An Extremely Inexpensive Multisampling Scheme

Tomas Akenine-Möller
Ericsson Mobile Platforms AB
Chalmers University of Technology
Technical Report No. 03-14

Note that this technical report will be extended later and submitted somewhere.

9 February, 2003, Version 0.5
24 February, 2003, Version 1.0
2 August, 2003, Version 1.1
15 August, 2003, Version 1.11

Abstract

We present a new sampling scheme for low-cost multisampling rasterization. The scheme builds upon sample sharing and the N-rooks sample scheme, and it achieves at most two shades in between fully outside and fully inside. Also, the slightly irregular pattern helps disguising aliasing effects. A simple visual evaluation shows that our new scheme gives similar quality to that of the Quincunx scheme. More importantly, our scheme uses only 1.25 samples on average per pixel, while Quincunx and FLIPQUAD uses 2 samples. As a consequence, our new scheme uses 62.5 percent of the memory and memory bandwidth as compared to Quincunx and FLIPQUAD.

1 Introduction

For mobile platforms, high image quality is of extreme importance because the display is small, and is often kept close to the eyes [1]. This implies that many of the components in a graphics engine must deliver results of high quality, and at the same time, the cost of implementation should be kept as low as possible. One important avenue for increasing the image quality is screen-based antialiasing schemes. Such schemes, decrease the, often disturbing, effect of the "jaggies," i.e., that pixels may appear to blocky. This effect is even more disturbing (and easily detected by the human visual system) during animation.

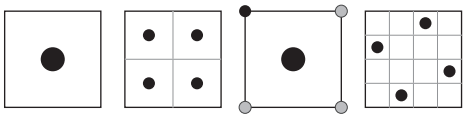

Figure 1: Simple, low-cost sampling patterns (left to right): one sample per pixel (no antialiasing), 2×2 super sampling (brute force), Quincunx sampling pattern (gray samples are borrowed from neighboring pixels), and rotated grid supersampling (RGSS).

This is sometimes referred to as the "crawlies." There are basically two types of schemes for screen-based antialiasing; *supersampling* and *multisampling*.

Supersampling is the bruteforce method. Such techniques sample an image at higher resolution, say, $2w \times 2h$, where $w \times h$ is the resolution of the final image. In a filtering step, the average of 2×2 pixels is computed and stored in the final image. Multisampling schemes share computations among samples, and is therefore smarter and can be cheaper, and definitely faster. For example, since the texture is supposed to be well-filtered, one can sometimes just use one texture sample per pixel, even though there are several sample points inside the pixel.

2 Previous Work

Some low-cost multisampling schemes are shown in Figure 1. A well-known multisampling scheme is called *Quincunx*, and was developed at NVIDIA [5]. The pattern is that of the "5" on a die. This clever scheme is shown in Figure 1, and because of the location of the samples, only two sample points are needed per pixel. The rest of them are obtained from its neighbors. A subsequent step then filters the five samples with the weights: 0.125 per corner sample and 0.5 per middle sample. It should be observed that the middle sample of Quincunx can be removed, and the cost per pixel is only one sample then. However, this does not give higher quality—rather, a lowpass effect is achieved, which is non-desirable. Quincunx can in the best case achieve four extra shades, but this seldom occurs. On average, only two shades are obtained in between fully inside and fully outside.

To the right in Figure 1, the rotated grid supersampling (RGSS) scheme is shown. This is one particular instantiation of Shirley's N-rooks scheme [7], which sees to it that that there is one and only one sample per row and column.

Akenine-Möller and Ström present a new scheme called FLIPQUAD [1]. This can be seen in Figure 2. This costs two samples per pixel and in the majority of cases it gives better quality than Quincunx. FLIPQUAD combines the good features of Quincunx (sampling sharing) and RGSS (one sample per row and column).

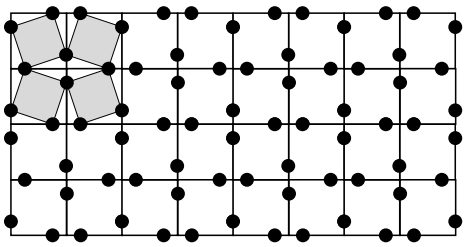

Figure 2: The FLIPQUAD multisampling pattern. Note that the pattern changes for every other pixel.

3 What Characterizes a Multisampling Scheme?

At first, I believed that for a sampling scheme to be called "multisampling", the scheme had to generate at least 2 samples per pixel. Sampling sharing can be used in order to increase the number of samples used to produce the final color, such as in, Quincunx and FLIPQUAD.

