

## GAPI Benchmark Suite.

Technical note. (Draft. May 17, 2000)

### ***Preliminary limitations and formulation of the problem.***

Why to call this Tests Suite (Tests) GAPI Benchmark Suite? It is proposed to test not only the one specific graphics chips with the help of low level commands directed to it, but a set of architecturally similar currently developed and future chips with a help of an abstract 3D graphics program interface (GAPI). These Tests uses the most prevalent (by the release of the first version of the Tests) GAPIs: OpenGL (version 1.2) by SGI<sup>1</sup> (former Silicon Graphics, Inc.), and Direct3D (version 8.0) by Microsoft<sup>2</sup>. So, the key feature is that both components – the specific implementation of the chip (or the specific implementation of the chip based 3D accelerator, to be more accurate) and the specific implementation of the GAPI (including libraries and directed to them device driver) – are being tested.

This Tests Suite needs metrics for evaluation criteria of 3D accelerators, based on the graphics chips being tested and working under the GAPI, with the following parameters:

- speed (performance)
- functionality
- visualization quality
- price

The final formula may look like this:

$$\text{Mark} = w_1 * \text{Speed} + w_2 * \text{Features} + w_3 * \text{Quality} + w_4 * \text{Cost}, \quad (1)$$

where  $0 < w[i] < 1$  – weight coefficients, and  $\sum w[i] = 1$ .

Who and why need these Tests? The proper choice of the weight coefficients  $w[i]$  completely depends on the answer on this question.

The following user categories may probably be interested in the Tests results:

- gamers, which use low-end 3D accelerators intended for games;
- professional artists, which use high-end 3D accelerators;
- game developers and the developers of different software which use 3D graphics.

The key feature for **gamers** is performance, then comes the price, functionality and the quality of rendered picture (image). Functionality and image quality are more important for **professional artists**, then comes performance and the price. And, finally, **developers** has the following priorities: performance, functionality, quality and the price.

---

<sup>1</sup> OpenGL is registered trademark of SGI.

<sup>2</sup> Misrosoft, Windows, Win32, Windows NT, DirectX, Direct3D, DirectDraw are either registered trademarks or trademarks of Microsoft Corporation.

The suggested system of priorities is not a final one and may be re-considered in future.

Each user from the previously mentioned categories can answer its question after the testing:

**the gamers** – can choose better 3D accelerator for their favorite game; **the artists** – can choose the best 3D accelerator for their work; **the developers** – can choose the proper way to write the program to use all the features of a 3D accelerator completely. (The developer is the only one who doesn't need to choose 3D accelerator, (s)he needs tests to get complete and reliable information on a 3D accelerator. (S)he only chooses the way to write an application with help of tests results).

**The main factor is the trust in the Tests.** There are a lot of tools, which can evaluate the performance and other characteristics of the system, available. But we cannot be absolutely sure that there will be no conflicts of interests behind these tests, and that all the tests are written and updated correctly. The only way to avoid such a situation is to write open-source tests for everybody and provide the opportunity to discuss them. Besides, testing principles should be clearly described in documentation, and documentation should be also available for every one. "Benchmarks should come from an unbiased, noncommercial source. Additionally, the source code for benchmarks should be readily available to everyone to insure that indeed no bias or mistake has been introduced" [2].

### ***Tests contents and presentation form***

As it has been mentioned above, the weakest spot of different tests is their closed source. The source should be completely opened to make the test a really popular tool, which gives the trustworthy results. Principles of evaluations should be available together with the sources and well documented.

All currently available tests of video subsystem can be divided into two main classes. They are application level tests and synthetic tests. Application level tests are often used in 3D accelerators reviews. They are usually based on popular 3D games like Quake, Unreal, Expendable. This choice is based on obvious results clearness. Really, if user needs 3D accelerator to play Quake, it makes a sense to run Quake on all available models and choose one with the highest FPS. But this approach has a few disadvantages.

Firstly, the results of these tests are almost useless for the other two above-mentioned categories of users: 3D artists and developers. Artists get no information about 3D accelerator functionality and developers can only suspect its hidden performance and other capabilities, which cannot be shown by the game-test. So, most of the information is taken on trust.

Secondly, gamers cannot get an exact answer either. Quake source is unknown. Besides, drivers may also be Quake-optimized. The mini-driver by 3dfx and TurboGL by Matrox are well known examples. nVidia staff (M.Gold) admitted that OpenGL ICD was optimized for better Quake performance, since it is one of the most popular games on the market. They also admitted the mistakes in Quake in spite of nVidia and id Software close contacts. It means that test results may significantly change when updates, patches and new drivers versions appear or the whole system is modified (RAM, HDD, etc.).

[4] suggests to create a viewset by an independent developer (ISV) with a help of sponsors. It might be a good solution but requires extra financial expenses and organization efforts. Anyway it is difficult to write an application, which uses all the features of a video subsystem. There is no such an application, so we need to use a whole application suite and it will produce a problem with weight coefficients.

