Polygon Scan Conversion

1 Triangles

Fast triangle scan-conversion is based on linearity of the edges. If an intersection of a scanline and an edge is known, then to get the intersection of that edge with the next scanline below, some constant (depending on the edge's slope) needs to be added to the x-coordinate of the current intersection (see Figure 1).

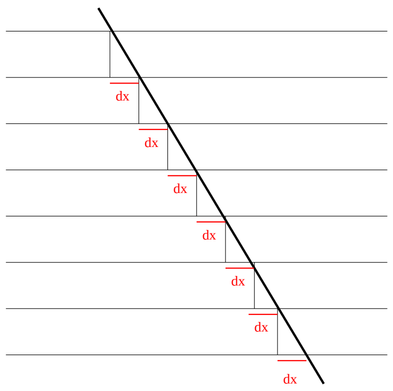
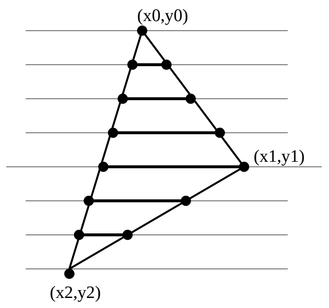


Figure 1: Intersections of a line with horizontal scanlines.

This leads to the algorithm in Figure 2. For simplicity, let's assume that the triangle's vertices have integer coordinates. The algorithm works by walking through the scanlines intersecting the triangle starting from the uppermost one until the lowermost is reached. Each such scanline intersects the triangle's boundary at two points (unless degeneracy like horizontal edge in the triangle appears – see next page). The x-coordinates of the intersection points are updated at each step by adding a constant (one over the edge's slope).



```
xleft = xright = x0;
dxleft = (x2−x0)/(y2−y0);
dxright = (x1−x0)/(y1−y0);
for y := y0 to y1+1 step −1 d
    plot all pixels between
      (xleft,y) and (xright,y);
    xleft −= dxleft;
    xright −= dxright;

dxright = (x2−x1)/(y2−y1);
for y:=y1 to y2 step −1 do
    plot all pixels between
      (xleft,y) and (xright,y);
    xleft −= dxleft;
    xright −= dxright;
```

Figure 2: Triangle scan-conversion.

Clearly, things change slightly if the vertex (x1,y1) is on the left of the edge joining the other two vertices. In this case, the increment that is applied to the intersection point on the left needs to be changed (Figure 3). Also, care must be taken of horizontal edges.

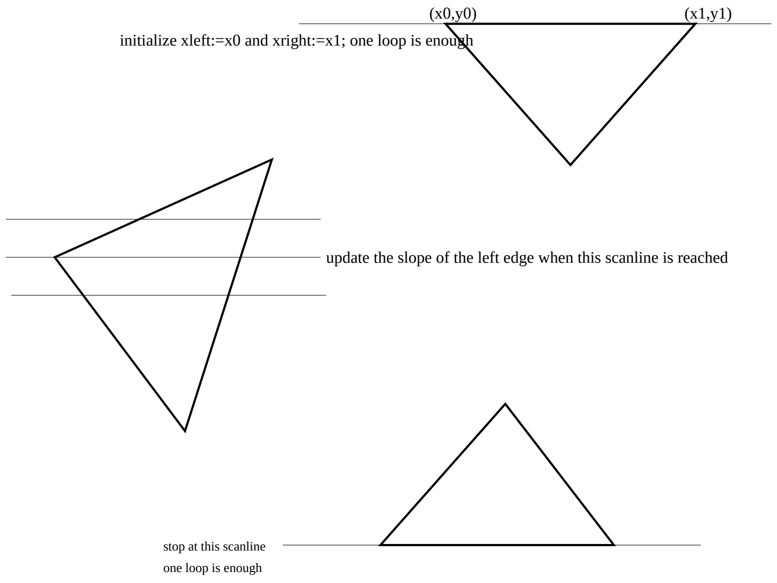

(x0,y0)
(x1,y1)
initialize xleft:=x0 and xright:=x1; one loop is enough
update the slope of the left edge when this scanline is reached
stop at this scanline
one loop is enough

Figure 3: Triangle ZOO: each triangle type has to be treated differently.

2 Linear interpolation while scan-conversion

A similar idea as that of adding a slope-related constant to update the intersection point applies to interpolated quantities. Linear functions change by a constant along any vector. Thus, in particular, if we precompute the change along horizontal and vertical unit vectors (Figure 4), we can use them to compute the value at a neighboring pixel from the available value at the current one.

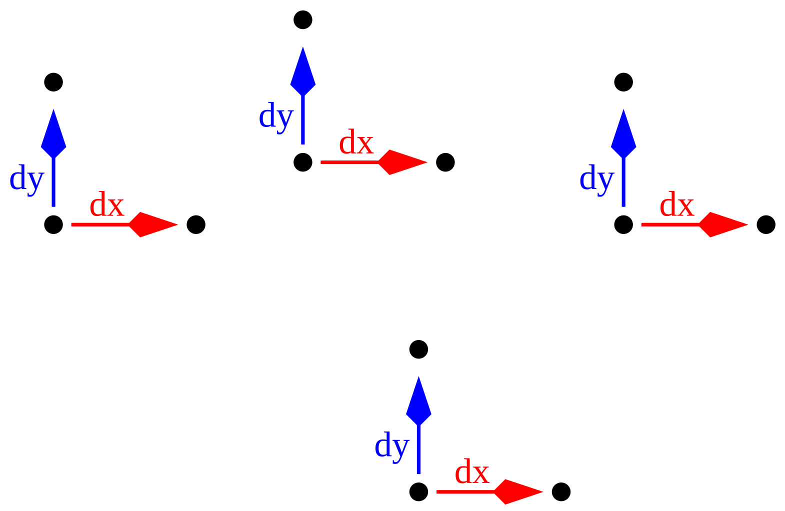


Figure 4: The amount of change of a linear function along the blue and red vectors is the same everywhere.