Selecting Graphics Hardware for Technical Applications in a Windows Environment

By Ken Severson and Howard Stroyan

Hewlett-Packard Technical Solutions Lab

Contact information:
Hewlett-Packard Company
MS 73
3400 E. Harmony Rd,
Fort Collins, Co 80528

Ken: 970-898-3529
severson@fc.hp.com
Howard: 970-898-3317
hstroyan@fc.hp.com
Fax: 970-898-7470

Background

HP is in a unique position in the technical workstation market place: in addition to designing and implementing high-performance 3D graphics subsystems for over 20 years, HP also selects and qualifies graphics for its Intel-based workstation and PC product lines. This knowledge yields insights into graphics and system design and their affect on application performance.

By looking at three representative applications with several graphics cards, it is possible to establish a correlation to what graphics attributes make the biggest difference in predicting application performance.

The intent of this paper is not to be an exhaustive survey of graphics options or applications available in the market, but rather to choose 3 cards with distinctly different designs in order to find the most salient characteristics for predicting application vs. benchmark performance.

Benchmarks

Graphics Benchmarks

There are many graphics benchmarks, and often graphics hardware vendors will try to push a new benchmark to show off the particular advantages of their implementation. Because all the significant 3D technical applications use OpenGL as the graphics Application Programming Interface (API), this analysis only looks at OpenGL based benchmarks.

The graphics benchmarks for this comparison are the SPECviewperf™ (abbreviated as viewperf) and the SPECglperf™ benchmarks sponsored by the OpenGL Performance Characteriazation (OPC) committee of SPEC. A complete set of information can be found at their website, www.specbench.org.

Viewperf is a set of tests in a synthetic graphics test fixture, using application-oriented data sets and tests. The data set and tests for each of the 5 applications represented is known as a "Viewset." The tests are measured in frames per second (higher is better), and the score for each Viewset is a geometric mean, using weights based on the application vendor's view of the importance.

There are five viewsets: AWadvs is based on Alias/Wavefront's Advanced Visualizer application. DX is based on IBM's Data Explorer, DRV is based on Intergraph's DesignReview, Light is based on Discreet Logic's Lightscape Visualization System. and ProCDRS is based on Parametric Technology Corporation's CDRS industrial design software.

SPECglperf (abbreviated as GLperf) is a script-oriented benchmark that measures the performance of OpenGL 2D and 3D graphics operations. These operations are low-level primitives (points, lines, triangles, pixels, etc.) rather than entire models such as those used in the Viewperf benchmark.

Graphics Application Workloads:

In order to reliably measure true application performance, a set of actual application benchmarks were chosen. The intent was not to be exhaustive and show correlation for all the various technical applications, but rather to establish a methodology for characterizing a few representative applications. We used three application workloads: PTC Pro/Engineer™ as representative of Mechanical Computer Aided Design (MCAD), Newtek's Lightwave3D™ to represent Digital Content Creation (DCC), and Unigraphics™ (also MCAD) because (at least up through Version 15) it uses a different OpenGL programming model.

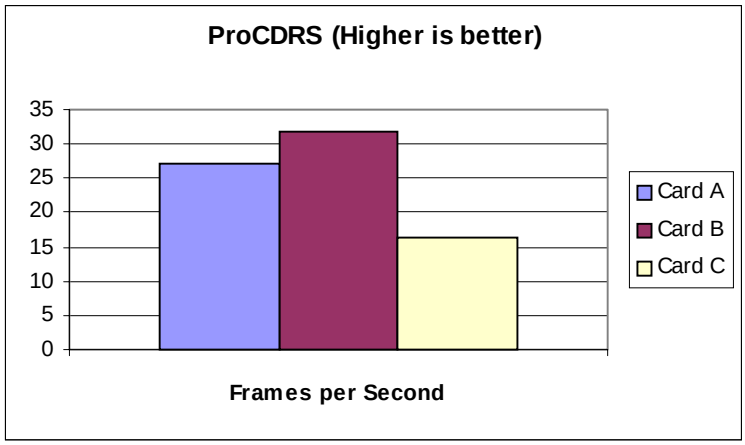
All the application benchmarks involve running the actual application, unlike the synthetic Viewperf benchmarks.

Methodology

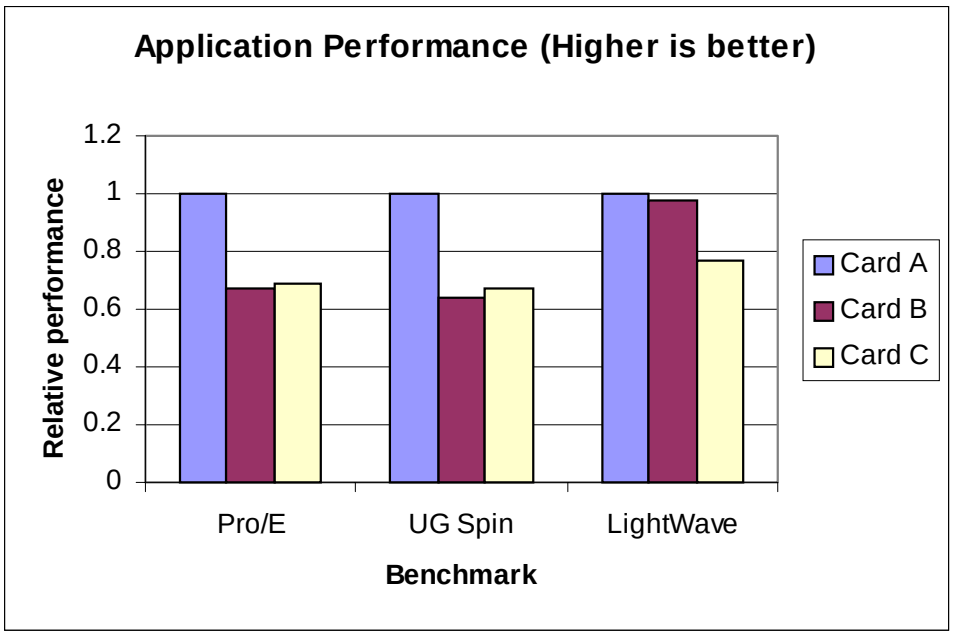
In order to correlate the application data to graphics benchmark data, all the data for the same application was taken on the same computer, changing only the graphics card and the corresponding graphics driver. The reference computer was clocked at 750MHz, with an Intel Pentium IIIe with 100Mhz front side bus. It was configured with 512MB of 100MHz SDRAM and a 10K RPM SCSI disk. The operating system was Windows NT Version 4.0, with Service Pack 5. The application measurements were taken with the graphics driver configured per the recommended settings for the application being measured. The GLperf measurements were all taken with the driver configured for the Pro/E application.

