# LINE DRAWING ALGORITHM: A COMPARATIVE ANALYSIS OF DDA, BRESENHAM, AND WU ALGORITHMS

**Zharnasek D.A.**, 2nd year student,
**Moyseyonok N. S.**, senior lecturer,
Belarussian State University, Minsk, Belarus

**Abstract**: The report discusses some of the most popular methods of this process, namely line rasterisation using the DDA-line algorithm, Bresenham's line algorithm and the Xiaolin Wu's line algorithm as examples. The text describes the characteristics of each of the methods and gives their action structures in a step-by-step format. It shows that Bresenham's algorithm remains the standard due to its speed and agility, while Xiaolin Wu's algorithm is the basic method when antialiasing is required. The main purpose of the article is to describe in simple words complex computer graphics algorithms, without which it is difficult to imagine simple things: from basic graphic editors to GPS maps.

**Key words:** algorithm, rasterization, digital differential analyzer, Bresenham's line algorithm, Xiaolin Wu's line algorithm.

The process of converting mathematical equations into graphic images has become a basis for different applications from our daily life. Rasterization is used for rendering a text character, drawing digital pictures, as well as creating navigational maps. It is a vital part that connects vector graphics and pixel-based displays, which is based on efficient algorithms.

Every day, without noticing, we encounter graphics. For example, when printing text, each letter is a separate object. The computer models the scene and then generates a final image that can be printed or displayed on a monitor. It is logical that first we work with vector graphics and then with raster graphics, i.e. with pixels. Methods of image conversion from vector graphics to raster (further rasterization) is a very important algorithm when working with computer graphics.

As with other methods, it is easier to start with a simple case. In the case of rasterization, it is the rasterization of a line. When working with horizontal, vertical and 45° inclined lines, the choice of raster elements is obvious. Otherwise, selecting the right pixels is already a complex and important task.

For further understanding of the algorithms, let us consider some notations and concepts. Let a segment have a beginning and an end (points $A(x_A, y_A)$ and $B(x_B, y_B)$). We will say that the segment belongs to the $I$-octant angle if vector $\overrightarrow{AB}$ belongs to this octant.

To unite a set of pixels into a single set, the concept of connectivity is used to determine which pixels are neighboring pixels. Two pixels are called 4-connected if $|x_2 - x_1| + |y_2 - y_1| \leq 1$ is satisfied for pixels $(x_1, y_1)$, $(x_2, y_2)$, $x_1, x_2, y_1, y_2 \in \mathbb{Z}$, and 8-connected if for the same pixels, two inequalities are satisfied: $|x_2 - x_1| \leq 1, |y_2 - y_1| \leq 1$.

DDA-line algorithm is one of the first segment rasterization algorithms proposed in the 1960s. The abbreviation DDA stands for digital differential analyzer. The algorithm always builds an 8-connected segment.

Let us consider a non-symmetric variant of this algorithm [1]. It consists of the following actions:

1. Round the end points. Introduce the following notations: $x_{begin} = [x_A], y_{begin} = [y_A]$ and $x_{end} = [x_B], y_{end} = [y_B]$.

2. Calculate the number of steps $L = \max\{|x_{end} - x_{begin}|, |y_{end} - y_{begin}|\}$. The number of pixels will be equal to $L + 1$.

3. If $L = 0$, then begin and end of line match. Return point $(x_{begin}, y_{begin})$ and end algorithm.

4. If $L > 0$. Create two secondary variables $x, y \in \mathbb{R}$. Primarily, they equal $x_A$ and $y_A$ respectively.

5. At each $i^{th}$ step, $i = \overline{2, L+1}$, change the value of secondary variables: $x := x + (x_B - x_A)/L,\ y := y + (y_B - y_A)/L$.

6. Coordinate of the $i^{th}$ pixel will be equal: $(x_i, x_i) = ([x + (x_B - x_A)/2], [y + (y_B - y_A)/2])$ , the value of the variable x is taken at step $i$.

