Efficient Triangle Coverage Tests for Stochastic Rasterization

Samuli Laine Tero Karras*

NVIDIA Research

Abstract

In our previous paper on stochastic rasterization [Laine et al. 2011], we stated that a 5D triangle coverage test consumes approximately 25 FMA (fused multiply-add) operations. This technical report details the operation of our coverage test. We also provide variants specialized for defocus-only and motion-only cases.

1 A Basic 5D Test

Let us denote the three moving input vertices as

$$\begin{aligned}\mathbf{v}_0(t) &= (x_0, y_0, w_0) + t \cdot (\Delta x_0, \Delta y_0, \Delta w_0)\\ \mathbf{v}_1(t) &= (x_1, y_1, w_1) + t \cdot (\Delta x_1, \Delta y_1, \Delta w_1)\\ \mathbf{v}_2(t) &= (x_2, y_2, w_2) + t \cdot (\Delta x_2, \Delta y_2, \Delta w_2)\end{aligned}$$

and the related clip-space circles of confusion (CoC) as

$$\begin{aligned}C_0(t) &= c_0 + t \cdot \Delta c_0\\ C_1(t) &= c_1 + t \cdot \Delta c_1\\ C_2(t) &= c_2 + t \cdot \Delta c_2\end{aligned}$$

A simple and efficient way for performing the coverage test is to instantiate the triangle at the given (u, v, t) coordinates[1], and perform a standard 2D coverage test against pixel coordinates (x, y). This leads to the following procedure. The number of FMAs on each line is shown on the right in parentheses.

$$\begin{aligned}
W_0 &= w_0 + t \cdot \Delta w_0 && (1)\\
W_1 &= w_1 + t \cdot \Delta w_1 && (1)\\
W_2 &= w_2 + t \cdot \Delta w_2 && (1)\\
C_0 &= c_0 + t \cdot \Delta c_0 && (1)\\
C_1 &= c_1 + t \cdot \Delta c_1 && (1)\\
C_2 &= c_2 + t \cdot \Delta c_2 && (1)\\
X_0 &= x_0 + t \cdot \Delta x_0 - W_0 \cdot x + C_0 \cdot u && (3)\\
X_1 &= x_1 + t \cdot \Delta x_1 - W_1 \cdot x + C_1 \cdot u && (3)\\
X_2 &= x_2 + t \cdot \Delta x_2 - W_2 \cdot x + C_2 \cdot u && (3)\\
Y_0 &= y_0 + t \cdot \Delta y_0 - W_0 \cdot y + C_0 \cdot v && (3)\\
Y_1 &= y_1 + t \cdot \Delta y_1 - W_1 \cdot y + C_1 \cdot v && (3)\\
Y_2 &= y_2 + t \cdot \Delta y_2 - W_2 \cdot y + C_2 \cdot v && (3)\\
d_{01} &= X_0 \cdot Y_1 - Y_0 \cdot X_1 && (2)\\
d_{12} &= X_1 \cdot Y_2 - Y_1 \cdot X_2 && (2)\\
d_{20} &= X_2 \cdot Y_0 - Y_2 \cdot X_0 && (2)
\end{aligned}$$

Here, W_i are the vertices' w coordinates interpolated according to t, and C_i are the circles of confusion. X_i and Y_i are the vertices' x and y coordinates, respectively, interpolated according to t, sheared according to W and pixel (x, y) coordinates, and adjusted by interpolated circles of confusion. Note that the resulting coordinates remain in clip space. The shear is designed so that it makes the clip-space ray towards pixel position $(x, y, 1)$ point in direction $(0, 0, 1)$. This is equivalent to screen-space translation that moves (x, y) to the viewport origin. What remains is testing the viewport origin against the three edge functions. We only need the constant terms d_{ij} of the edge functions, and these give the edge evaluation results directly.

The calculation of the edge evaluations in the end are equivalent to a 3D ray vs triangle test, where the ray is always from $(0, 0, 0)$ towards $(0, 0, 1)$, and the triangle lies in 3D clip space. The formulas for the 3D determinants (corresponding to volumes between ray and triangle edges) simplify to the 2D determinants above. The shear has no effect on volumes, so perspective-correct barycentric coordinates can be obtained directly by dividing d_{ij} by their sum.

In total, this method requires 30 FMAs. We assume that the comparison against zero is free, i.e., performed by the hardware, after d_{ij} have been computed, so no other instructions are required. For samples that pass the coverage test it is also necessary to check that the z depth of the hit point is within the view frustum, but this is not a part of the coverage test itself.

