Sub-pixel accurate rasterization of polygons using the Amiga blitter

The Amiga has a blitter coprocessor that can assist in drawing filled polygons. In this document it is explained how the blitter can be used to do sub-pixel accurate rasterization. Before reading further you may want to read the "Drawing lines using the Amiga blitter" [1] document in order to understand how the blitter is programmed by setting the initial accumulator and accumulator increments to draw lines.

Rasterization

The problem that we want to solve is illustrated in Figure 1. The figure contains a grid of 7x9 pixels. The dots are the pixel centers. A triangle is drawn in darker blue that consists of three vertices and three edges. The vertices are labeled v1, v2 and v3.

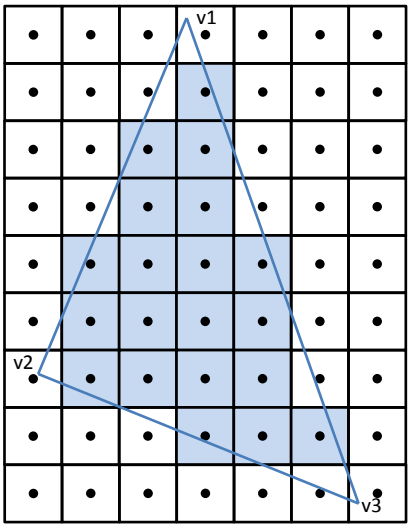

Figure 1: Pixels covered by the triangle are shown in blue.

Our goal is to set the bits in the frame buffer corresponding to only those pixels whose centers are inside the triangle, shown in blue in the figure. Note that if the rasterization hadn't been sub-pixel accurate, then the vertices would first snap to their closest pixel centers (which have integer coordinates), before determining which pixels are covered. Doing so would change the shape of the triangle slightly.

Drawing a filled polygon using the blitter is done in two steps:

1. Drawing the outline (the edges) using the line draw mode.
2. Filling the inside of the polygon using the area fill mode.

Area fill

How to program the blitter to do area fill is not explained in detail here, but enough details are explained so that we can determine how the lines should be drawn in the first step. When doing area fill the blitter works one row at a time, going **from right to left**. The following pseudo-code describes the algorithm that the blitter uses to do area fill.

```
boolean fill = false;
for (int x = W-1; x >= 0; x--) {
        should_fill = fill;
        if (pixel[x]) fill = !fill;
        if (should_fill) pixel[x] = !pixel[x];
}
```

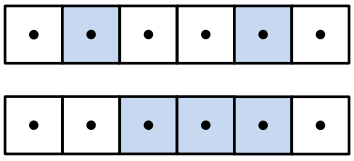

Figure 1: Before (above) and after (below) exclusive mode area fill.

When the blitter encounters a pixel that is set it toggles the fill mode. If the fill mode was enabled before encountering this pixel, the pixel it toggled from set to clear, or from clear to set. Figure 2 shows an example of a row before (above) and after (below) filling.

Drawing the outline

Now we can determine how the lines should be drawn in the outline step, in order to get the desired result after the area fill step. The left half of Figure 3 is the output from the outline step required to get the desired result after the fill step, seen in the right half of Figure 3.

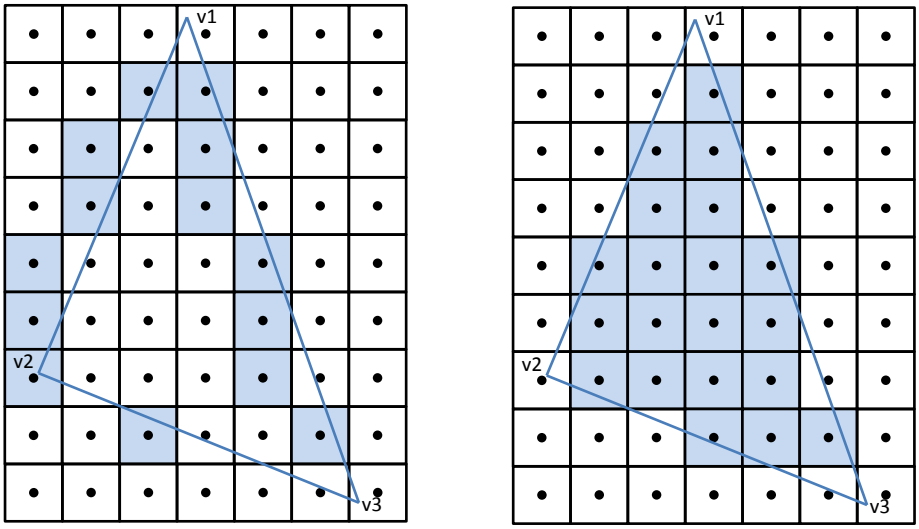

Figure 3: The output from the outline step (left) and the corresponding output from the fill step (right).