We tried as much as possible to treat each graphics subsystem as a “black box,” using only the externally measurable characteristics in our attempts to correlate the data. In order to gain further data on all the cards, we varied the CPU frequency by down-clocking the CPU from 750MHz in 50MHz increments to as low as 500MHz, but changing no other system attributes.

Which card is the best?



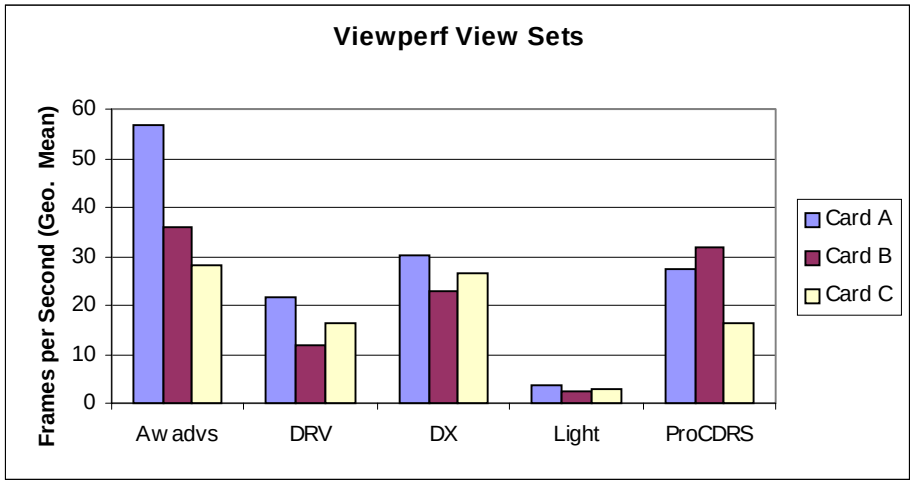
The chart shows the three graphics cards and their ProCDRS Viewperf scores. Because ProCDRS is the best-known and most-often touted benchmark, we look at it first. From the ProCDRS score, one would conclude that Card B is the best choice, assuming the price is the same.



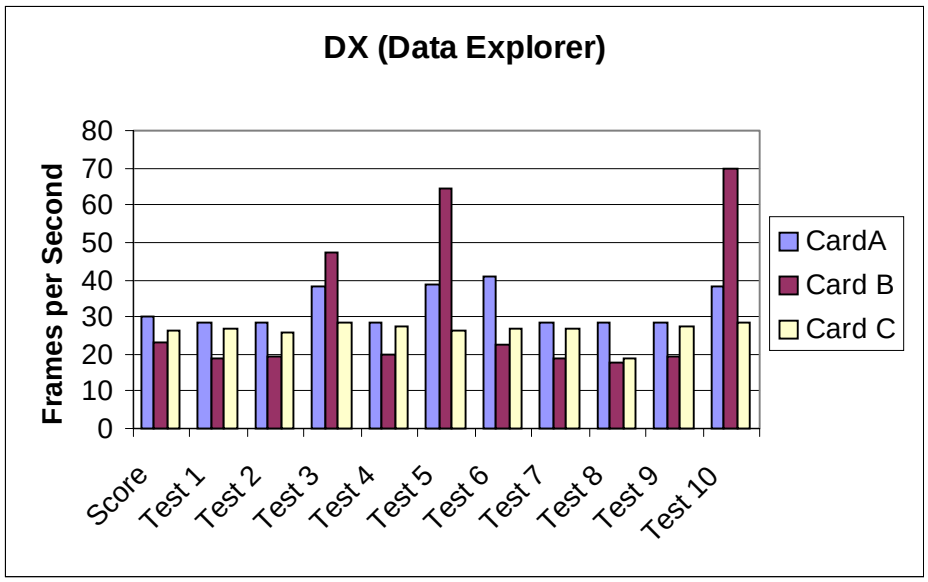
But now lets look at the application results. The above chart shows that Card A is the overall performance winner, and that Card C also beats Card B in Pro/E and Unigraphics performance, even though Card C is well behind Card B in ProCDRS performance. Why the discrepancy? What is wrong with ProCDRS as a predictor of graphic application

performance? By answering this question, new insights into the strengths and limitations of benchmarks in predicting application performance are possible.

The entire set of viewperf benchmarks yield a more complete picture:



Note that Card A is the winner in all but ProCDRS, and Card C is in second place in three out of the five benchmarks. This is more consistent with the application data we were seeing. Drilling a step deeper into the DX (Data Explorer) benchmark, interesting characteristics begin to emerge.



Card C is ahead of Card B in the overall benchmark score, but Card B is the winner in DX tests 3, 5, and 10. By looking at the www.specbench.org web site, the three DX tests in which Card B is the winner have an important attribute in common with ProCDRS: the three tests are display list mode, and all the ten tests of ProCDRS are display list mode. The other 7 tests in DX are immediate mode. This distinction is explained in the next section.

3D Graphics with OpenGL

Display List and Immediate Mode

There are two fundamental programming models used by application programmers with OpenGL: Display List and Immediate Mode. Many applications use a mixture, but most are predominately one or the other. Because the performance characteristics of Card B are quite different in immediate mode than display list, it is important to make the distinction between the two rendering modes.

In immediate mode, the application calls OpenGL to draw a primitive (e.g., line or polygon) or modify the hardware state (e.g., line color or a transformation matrix). The operation is performed, for all intents, "immediately," and the driver does not keep a copy of the primitives.

