Anti-Aliasing in Triangular Pixels

Andrew Glassner
Xerox PARC
3333 Coyote Hill Road
Palo Alto, CA 94304
glassner@parc.xerox.com
415/494-4467

Most graphics rendering takes place in a square pixel grid. This is because the boundaries of the square grid are all vertical and horizontal lines, making sampling, filtering, and reconstructon easy.

But it is well-known that the square grid is not the most uniform in terms of sampling densities. In two dimensions, the triangular lattice shown in Figure 1a is more isotropic. This lattice permits two tilings: hexagonal and triangular, shown in Figures 1b and 1c. Both unit cells have been used for image processing, and to a lesser extent, computer graphics (note that the arrangement of phosphor triples on the inside of a CRT monitor is a lattice like that of Figure 1).

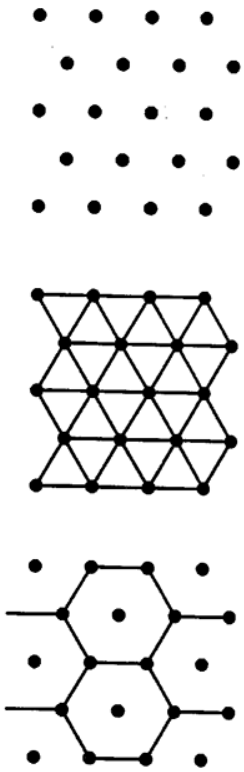

Figure 1

This note presents an inexpensive anti-aliasing technique for triangular pixels.Figure 2 shows a tiling of the plane by equilateral triangles, and a polygon passing through each triangle. We wish to know the area occupied by the polygon within each triangle.

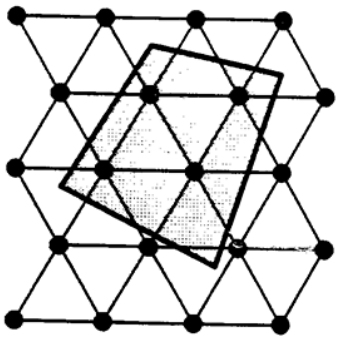

Figure 2

We can parameterize the pre-filtered contribution of the polygon to each element of the grid by building a look-up table based on the vertices of the polygon and its intersection with the edges of the triangle. A simple box filter simply returns the amount of area within the triangle; higher-order filters may apply a weighting function (e.g. a Gaussian) first.

Our analysis is inspired by the square-pixel anti-aliasing technique of [Abram, Westover, Whitted]. We first detect the cases when the polygon is either completely within the triangle, or the triangle is completely surrounded by the polygon. In the first case, the area of overlap is the area of the polygon; in the latter it is the area of the triangle.

We will focus on two cases: no vertices of the polygon are within the triangle (called the V0 case), and one vertex of the polygon is in the triangle (called the V1 case).

In the following geometry, we assume the side length of the triangular pixel is 1. In each example, we consider the two points where the edges of the polygon intersect the edges of the triangle. We call these points **A** and **B**. We parameterize these by their distances from a distinguished vertex; these distances are called α and β respectively. We start with the V0 case.

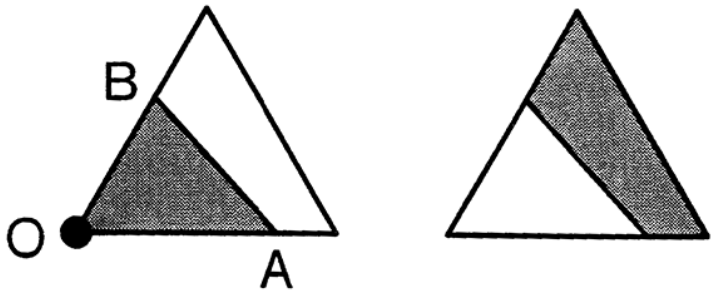

Figure 3

Figure 3 shows the two possible V0 configurations. In 3a, the area to be counted is the shaded triangle. Without loss of generality, we place one vertex at the origin (**O**), and align the edge containing one of the intersection points with the X axis. We may then identify the two points by their distances α and β from the surrounded vertex: we label the coordinates **A**=(α, 0) and **B**=($\frac{\beta}{2}$, $\frac{\beta\sqrt{3}}{2}$). Since for a triangle, Area $= \frac{1}{2}bh$, we find

$$A = \alpha\beta\frac{\sqrt{3}}{4} \qquad \text{(Case V0, Convex)}$$

When the vertex at the origin is part of the area to be counted, we call that the *convex* case. If the vertex is outside the area, we call that the *concave* case. This notation is prompted by the geometry of the Case V1 figures, below. The area in Figure 3b is just the area of the triangle $\left(A = \frac{\sqrt{3}}{4}\right)$ minus this amount:

$$A = \frac{\sqrt{3}}{4}(1 - \alpha\beta). \qquad \text{(Case V0, Concave)}$$

This concludes type V0.

The V1 case comes in three types, called Types 1, 2, and 3, illustrated in Figures 4, 5, and 6. Note that Types 2 and 3 each have the two configurations we saw for the V0 case. In each type, we place the distinguished vertex at the origin.

