A Family of Inexpensive Sampling Schemes

Jon Hasselgren[‡] Tomas Akenine-Möller[‡] Samuli Laine[*†]

[‡]Lund University, [*]Helsinki University of Technology/TML, [†]Hybrid Graphics, Ltd.

Abstract

To improve image quality in computer graphics, antialiasing techniques such as supersampling and multisampling are used. We explore a family of inexpensive sampling schemes that cost as little as 1.25 samples per pixel and up to 2.0 samples per pixel. By placing sample points in the corners or on the edges of the pixels, sharing can occur between pixels, and this makes it possible to create inexpensive sampling schemes. Using an evaluation and optimization framework, we present optimized sampling patterns costing 1.25, 1.5, 1.75, and 2.0 samples per pixel.

1. Introduction

For computer generated imagery, it is most often desirable to use antialiasing algorithms to improve the image quality. Aliasing effects occur due to undersampling of a signal, where a high frequency signal appears in disguise as a lower frequency signal. Antialiasing algorithms in screen space reduce these image artifacts by raising the sampling rate and computing the color of a pixel as a weighted sum of a number of associated sample points' colors. Such algorithms are often divided into two categories: *supersampling* and *multisampling*.

Supersampling includes all algorithms where the scene is sampled in more than one point per pixel and the final image is computed from the samples. In multisampling techniques, the scene is also sampled in more than one point per pixel, but the results of fragment shader computations (e.g. texture color) is shared between the samples in a pixel.

When using multisampling or supersampling, the positions and weights of the different sample points play a major role in the final image quality. Work has already been done to find the best sampling schemes for certain numbers of samples per pixel but so far no one has studied those that only cost between 1.25 and 2 samples per pixel. This family of sampling schemes is particularly interesting when designing hardware with strict performance and memory limitations, such as graphics hardware for mobile platforms [2].

The second author of this paper has written a technical report [3], where he presented a sampling scheme, called *FLIPTRI*, that costs only 1.25 samples per pixel on average. The purpose of this paper is to present that scheme to a broader audience, as well as taking the idea one step further and explore all sampling schemes that cost between 1.25 and 2.0 samples per pixel. We use the optimization and evaluation framework by Laine and Aila [4] to evaluate the quality of the sampling patterns and to compute optimized sample coordinates and weights.

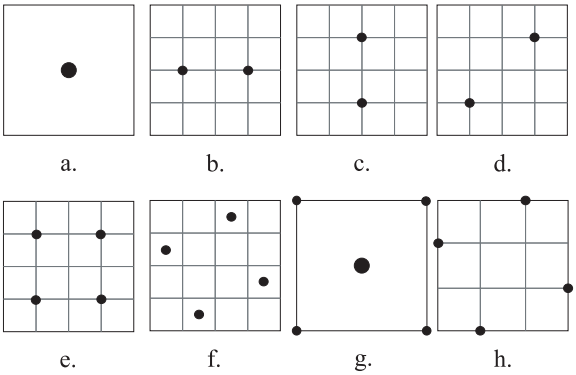

Figure 1: *A collection of sampling schemes. From top left to bottom right: a) Centroid sampling, b) Kyro horizontal, c) Kyro vertical, d) Diagonal, e) 2×2 box pattern, f) Rotated Grid Supersampling (RGSS), g) Quincunx, h) FLIPQUAD.*

This is the original manuscript (author's version) of the article originally submitted for publication. The definitive, revised version was published in Computer Graphics Forum 24, 4 (2005), pp. 843–848.

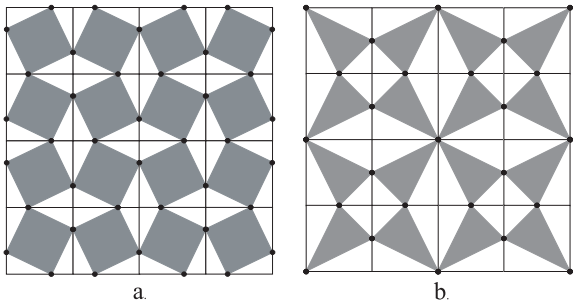

Figure 2: *The a) FLIPQUAD and b) FLIPTRI sampling patterns repeated and reflected over* 4×4 *pixels. For illustration purposes, shaded regions are added to emphasize the sample points that are used for reconstructing the pixels.*

2. Previous Work

The first attempts to produce inexpensive antialiasing in graphics hardware were simple solutions that performed poorly in many cases. This includes, for example, the Kyro hardware described by Akeine-Möller and Haines [1] that uses completely *horizontal* or *vertical* sampling schemes (Figures 1b and 1c) to perform supersampling with two samples per pixel. Placing the samples diagonally instead (Figure 1d) gives a slight visual improvement. The 2×2 *box pattern* (Figure 1e) was also implemented on Kyro, as well as on GeForce 3, 4 and FX [8] and probably on most of the other consumer level hardware. The popularity of 2×2 box pattern is obviously due to its simple implementation. However, it should be noted that the *Rotated Grid Supersampling* (RGSS) pattern (Figure 1f) delivers much better quality with equal cost.