In display list mode, the application tells OpenGL to create a copy of the data that it is sending. The creation of the list is called "compiling" the display list, and drawing from the display list is known as "executing" the display list. Because the OpenGL driver can store the information in a format that is the most convenient for the hardware and the number of OpenGL procedure calls when drawing is reduced, the graphics throughput for display lists are almost always greater than or equal to Immediate mode rendering. Display lists are very effective when the same data is to be displayed multiple times. Good examples are an architectural walk-through of a building or a fly-through of a complex (but static) mechanical assembly.

Nevertheless, there are compelling reasons for some applications to use immediate mode. In general, immediate mode is an easier programming paradigm for an application programmer.

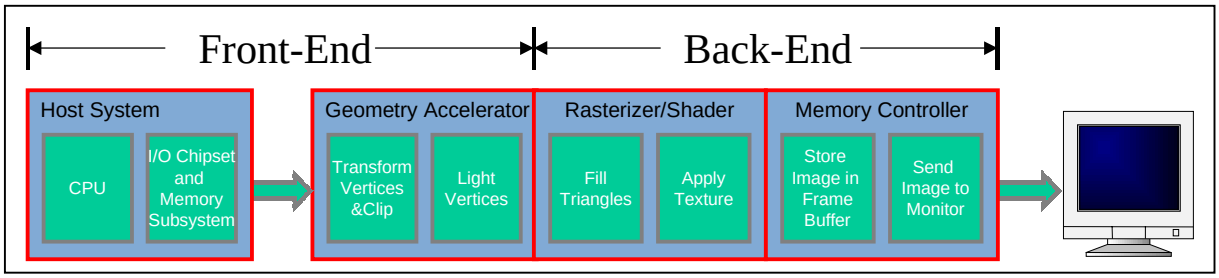
For a large model, the display list itself could use many megabytes and is a duplicate of the model already in the application controlled memory. So an immediate mode application may require less memory than display list, and this could be the difference between being able to work on the model at all in a system with a maximum memory configuration of 2 GB.

Immediate mode is useful if the data changes significantly from frame to frame – for example, when displaying time-varying scientific data, or a complex animation, or with varying levels of detail.

Examples of applications that run on Windows NT that are primarily Display List include Unigraphics™(at least up through Version 15), CoCreate SolidDesigner™, and SDRC's IDEAS™ suite of applications. Examples of immediate mode applications are PTC Pro/E, Kinetix 3D Studio Max™, Avid SoftImage™, and Alias/Wavefront Maya™. The same company may produce applications with different rendering characteristics: while PTC Pro/E is primarily immediate mode, PTC CDRS is a display list application.

The Graphics Pipeline and Potential Bottlenecks

The conceptual graphics pipeline is shown in the following figure:



In practice, the Geometry Accelerator (GA) portion is implemented either in graphics hardware (typically mid-range to high-end graphics HW) or in software in the host CPU (typically entry-3d to mid-range graphics HW). Intel's SSE (Streaming SIMD Extensions) introduced with the Pentium III are especially useful for the GA portion of the pipeline when it is implemented in the CPU.

To draw lines, the GA typically only transforms and clips. It can also do the calculations necessary for depth cueing and other effects. Lighting vertices of lines is possible, although less frequently used.

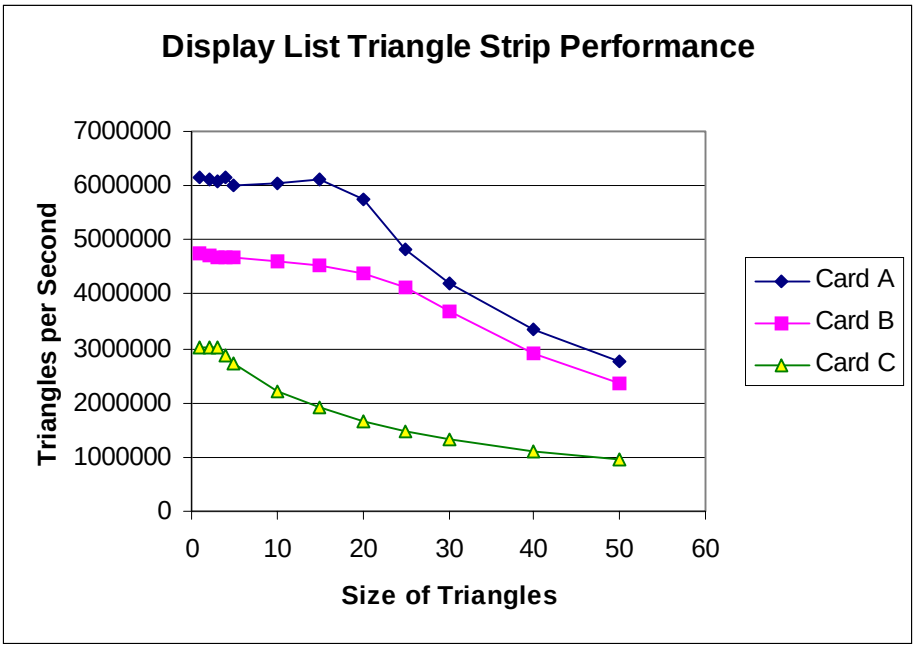
A graphically intense application will often be limited in performance by a "bottleneck" at some specific point in the graphics pipeline. What this means is that all other elements of the pipeline will be under-utilized while the bottlenecked element is saturated.

Bottlenecks are possible anywhere in the pipeline, including the host. For actual applications, the bottleneck can in fact be how fast the application feeds the graphics information to the graphics subsystem. When the bottleneck is in the host system, the host interface, or in the geometry accelerator, the limit is considered a "front-end" limit. A typical way to quantify a front-end limit is as a maximum number of primitives (vectors per second or triangles per second) that can be processed.