2 Optimizations for the 5D Test

The optimized 5D coverage test is based on the assumption that CoC is linear in w in clip space, corresponding to the ubiquitous thin-lens model. This linear dependence can be specified using two constants A, B so that

$$C = A + B \cdot w$$

We can now remove the interpolation of C_i saving 3 FMAs, but we need to replace the corresponding terms in further rows by $(A + B \cdot W_i)$, as follows.

$$\begin{aligned}X_0 &= x_0 + t \cdot \Delta x_0 - W_0 \cdot x + (A + B \cdot W_0) \cdot u\\ X_1 &= x_1 + t \cdot \Delta x_1 - W_1 \cdot x + (A + B \cdot W_1) \cdot u\\ X_2 &= x_2 + t \cdot \Delta x_2 - W_2 \cdot x + (A + B \cdot W_2) \cdot u\end{aligned}$$

Expanding the product at the end yields

$$\begin{aligned}X_0 &= x_0 + t \cdot \Delta x_0 - W_0 \cdot x + A \cdot u + B \cdot u \cdot W_0\\ X_1 &= x_1 + t \cdot \Delta x_1 - W_1 \cdot x + A \cdot u + B \cdot u \cdot W_1\\ X_2 &= x_2 + t \cdot \Delta x_2 - W_2 \cdot x + A \cdot u + B \cdot u \cdot W_2\end{aligned}$$

and collecting terms with W gives

$$\begin{aligned}X_0 &= x_0 + t \cdot \Delta x_0 + W_0 \cdot (B \cdot u - x) + A \cdot u\\ X_1 &= x_1 + t \cdot \Delta x_1 + W_1 \cdot (B \cdot u - x) + A \cdot u\\ X_2 &= x_2 + t \cdot \Delta x_2 + W_2 \cdot (B \cdot u - x) + A \cdot u\end{aligned}$$

This allows us to precalculate $(A \cdot u)$ and $(B \cdot u - x)$. However, while precalculation of $(A \cdot u)$ removes two multiplications, it does not reduce the number of instructions in computing X_i because we would have two terms without multiplication. Hence it is better to precalculate only one of the terms as follows.

$$\begin{aligned}E_x &= B \cdot u - x\\ X_0 &= x_0 + t \cdot \Delta x_0 + W_0 \cdot E_x + A \cdot u\\ X_1 &= x_1 + t \cdot \Delta x_1 + W_1 \cdot E_x + A \cdot u\\ X_2 &= x_2 + t \cdot \Delta x_2 + W_2 \cdot E_x + A \cdot u\end{aligned}$$

*e-mail: {slaine,tkarras}@nvidia.com

NVIDIA Technical Report NVR-2011-003, September 2011.

[1] To ensure that the CoCs appear circular on the screen, we need to account for the aspect ratio of the viewport. This can be done, e.g., by scaling u or v according to the aspect ratio in the sample generation phase. In the coverage tests that assume thin-lens CoCs (Section 2), this scaling can also be accomplished by having separate CoC coefficients A, B for x and y axes.

Let us now look at the entire coverage test exploiting this optimization.

$$
\begin{aligned}
W_0 &= w_0 + t \cdot \Delta w_0 && (1)\\
W_1 &= w_1 + t \cdot \Delta w_1 && (1)\\
W_2 &= w_2 + t \cdot \Delta w_2 && (1)\\
E_x &= B \cdot u - x && (1)\\
E_y &= B \cdot v - y && (1)\\
X_0 &= x_0 + t \cdot \Delta x_0 + W_0 \cdot E_x + A \cdot u && (3)\\
X_1 &= x_1 + t \cdot \Delta x_1 + W_1 \cdot E_x + A \cdot u && (3)\\
X_2 &= x_2 + t \cdot \Delta x_2 + W_2 \cdot E_x + A \cdot u && (3)\\
Y_0 &= y_0 + t \cdot \Delta y_0 + W_0 \cdot E_y + A \cdot v && (3)\\
Y_1 &= y_1 + t \cdot \Delta y_1 + W_1 \cdot E_y + A \cdot v && (3)\\
Y_2 &= y_2 + t \cdot \Delta y_2 + W_2 \cdot E_y + A \cdot v && (3)\\
d_{01} &= X_0 \cdot Y_1 - Y_0 \cdot X_1 && (2)\\
d_{12} &= X_1 \cdot Y_2 - Y_1 \cdot X_2 && (2)\\
d_{20} &= X_2 \cdot Y_0 - Y_2 \cdot X_0 && (2)
\end{aligned}
$$

The speedup is rather disappointing, as we have succeeded in removing only one FMA, yielding 29 FMAs in total. However, we can now exploit the fact that all computations of X_i share a common term $(A \cdot u)$, and similarly for Y_i. The idea is to compute X_0 as above, but instead of X_1 and X_2 calculate their differences to X_0 (respectively for Y_1 and Y_2). This corresponds to operating on the side vectors of the triangle instead of position vectors of its vertices.

