

Chapter – 2 Line Generation

2.1 Geometry of Line

Any two points specified in the Plane will define a line, which means that for defining the line we must have to specify two points.

In the figure 2.2, two points $P_1(x_1, y_1)$ and $P_2(x_2, y_2)$ define a line l_1 . To define the line we need an equation. A random point $P(x_i, y_i)$ is said to be on the line l_1 if and only if it satisfies the equation of the line. The equation of a straight line can be derived with the help of slope concept of a straight line. The **slope** is the rate at which an ordinate of a point on a plane changes with respect to a change in the horizontal coordinate. The notion slope is denoted by m . Thus,

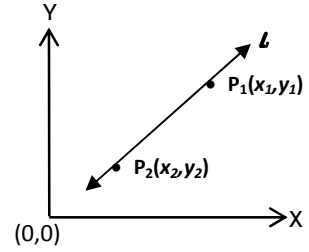


Fig:2.1- A Straight Line

$$m = \frac{\text{vertical distance between two points}}{\text{horizontal distance between two points}} \quad \therefore \quad m = \frac{y_2 - y_1}{x_2 - x_1}$$

So the slope of any line can be one of four types: the straight line 1) with positive slope, 2) with negative slope, 3) with zero slope and 4) with undefined slope.

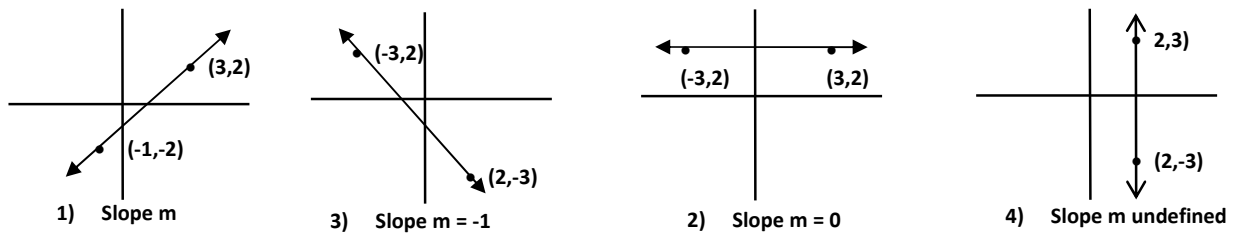


Fig:2.2 – Possible Slope values

As per previous discussion to derive a line equation the concept of slope is being used. The slope of any line between two points will remain unchanged (constant). According to this Let assume that $P_1(x_1, y_1)$ and $P_2(x_2, y_2)$ constitute a line as well as a point $P(x, y)$ is lying on the same line and the slope remain same at any point of the slope.

So the slope can be written as follows:

$$m = \frac{y_2 - y}{x_2 - x} \quad \text{and also} \quad m = \frac{y_2 - y_1}{x_2 - x_1}$$

$$\frac{y_2 - y}{x_2 - x} = \frac{y_2 - y_1}{x_2 - x_1}$$

$$\begin{aligned} \frac{y - y_1}{x - x_1} = m &\Rightarrow y - y_1 = m(x - x_1) \Rightarrow y = mx - mx_1 + y_1 \Rightarrow y \\ &= mx + (y_1 - mx_1) \end{aligned}$$

We can assume that $b = y_1 - mx_1$ then the equation of straight line can be given as $y = mx + b$ (1)

2.2 Frame buffer

Pixel (picture element) is the smallest addressable screen element. Each pixel has a name or address. The names correspond to the coordinates, which identify points. The display screen is considered as a grid (or array) of pixels. Each row and column can be numbered conveniently to represent the screen coordinates. The coordinate (i, j) gives the column and row of a pixel. Each pixel is centered at its coordinates. Line segments are drawn by setting the intensities (brightness) of a string of pixels between a starting pixel and an ending pixel. The maximum number of distinguishable points, which a line may have, depends on the resolution of the display device.

Frame buffer is an array (stored in the computer's memory), which contains an internal representation of the image to be displayed. It collects and stores the pixel values for use by the display device. The graphics display device accesses this array to determine the intensity at each pixel should be illuminated (displayed).

2.3 Line Generation (DDA) Algorithms

The digital differential analyzer (DDA) is a scan conversion line algorithm based on calculation either Dy or Dx . The line at unit intervals is one coordinate and determine corresponding integer values nearest line for the other coordinate. Consider first a line with positive slope.