Size can matter

If the bottleneck is in the Rasterizer/Shader or Memory Controller portion of the pipeline, this is considered a “back-end” limit. Generally, small triangles and short lines expose the front-end limits, and large triangles and long lines expose back-end limits. The pixels between the vertices of the lines are drawn in the Rasterizer/Shader, and the interior of the transformed vertices of a triangle are filled. The amount of time that this takes is proportional to the projected size of the triangle or line, and the simplest measure of this is the area in pixels for a triangle, and the length in pixels of the line. With all of today's graphics implementations, the rasterization is accomplished in parallel with the transformation. A possible exception is during state changes (e.g., changing the primitive's color), where some graphics systems require the graphics pipeline to be "flushed," which will stall the pipeline until the rasterization is complete.

To see the effect of front-end versus back-end limits, the following measurements of lit, smooth-shaded, z-buffered triangle strips were taken with the GLperf tool:



From this chart, both Card A and Card B are front-end limited for triangles that are between 1 and approximately 18 pixels in size. For triangles greater than 18 pixels, the performance is back-end limited. For Card C, the back-end is the limit for triangles 5 pixels and larger. The chart also establishes the Card A front-end limit to be just over 6 Million triangles per second, and the back-end limit peaks at over 180 Million pixels per second. This type of data is shown in the charts in the next section.

Primitive-level performance

Primitives are the basic types of simple objects from which the complex 3D renderings are created. Two of the most important primitives are triangle strips (for creating shaded renderings) and line strips (for creating wireframe renderings). Examples of other primitives include points, line loops, line strips, and triangles.

Typical graphics vendors are not consistent with what they quote for peak triangle rates and vector rates. They vary parameters such as the size of the projected triangle, the number of triangles in a triangle strip, and whether the quoted numbers include back-faced culled primitives (which double the numbers quoted). Rather than use those "marketing impaired" numbers, we measured the actual performance for each card on the reference computer.

Table 1. Smooth-shaded Triangle Strip performance

	Immediate Mode Front-end (Tri/s)	Display List Front-end (Tri/s)	Back-end (fill) (pixels/s)
Card A	6.1M	6.2M	182M
Card B	3.0M	4.7M	176M
Card C	3.0M	3.0M	83M

Table 2. Vector (wireframe) performance

	Immediate Mode Front-end (Vectors/s)	Display List Front-end (Vectors/s)	Back-end (fill) (pixels/s)
Card A	9.5M	17.4M	58M
Card B	6.2M	18.8M	62M
Card C	2.9M	3.0M	37M

Table 3. Anti-aliased (smooth) Vector performance

	Immediate Mode Front-end (Vectors/s)	Display List Front-end (Vectors/s)	Back-end (fill) (pixels/s)
Card A	6.5M	6.8M	39M
Card B	6.2M	14.7M	46M
Card C	2.3M	2.3M	19M

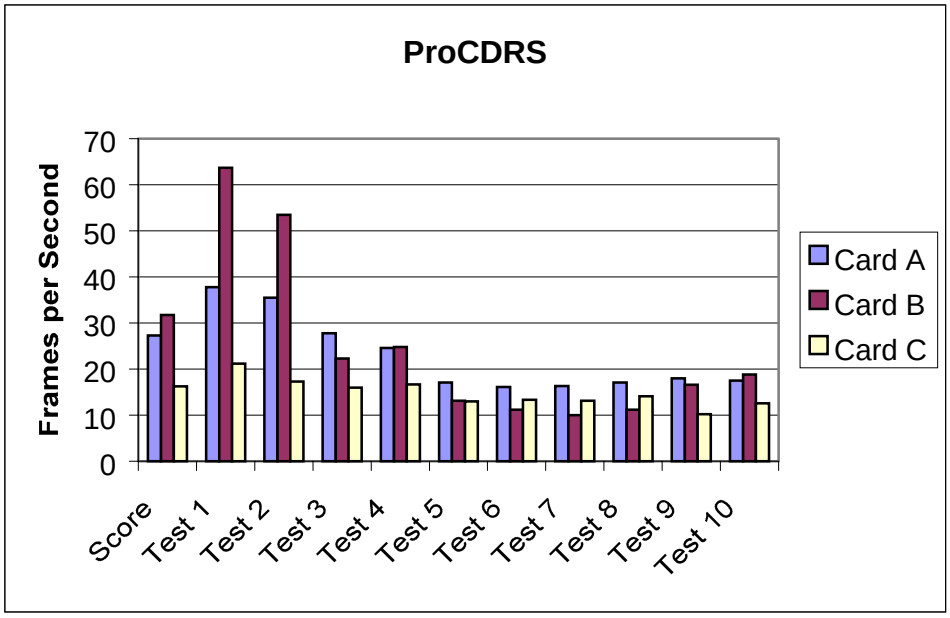
In all of the values of the three tables, a larger value is better. Now we can begin to see the relative strengths and weaknesses of the 3 cards. Card B is weaker than Card A in immediate-mode. Card A generally outperforms Card B in both vector and triangle front-end, with the one exception of display list anti-aliased vectors. Card A and B have similar back-end (fill) rates. Card C is generally in 3rd place, with the one exception of immediate mode triangle front-end (where, to higher precision, Card C transforms 3.02 M tri/s, and Card B transforms at 2.98 M tri/s).

If an application is characterized as typically having a bottleneck in one of these basic performance areas then measurements like this could be used to provide insight in to the expected application performance prior to more detailed analysis, but often applications will exhibit a more complex pattern of operation.

Inside the numbers: What does Viewperf measure?

The ProCDRS Viewset

ProCDRS is based on the workload of an industrial design application that was originally part of the Evans & Sutherland family of products, but later was acquired by PTC. The current version of the application is PTC CDRS 2000i.