In the outline step we should, for each edge and for each row, set the pixel immediately to the left of the edge. Note that in the example in Figure 3, the first and last rows should have no pixels set in the output from the outline step, although there are edges that cover those rows. This is accomplished by drawing lines in exclusive-or mode; when the edge from v1 to v2 is drawn, the pixel in the 3rd column in the 1th row is set, but that pixel is then cleared when the edge from v1 to v3 is drawn.

Deriving the values used to program the blitter

Consider the edge between v2 and v3, seen in Figure 4. After the outline step, the pixels that are highlighted in blue should be set. Let (x2, y2) be the coordinates for vertex v2, and (x3, y3) be the coordinates for v3. Pixel centers have integer coordinates.

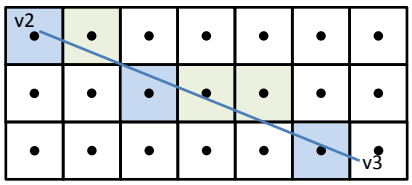

Figure 4: The pixels in blue should be set for the edge between v2 and v3.

Drawing the line is done using the ONEDOT mode of the blitter. In the ONEDOT mode, when the blitter steps along the line it only sets the first pixel in every row, and then leaves the subsequent pixels in that row clear. In Figure 4, the blitter working in ONEDOT mode steps along the highlighted pixels (both blue and green) but only sets the blue pixels.

As the edge is seen to lay in octant 1 (refer to [1]) the major direction is to the right and the minor direction is down. First we determine which rows are covered by the edge. The integer y coordinates for the first and last pixels covered by the edge are obtained as Y2 = floor(y2 + 1) and Y3 = floor(y3). If Y3<Y2 then no pixels are covered by the edge and the line drawing is skipped.

Let X2 be the integer x coordinate such that (X2, Y2) is the coordinates for the pixel center of the first pixel that should bet set. From the line equation, the integer coordinate X2 as found as:

$$X2 = \text{floor}(x2 + (Y2 - y2)*dx/dy)$$

The width (dx) and height (dy) of the line are dx=x3-x2 and dy=y3-y2.

Let's assume that the pixel in the top-left corner of Figure 4 has coordinates (X2, Y2) = (0, 0). After the blitter has set the first pixel at (0, 0) it must decide if it should take a step in the major direction only, or in the combined major and minor direction. By inspecting Figure 4 we reason as follows. If a step is taken in the combined major and minor direction then the pixel at (1, 1) will be set next. But pixel (1, 1) is not the pixel immediately to the left of the line, which we see as pixel (2, 1) is also to the left of the line.

Therefore, if the pixel (X, Y) was the pixel that was just set, the blitter should take a step in the combined major and minor direction only if the pixel center at (X+2, Y+1) is to the right of (or on) the line, that is if:

$$X+2 \geq x2 + ((Y+1)-y2)*dx/dy$$

By reordering terms we have:

$$(X-x2+2)*dy - (Y-y2+1)*dx \geq 0$$

The initial accumulator should therefore be set to: (X2-x2+2)*dy - (Y2-y2+1)*dx. The increment of the accumulator for a step that is taken in the major direction is dy, and the increment for a step that is taken in the combined major and minor direction is dy - dx.

As noted in [1] the initial accumulator and accumulator increments must be multiplied by two before they are written to the blitter registers, and the correct values are then (X2 and Y2 are found as above):

- acc = 2*((X2-x2+2)*dy - (Y2-y2+1)*dx)
- inc_maj = 2*dy
- inc_majmin = 2*(dy - dx)

The number of steps that the blitter should step along the line, and that should be programmed in the BLTSIZE register, is dX = X3 - X2 + 1, where X3 is found in the same way as X2:

$$X3 = \text{floor}(x2 + (Y3 - y2)*dx/dy)$$

Note that only the case for octant 1 is covered here; the other octants are handled similarly.

There are some details that should be handled in an implementation. Numbers are expressed in fixed-point, typically 12.4 (12 bits for the integer part and 4 bits for the fractional part). When dividing numbers, care must be taken to do rounding properly. Refer to the source code for all details.