2.3.1 VECGEN Algorithm

The concept of drawing a line or a curve, stepping along a row index (y) or a column index (x) to determine the corresponding column index or row index respectively is known as Digital Differential Analyzer (DDA). The line equation works well for gentle slope but when the slope is sharp the role of x and y are interchanged as well as slope m replaced by $1/m$. This method is known as VECGEN method.

Step: 1 If the slope is less than or equal to 1, the unit x intervals $dx=1$ and compute each successive y values.

$$\begin{aligned} dx &= 1 \\ m &= dy / dx \\ m &= (y_2 - y_1) / 1 \\ m &= (y_k + 1 - y_k) / 1 \\ y_{k+1} &= y_k + m \quad \text{----- (2)} \end{aligned}$$

Subscript k takes integer values starting from 1, for the first point and increment by 1 until the final end point is reached.

- m -> any real numbers between 0 and 1
- Calculate y values must be rounded to the nearest integer

Step: 2 If the slope is greater than 1, the roles of x and y at the unit y intervals $dy=1$ and compute each successive x values.

$$\begin{aligned} dy &= 1 \\ m &= dy / dx \\ m &= 1 / (x_2 - x_1) \\ m &= 1 / (x_{k+1} - x_k) \\ x_{k+1} &= x_k + (1 / m) \quad \text{----- (3)} \end{aligned}$$

- Equation 2 and Equation 3 that the lines are to be processed from left end point to the right end point.

Step: 3 If the processing is reversed, the starting point at the right

$$dx = -1$$

$$m = dy / dx$$

$$m = (y_2 - y_1) / -1$$

$$y_{k+1} = y_k - m \quad \text{-----} \quad (4)$$

Intervals $dy=1$ and compute each successive y values.

Step: 4 Here, $dy=-1$

$$m = dy / dx$$

$$m = -1 / (x_2 - x_1)$$

$$m = -1 / (x_{k+1} - x_k)$$

$$x_{k+1} = x_k + (1 / m) \quad \text{-----} \quad (5)$$

Equation 2 and Equation 5 used to calculate pixel position along a line with -ve slope.

This algorithm can also be written in following form:

Step-1: Input two line end points $A(x_a, y_a)$ and $B(x_b, y_b)$ and start with the left endpoint

$A(x_a, y_a)$ and declare Δx , Δy , step, k , x_{inc} , y_{inc} , x , y .

Step-2: Calculate $\Delta x = x_b - x_a$; $\Delta y = y_b - y_a$ [find the difference of end points]

Step-3: If $abs(\Delta x) > abs(\Delta y)$ then

$$\text{step} = abs(\Delta x)$$

else

$$\text{step} = abs(\Delta y)$$

Step-4: We have $x_{inc} = \Delta x / \text{steps}$ and $y_{inc} = \Delta y / \text{steps}$ [Find total intermediate points]

Step-5: $\text{setPixel}(\text{round}(x_a), \text{round}(x_a), 1)$ [Plot the first point]

Step-6: $x = x + x_{inc}$; $y = y + y_{inc}$; $\text{setPixel}(\text{round}(x), \text{round}(y), 1)$ [Plot all points]

Step-7: Repeat step-6 for step times

Advantage:

- 1) It is a faster method for calculating pixel position than simple line algorithm.
- 2) It eliminates continuous multiplication in equation " $Y=mx+b$ " so that appropriate increments are applied on x or y .

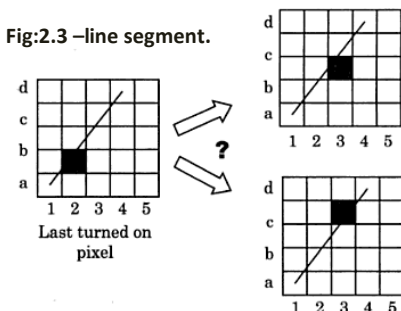
Disadvantage:

- 1) Round off error in successive additions can cause the pixel positions to drift away from the true line path for long line segments.
- 2) The rounding operations and floating point arithmetic are still time consuming.