Then, synthetic tests are required, since application level tests can't describe the real situation consistently.

Synthetic tests cope with a task of functionality evaluation of a 3D accelerator and are able to represent visualization quality. But performance evaluation is very difficult. Synthetic tests are always criticized for the incorrect model of subsystem loading – so, the tests results are irrelevant and cannot be reliable. Generally this statement is right. But when we deal with games and a classic implementation of Graphics rendering pipeline we have an ability to model an adequate behavior of a video subsystem. Here we have to take into account the real quantitative characterization of the graphics part of contemporary 3D games and the real development experience. Video subsystem performance is ever increasing and developers need a tool to distribute effectively additional free resources. Image quality tests become more and more important. Modern graphics chipsets are able to provide high fps rating and now it's time for picture quality [9].

What scenes should be built? [3] claims “seeing is believing” and suggests to build “the most amazing 3D scenes ever done in real-time on PC”. We believe, developers can build even a better scene if they have results and information from the Tests. There is no need to build large and complex scenes to get reliable information about graphics chip features and capabilities. A proper loading of each stage of graphics pipeline with primitives and per-fragment operations is enough. The correctness of this supposition will be proved on practice.

**The general idea** of suggested approach is the following: nobody will organize the fairytale show on the smooth surface of water with diving and other tricks. The mission of this Test Suite is to explore the river bottom, to mark the banks of the river and the fairway. The Tests only mark zone of action and anyone can use the test info in his/her own way.

To calculate each parameter of evaluation criteria of 3D accelerators we suggest the following suite of tests:

- stress/performance tests;
- conformance tests;
- visualization quality tests.

### **Performance tests.**

The techniques used in the performance tests are based on the SGI [1] guidelines. The traditional performance optimization and tuning is based on search of the code bottlenecks and their elimination. (You know, the 10% of the program code spends 90% of its run time). The technique we offer loads a video chip up to the limits uncovering the weak spots (bottlenecks) in the hardware and software (drivers) implementation. It is known that graphics rendering pipeline consists of three conceptual processing stages: the CPU subsystem, the geometry subsystem, and the raster subsystem. The Test System is used to check scrupulously and effectively each stage of graphic rendering. The metrics is created in the following way: the loading CPU is measured, the number of polygons in the scene and number of pixels (or/and overdraw ratio) is counted up at the fixed fps (frames per second) rate.

We propose to use two fixed fps rates, namely 30fps and 60fps. The modern systems can steam up even to 85-100fps but that frame rate makes problems with the synchronization of the display. Turning off the VSYNC signal basically distorts a picture and makes tests results incorrect and ambiguous. The lower limit of 30fps is the minimum, below which the animation in 3D applications loses smoothness and looks unnaturally. The high bound of 60fps is that threshold beyond which, according to medical data, the human eye ceases to distinguish separate frames and perceives all as a continuous visual dataflow. “The frequency

at which that occurs is called critical flicker frequency (CFF) or flicker fusion frequency (FFF). The CFF for human vision can vary with several environmental and psychological factors, but in general it varies between 50 and 70 fps” [11].

The guidelines [1] suggest performing tests in the single-buffer mode only, for the accuracy of computations was not affected by time of buffers swapping. On the other hand, most of 3D applications use the above-mentioned technique of double-buffering for smooth animation. Taking in account these facts we suppose to measure and log buffer switch time.

In order to reduce the influence of driver implementation errors to the final results as much as possible, the Test System should contain the code of initialization for all default states and values according to GAPI specifications.

**CPU stage:** It is rather difficult to specify the load on the processor in the multitask OS, especially in case of multiprocessor environment. It is important to take into account the fact that some systems employ the CPU for calculations for geometry stage of graphics pipeline using SIMD (SSE, 3D Now!) instruction. We offer following solution – simply to count idle cycles of CPU. The lower the CPU is loaded the more effective the graphic subsystem functions are. To make the estimation more adequate, one should avoid overloading CPU and reduce count of background and other parallel tasks. The minimum load to the CPU will point on well-optimized implementation of the 3D accelerator driver.

**Geometry stage:** RGB mode; simple and connected primitives (tristrips, trifans, quadstrips) 8, 12 to 16 primitives in a sequence; various way of primitives processing (direct, arrays, indexed arrays, display lists); control size of the data sent per vertex; various amount and types of light sources – up to 8 (OpenGL 1.2) local, infinite light, fog; control material type; normalization.

**Raster stage:** frame resolution control, color buffer depth control, smooth shading (flat shading in linked primitives works differently in OpenGL and D3D); Z-buffer control depth control; alpha-channel control; texturing execution control (size of textures and used color depth); multitexturing implementation (up to 3 textures, e.g. EMBM); animated textures (fire, water, fog).

The final spread of tasks will be discussed and then will be registered in the detailed specification document.

It is supposed to arrange all test benches to groups according to the load value. Numbers assigned to the test groups will ascend in accordance with test difficulties. For each stage of graphic pipeline the separate group will be assigned.