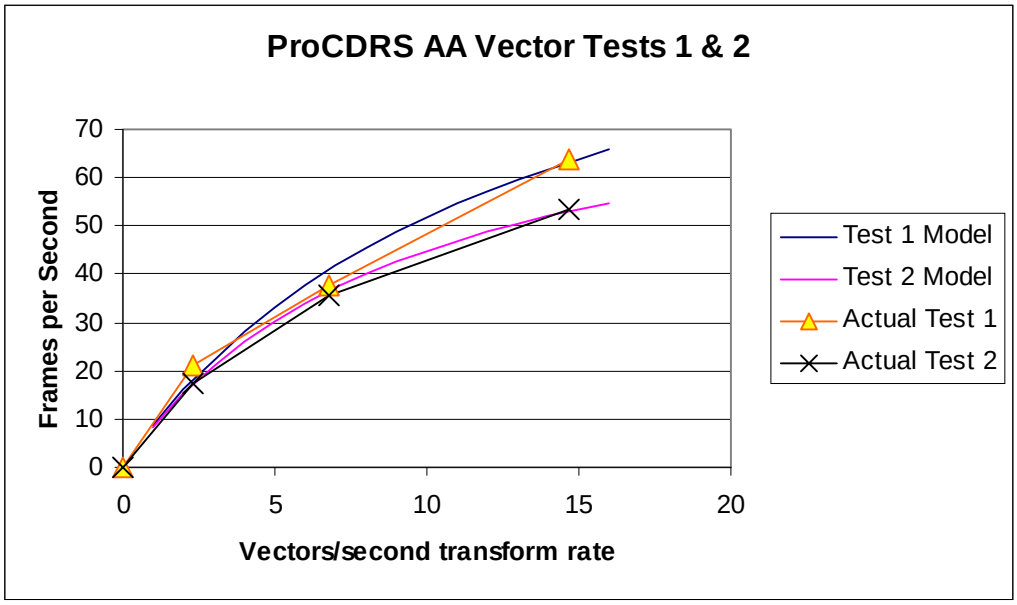


Because anti-aliased wireframe is important to the styling engineer, half (Tests 1 and 2) of the score in this benchmark is for anti-aliased (AA) vector rendering. Anti-aliased lines

mitigate the “stair-step” appearance of normal lines, and are sometimes referred to as "smooth" vectors.

Test 1 spins the model, with no zooming and very little clipping. Test 2 is a "fly through" that zooms in on the model, and moves around. Because of the zooming, Test2 should show a higher correlation to AA vector fill rate since the average vectors are longer, and could also show some affect on the efficiency of the clipping since many vectors are off screen in a given frame.

Using a tool to measure the distribution of line lengths, we found over 90% of the lines to be less than 10 pixels in Test 1, with the mode around 4 pixels. This would imply that the largest contribution to performance is the front-end of the graphics pipeline. Test2 has a similar distribution, but due to zooming, has a higher percentage of long vectors. For Test 2, the correlation should also be strongest for front-end performance. Using a simplistic model that treats the double-buffer clear and swap time, combined with the time waiting for long lines to fill as a constant, we get a reasonable correlation between the front-end measured performance and actual results in tests 1 and 2 as shown in the following chart. The data points are from the 3 cards, and knowing that 0 vectors per second would yield 0 frames per second.



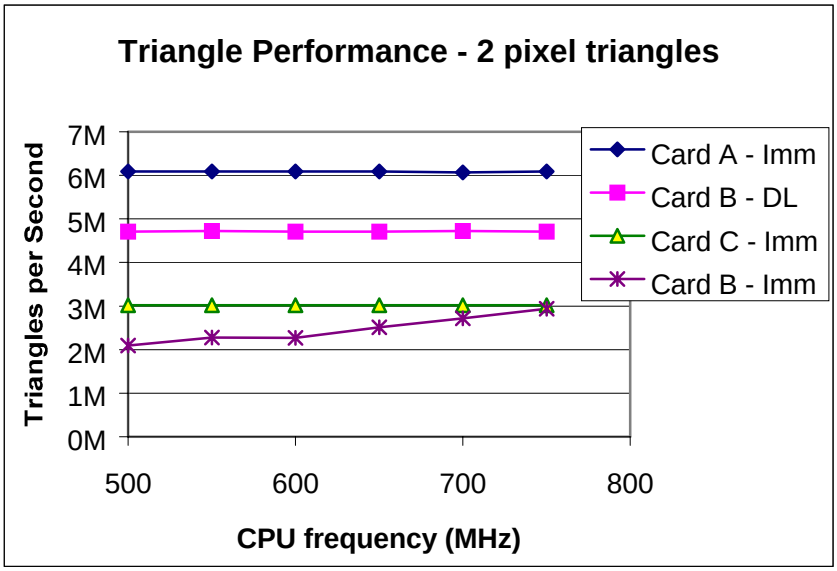
Especially for higher frame rates, per-frame costs like double-buffer switching become more significant, and can even dominate a benchmark. Since generally 20 to 30 frames per second is more than adequate, simple benchmarks with high frame rates can end up measuring double-buffer swap and buffer clear times, rather than the intended graphics capability.

This data suggests that the ProCDRS benchmark could be replaced by a simple test of display-list, anti-aliased vectors. The benchmark adds very little value above the simpler

AA vector primitive benchmark. There is nothing wrong with ProCDRS as a benchmark per se, and it is likely a good indicator of the graphics performance for someone who uses the PTC CDRS 2000I application. However, inferring any more than that from the benchmark can be very misleading. It is not a balanced measure of overall graphics performance.

The DX and DRV Viewsets

As shown earlier, the DX viewset uncovered interesting characteristics regarding display list and immediate mode. Taking a look at DX in more detail, it turns out that Test 3 is identical to Test 1, except that Test 1 is immediate mode and Test 3 is display list. However, the above front-end and back-end limits would still suggest that Card B should have an advantage over Card C in immediate mode, or at worst be equivalent. Further insight is gained by looking at the triangle performance, varying the CPU frequency of the system (without changing the processor).