Denoting difference values with primes, the calculation of, e.g., X_1' is then carried out as follows

$$
\begin{aligned}
X_1' &= x_1 + t \cdot \Delta x_1 + W_1 \cdot E_x + A \cdot u\\
&\quad - (x_0 + t \cdot \Delta x_0 + W_0 \cdot E_x + A \cdot u)\\
&= (x_1 - x_0) + t \cdot (\Delta x_1 - \Delta x_0) + (W_1 - W_0) \cdot E_x
\end{aligned}
$$

For this to be efficient, we need to form t-dependent side vectors at triangle setup. We therefore need

$$
\begin{aligned}
x_1' &= x_1 - x_0 \qquad & \Delta x_1' &= \Delta x_1 - \Delta x_0\\
y_1' &= y_1 - y_0 \quad , & \Delta y_1' &= \Delta y_1 - \Delta y_0\\
w_1' &= w_1 - w_0 \qquad & \Delta w_1' &= \Delta w_1 - \Delta w_0
\end{aligned}
$$

and similarly for $x_2', \Delta x_2'$, etc. Now we can compute, e.g., X_1' as follows

$$
\begin{aligned}
W_1' &= w_1' + t \cdot \Delta w_1' && (1)\\
X_1' &= x_1' + t \cdot \Delta x_1' + W_1' \cdot E_x && (2)
\end{aligned}
$$

which is one FMA less work than before.

Evaluating the edge functions is equivalent to evaluating three scalar triple products where the last vector is always $(0, 0, 1)$. Denoting the translated, sheared vertex positions as $V_i = (X_i, Y_i)$, each result is obtained directly from the well-known two-dimensional analog of cross product[2] between V_i. With some notational sleight of hand, our original calculation can therefore be symbolically represented as

$$
\begin{aligned}
d_{01} &= V_0 \times V_1\\
d_{12} &= V_1 \times V_2\\
d_{20} &= V_2 \times V_0
\end{aligned}
$$

Now instead of V_1 and V_2 we compute $V_1' = (X_1', Y_1')$ and $V_2' = (X_2', Y_2')$, so that $V_1 = V_0 + V_1'$ and $V_2 = V_0 + V_2'$. Therefore, the cross products above can be written as

$$
\begin{aligned}
d_{01} &= V_0 \times (V_0 + V_1')\\
d_{12} &= (V_0 + V_1') \times (V_0 + V_2')\\
d_{20} &= (V_0 + V_2') \times V_0
\end{aligned}
$$

Expanding the cross products, reordering the calculations and simplifying yields

$$
\begin{aligned}
d_{01} &= V_0 \times V_1'\\
d_{20} &= V_2' \times V_0\\
d_{12} &= V_1' \times V_2' - d_{01} - d_{20}
\end{aligned}
$$

The entire coverage test can now be written as follows.

$$
\begin{aligned}
W_0 &= w_0 + t \cdot \Delta w_0 && (1)\\
W_1' &= w_1' + t \cdot \Delta w_1' && (1)\\
W_2' &= w_2' + t \cdot \Delta w_2' && (1)\\
E_x &= B \cdot u - x && (1)\\
E_y &= B \cdot v - y && (1)\\
X_0 &= x_0 + t \cdot \Delta x_0 + W_0 \cdot E_x + A \cdot u && (3)\\
X_1' &= x_1' + t \cdot \Delta x_1' + W_1' \cdot E_x && (2)\\
X_2' &= x_2' + t \cdot \Delta x_2' + W_2' \cdot E_x && (2)\\
Y_0 &= y_0 + t \cdot \Delta y_0 + W_0 \cdot E_y + A \cdot v && (3)\\
Y_1' &= y_1' + t \cdot \Delta y_1' + W_1' \cdot E_y && (2)\\
Y_2' &= y_2' + t \cdot \Delta y_2' + W_2' \cdot E_y && (2)\\
d_{01} &= X_0 \cdot Y_1' - Y_0 \cdot X_1' && (2)\\
d_{20} &= X_2' \cdot Y_0 - Y_2' \cdot X_0 && (2)\\
d_{12} &= (X_1' \cdot Y_2' - d_{01}) - (Y_1' \cdot X_2' + d_{20}) && (2^*)
\end{aligned}
$$