2.3.2 Bresenham's Algorithm

The line generating algorithm developed by **Bresenham** is accurate and faster; which converts scan lines using incremental integer calculations that can be used to draw not only lines but also used to display circles and other curves.

Fig:2.3 –line segment.



As we can see in fig:2.3 last turned on pixel is $(2,b)$ and the problem is which pixel should be turned on next, $(3,c)$ or $(3,d)$? According to the Bresenham's algorithm the pixel closest to the line to be drawn will be chosen. For this he has defined a decision parameter, so let understand what is this decision parameter?

Let us assume that we are drawing a line $y = mx + b$ that passes through a point (x_0, y_0) . Here $0 < m < 1$. Let us also assume that the last pixel turned on is (x_k, y_k) and the decision to be made is for the next step that is for the vertical distance $x_k + 1$ as in fig:2.4.

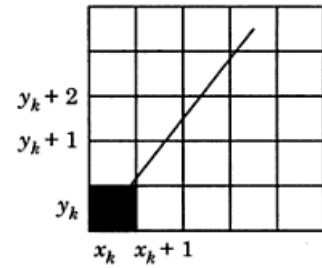


Fig:2.4 – Drawing a line.

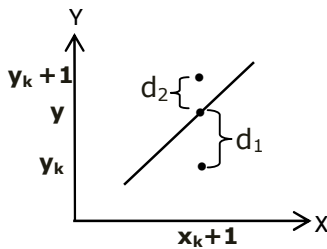


Fig:2.5 – Distance between Points

Now let us assume a vertical row of pixels which passes through horizontal distance $x_k + 1$. There are three vertical points, $(x_k + 1, y_k)$, $(x_k + 1, y)$ and $(x_k + 1, y_k + 1)$, fall on this assumed line. In addition the assumed distance between $(x_k + 1, y_k)$ and $(x_k + 1, y)$ is d_1 and distance between $(x_k + 1, y)$ and $(x_k + 1, y_k + 1)$ is d_2 in above figure.

The equation of this vertical line is,

$$y = m \cdot (x_k + 1) + b$$

Therefore,

$$d_1 = y - y_k = m \cdot (x_k + 1) + b - y_k$$

$$d_2 = y_k + 1 - y$$

$$= y_k + 1 - (m \cdot (x_k + 1) + b)$$

$$= y_k + 1 - m \cdot (x_k + 1) - b$$

The difference among these two points can be obtained as $d_1 - d_2$,

$$d_1 - d_2 = [m \cdot (x_k + 1) + b - y_k] - [y_k + 1 - m \cdot (x_k + 1) - b]$$

$$= m \cdot (x_k) + m + b - y_k - y_k - 1 + m \cdot (x_k) + m + b$$

$$= 2m \cdot (x_k) + 2m + 2b - 2y_k - 1$$

$$= 2m \cdot (x_k + 1) + 2b - 2y_k - 1$$

So,

$$d_1 - d_2 = 2m \cdot (x_k + 1) + 2b - 2y_k - 1$$

$$= 2 \frac{dy}{dx} \cdot (x_k + 1) + 2b - 2y_k - 1 \quad [\text{after substituting } m = \frac{dy}{dx}]$$

$$dx (d_1 - d_2) = 2 dy \cdot (x_k + 1) + 2b \cdot dx - 2y_k \cdot dx - 1 \cdot dx$$

$$= 2 dy \cdot x_k + 2 dy + 2b \cdot dx - 2y_k \cdot dx - dx$$

$$P_k = 2 dy \cdot x_k - 2y_k \cdot dx + c \quad \dots\dots\dots [2]$$

Where $c = 2 dy + 2b \cdot dx - dx$ and it is independent of the positions of pixel, and P_k is a decision parameter for the k^{th} step,

The sign of decision parameter is same as the sign of $d_1 - d_2$, as for our case $0 < m < 1$, that is $dx > 0$.

If P_k is positive, that is $dx(d_1 - d_2) > 0$, and $d_1 > d_2$. Therefore, the upper pixel at position $y_k + 1$ is closer to the line than the pixel y_k , hence the pixel at position $y_k + 1$ will be activated. In case of negative P_k , that is $dx(d_1 - d_2) < 0$, ($d_2 > d_1$) and therefore the pixel at position y_k will be activated.