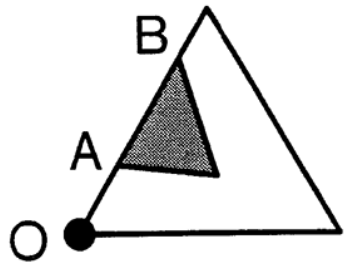

Figure 4

We start with Type 1. In Figure 4, the two edges of the polygon intersect the triangle on the same side, at distances α and β from the origin. These points have coordinates **A**=$\left(\frac{\alpha}{2}, \frac{\alpha\sqrt{3}}{2}\right)$, **B**=$\left(\frac{\beta}{2}, \frac{\beta\sqrt{3}}{2}\right)$, and **V** =($V_x$ V_y). To find the area of triangle (**V, A, B**) we will use the general polygon area formula

$$A = \frac{1}{2}\sum_{i=1}^{n}\left(x_i\, y_{(i+1) \bmod n} - y_i\, x_{(i+1) \bmod n}\right)$$

which may be expanded in the triangular case (n=3) to

$$A = \frac{1}{2}\left(x_1y_2 + x_2y_3 + x_3y_1 - y_1x_2 - y_2x_3 - y_3x_1\right)$$

Assigning the point **A** to point 1, point **B** to point 2, and point **V** to point 3, and simplifying, we get

$$A = \frac{\alpha - \beta}{4}\left(V_x\sqrt{3} - V_y\right) \qquad \text{(Case V1, Type 1)}$$

This concludes Type 1.

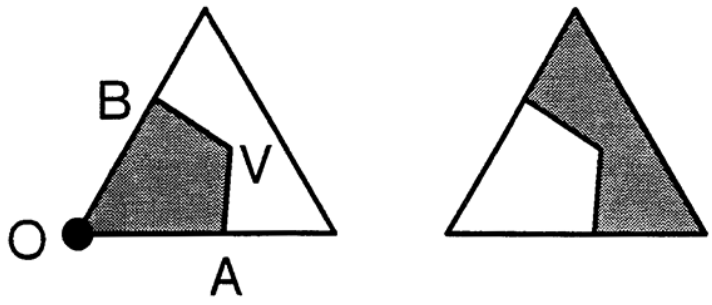

Figure 5

Type 2 is illustrated in Figure 5. This type is identified by a single vertex **V** in the triangle (thus case V1), the two polygon edges passing through two triangle edges, and the vertex **V** outside the triangle (**O, A, B**). We can detect this situation by finding the distance from the midpoint **M** to **V**, and comparing that to the distance from the origin **O** to **V**. If |**MV**| < |**OV**|, then we have Figure 5, otherwise Figure 6.

We find the area in Figure 5 by first finding the area of the triangle (**O, A, B**), and then adding the area of triangle (**A, V, B**). As we saw from case V0, the area of "inner" triangle (**O, A, B**) is

$$A_i = \frac{\sqrt{3}}{4}\alpha\beta.$$

Using the triangle area formula above, we find that the area of "outer" triangle (**A, V, B**) is

$$A_o = \frac{1}{4}\left(\alpha\beta\sqrt{3} - \beta\ V_y\ - \beta\ V_x\sqrt{3} - 2\alpha\ V_y\right)$$

The area of the complete quadrilaterial is thus

$$A = A_i + A_o. \qquad \text{(Case V1, Type 2, Convex)}$$

Note that for the concave case, we subtract this area from the triangle:

$$A = \frac{\sqrt{3}}{4} - (A_i + A_o) \qquad \text{(Case V1, Type 2, Concave)}$$

This ends Type 2.

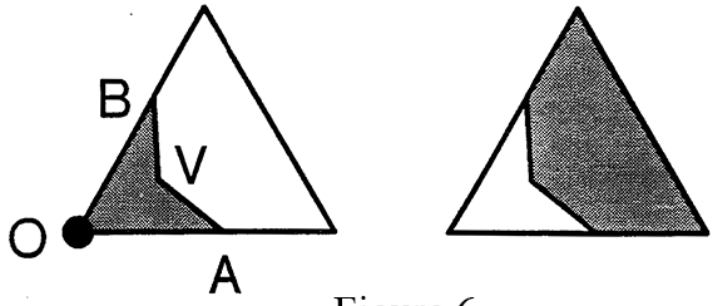

Figure 6

Type 3 is shown in Figure 6. It follows much the same argument, only we subtract the triangle (V, α, β) from the other:

$$A = A_i - A_o. \qquad \text{(Case V1, Type 3, Convex)}$$

$$A = \frac{\sqrt{3}}{4} - (A_i - A_o) \qquad \text{(Case V1, Type 3, Concave)}$$

This concludes the case analysis. Note that in Case V1, Types 2 and 3 Concave find the area of pentagonal regions by finding the complementary quadrilateral within a triangle, and decomposing that quadrilateral into triangles. This decomposition is a general result for this situation.

Note that each analysis above assumed that the triangle was in a canonical position. For a tiling such as that in Figure 2, half of the triangles will be vertically mirrored. One may simply reflect the y-coordinate of each point to align those triangles with the ones above. A further reflection or rotation of $2\pi/3$ or $4\pi/3$ may be required to move the included vertex and point A into proper position. One may explicitly perform this rotation on the points of intersection with the edges (and the included vertex if necessary), and then apply the formulae above. An alternative is to simply form new sets of equations with these rotations built in, and then select the appropriate set for each case.

This note uses included areas as an anti-aliasing meaure. This implies that we are using a box filter. A box filter is better than no filter at all, but it is not ideal. One could extend this technique by applying the ideas of [Abram, Westover, Whitted] to build pre-computed anti-aliasing tables based on parameterized intersections with the triangle edges.

We consider polygons which include more than one vertex in the triangle to be sufficiently rare, and sufficiently hard, that traditional (and more expensive) area-estimation techniques such as low-resolution scan conversion and analytial area computation should be considered, rather than building many more case analyses.

These techniques are also applicable to text filtering, drawing thick lines, and other scan-conversion operations with triangular pixels.

References

[Abram, Westover, Whitted] Abram, Greg; Westover, Lee; Whitted, Turner, "Efficient Alias-free Rendering using Bit-masks and Look-up Tables," Computer Graphics, Vol. 19, No. 3, (Proc. Siggraph '85), July 1985, pp. 53-9