The total cost of this calculation is 25 FMAs, assuming that comparing d_{12} against zero is in reality performed by comparing the two terms it is composed of against each other. The comparison of two floating-point terms is much cheaper in hardware than a full floating-point subtraction, so this is a more efficient solution. Only if the sample is covered, the subtraction has to be carried out in order to calculate barycentric coordinates. Note that the use of side vectors instead of position vectors for $\mathbf{v}_1$ and $\mathbf{v}_2$ does not increase the amount of input data required.

3 Defocus-Only Coverage Test

By setting $t = 0$ in the above coverage test and optimizing, we obtain a defocus-only coverage test as follows.

$$
\begin{aligned}
E_x &= B \cdot u - x && (1)\\
E_y &= B \cdot v - y && (1)\\
X_0 &= x_0 + w_0 \cdot E_x + A \cdot u && (2)\\
X_1' &= x_1' + w_1' \cdot E_x && (1)\\
X_2' &= x_2' + w_2' \cdot E_x && (1)\\
Y_0 &= y_0 + w_0 \cdot E_y + A \cdot v && (2)\\
Y_1' &= y_1' + w_1' \cdot E_y && (1)\\
Y_2' &= y_2' + w_2' \cdot E_y && (1)\\
d_{01} &= X_0 \cdot Y_1' - Y_0 \cdot X_1' && (2)\\
d_{20} &= X_2' \cdot Y_0 - Y_2' \cdot X_0 && (2)\\
d_{12} &= (X_1' \cdot Y_2' - d_{01}) - (Y_1' \cdot X_2' + d_{20}) && (2^*)
\end{aligned}
$$

The cost of this test is 16 FMAs, with the same reservations as above regarding the comparison between d_{12} and zero. If we set $t = 0$ in the basic coverage test shown in Section 1, we obtain a test that supports arbitrary per-vertex CoCs but requires 18 FMAs.

4 Motion-Only Coverage Test

If there is no defocus, CoC coefficients A and B are zero, leading to the following solution.

[2] To assure ourselves that this two-dimensional analog of cross product follows the same algebraic rules as the usual three-dimensional cross product, we note that $\mathbf{u} \times \mathbf{v} = a$ in two dimensions is equivalent to $(\mathbf{u}_x, \mathbf{u}_y, 0) \times (\mathbf{v}_x, \mathbf{v}_y, 0) = (0, 0, a)$ in three dimensions. Therefore, any identities involving three-dimensional cross products are also valid for the two-dimensional analog.

$$
\begin{aligned}
W_0 &= w_0 + t \cdot \Delta w_0 && (1)\\
W_1 &= w_1 + t \cdot \Delta w_1 && (1)\\
W_2 &= w_2 + t \cdot \Delta w_2 && (1)\\
X_0 &= x_0 + t \cdot \Delta x_0 - x \cdot W_0 && (2)\\
X_1 &= x_1 + t \cdot \Delta x_1 - x \cdot W_1 && (2)\\
X_2 &= x_2 + t \cdot \Delta x_2 - x \cdot W_2 && (2)\\
Y_0 &= y_0 + t \cdot \Delta y_0 - y \cdot W_0 && (2)\\
Y_1 &= y_1 + t \cdot \Delta y_1 - y \cdot W_1 && (2)\\
Y_2 &= y_2 + t \cdot \Delta y_2 - y \cdot W_2 && (2)\\
d_{01} &= X_0 \cdot Y_1 - Y_0 \cdot X_1 && (2)\\
d_{12} &= X_1 \cdot Y_2 - Y_1 \cdot X_2 && (2)\\
d_{20} &= X_2 \cdot Y_0 - Y_2 \cdot X_0 && (2)
\end{aligned}
$$

The total cost is 21 FMAs, and there is no benefit in using side vectors instead of position vectors for the vertices of the triangle.

References

LAINE, S., AILA, T., KARRAS, T., AND LEHTINEN, J. 2011. Clipless dual-space bounds for faster stochastic rasterization. *ACM Trans. Graph. 30*, 4, 106:1–106:6.