Now for the $(k + 1)^{\text{st}}$ step,

$$P_{k+1} = 2 dy \cdot x_{k+1} - 2y_{k+1} \cdot dx + c$$

$$P_{k+1} - P_k = [2 dy \cdot x_{k+1} - 2y_{k+1} \cdot dx + c] - [2 dy \cdot x_k - 2y_k \cdot dx + c]$$

$$\begin{aligned}
 &= 2 dy \cdot x_{k+1} - 2y_{k+1} \cdot dx + c - 2 dy \cdot x_k + 2y_k \cdot dx - c \\
 &= 2 dy \cdot (x_{k+1} - x_k) - 2dx (y_{k+1} - 2y_k) \\
 &= 2 dy - 2dx (y_{k+1} - 2y_k) \quad [\text{since } x_{k+1} - x_k = 1] \\
 P_{k+1} - P_k &= 2 dy - 2dx (y_{k+1} - 2y_k) \\
 P_{k+1} &= P_k + 2 dy - 2dx (y_{k+1} - 2y_k)
 \end{aligned}$$

Where the value of $(y_{k+1} - y_k)$ will be 0 or 1 respectively for positive and negative value of decision parameter P_k . From equation [2], the initial value of the decision parameter is given as:

$$P_k = 2 (dy - dx) \dots (3)$$

Here, it is noticeable that for making a decision regarding the pixel to be activated, no floating point operation is needed. Other integer coordinate values of the next point in line can be obtained by the following algorithm and so the line is created.

Bresenham's Line-Drawing Algorithm for $|m| < 1$

Step 1: Input the two line endpoints and store the left endpoint in (x_0, y_0)

Step 2: Load (x_0, y_0) into the frame buffer; that is, plot the first point.

Step 3: Calculate constants dx , dy , $2dy$, and $2dy - 2dx$, and obtain the starting value for the decision parameter as

$$P_0 = 2dy - dx$$

Step 4: At each x_k along the line, starting at $k = 0$, perform the following test:

If $P_k < 0$, the next point to plot is $(x_k + 1, y_k)$ and

$$P_{k+1} = P_k + 2dy$$

Otherwise, the next point to plot is $(x_k + 1, y_k + 1)$

$$P_{k+1} = P_k + 2dy - 2dx$$

Step 5: Repeat step 4 dx no. of times.

2.4 Line Styles

Generally, solid lines, dashed lines, dotted lines and thick lines are used as different line styles in graphics applications. Dashed lines and dotted lines are generated by just a minor modification in our standard DDA algorithms.

A dashed line can be generated by inter-dash spaces between the dashes. Generally, dashes and inter-dash spaces are equal in length whereas a dotted line can be generated by very short length dashes. By using different characters in this method, different line styles can be created.

2.4.1 Thick Lines

The thick line primitive is more interesting. A line or a line segment generated by the standard DDA algorithm is a single pixel width. Let us assume a line with a gentle slope, that is, a slope less than 1. If you need such a line segment with a thickness of three pixels, you need to put there a vertical span of three pixels for each value of x . Thus, a line with any width w , can be generated just by putting a vertical span of $(w/2)$ pixels on both the sides of the central line as shown in fig:2.6

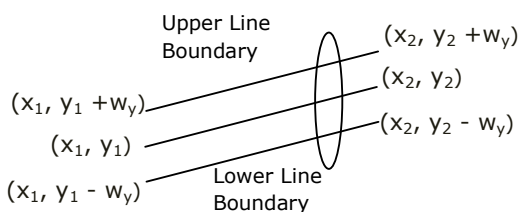
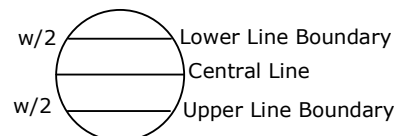


Fig. 2.6- Thick Line Generation



Let $p_1(x_1, y_1)$ and $p_2(x_2, y_2)$ be the end points of a line segment with width w , which is to be drawn. Here two parallel line segments will be drawn from a central line (x_1, y_1) and (x_2, y_2) at a distance $(w/2)$.