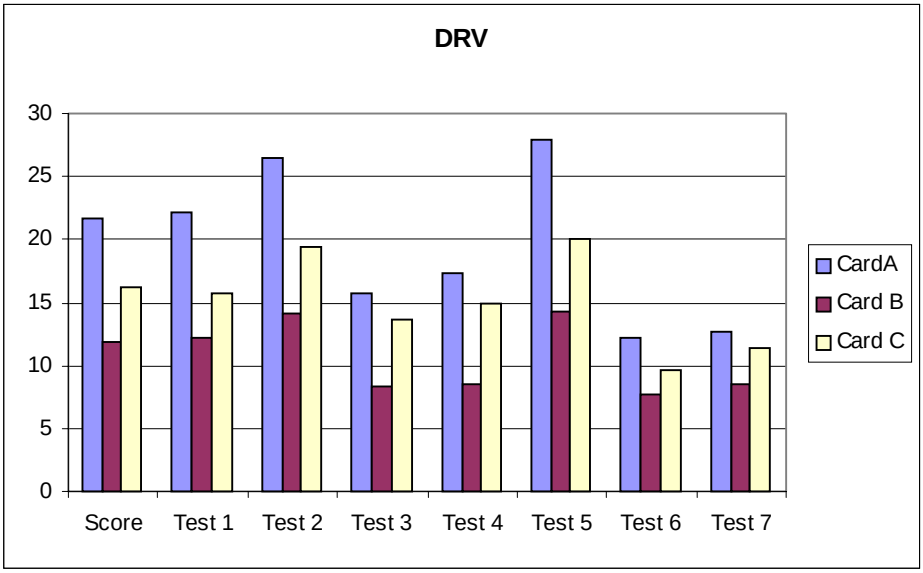


From this graph, it is apparent that the CPU is kept much busier with Card B in immediate mode. Because the GLperf tool has very low overhead, this CPU activity for Card B is very likely work that the driver needs to do to feed the card with immediate mode that is not needed in display list mode, and is not needed for Card A or Card C. The fact that Card C is faster than Card B for DX-Test 1 and is faster at the Pro/E benchmark (which is also immediate mode) indicates that DX comes closer than GLperf to an actual application in the amount of work that it needs the CPU to do.

Because they don't consume the CPU, Card A and Card C are able to do more in parallel with the application than Card B. This allows the application to begin preparing the next

packet of data sooner, and in immediate mode, the rate at which the application can feed the graphics can easily be the limit for high performance 3D graphics.

This result is also seen with DRV, which is a purely immediate mode benchmark. Especially in single processor mode, DRV can be a good indicator of immediate mode application performance. Note that in all of the DRV sub-tests, Card C outperforms Card B.



The AWAdvs and Light Viewsets

The AWadvs viewset is entirely immediate mode. The highest weighted test (>40%) uses trilinear mip-mapped textures and small triangle areas. In looking at the sub-tests, Card A is the winner across the board. Card B beats Card C in most of the tests, with the exception being the 2 flat shaded tests. The reason the result is different than for the other Viewperf immediate mode tests is that the average triangle size is around 10 pixels, which is where Card B fills faster than Card C.

The Light viewset has a low frame rate across the board. In all four subtests, Card A is the fastest, Card C is in the middle, and Card B is the slowest. This would be a typical result for an immediate mode benchmark with small triangles.

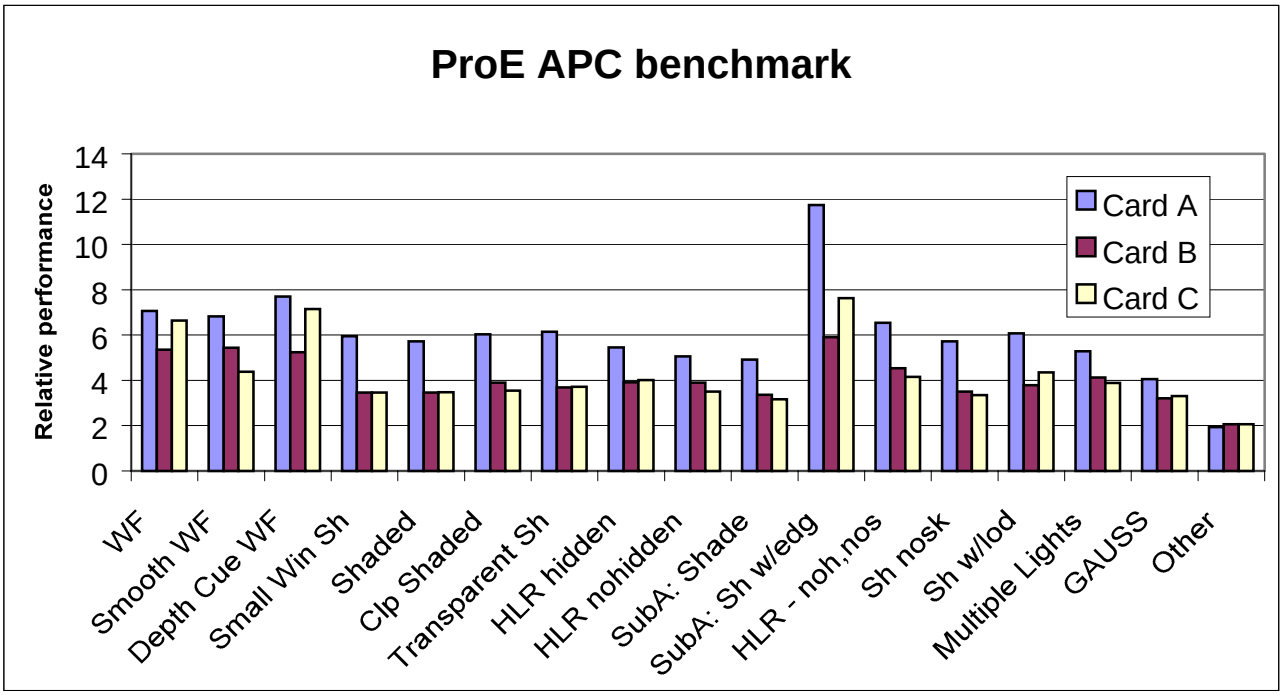
Application performance benchmarks:

Pro/E APC benchmark

The APC Pro/E application benchmark, like Viewperf, was sponsored by SPEC, as part of the Application Performance Characterization (APC) project. Unlike Viewperf and GLperf, APC benchmarks are composed of workloads that are run through the actual application being measured. Running this benchmark requires a license to the PTC Pro/E application. The model is a copy machine. Parts of the benchmark deal with a sub-assembly, the "fuser" that fuses the toner to the paper in the copier.