RGSS is a scheme of the more general *N-rooks* family presented by Shirley [9]. Assuming that the pixel is divided into an $n \times n$ uniform grid, an N-rooks pattern satisfies the criterion that no two sample points are placed on same row or column. This is analogous to placing n rooks on an $n \times n$ chessboard without letting any two rooks capture each other. Sampling schemes fulfilling the N-rooks criterion perform well for near-horizontal and near-vertical edges.

An inexpensive multisampling scheme exploiting sample sharing between adjacent pixels is the *Quincunx* scheme [7] used in NVIDIA graphics hardware (Figure 1g). It resembles the five on a six-sided die and is horizontally and vertically symmetric. Therefore, it can be repeated for every pixel and the sampled colors from the sample points in the corners can be shared by four pixels resulting in a total cost of two samples per pixel. A weakness of the Quincunx pattern is that it does not fulfill the N-rooks criterion. The weights are 0.125 for the corner samples, and 0.5 for the center sample.

The *FLIPQUAD* [2] scheme, shown in Figure 1h, is an example of a multisampling scheme based on the N-rooks criterion. All of the sample points can be shared with neighboring pixels when the pattern is horizontally and vertically reflected for different pixels as shown in Figure 2a. Therefore the cost is only two samples per pixel.

Naiman [6] has presented a study where several test subjects were instructed to identify the more jagged edge from a set with two edges rendered at different resolutions. The result of this study can be interpreted as an estimate of the importance of antialiasing for lines with different slopes. It suggests that people are most sensitive to lines nearly horizontal or vertical and to lines with a slope near 45°. To that end, we suggest that one uses the *N-queens* sample point positioning strategy, where the sample point positions fulfill the N-rooks criteria with an additional requirement that no two sample points are allowed on the same diagonal. Both RGSS and FLIPQUAD fulfill this condition, which indicates that they should perform well on edges near 45° as well.

Laine and Aila [4] define an error metric, E_2, for evaluating and optimizing sampling patterns. The metric takes into account the slope-specific acuity factors in the study by Naiman [6] and uses a high-quality reconstruction filter [5] for computing the reference value for the final color of each pixel. The error is evaluated by sweeping a set of lines with different slopes over the sampling pattern and comparing the values given by a sampling pattern to the exact reference value. To correct for human perceptual ability, the slope-specific supersampling errors are weighed based on the acuity measurements in Naiman's study. After this, the maximum error among the different slopes is chosen. This error metric allows us to perform various computer driven searches for best patterns fulfilling a set of restrictions.

3. Notation

We use the same $P(s, r, n)$-notation as Laine and Aila [4] to describe a specific class of sampling patterns. The s parameter represents the number of pixel-sized sample point sets that form the periodically repeating pattern, r is the number of pixels used by the reconstruction filter, and n is the average number of samples taken for each pixel. As a special case, notation r^+ means that the reconstruction filter may also use samples from sample points located on the boundary of the reconstruction region. For instance, if r is 1^+, the reconstruction filter uses samples from sample points in and on the border of a single pixel. In this study we are only concerned with the $P(4, 1^+, n)$ family of sampling patterns where $n \in \{1.25, 1.5, 1.75, 2.0\}$. Note that the FLIPQUAD scheme belongs to this class.

4. The FLIPTRI Sampling Scheme

In this section, we present the FLIPTRI sampling scheme, which previously only has been described in a technical report [3]. The FLIPQUAD scheme assumed that the sampling pattern for a pixel is reflected through the pixel edges in order to ensure that sample sharing between pixels can occur.

