06 Basic or Advanced Editions may not be used for any commercial purposes like system testing at PC assembly companies, OEM's or IHV's.

3DMark06 Demo

The 3DMark demo has always been an important part of the product. We have always used it as an opportunity to share our excitement about the latest 3D graphics technology. The goal of the demo has always been to simply have some fun and to showcase some very compelling scenes. Getting feedback on the demo has always been exciting; it is not uncommon to see the 3DMark demo running in loop mode at tradeshow exhibits. The 3DMark06 demo contains content from all four graphics tests, featuring significantly longer versions of them. All tests include the use of rich sounds.

Figure 20: The Demo

The Basic Edition only runs the first part of the demo, while both Advanced and Professional Editions run all parts. Please note that only the Professional Edition is authorized to be shown and used in public places such as shops, fairs etc. The Professional edition has the *loop demo* option available.

3DMark06 Mini-Game

Since 3DMark2001 (which included a mini-game) users have been asking for a new mini-game. 3DMark06 brings back this unique extra feature, and this time more like a real game than the one in 3DMark2001. The game is based on the CPU tests, and uses Futuremark's own mini-game engine. The scene is not rendered in HDR since the CPU tests were designed with SM2.0 in mind. However, in order to give gamers something extra, the game uses our new dynamic shadow implementation.

Figure 21: The Mini-Game

Using any of the SM2.0 profiles, the game will use CSM, but only with 4 rotated pointsamples / PCF / FETCH4 filtering to smooth the shadow edges. The SM3.0 profile will also use CSM, but with the same 16 sample kernel that is randomly rotated for each pixel as used in the HDR/SM3.0 graphics tests. Choosing any of the SM2.0 profiles may increase the FPS on any available SM3.0 hardware due to the heavy 16 sample filtering for the shadows. The controls of the game are:

W: Forward Thrust
S: Reverse Thrust
A: Strafe Left
D: Strafe Right
Mouse Move: Aim Turret
Left Mouse Button: Fire
Right Mouse Button: Toggle Turret Elevation
**Space**: Pause / Resume
ESC: Abort game

When the turret is in the higher position, the recoil is much higher and aiming is more difficult, but the view is better. The game time is 5 minutes, during which the player should try to shoot down as many speeders as possible. Each destroyed speeder earns the player 30 points. When the time is up, a score dialogue appears and the game score can be submitted to the ORB for the Game Hall of Fame, which is posted on Futuremark's website.

Online ResultBrowser (ORB)

The utility of a benchmark result can be rather limited in isolation. Without a mechanism to compare your system to others, it is difficult to say if your system is a high-end state-of-art PC, a mid-end system, or even a low-end beige box. To allow benchmark users to come together to compare and analyze results, we provide a web service called the Online ResultBrowser or ORB.

The ORB has become Futuremark's most popular online service. It provides the users a web application to manage and compare benchmark results. The ORB database contains over 30 million results. After running the benchmark, the user can choose to *submit* the results to the ORB. We enforce data privacy, so no one except the user will be able to see the individual results. Futuremark also verifies the uploaded results for accuracy.
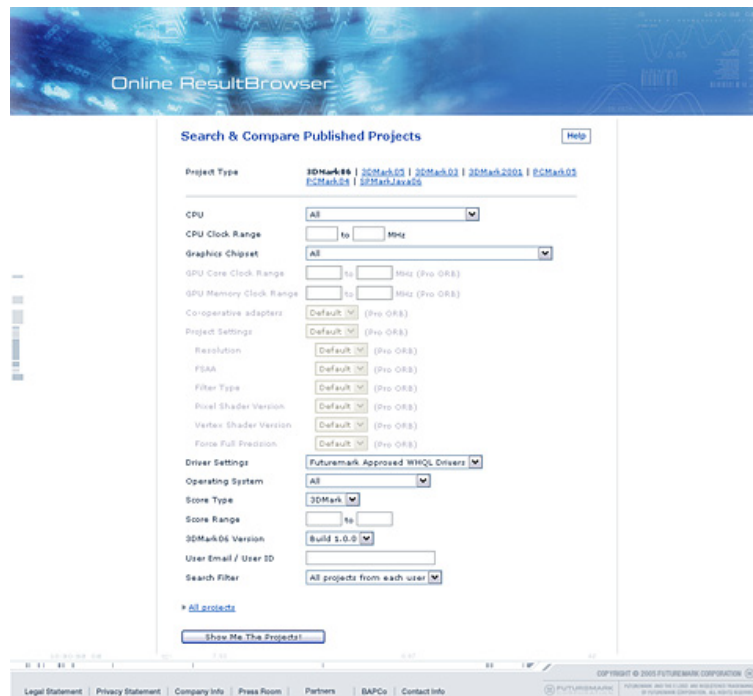
Figure 22: 3DMark06 Online ResultBrowser (ORB)

The ORB helps the user by adding meaning to the benchmark data; it allows the user to compare the results with those from other PCs. The user is able evaluate the PC's relative performance - determine the PC's weaknesses and its strengths. Of course, users can decide to share their results by explicitly *publishing* them. This allows them to show their results to the rest of the world. For many users, their 3DMark performance results are a point of pride. For some, their position in our top performing PC rankings is the source of a bit of fame. Many proud PC owners include ORB URL links to their published benchmark result in their email signatures.