It is not difficult to see that the result of the algorithm will be a set of points of the form:

$(x_i, y_i) = ([x_A + (i-1)(x_B - x_A)/L],\ [y_A + (i-1)(y_B - y_A)/L]),\ i = \overline{1, L+1}.$

In this approach, the points $(x_1, y_1)$ and $(x_{L+1}, y_{L+1})$ match $(x_{begin}, y_{begin})$ and $(x_{end}, y_{end})$ respectively.

There is also another variation of the algorithm, the symmetric one. Unlike its non-symmetric "sibling" (Figure 1), a different formula for auxiliary variables is used:

$x \coloneqq x + (x_{end} - x_{begin})/L\,,\ y := y + (y_{end} - y_{begin})/L.$

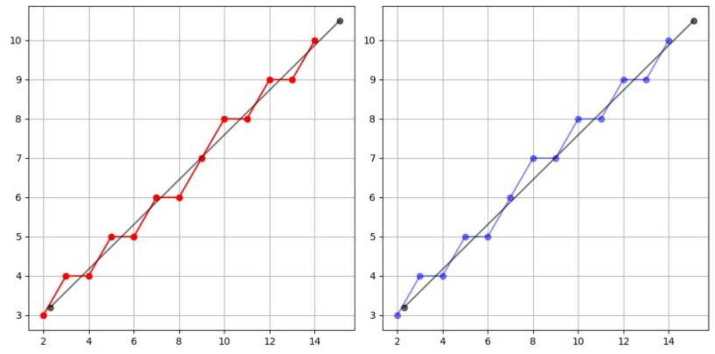

Figure 1 – symmetric (red) and non-symmetric (blue) DDA-line algorithms

The DDA-line algorithm, although it is simple, is not currently used in modern computer graphics because of its low performance. It is an illustrative example of a rasterization algorithm.

As mentioned above, the DDA-line algorithm is not optimal. This is due to the work with non-integer numbers. To solve this problem in 1962 Jack Elton Bresenham invented another algorithm. The main idea is to calculate the ordinate, the pixels closest to the original segment. The algorithm works for segments that are in the 1st octant angle, otherwise it is necessary to make the appropriate transformation to move the segment to the desired octant angle.

The algorithm consists of the following steps [2]:

1. Let the beginning of the line be located at the origin and the end be $(x_B, y_B), x_B, y_B > 0$. Mark the origin as the first pixel.

2. Introduce secondary variables: $i \coloneqq 1, d_1 = 2y_B - x_B$.

3. For the $(i+1)^{th}$ pixel, calculate $d_i$ using the formula:

$$d_i = \begin{cases} d_{i-1} + 2(y_B - x_B), & d_{i-1} \geq 0, \\ d_{i-1} + 2y_B, & d_{i-1} < 0. \end{cases}$$

4. Calculate the ordinate increment of the current pixel:

$$\Delta y_i = \begin{cases} 1, & d_{i-1} \geq 0, \\ 0, & d_{i-1} < 0. \end{cases}$$

5. Calculate the coordinate of the next pixel: $(x_{i+1}, y_{i+1}) = (x_i + 1, y_i + \Delta y_i)$.

6. If $i = x_B$, then the algorithm completes its work by returning the set of received pixels. Otherwise $i \coloneqq i + 1$ and go back to step 3rd.

The algorithm results in an 8-connected segment, since the increment does not exceed one ($|x_2 - x_1| \leq 1, |y_2 - y_1| \leq 1$).

It is also worth noting that in case the initial coordinates are not integers, they can be pre-rounded and then we can start the algorithm.

This algorithm is still used half a century later due to its simplicity and speed. Bresenham's line algorithm has influenced the work of modern video cards, as well as the appearance of new rasterization algorithms. One of the algorithms similar in time to Bresenham's algorithm is the Xiaolin Wu's line algorithm (further Wu's line algorithm) for rasterization with antialiasing.

To understand how rasterization with antialiasing works, we need to introduce some additional definitions. Grayscale is a real number $g(x, t)$ from [0; 1] or integer number from $[0; M - 1]$. The intensity of a pixel $(x, y)$ is the level of influence of some object on its colouring. Refer the intensity by $I(x, y)$, and $I$ takes a real value in the range from [0 ; 1]. Then the grayscale for each pixel can be found as: $g(x, y) = (i - 1)g_{bg} + g_0$. Assuming that $g_0$ (segment colour) is white, and $g_{bg}$ (background colour) is the darkest pixel or the smallest number, get that $g(x, y) = (i - 1)g_{bg}$.