Sample point position	Cost
Corner	0.25
Edge	0.5
Pixel	1

Table 1: *Cost of differently placed sample points in the reflected $P(4, 1^+, n)$ class*

We continue to use that approach, and note that if a sample point is placed in the corner of a pixel, then the cost of that sample point is 0.25 samples per pixel, since the corner is shared by four pixels. Sample points on pixel edges are, in general, shared by two pixels, and so the cost is 0.5 samples per pixel. Finally, a sample point placed inside the pixel costs one sample per pixel. Table 1 summarizes these costs. At this point, it is simple to verify the costs for a given scheme. For FLIPQUAD, the cost is $4 \times 0.5 = 2$, and for Quincunx it is $1 + 4 \times 0.25 = 2$ samples per pixel.

To the best of our knowledge, all previously presented supersampling schemes cost at least two samples per pixel. In that sense, the FLIPTRI scheme is quite radical since it costs less than that. This is achieved by placing one sample point at a corner, and two sample points on different edges, as shown in Figure 2b, giving the scheme a total cost of $0.25 + 2 \times 0.5 = 1.25$ samples per pixel. As can be seen, by placing the edge sample points on the edges that do not share the corner sample point, the N-rooks property is fulfilled. Due to Naiman's study [6], one can expect that a similar and slightly better scheme can be obtained by offsetting the sample points positioned on the edges. This way, the error can be moved to angles for which the eye is less sensitive.

5. Sampling patterns

Here, we further explore the *design space* of inexpensive sampling patterns in order to verify the efficiency of the FLIPTRI and FLIPQUAD patterns, but also to produce and evaluate new sampling schemes that cost 1.5 and 1.75 samples per pixel.

5.1. Initial Pattern Generation

Given the costs for different sample point placements (corner, edge or center) in Table 1, it is quite straightforward to write a computer program that generates all possible unique configurations of placements for $P(4, 1^+, n)$ sampling patterns. At this point, we are not interested in the actual coordinates and weights of the sample points but only in deciding if the sample point is placed in a specific corner, on a specific edge or somewhere inside the pixel.

The set of unique configurations of sample point placements is small when $n \leq 2$. Therefore we could use a brute-force algorithm that generates all possible placement configurations with the specified cost and discards a configuration if it is a rotated (90°, 180° or 270°) and/or reflected version of an already generated candidate.

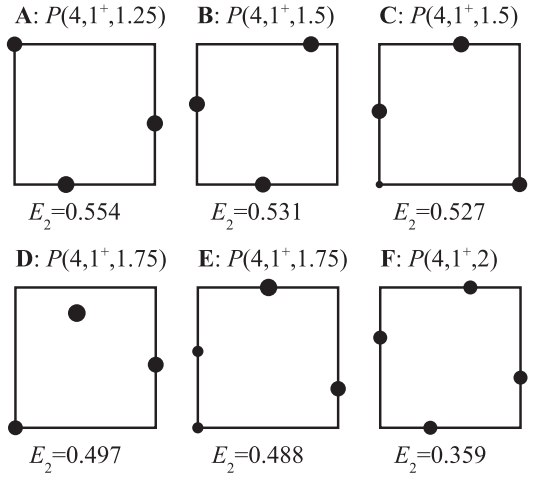

Figure 3: *The figure shows the best schemes in terms of the error metric E_2 in each class. The area of each circle is proportional to the weight of the corresponding sample point. For $n = 1.5$ and $n = 1.75$, two patterns were found with similar error measures, but with different number of samples used during reconstruction. Schemes A and F are versions of the FLIPTRI and FLIPQUAD schemes respectively, that have been optimized using the E_2 error metric (see section 5.2)*

5.2. Pattern Ranking and Optimization

The number of placement configurations with unique properties, in our target families, are only 94 in total and were therefore quite easily manageable. Once generated, we manually examined each configuration and removed those that would clearly not perform well. For example, patterns with all sample points placed along a single edge could be safely removed from further consideration.

The patterns passing this preliminary culling phase were processed by the error metric-based optimizer by Laine and Aila [4] to find an optimal set of weights and coordinates for the sample points and also an estimation of the E_2 error for a given pattern. The error was used to rank the patterns in each category.

6. Results

We present a summary of the most interesting sampling schemes, that we found, in Figure 3, and more exact pattern descriptions in Appendix A. In the cases where two patterns in the same class had similar errors, but different reconstruction costs (number of samples used to compute the final color), both alternatives are presented.