The ORB is the user's virtual benchmark laboratory. Users can experiment with different system configurations such as increasing the RAM or upgrading the CPU. They can get the new PC performance by searching through results published by others. Users can also maintain their own performance track record by submitting multiple projects to the ORB. They can assemble custom multi-compare sets to compare their PC to multiple other configurations. As the ORB is an online service, Futuremark continues to improve it by adding new functionality. The benchmark data collected by Futuremark is used for generating statistics and recommendations for the user community. This means that every result submitted helps all users to select reliable hardware upgrades. The ORB aids in making buying decisions; before spending money, the user can validate expectations of different hardware options.

System Information

Associated with each benchmark result is the complete profile of the target PC; we call this the *System Information*. The system information not only provides detailed configuration information (CPU speed, RAM, graphics chipset, etc.) to go along with the performance data, but also gives the system state (open applications, AGP mode, free system memory, etc.).
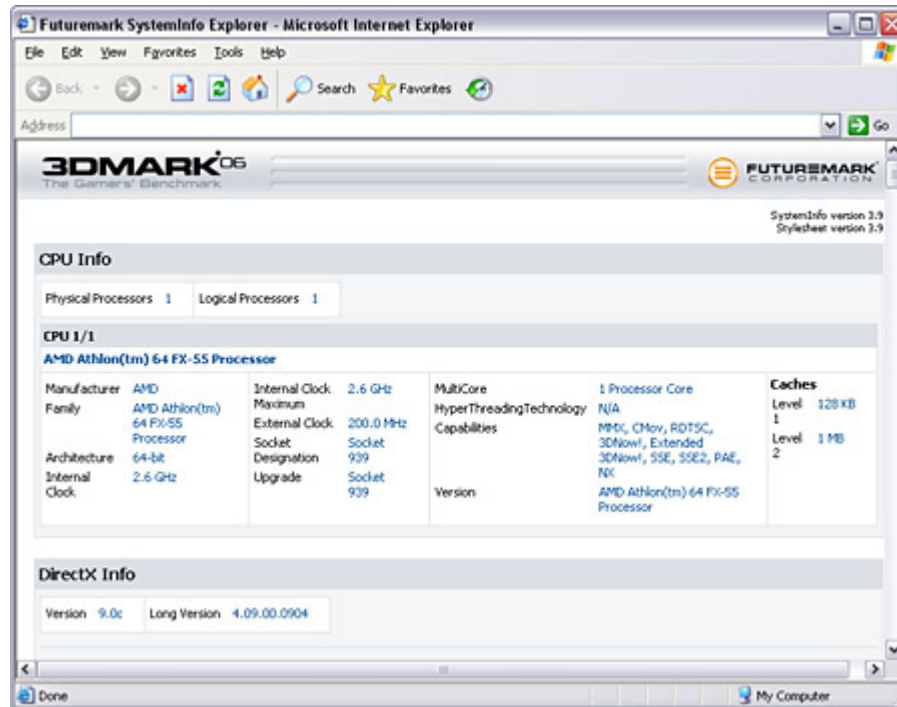

Figure 23: System Info

The complete system information consists of over 300 fields. These include:

- **CPU** Information – clock speeds, internal and external caches
- **DirectX** Information – display drivers, direct draw/show attributes, texture formats, 3D capabilities
- **Memory** Information – memory arrays and modules
- **Motherboard** Information – bios, card slots, system devices
- **Monitor** Information – monitor attributes
- **Power Supply** Information – batteries
- **Operating System** Information – version, installed service packs
- **Open Processes** Information – applications, processes
- **Logical Drives** Information – local and network logical drives
- **Hard Disk** Information – disk drive attributes

Note that no private information is ever collected.

The ORB uses the system information to enable the search and compare functionality. System information also allows Futuremark to verify the accuracy of published benchmark results. These increasing numbers of system information records are used to provide information back to the users in the form of lists of most popular and powerful hardware components shown on our web site.

Conclusion

Over the last six years, 3DMark has become a trusted standard in 3D graphics benchmarking. Futuremark's latest version, 3DMark06, continues the tradition of providing an easy-to-use tool for benchmarking next generation hardware. This time, powered by the latest DirectX 9 features and graphics hardware, we showcase 3D graphics likely to be seen over the next year and a half. More importantly we provide a neutral and transparent benchmarking tool to allow the user to evaluate the capabilities of the latest hardware. As graphics hardware becomes more powerful, especially with new sophisticated shader support, stunning 3D graphics will become accessible to more applications. We believe 3DMark06 will serve as a highly dependable tool for the benchmarking professional in this new environment.

Any Questions Unanswered?

If you have any questions about the 3DMark06, or need more information about anything of the above, please don't hesitate to contact us at:

Futuremark Corporation, http://www.futuremark.com

Mr. Nicklas Renqvist
Benchmark Development Architect
Email: nick@futuremark.com

Futuremark Corporation
Kappelitie 6 D
FIN-02200 Espoo
FINLAND
Tel: +358-20 759 8250
Fax: +358-20 759 8251