In this workload, Card A is by far the best performer. Because Pro/E is an immediate mode application, the strengths of Card A in immediate mode come to play. It is useful to go a step further and look into the components that make up the benchmark.

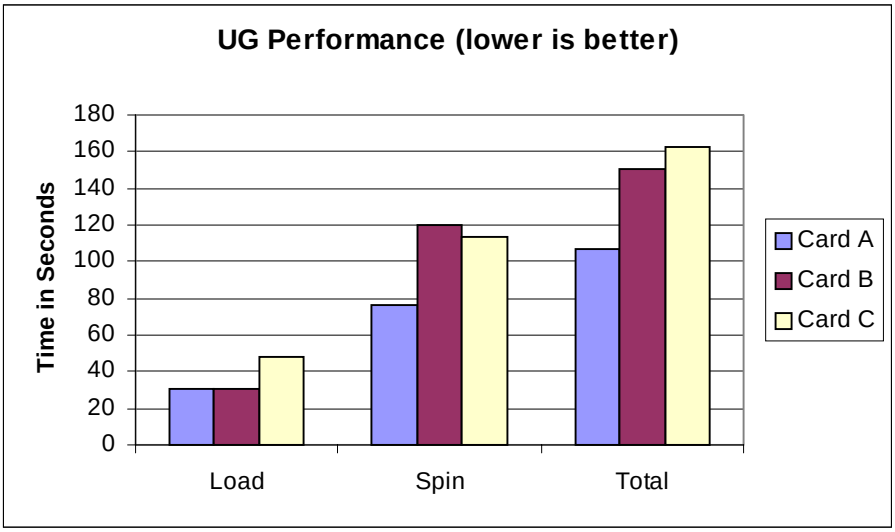
The next figure shows the individual tests that make up the benchmark. All the sub-tests are weighted equally. Card A is ahead in all of the sub-tests. Card C beats out Card B in the overall score by a small amount. This is due to the superior performance in the wireframe test, the depth-cued wireframe test, and the Sub-Assembly Shaded with Edges (SubA: Sh w/edg) tests. This makes up for the smooth wireframe (anti-aliased vector) test where Card B beats Card C significantly.



In summary, the characteristics of this application benchmark match up reasonably well with the trends shown in the DRV and the immediate mode parts of the DX benchmark. Where Card B does the best is in anti-aliased vectors, which is predicted by the immediate mode primitive results.

A Unigraphics benchmark

This workload is not a publicly available benchmark, but it was used because Unigraphics is a popular design application and, as measured with Version 14, it exhibits characteristics typical for display-list based applications. At the time of this writing, an



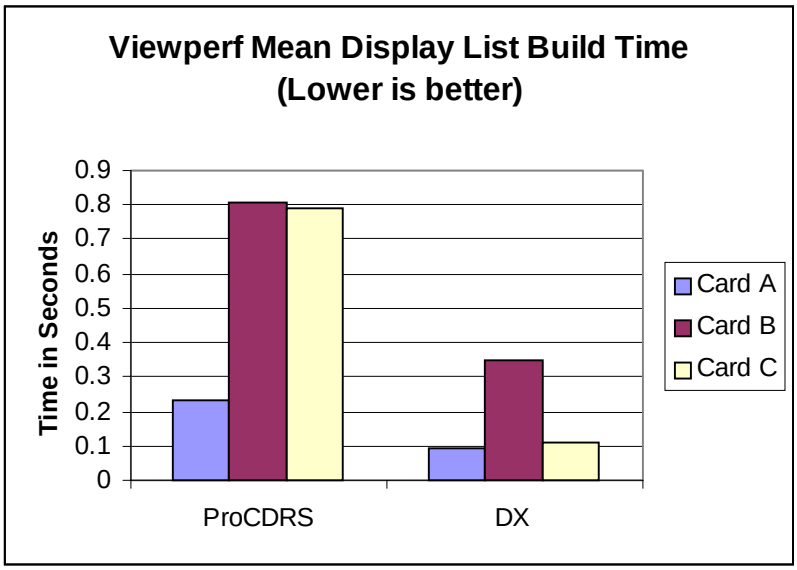
APC benchmark for Unigraphics is under development, but was not yet available. The benchmark used instead has 2 parts. The "spin" results were shown in the application summary above, and measured "load" time. The chart shows both, and the total time spent by both spin and load.

In the results, while Card C is better than Card B in spin performance, it is slower at load. Because Unigraphics is a display list application, it builds the display list as it is loading the part from the file system. This accounts for the large variation between Card C and Cards A and B.

Display list build time is an oft-neglected part of graphics performance. In evaluating benchmarks, it is important to establish an appropriate weight for display list build performance. If a display list is built once and rendered hundreds of times, it may be worth waiting longer for the display list to build. If the display list build is part of a "load" operation that takes minutes anyway, it might be the time that a typical user might check email or perform some other administrative function. In this scenario, they might prefer an increase in interactivity and thus tolerate a longer display list build time to optimize the display list for rendering.

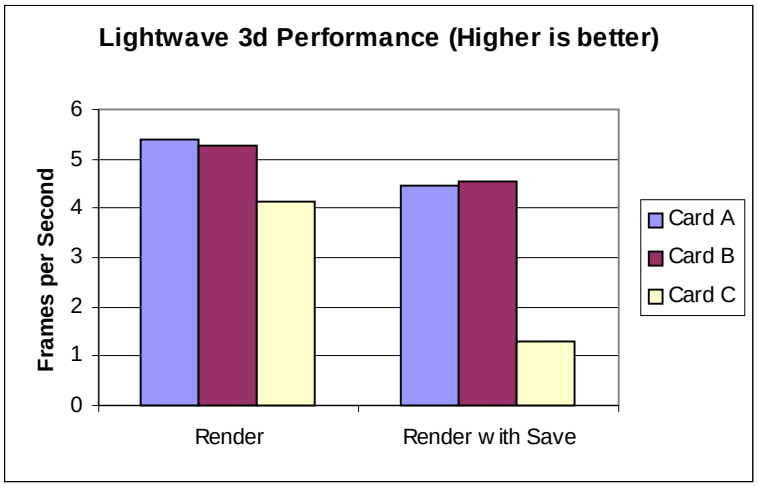
Although Card C is slower at display list build for this particular workload, this does not mean that it is always slower. A not-as-well publicized aspect of the Viewperf benchmarks are the Display List Build (DLB) times reported for both the ProCDRS and the display list portions of the DX benchmark.