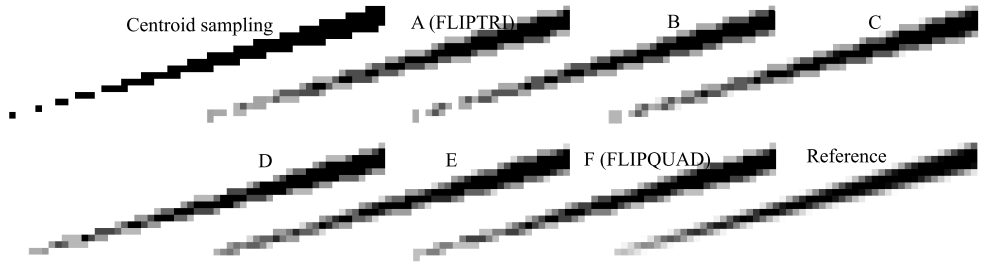

Figure 4: *A thin polygon rendered using the sampling schemes, A through F, from Figure 3. The upper left polygon uses a single sample point per pixel while the reference polygon is supersampled with 1024 jittered grid sample points and uses a* 4×4 *pixel Mitchell-Netravali reconstruction filter.*

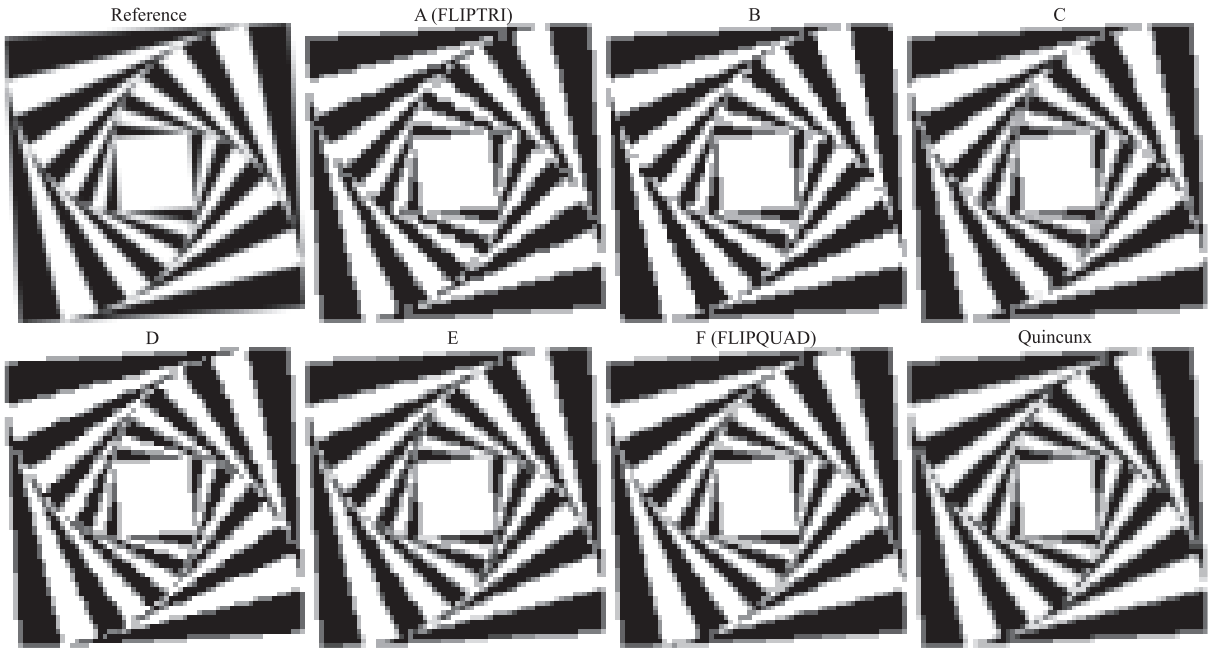

Figure 5: *A spiral test image rendered using the sampling schemes, A through F, from Figure 3. The reference image is supersampled with 1024 jittered grid sample points and uses a* 4×4 *pixel Michell-Netravali reconstruction filter. An image rendered using the Quincunx pattern is also included for comparison.*

6.1. Evaluation

We have evaluated our results both visually and experimentally. For the visual evaluation we simply implemented supersampling using the sampling patterns in Figure 3 and rasterized a number of test images. Figure 4 shows the results of drawing a thin polygon with the respective schemes as well as using centroid sampling and a high quality supersampling pattern for reference. Figure 5 shows a more complex test image.

We have also performed experimental evaluation by rasterizing two synthetic animations. One showing a rotating triangle and the other showing a translating circle. The animations were rasterized using each of our schemes and we compute the per-pixel deviation, as compared to a reference animation. The reference filter was identical to the references used in Figure 4 and 5 but with 256 sample points for performance reasons. Since the evaluation result of the two animations were almost identical, we only present figures for the translation animation in this paper.