The metrics to rate the performance is represented by a functional of sort:

$$F(\text{polys/sec}, \text{Koverdraw}, \text{Ntextures}, \text{Nlights}), \text{ or } F(\text{Ngr\_cpu}, \text{Ngr\_geom}, \text{Ngr\_raster}), \quad (2)$$

where the exact appearance of the formula will be refined during the real tests.

## The conformance tests

There is a standard conformance test set for OpenGL GAPI. Probably, it would be useful to add some tests to ensure correct function implementation from the OpenGL 1.2 core and to remove some irrelevant tests. The final test set should meet the [8] requirements as closely as possible. It is necessary to mention that conformity tests should not initialize all default states and values.

As to check of conformity to the Direct3D GAPI, the set of tests has to be made according to a standard manual included in the MS DirectX SDK. All the other draft or beta or “internal” specifications should be ignored completely. Updates of tests should be performed following the corrections of the DirectX SDK papers (if any will be published) but is not more often than the release of scheduled DirectX versions (see below).

### **Visualization Quality Tests.**

It is supposed that each of tested 3D of accelerators will proceed N of the standard predefined scenes in 1024x768 display resolution with maximum for given 3D Accelerator color depth, Z-buffer depth, texture size and all the other features provided for video quality enhance. Then the screen capture should be performed and these taken resulting bitmaps should be analyzed using predefined procedural steps. Another approach is to compare SW and HW rendering results.

### **The Representation Form.**

The Test System is represented as a set of sources written in the C++ language. The project is originally oriented to the Win32 platform and is used within MSVC++6.0 environment. The choice of the representation form of the project is stimulated by popularity and availability of the platform and the toolkit. It would be useful to develop the system meta-API, which will serve as a mediator between tests and OpenGL 1.2 and Direct3D 8.0 GAPI. It is possible, however, that the entirety, expandability and convenience of operation of this intermediate API will prove to be ineffective. Therefore, as the alternative, we suggest that a separate set of tests for each of base GAPI should be developed.

All source files should have unified specifications and must be documented and well commented. The file headers should contain the references to the web sites where necessary specifications, description of technique, standard of coding and other working documentation are placed. The conditions of usage of the source texts and their modification should be formulated in the License Agreement. All corrections should be logged completely in the file, using the following fields: name/initials/nickname, date (yyyymmdd), reason. (See Coding Standard for details).

### ***The new versions release.***

The new versions should be issued regularly, but no more frequently than one release per six months. We propose, that a version name should include the name of the suit itself, the year ‘name’ and a letter denoting the period of the year – [yyyy] [A|B], for example, <Suite Name> 2001B (second half-year). Every new version release should be bundled to the issue of a new version of MS DirectX GAPI and video chips evolution.

### ***Known issues.***

All the issues and bottlenecks, as well as the knowledge base, should be systematized and published. The solution of an issue should be placed in “ Promo Proposals ... ”.

The questions of the day are:

1. Weight coefficients choice  $w[i]$  from (1)
2. Models selection (static, animated, simple, complex)
3. Procedural scripts for viewset description (workloads - [6]).
4. Metrics of visualization quality rating. (It is possible to estimate only in matching.)
5. Variants of test benches. Definition and specification (2).
6. reserved...

**Sources:**

- [1] R. Kempf, and J. Hartman. *OpenGL on Silicon Graphics Systems*. SGI, 1996  
<http://techpubs.sgi.com/library/manuals/2000/007-2392-001/pdf/007-2392-001.pdf>
- [2] Van Smith. *A Proposal for Open-Source Benchmarks*.  
<http://www6.tomshardware.com/column/00q2/000414/index.html>
- [3] 3DMark2000. The gamer's global performance diagnostic.  
<http://www.MadOnion.com/3Dmark>
- [4] The OpenGL Performance Characterization Project Committee Rules. Version 1.1, June 10, 1999. <http://www.spec.org/gpc/opc.static/rulesv11.htm>
- [5] SPECglperf™ Graphics Benchmark. <http://www.spec.org/gpc/opc.static/glperf.htm>
- [6] Game Benchmark Development. A White Paper. Intel Corp. Rev 0.0, October 27, 1998.
- [7] OpenGL ARB, J. Neider, T. Davis, D. Shreiner, and M. Woo. *OpenGL Programming Guide, Third Edition*. Addison Wesley Longman, Inc., 1999. ISBN 0-201-60458-2
- [8] The OpenGL Graphics System: A Specification (Version 1.2.1).  
<ftp://sgigate.sgi.com/pub/opengl/doc/opengl1.2/opengl1.2.1.pdf>
- [9] John Carmack's .plan, 04/29/2000. <http://finger.planetquake.com/>
- [10] Foley, J. D., A. vanDam, S. K. Feiner, and J. F. Hughes, *Computer Graphics: Principles and Practice*. 1990.
- [11] Steven Schwartz, *Visual Perception: A Clinical Orientation*, 1<sup>st</sup> ed. Norwalk, Conn.: Appleton & Lange, 1994.