

Benchmarking Scene Quality

Quantitative Benchmarks Can Use a Perfect Rendering as a Standard

by Roy Latham, rlatham@cgsd.com

To date, benchmarks of graphics accelerators have dealt with polygon transformation rates and pixel fill rates. When reduced to these two numbers, products differing by one or two orders of magnitude in price often appear comparable in performance. There is another dimension to graphics accelerator performance beyond polygon transformation rates and pixel fill rates. This extra dimension explains why much more expensive products can compete successfully with less expensive products that offer similar benchmarks for polygons and pixel fill.

The added dimension is the feature set, the list of things that the graphics accelerator is capable of doing. Some users of graphics engines would, for the sake of simplicity, like to get performance down to a single number, just as some computer users are content using the processor clock speed or a single processor benchmark as a sole measure of performance. The "single measure" approach may or may not be appropriate for computer users, but it certainly is not appropriate for graphics engines. Graphics engines are inherently multi-featured. Reducing everything to a single number, or even a couple of numbers, makes as much sense as reducing the specifications for an automobile to horsepower and paint color — insufficient to distinguish a sports car from a city bus.

In the case of graphics accelerators, users may be helped by having a checklist of features. Going down the checklist will reveal why one machine costs a lot more than another, because it offers many more features. The list would presumably help users decide what features they need and which they do not need for their particular application, so they can find a product that will do the job without paying for unneeded features.

At last summer's IMAGE Conference, a special interest group for PC simulation was formed. One of the tasks undertaken was building a list of image generator features. Steve Hadfield (shadfiel@es.com) at E&S volunteered to coordinate the effort. This newsletter, *Real Time Graphics*, began assembling a list of features accessible through graphics API's. The list of about 120 features was circulated in the industry, and was ultimately published in the December issue as part of a survey of available run time software products. Not every image generator feature is accessible through the API, however. Features like antialiasing, pre-distortion for dome projection, and calligraphic light points may be part of the graphics engine design but not explicitly controlled through the API. I recently drafted a list of 82 hardware features and sent a copy to Steve Hadfield for circulation.

Numerical measures of scene quality would shape the industry.

Making a checklist of features reveals that many features have a qualitative or quantitative aspect, beyond merely being "present" or "not present." For example, there are wide variations in the quality of full screen antialiasing. In principle, most of the 200 features could be benchmarked in terms of quality or in terms of performance at a selected quality level. In practice, it is quite unlikely that anyone would invest in developing and applying a comprehensive suite of benchmarks for all 200 features.

As a practical matter, it may be viable to focus on benchmarking a subset of the features related to image quality.

This subset might include quality benchmarks for texture antialiasing, polygon antialiasing, smooth shading, transparency, and fading. This subset would be of broad interest to many people, and moreover, companies pushing image quality would have a vested interest in sponsoring quality benchmarks. Quality benchmarks would provide something concrete to which a vendor could point in order to justify a higher price for a product. From the users viewpoint, the quality benchmarks would help in product selection, and it would also encourage competition in the area of quality.

Currently, there are some benchmark databases designed to help users appraise scene quality. The idea is to allow the user to compare renderings on different hardware and make trade-offs subjectively. Subjective comparisons are valid for many purposes, and there is no reason why the practice should not continue. However, subjective comparisons have limitations as well. Images that look "about the same" on a CRT display in an office may show dramatic differences on a wide field-of-view collimated display with controlled viewing conditions. Moreover, the particular scenes viewed for comparison may not show a problem that is prominent in a different scene. Finally, a user might like to narrow a selection of products from, say, fifty candidates to five candidates based on numerical scores, rather than assembling and viewing fifty images.

There are challenges in making quantitative benchmarks of image quality, but for the special cases of scene quality mentioned, I believe the challenges can be overcome to provide practical results. The method proposed for quantitative benchmarking has the following four elements:

1. A series of benchmarks is designed to test individual features and features in combination. That way if a product has a poor implementation of, say, transparency, but an excellent implementation of texture, a prospective user can judge if the product profile suits a particular application.
2. The reference image for comparison is a “nearly perfect rendering of the scene.” In other words, the scene is assumed to exist as uniquely derived from the mathematical expression of the database and the viewing parameters.
3. The difference between the reference image and the test image is computed in e-units. E-units are distances in the Lab color space, which are derivable from RGB values, taking into account the color sensitivity of human perception that makes some differences more noticeable than others. E-units are used conventionally in industry for quality control work involving colors.
4. Measures of overall quality will include the r.m.s. difference between the test and reference image. Using the root-mean-square difference is not completely justified by theory, but it has at least a long history of use in image processing work, and it seems as plausible as anything.

The concept of a theoretically perfect (or in practice, near perfect) rendering is bound to be controversial. First, understand that a theoretically perfect rendering is a rendering of a digital database. We are not attempting to measure how well the database corresponds to anything in the real world. The real world does not offer polygons with infinitely sharp edges and perfectly uniform colors, but there is no problem representing such things in a mathematical database.

What is the theoretically perfect rendering? It is the rendering that a majority of a committee of reasonable graphics people agree is the perfect rendering. If no agreement is reached on the perfect rendering of a feature, then that feature cannot be benchmarked in the quality standard. The next question is: How will a committee agree on *anything* as being “perfect”?