The Wu's line algorithm provides a simple and efficient means of smoothing lines and curves based on visual error correction. This approach, although based on visual representation, is equivalent to convolution of a rectangular filter over two pixels located between lines, where the original signal has support in the 'true' pixel for each line.

Wu's line algorithm for rasterizing a black segment sound the following way [3]:

1. Let the beginning of the line be located at the origin and the end be $(x_B, y_B), x_B, y_B > 0$. Mark the origin as the first pixel.

2. The coefficient $N$ of raster grid splitting and the number of grey shades $M$ are determined.

3. Introduce an iteration variable $D \in \mathbb{N}_0$. The starting value is D:=0. Each step will increase it by $d = [Ny_B/x_B]$ .

4. Introduce secondary variables $x_1, x_2, y_1, y_2 \in \mathbb{N}_0$. They will store respectively the abscissa and ordinate of the current start point and the abscissa and ordinate of the current end point. Assign values to variables: $x_1 \coloneqq 0, x_2 \coloneqq x_B, y_1 \coloneqq 0, y_2 \coloneqq y_B$.

5. The pixels at the beginning and end of the segment are black: $g(0,0) = g(x_B, y_B) = 0$.

6. The abscissas of the current start and end are approaching each other, i.e. $x_1 \coloneqq x_1 + 1, x_2 \coloneqq x_2 - 1$. If after this operation it turns out that $x_1 > x_2$, terminate the algorithm.

7. $D \coloneqq D + d$.

8. If $D \geq N$, then the ordinates of the current start and end points converge: $y_1 \coloneqq y_1 + 1, y_2 \coloneqq y_2 - 1$, and the iteration variable itself is redefined: $D \coloneqq D - N$.

9. The colour of the new current end and start points is determined: $g(x_1, y_1) = g(x_2, y_2) = [DM/N]$. Also define two adjacent pixels: $g(x_1, y_1 + 1) = g(x_2, y_2 - 1) = M - 1 - g(x_1, y_1)$.

10. Return to step 6th.

This algorithm is widely used in various fields of graphic visualisation due to its ability to produce smooth and visually pleasing lines. In graphic editors such as Paint, Adobe Photoshop and Inkscape, the Wu’s algorithm allows you to achieve high quality rendering while minimising the effect of pixelation. This is especially important when working with slanted lines and fine details, where antialiasing makes the image look more natural and professional. This makes such programmes indispensable for designers, artists and users involved in image processing.

Line drawing algorithms play an important role in computer graphics and various programming areas. The considered algorithms, such as the DDA-line algorithm, Bresenham's line algorithm and Wu's line algorithm, demonstrate different approaches to solve the introduced problem. They differ in their computational efficiency, ease of implementation and quality of the result obtained. Bresenham's algorithm deserves special attention due to its simplicity, speed and lack of floating-point operations, which makes it standard for raster graphics tasks. Wu's line algorithm, based on Bresenham's, helps in antialiasing work without compromising on speed and simplicity.

The study of algorithms for drawing line helps to understand the basic principles of graphical systems and lays the foundation for more complex operations, such as the construction of figures and the realisation of three-dimensional objects.

**References:**

1. DDA Line generation Algorithm in Computer Graphics [Electronic resource]. – Mode of access: https://www.geeksforgeeks.org/dda-line-generation-algorithm-computer-graphics. – Date of access: 18.11.2024.

2. Flanagan C. The Bresenham Line-Drawing Algorithm [Electronic resource] / C. Flanagan. – Mode of access: https://www.cs.helsinki.fi/group/goa/mallinnus/lines/bresenh.html. – Date of access: 18.11.2024.

3. Wu, X. An efficient antialiasing technique / X. Wu // Computer graphics. – 1991. – Vol. 25, No. 4. – p. 143–152. – DOI: 10.1145/127719.122734.