It should be noted that this measurement does not take any psychovisual aspects into account, but can still be used as a rough measurement of quality. We can use the deviations to

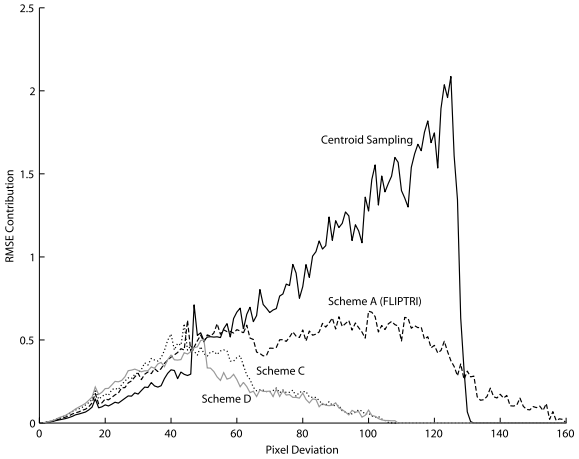

Figure 6: *Histogram of the squared pixel error. This figure shows a comparison between the* 1.25*,* 1.5 *and* 1.75 *sample shemes. Sampling with a single centroid sample is also included for reference. The RMSE contribution is computed as* $C(i)i^2/n$

	E_2	RMSE	max dev.
Centroid sampling	1.256	9.810	131
Diagonal	0.692	6.500	126
Scheme A (FLIPTRI)	0.554	7.548	159
Scheme B	0.531	5.139	114
Scheme C	0.527	4.830	107
Scheme D	0.497	5.000	107
Scheme E	0.488	4.574	108
Quincunx	0.484	4.400	95
Scheme F (FLIPQUAD)	0.359	3.759	76

Table 2: *The errors of our sampling schemes, using different metrics. The "Diagonal" scheme is the diagonal super sampling scheme from Figure 1d.*

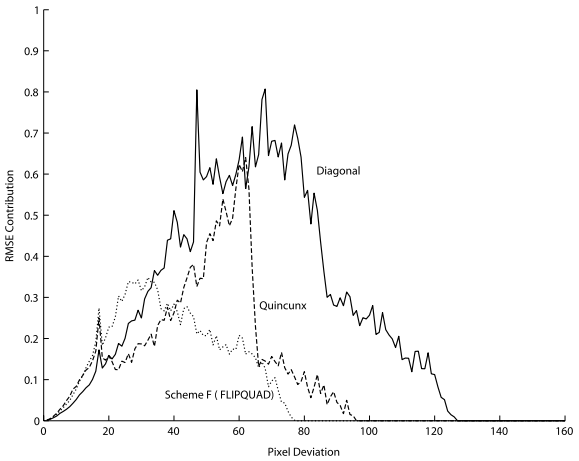

Figure 7: *Histogram of the squared pixel error. This figure shows how scheme f (FLIPQUAD) compares to Quincunx and the Diagonal sample scheme from Figure 1d.*

compute the Root Mean Square Error (RMSE) as

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=0}^{255} C(i)i^2} \qquad (1)$$

Where $C(i)$ is the number of pixels with a given deviation, i, from the reference animation. The normalization constant n is the total number of pixels in the animation. In Figure 6 and 7, we present plots of the $C(i)i^2/n$ values of different patterns. The plots can be seen as the scheme's penalized histogram of the pixel deviations, and the area below each curve is approximately equal to the Mean Square Error (MSE).

For Figure 6, we settle for comparing our own sampling schemes, since there are no other sampling schemes with similar costs. It should be noted that FLIPTRI (scheme A) actually has a worst case deviation higher than that of the centroid sampling scheme. This is because of the placement of sample points in the scheme. Looking at the very center of Figure 2b, we see that the sample point in the corner will be further away from its nearest neighbors than in the case of centroid sampling. It is this distance that causes the greater maximum deviations.

In Figure 7, we compare FLIPQUAD (scheme F) with other sample patterns costing 2 samples per pixel. The behavior of FLIPQUAD is clearly preferable to both Quincunx and normal super sampling using two diagonal samples. The Quincunx plot has a behavior that is very similar to the centroid sampling curve, but the four extra "low-pass" sampling points removes a substantial part of the larger errors.

Finally, we present a summary of the errors in Table 2. We see that even though the E_2 metric does not match the RMSE metric, we still get similar performance with the exception of FLIPTRI and scheme C. We have already motivated the behavior of FLIPTRI. Scheme C is given an unusually low RMSE when we compare it to schemes B and D, but this is not very surprising if we look at the design of the sample schemes. Scheme C has an additional sample point which lowers the RMSE but does not significantly affect the perceptually important nearly horizontal and nearly vertical edges.