Agreement may actually be easy to get on many features, in fact, surprisingly easy. Let us start with perspective transformation of the image from the 3-D database to screen coordinates. Is there any controversy on how to do a perspective transformation? The equations are given in every standard textbook, they are derived from the laws of geometry, and they are the same for everyone. Anyone who uses the right equation with enough precision in the calculations will get exactly the same answer (to some number of digits of precision). If you do not get the same answer, you have made an error. There is, I claim, no controversy on the theoretically perfect way of doing perspective transformation.

Similarly, there is agreement on the mathematics of flat shading and smooth shading with direct and ambient light sources. There is mathematical certainty on occlusion calculations as well. Every surface on the screen can be defined by polygonal boundaries with precisely defined vertices.

The theoretically determined visible polygons in screen coordinates must be sampled to make a raster image. There is exactly one theoretically correct way of doing that. The theoretical image is filtered by convolution with a sinc function and then sampled at the

Continued on page 8



This high-resolution image is a photograph, but conceptually it could have been a rendering of a polygon database. For our example, it is the original “perfect” image.



The original image (a) has been point sampled to produce a low-resolution rendering. The result corresponds to rendering the image without antialiasing.



Here, the original image (a) was first filtered with a nearly ideal 4 x 4 pixel low pass filter before point sampling to the same low resolution as (b). The image is comparable to a perfectly antialiased rendering at the low resolution.



The absolute values of the differences between (b) and (c) correspond to the errors in each pixel in the rendering in (b). The r.m.s. of these errors is a measure of the image quality, and note that it can be computed without displaying any of the images.

Benchmarking Scene Quality

Continued from page 7

pixel spatial frequencies. Theory says that will preserve the maximum amount of sharpness without aliasing.

Well, actually, this is one of those “near perfect” compromises. The filter kernel should theoretically extend to infinity, and there is a problem with the high frequencies generated by the image ending in rectangular borders. In practice, if the filter kernel is something like 8 x 8 pixels, the results will probably be indistinguishable to a human observer from anything better. Since the digital database is defined outside the screen clipping boundaries, the image can be computed on a larger screen area, filtered, and then clipped smaller. This will be close enough to perfect.

For our theoretically perfect image, texture begins as an array of sampled points. The projection of the sampled points onto the polygon in the scene has an exact formulation, and if the points appear closer than pixel spacing after projection, there is a unique way of filtering them to best prevent aliasing. If the projected texture samples are farther apart than the pixel's sample points, the texture values must be interpolated. There is one theoretically correct way of reconstructing the texture image from the sample points, and that is how the interpolation should be done.

Transparency in the scene is computed according to the formula of one minus alpha times the first color plus alpha times the second color, iterated for as many layers of transparency as there are in the stack.

Exponential haze fading may or may not match atmospheric haze in a real world scene. The exponential function is derived from an assumption of uniform particle scattering in the air. However, whether it matches the real world or not, the formula for exponential fading is precisely exponential, and so conformance can be tested.

Nothing in the quality benchmarking prohibits a vendor from adding features beyond those included in the benchmarks. One may provide a haze fading function that is in fact more realistic

than exponential fading; one just cannot claim it to be exponential fading. The user will then evaluate the exceptional implementation.

Implementation of the quality tests will raise additional issues. For example, suppose a test image is uniformly too dark. All the pixels have an error, so the quality will be judged poor, but the details of antialiasing may be nonetheless implemented correctly. Or perhaps the vendor has made a small error in setting up rotation, so that the image is computed a few pixels out of registration with the reference image. Perhaps the quality test should first attempt to achieve registration in scale, position, and brightness before proceeding to the pixel by pixel comparisons.

*Whether it matches
the real world or not,
the formula for
exponential
atmospheric haze is
precisely exponential,
so conformance can
be tested.*

There is also the possibility that quality tests on static images will not provide an effective measure of quality for sequences of images. The static image quality tests will probably carry over well to moving sequences, but there might be exceptions. For example, an implementation might use a dither matrix attached to screen coordinates, with the matrix changing on successive frames. For the present, such cases can be handled with the inevitable footnotes that accompany benchmark test results. Ultimately, successive images can be compared to derive more quality measures.

Features must be tested individually because we want products to get credit for whatever is implemented, without demanding a full feature set to achieve a

score. Also, it will be useful to run performance benchmarks with various features enabled to see, for example, if pixel fill rates suffer when antialiasing is enabled.

Features also need to be tested in combination, because additional problems often occur only in combination. Getting Z-buffering, antialiasing, and transparency to all work in combination is much more difficult than getting each of the features to work separately.

If the “perfect renderer” can be packaged to accept databases in a common format, then users could run their own test for combinations of features.

The quantitative tests using r.m.s. e-units does not take into account every aspect of human perception that relates to quality. In particular, defects that produce apparent contours are more distracting than continuous artifacts. Imagine Product A with an implementation of haze fading using too few bits, so that contoured steps appear in the image. Product B might have a poor fit to the exponential curve, but appear completely smooth. Product B could then yield a higher r.m.s. error, while nonetheless looking less objectionable to the average user.

One avenue is to improve the quality metric by including an edge-detection mechanism in the evaluation of the errors over the image. Another possibility is to provide a map of the errors produced by comparison of the reference and test images to help the user understand unusual results. While the r.m.s. e-unit metric should work well most of the time, there may be exceptions.

Overall, despite the challenges and limitations, quantitative image quality benchmarks would, together with a feature checklist, help many users find the best products for their applications. Traditional performance benchmarks have many limitations as well, but they still are nonetheless invaluable in helping users identify important differences among products. Quality benchmarks would also help drive product development towards improved image quality, a direction that would benefit many in the graphics community. ▲