How few generated samples can be used per pixel in a multisampling scheme? We already know that two samples can generate a multisampling effect. Can a sampling scheme that only uses one sample per pixel be a sampling scheme? No. Either it would merely create a lowpass effect of the rendered image, or simply an offsetting effect on the samples. What about if a scheme uses in between one and two samples per pixel? We will show that such a scheme is possible to create, and that it indeed gives a multisampling effect.

4 FLIPTRI: New Multisampling Scheme

Our new scheme is based on a new sampling pattern, shown in Figure 3. The advantages with this pattern are several:

- The three samples per pixels are located on unique x- and y-coordinates in order to get a similar effect as in the N-rooks sampling patterns.

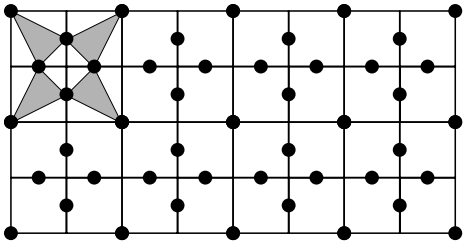

Figure 3: FLIPTRI: our new sampling pattern for multisampling. Note that the pattern changes for every other pixel.

- Except for pixels fully inside, or fully outside, two shades are obtained, which is what Quincunx achieves on average.
- The irregularity of the scheme breaks up symmetry somewhat, which increases the apparent quality (as do FLIPQUAD).
- The traversal cost is a bit smaller than FLIPQUAD.
- Can easily be implemented with Pineda's edge functions [6].
- Smaller filtering cost than Quincunx and FLIPQUAD since FLIPTRI only uses 3 samples to produce the final color instead of 4.
- Most importantly, it uses only 1.25 samples on average per pixel, which reduces memory costs and bandwidth costs with 37.5 percent as compared to FLIPQUAD and Quincunx.

While only using 1.25 samples per pixel is an advantage in terms of bandwidth usage, it obviously also affects the quality, as will be shown in the next section.

5 Discussion

Evaluation and especially comparison of multisampling schemes is very difficult, and there does not seem to exist a straightforward procedure for this. One reason is that the still is a debate on how large a kernel is needed to produce the final color of a pixel. In our case this is not really relevant as our goal is to reduce bandwidth requirements as much as possible. However, for low-cost sampling schemes, evaluation may be harder because discrepancy [2, 7] does not seem to work well for such schemes.

Keller and Heidrich [3] use Fourier analysis to evaluate a scheme that uses a locally (over a set of pixels) randomized scheme that is repeated over the array of pixels. Compared to using a randomized pattern inside a single pixel, and using this pattern over all pixels, Fourier analysis can easily be used to determine that the former scheme is better. However, for more regular schemes such as Quincunx, FLIPQUAD, and FLIPTRI, Fourier analysis is not very useful. This stems from that periodicity in the time (spatial) domain means that we have the same sort of periodicity in the frequency domain. Furthermore, a rotation in the spatial domain gives rotation in frequency domain. Therefore, we conclude that Fourier analysis is not the tool for low-cost schemes.

One might argue that computing the PNSR between a low-cost scheme and a high-quality rendered image would be a better way of evaluating a scheme. However, this is not entirely reasonable either, since the human visual system seem to be more sensitive to bad sampling on near-horizontal and near-vertical edges. In our previous study [1], we empirically found that near-45° edges are next-to-most annoying. Such information is hard to integrate into a PNSR-based approach.

