Directly Rasterizing Straight Line by Calculating the Intersection Point

Hua Zhang[1,2], Changqian Zhu[1], Qiang Zhao[2], and Hao Shen[2]

[1] Department of Computer and Communication Engineering,
Southwest Jiaotong University, Chengdu, Sichuan, 610031 P.R.China
cqzhu@home.swjtu.edu.cn
[2] Institute of Computer Applications, China Academy of Physics Engineering,
P.O. Box 919 Ext. 1201, Mianyang, Sichuan, 621900 P.R.China
{hzhang, zhaoq, shh}@caep.ac.cn

Abstract. In this paper, a method of line scan-conversion is presented by calculating the intersection point of straight line with the middle scan line of screen. With this method, the pixel's screen coordinate of spatial line could also be obtained. Moreover, raster precision is exactly the same as that of Bresenham's middle algorithm; and more than or at least one pixel could be obtained at a computational effort of one inner loop. However, to Bresenham's middle algorithm, only one pixel is obtained in one inner loop's calculating.

1 Introduction

Line scan-conversion is to raster a spatial straight line onto a 2D raster display. There is a famous and classical algorithm, named as Bresenham's middle algorithm [1]. In Bresenham's algorithm, the next point is obtained by checking the sign of error, and only integer computation is used in inner loop calculating. With regard to Bresenham's middle algorithm, there are many fast algorithms and implementations, such as in [2], [3], [4], [5], [6], [7], and [8]. Some of these methods emphasize on the distribution of line's direction; others require that the line drawn is not to be very short. Chen [9] et al. presented statistical results of actual applications for line scan-conversion. In those statistical results, about 38.58 percent of lines are not horizontal, vertical and diagonal; and about 87.79 percent of lines are not larger than 17 pixels. In this paper, a method independent of line direction and line length is presented. We also assume that the line's slope has the rang of 0 to 1, which is an assumption often made when discussing line scan-conversion. Other slope's line can be handled by reflections about the principal axes.

2 Line Rasterization

In fig. 1, for a line's slope with the range of 0 to 1(when the slope is equal to zero, this is a special case that will be discussed in following section), two regions could be obtained by subdividing the first octant with line 2, which has the slope of half one. Our algorithm of line rasterization in each region could be represented as following steps.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3516, pp. 826 – 829, 2005.

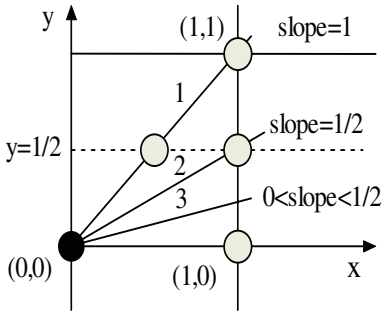

Fig. 1. Basis of line rasterization

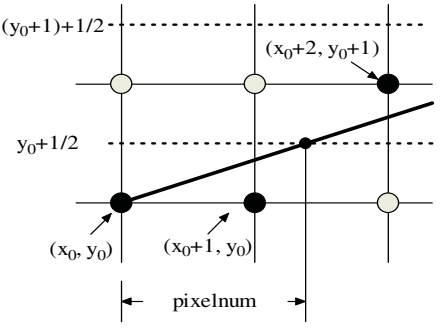

Fig. 2. An example of line rasterization

For a fixed two different end points with coordinates of (***x0***, ***y0***) and (***xn***, ***yn***), the line equation could be obtained, as in:

$$y = \frac{yn - y0}{xn - x0} x + (y0 - \frac{yn - y0}{xn - x0} x0) \tag{1}$$

Coordinates of intersection point between the line and ***y=y+1/2*** could be obtained by submitting ***y=y+1/2*** to the upper line equation. Now, submitting ***y=y+1/2*** into (1) and rearranging terms, the ***x*** component of intersection point could be represented as following (see fig. 2):

$$x = \frac{(xn - x0)(2(y - y0) + 1) + 2(yn - y0)x0}{2(yn - y0)} \tag{2}$$

Assume ***x*** coordinates of current start pixel obtained is ***xc***. Let the integer variable ***pixelnum*** be the distance of ***xc*** and ***x*** (see fig. 2). Now, the distance of ***x*** and ***xc*** is as following:

$$pixelnum = x - xc$$

Now, ***pixelnum*** pixel(s) could be written by increasing in ***x*** direction. For example, in fig. 1, line 1, 2, and 3 has ***pixelnum*** of 0, 1, and 2, respectively. In line 1, there is no pixel is written in ***x*** direction from the start pixel. In line 2, there is one pixel (1, 0) is written in ***x*** direction from the start pixel. In line 3, there are two pixels, (1, 0) and (2, 0), written in ***x*** direction from start pixel. Note when ***x*** coordinates exceed ***xn***, writing pixels is completed.

When the first loop is completed, point (1, 1) is selected as a new start point for line 1, because vertical distance from (1, 1) to line 1 is not greater than vertical distance from (1, 0) to line 1. Similarly, point (2, 1) and (3, 1) is selected as the new start point for line 2 and line3, respectively.