As shown in fig:, the coordinates of end points of the upper and lower line boundaries are $[(x_1, y_1 + w_y), (x_2, y_2 + w_y)]$ and $[(x_1, y_1 - w_y), (x_2, y_2 - w_y)]$ respectively. Where the w_y is as below:

$$w_y = \frac{w-1}{2}$$

In the above equation, $w-1$ is taken instead of w . the reason being, on both sides the line boundaries are drawn which are of 1 pixel in length. Therefore, from both the sides half of the pixel width will be deducted to get the exact width w . for a line segment with steep slope, that is slope greater than 1, the role of x and y in equation () will be interchanged. Thus, for steep slope, the value of w_y will be

$$w_y = \frac{w-1}{2} \cdot \frac{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}}{|x_2 - x_1|} \dots\dots (4)$$

In the above equation, $w - 1$ is taken instead of w . The reason is we are considering the boundaries which are drawn on both the sides are of 1 pixel in length. So from both side 1 deducted to get its original width w . When the slope is steep that is greater than 1, the role of x and y in the above equation will get change and the equation would be written as follows:

$$w_x = \frac{w-1}{2} \cdot \frac{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}}{|y_2 - y_1|} \dots\dots (5)$$

In this case, pixels with horizontal span are activated simultaneously.

2.4.2 Line Caps

When we construct the thick lines the issue is generated for its ends whether it should be vertical or horizontal. To clear this issue line caps can be used. There are three basic techniques or caps which are used in graphics applications: butt cap, round cap, and projecting square cap. (See in fig.2.7)

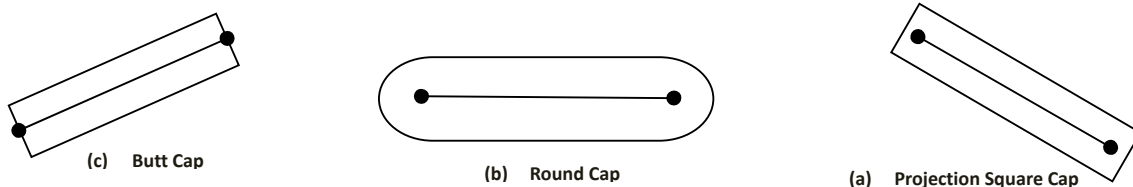


Fig.2.7 - Line Caps

In the butt cap the end positions of the thick lines are adjusted in such a way that the line segment appears with square ends. The ends of such thick lines are perpendicular to the original line segment.

In the round cap, the upper and lower line boundaries are joined by a circular arch with a semi-circle of the diameter of the width of the thick line segment.

In the projecting square cap, the portion of line segment is just extended to give a line segment an effect of square.

2.4.3 Thick Line Segments

When two thick line segments are connected with each other, they create problems at the joining end points. It creates an angle at the joining end. Joint of two such lines is shown in the figure besides:

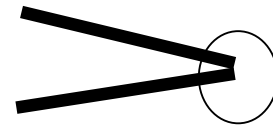


Fig.2.8 – Joint of two thick line segments

The discreteness that occurs while joining two thick lines can be removed from the poly-lines by adding some additional portion to the joint as some specific types of joining process. There are three joints: Miter Joint, Round Joint, Bevel Joint.



Fig.2.9 – types of line joints

The miter joint is one in which the exterior boundaries of two thick line segments are extended up to their intersection point – the point where these two line segments meet each other, if they are extended. (See figure 2.9)

In the round joint, the exterior boundaries are connected with a circular arch with a diameter of total thickness of two thick lines, which are joined. (See figure 2.9)

The bevel joint is one where butt cap is used to draw line segment and the little triangle that is created at the joint is filled with color of these two line segments. (See figure 2.9)

2.4.4 Pen Styles

There are many graphics applications, which make use of different pens and brushes of different shapes. Different color is also used with different shapes of pens and brushes. Some of the brushes and pens are shown in the following figure.

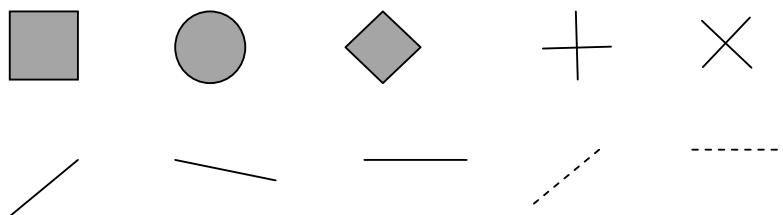


Fig.2.10 – Different brush shapes