7. Conclusion and Future Work

In this paper, we have generated a family of inexpensive sampling schemes. It is difficult to compare and evaluate sampling schemes, so we present visual results along with comparisons using the RMSE metric.

Our evaluation shows that the modified FLIPQUAD pattern clearly stands out in terms of quality. It is also along with FLIPTRI the most cost efficient filter, in terms of $quality/cost$, as far as the E_2 metric is concerned. However, in the case of FLIPTRI, we have to sacrifice the performance of certain edges yielding relatively bad worst case performance. Our other schemes, costing 1.5 and 1.75, samples per pixel also have very low cost and eliminates any chance of a worse performance than normal centroid sampling. In fact they are very close to Quincunx in terms of performance, with respect to both the E_2 and RMSE metrics.

It should also be noted that extra degrees of freedom, in the sample patterns, help in cases not taken into account by the E_2 model. An example of such a case is shown in Figure 4 where the tip of the triangle is thinner than one pixel and therefore do not fall within the assumptions of the E_2 metric. The tip is broken into a few disjoint pieces for the more inexpensive schemes, A-C, while the more expensive schemes, D-F, gives a better visual result.

An open issue with the suggested sampling schemes is where to locate the shader sample point to allow sharing data between sample points in order to avoid texture blurring. For FLIPQUAD this problem has already been addressed [2], but the placement for the other schemes, such as FLIPTRI, are less obvious. It would also be interesting to explore the $P(4, 4^+, n)$ family for small values of n since they potentially provide higher quality.

Acknowledgments

Funding from Swedish Research Council and Swedish Foundation for Strategic Research.

Appendix A: Sample positions and weights

The following table lists the sample point coordinates and weights for the sampling patterns in Figure 3. Each row contains the description of one sample point: x, y, $weight$. The origin $(x = 0, y = 0)$ is located at the center of the pixel. The width and height of a pixel are both 1.0, i.e., the borders are at $x = \pm 0.5$ and $y = \pm 0.5$.

A: $P(4, 1^+, 1.25)$	**B**: $P(4, 1^+, 1.5)$
-0.500, 0.500, 0.299	-0.500, 0.073, 0.335
-0.133, -0.500, 0.360	-0.030, -0.500, 0.331
0.500, -0.064, 0.341	0.313, 0.500, 0.334

C: $P(4, 1^+, 1.5)$	**D**: $P(4, 1^+, 1.75)$
-0.500, 0.022, 0.306	-0.500, -0.500, 0.280
-0.500, -0.500, 0.068	-0.063, 0.318, 0.397
0.083, 0.500, 0.338	0.500, -0.050, 0.323
0.500, -0.500, 0.288	

E: $P(4, 1^+, 1.75)$	**F**: $P(4, 1^+, 2)$
-0.500, -0.500, 0.158	-0.500, 0.143, 0.250
-0.500, 0.045, 0.156	0.500, -0.143, 0.250
0.004, 0.500, 0.380	0.143, 0.500, 0.250
0.500, -0.222, 0.306	-0.143, -0.500, 0.250

References

[1] Akenine-Möller, Tomas, and Eric Haines, *Real-Time Rendering*, 2nd edition, June 2002.

[2] Akenine-Möller, Tomas, and Jacob Ström, "Graphics for the Masses: A Hardware Architecture for Mobile Phones," *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2003)*, vol. 22, no. 3, pp. 801–808, July 2003.

[3] Akenine-Möller, Tomas, *An Extremely Inexpensive Multisampling Scheme*, Chalmers University of Technology, Technical Report No. 03-14, August 2003.

[4] Laine, Samuli, and Timo Aila, "A Weighted Error Metric and Optimization Method for Antialiasing Patterns," to be released as a tech report, 2005.

[5] Mitchell, Don P., and Arun N. Netravali, "Reconstruction filters in computer-graphics," *Computer Graphics (SIGGRAPH '88 Proceedings)*, pp. 221–228, vol. 22, no. 4, August 1988.

[6] Naiman, Avi C., "Jagged edges: when is filtering needed?," *ACM Transactions on Graphics*, vol. 17, no. 4, pp. 238–258, 1998.

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

[8] NVIDIA Corp., *The GeForce 6 Series of GPUs: High Performance and Quality for Complex Image Effects*, Technical brief, 2004.

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