The following chart shows the average display list build time per test for the ProCDRS and the three display-list tests in DX. The large disparity in times between cards B and C is most likely due to differences in the characteristics of the display lists required.



A Lightwave3D Benchmark

This benchmark was created using an example animation distributed with the NewTek Lighwave installation CD, measuring the OpenGL preview of the animation. The animation is of objects on a desk in an office.



In the first benchmark reported in the first application summary chart, the time to render the OpenGL scene of 70 frames is recorded. In a second benchmark not included in the summary, an option to save the resulting animation in a file is included. Since the system is the same for all the measurements, the performance delta beyond a constant factor for writing the file is due to the time that it takes to read the resulting images from the graphics card. This function in OpenGL is known as Read Pixels.

The data shows that Card A and Card B do well at the rendering portion, but Card B has the advantage when the file-save function is included. Card C is weakest at both the scene rendering and the file-save read-back.

To corroborate this result, it is possible to measure the pixel read capability with the GLperf tool. This results in the following table for pixel formats: RGB and RGBA (most DCC applications use RGBA).

Table 4. Pixel Read Performance – 512x512 image

	RGBA Read (pixels/s)	RGB Read (pixels/s)
Card A	26.2M	28.6M
Card B	30.6M	40.7M
Card C	2.1M	1.9M

Other Characteristics of Graphics Options

Features

All three cards are similar in feature set. They all support double-buffer true color (8-bits each for red, green, and blue), and support OpenGL 1.1. At least one claims to be OpenGL 1.2, but that is impossible to verify as OpenGL 1.2 is not yet supported by Windows NT.

Cards A and C both have stereo connectors for synchronizing stereo devices. Card C also has a digital flat panel connector.

Performance

When the feature set is the same, the first level distinguishing characteristic between graphics devices is performance in two areas: how much geometry acceleration and how much fill acceleration. Other contributions to performance are: design and efficiency of the host interface, and the driver (OpenGL software layer) overhead. This can also vary between efficiency in handling changes in attributes or other context information, as well as efficiency with context swaps.

How much memory do you need on board the graphics card for performance? Except for texturing, memory over and above 16 MB is more of a question of functionality than performance. To run at full performance, most applications require enough memory for double buffering in a true color buffer, (2 buffers each at least 24 bits deep), and at least 24-bits of z-buffer. At 1280x1024, this requirement is met with a minimum of 12MB. If you add in features like stencil planes, alpha planes (for destination transparency operations), and texture memory, it is easy to see why many of today's 3D graphics cards have 32MB or higher. Of the cards in this comparison, Card A has 34MB, but 18MB is dedicated to double buffering and z-buffer and 16MB dedicated to texture maps, and Card C has 32MB with a flexible allocation scheme: the higher the resolution, the less texture memory is available. The amount of texture memory only matters if you need more than what is in the card: swapping textures in and out is analogous to not having enough system memory and swapping to disk.

Another use of memory is as an on-board cache for display list. It is our experience that this gives mixed results. It can help benchmark performance for benchmarks like ProCDRS because of the relatively small data sets, but give almost no performance improvement for the larger data sets present when running real applications with real workloads. The time needed to load the on-board cache negates the benefits of a cache when all of the data cannot fit in the on-board display list memory.

Graphics Benchmarks and Application Performance

There is no substitute for benchmarking the actual application you plan to use. When benchmarking applications to evaluate systems, the following are recommended:

1) Use workloads that are representative of the actual data used.
2) Segment the benchmark to separate non-interactive operations (such as loading a model) from interactive operations (editing or zooming in on the model).
3) Weight the interactive portions of the benchmark the way an end-user of the application would.
4) Use the graphics driver settings recommended for the application.
5) Don't bias the evaluation by previous work patterns - some ways of interacting with the model might have been avoided in the past because the graphics subsystem wasn't fast enough, but might mean substantial productivity benefits with today's high performance 3D graphics.

Conclusion

Application benchmarks are the best way of seeing how the application will perform with various graphics cards. Nevertheless, graphics benchmarks such as Viewperf can be reasonable predictors of application performance as long as the characteristics of the viewset have enough similarities to the applications of interest. This is especially true if

one drills behind the single number "score," and looks at the individual tests that make up the score.

In particular, single processor DRV and the immediate mode portion of the DX benchmark are reasonable predictors of Pro/E performance. ProCDRS is not a good predictor for the three applications examined, which is a direct result of the high weight given to display-list anti-aliased vector performance.

Primitive-level benchmarks like GLperf are also not a good predictor of application performance, but instead are a very useful tool for understanding the fundamental characteristics of the device. It is a good tool for correlation, not prediction.

It is also important to look for areas that are critical to users of certain applications. For example, display list build time may or may not be important to users of Unigraphics, SDRC, and CoCreate SolidDesigner. If one drills deep enough into the Viewperf results, the display list build time is available, but it does not appear in the weighted score. Some users of Lightwave3d and SoftImage will find save to file important. If so, then Pixel read-back is an important area to benchmark, which is not measured anywhere in Viewperf.

In summary, although the desire will always be there to characterize graphics performance with a single number, finding a meaningful single number remains elusive. Because applications differ in their use of graphics, and different workloads can behave very differently even with the same application, doing a thorough evaluation of graphics and system offerings will continue to be a complex activity. By understanding more of what is behind the benchmark information that is publicly available, the process can be simplified. At the very least, that understanding will make it easier to choose a good short list of what to evaluate in more detail.