Repeating the upward processes until the line reach the end of line, the process of line rasterization is completed. The **C** code could be represented as following:

```
//The line end points are (x0,y0) and (xn,yn), assumed not equal. The slope of this line has
//the rangeof 0 to 1, except 0.

LineRasterization (X0, y0, xn, yn){
  x=x0;y=y0;deltax=xn-x0;deltay=yn-y0;
  Pixelnum=0;pixelseq=0;
  While(x<=deltax){
     pixelnum=( deltax*(2*(y-y0)+1)+
              2*x0*deltay ) /(2*deltay) - x;
     For(pixelseq=0;pixelseq<=pixenum;x++){
        if(x>deltax) return ;
        writepixel(x,y);
        pixelseq++;
     }
     y++;
  }
}
```

2.1 Example

Consider line from (0, 1) to (6, 4). Rastering this line yields initial calculations: x=0, y=1, *deltax*=6, *deltay*=3, *pixelnum*=0, and *pixelseq*=0.

After running through the main loop and inner loop, coordinates could be shown in table 1.

Table 1. Coordinates of Rasterization

x	*pixelnum*	*writepixel*	Bresenham's results
0	1	(0, 1)	(0, 1)
		(1, 1)	(1, 1)
2	1	(2, 2)	(2, 2)
		(3, 2)	(3, 2)
4	1	(4, 3)	(4, 3)
		(5, 3)	(5, 3)
6	1	(6, 4)	(6, 4)

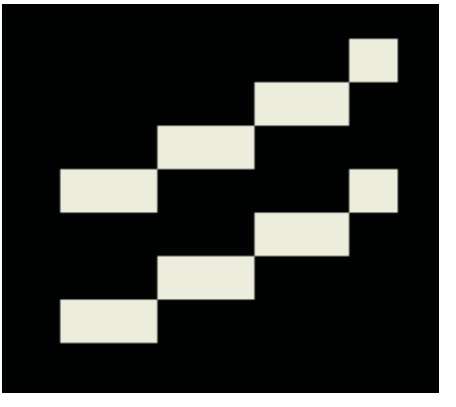

Fig. 3. Result of Rasterization

From these results, ***x*** is increased in each main loop; two pixels are written in each inner loop; and the coordinates of our algorithm for rasterization are the same as those of Bresenham's middle algorithm. Fig. 3 is a rasterized result for both Bresenham's algorithm and our algorithm. Note that the upper line has coordinates of (0, 4) and (6, 7), which are translated 3 pixels in ***y*** direction, and every pixel is represented by

20×20 pixels on screen. The upper line is rasterized by Bresenham's middle algorithm. The lower line is rasterized by our method. From fig. 3, same rasterized result could be noticed.

2.2 Slope Equal to Zero

When the slope of line is equal to zero, that is to say, $yn - y0 = 0$, the upper algorithm could not be used to raster the line. In this case, the line is parallel to horizontal axis. Therefore, the line could be rasterized from $x0$ to xn only with the increment of **1** in x direction(y is constant in the rasterization process).

3 Conclusions

A method of line scan-conversion is presented by directly calculating the intersection point of straight line with the middle scan line of screen. From this method, the coordinates of actual pixels on screen could be obtained. Moreover, the rasterized result is exactly the same as that of Bresenham's algorithm, and at least one pixel could be obtained in one inner loop's running. How fast about our method in actual applications is a further work, which we propose for future work.

References

1. Bresenham, J.E.: Algorithm for Computer Control and Digital Plotter. IBM Syst. J. Vol. 4, No.1, April (1965) 25-30
2. Dorst, L., Semulders, A.W.M.: Discrete Representation of Straight Lines. IEEE Trans. on Patter Analysis and Machine Intelligence, Vol. PAMI-6, No. 4, July (1984) 450-463
3. Foley, J.D., van Dam, A., Feiner, S.K., Hughes, J.F.: Computer Graphics: Principles and Practic*e*. 2nd edn. In C, Addition-Wesley (1996) 72-80
4. Rokne, J.G., Wu, X.: Fast Line Scan-conversion. ACM Trans. on Graphics, Vol. 9, No. 4, Oct. (1990) 370-388.
5. Wu, X., Ronkne, J.G.: Double-step Incremental Generation of Lines and Circles. Computer Vision, Graphics and Image Processing, Vol. 37, No. 3, March (1987) 331-344
6. Suenaga, Y., Kamae, T., Kabayashi, T.: A High-speed Algorithm for Generation of Straight Lines and Circular Arcs. IEEE Trans. Computers, Vol. C-28, No. 10, Oct. (1979) 728-736
7. Gill, G.W.: N-step Incremental Straight-Line Algorithm. IEEE Computer Graphics and Applications, Vol. 5, No. 3, May (1994) 66-72
8. Brons, R.: Linguistic Methods for the Description of a Straight Line on a Grid. Computer Graphics and Image Process, Vol. 9 (1979) 183-195
9. Chen, J.X., Wang, X., Bresenham J.E.: The Analysis and Statistic of Line Distribution. IEEE Computer Graphics and Applications, Vol. 22, No. 6 (2002) 100-107