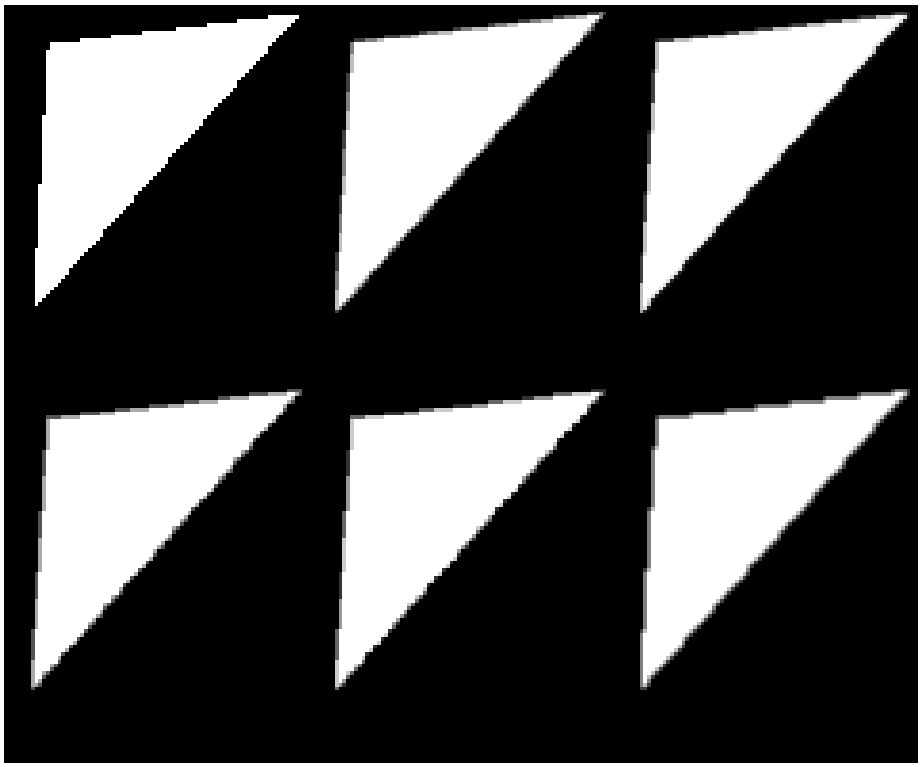

Figure 4: Left to right, top to bottom: One sample per pixel, FLIPQUAD, Quincunx, FLIPTRI (weights=0.33), FLIPTRI (weights=0.2 and 0.4), and one sample per pixel with the final color computed as the average of the 2×2 pixels.

6 A Preliminary Visual Evaluation

Due to the lack of a good evaluation procedure, we simply present some preliminary images here. The key idea here is just to show that FLIPTRI in fact can give something that is better than just using one sample per pixel. A white triangle rendered on black background is shown in Figure 4.

As can be seen, the FLIPTRI scheme gives a multisampling effect, and one can easily see that in the lower right corner, a low-pass effect is achieved. This is clearly not what we are after here. Using the same weights (0.33) for all three samples within a pixel seems to work best. For near-horizontal and near-vertical edges, the FLIPTRI scheme seem to work at least as good as Quincunx. However, the edge that is closer to 45° appears to contain more noise (instead of a regular staircase effect as for Quincunx).

7 Conclusion and Future Work

In this paper, we have simply presented a scheme that is very inexpensive, and we have shown that it gives a multisampling effect. Many questions remain open. For example, which weights should be used for the samples? It seemed that using the same weight (0.33) for each sample worked best. It is likely that Molnar's criteria [4] could be used for this. Another question that is left open-ended is the traversal order. We also need to determine a sample location for texturing. One solution is to take a texture sample at each sample inside the pixel. However, this will increase the texture bandwidth with 33%, which is not desirable, especially for mobile platforms. Another solution would be to take the texture sample in the center of the pixel. Assume that the sample locations for a certain pixel are $(0, 0)$, $(0.5, 1.0)$, and $(1.0, 0.5)$. Then it may be reasonable to scale down the samples that are used by two pixels by a factor of 0.5. This would give: $((0, 0) + 0.5(0.5, 1.0) + 0.5(1.0, 0.5))/3 = (0.375, 0.375)$. However, this position would be different for the neighboring pixels, which does not make sense, since the texture shift will be different for different pixels. Yet another idea is to do a texture lookup in each pixel corner where the also is a sample, and another texture lookup in the pixel corner which is surrounded by four samples. Perhaps the most important question is on how to evaluate low-cost multisampling schemes.

References

[1] Akenine-Möller, Tomas, and Jacob Ström, "Graphics for the Masses: A Hardware Architecture for Mobile Phones", submitted to *SIGGRAPH 2003*, January 2003.

[2] Dobkin, David P., David Eppstein, and Donald P. Mitchell, "Computing the Discrepancy with Applications to Supersampling Patterns," *ACM Transactions on Graphics*, vol.15, no. 4, pp. 354–376, October 1996.

[3] Keller, Alexander, and Wolfgang Heidrich, "Interleaved Sampling," *Rendering Techniques 2001*, pp. 269-276, 2001..

[4] Molnar, Steven, *Efficient Supersampling Antialiasing for High-Performance Architectures*, Technical Report, TR91-023, April 1991.

[5] *HRAA: High-Resolution Antialiasing Through Multisampling*, Technical brief, NVIDIA Corp., 2001.

[6] Pineda, Juan, "A Parallel Algorithm for Polygon Rasterization," *Computer Graphics (SIGGRAPH '88 Proceedings)*, vol. 22, no. 4, pp. 17–20, August 1988.

[7] Shirley, Peter, *Physically Based Lighting Calculations for Computer Graphics*, PhD thesis, University of Illinois at Urbana Champaign, December 1990.