include pulse width modulation (PWM), pulse density modulation (PDM), resistor ladder networks, and sigma-delta modulation.

- **Resolution:** DACs have a specified resolution, which represents the number of discrete levels or steps the converter can produce in the analog output signal. Higher-resolution DACs can provide more accurate and finely graded analog signals.
- **Sampling Rate:** DACs are often designed to work with specific sampling rates, which determine the rate at which the digital input signal is converted into an analog output signal. The sampling rate affects the maximum frequency that can be accurately reproduced by the DAC.

Solved Numerical Problems

1. A system with 24 bits per pixel and resolution of 1024 by 1024. Calculate the size of frame buffer (in Megabytes)

Solution:

Resolution = 1024 * 1024

Total number of pixel = 1024 * 1024 = 1048576 pixels

Bits per pixels' storage = 24 bits

Therefore, total storage required in frame buffer:

= 1048576 * 24

= 25165824 bits

=25165824/8 Byte

= 25165824/(8 × 1024) Kb

= 25165824/(8×1024*1024) Mb

= 3 Mb

2. How Many k bytes does a frame buffer needs in a 600 × 400 pixel?

Solution:

Suppose, n bits are required to store 1 pixel.

Then, the size of frame buffer = Resolution * bits per pixel

= (600 * 400) * n bits

= 240000 n bits

= 240000 n kb / (8 * 1024)

= 29.30 n k bytes.

3. Consider a RGB raster system is to be designed using 8 inch by 10-inch screen with a resolution of 100 pixels per inch in each direction. If we want to store 8 bits per pixel in the frame buffer, how much storage in bytes do we need for the frame buffer?

Solution:

Size of screen = 8 inch * 10 inch

Pixels Per inch (Resolution) = 100

Then total number of pixels = 8 * 100 * 10 * 100

= 800000 pixels

Bits per pixels' storage = 8

Therefore, total storage required in frame buffer

= 800000 * 8 bits

= 6400000 bits

= 6400000/8 bytes

= 800000 Bytes

4. Find out the aspect ratio of the raster system using 8 × 10 inches' screen and 100 pixels/inch.

Solution:

We know that, Aspect ratio = Width / Height

= (8 × 100) / (10* 100)

= 4 / 5

So, aspect ratio = 4: 5

5. What is the time required to display a pixel on the monitor of size 1024 * 768 with refresh rate of 60 Hz?

Solution:

Refresh Rate = 60Hz i.e. 60 frames per second

Total number of pixel in one frame = 1024 * 768 = 786432 pixels.

60 frames need 1 second

1 frame need $\frac{1}{60}$ second

i.e. 786432 pixels need $\frac{1}{60}$ second

Hence, 1 pixel need $\frac{1}{60 \times 786432}$ second

$= \frac{10^9}{60 \times 786432}$ ns = 21.19 ns

6. **If the total intensity achievable for a pixel is 256 and the screen resolution is 640 * 480. What will be the size of frame buffer?**

Solution:

1 pixel = 256 different intensity level

Resolution = 640 * 480

Let X be the number of bits required to represent 256 different intensity level

Then, $2^X = 256$

Therefore, X = 8 bits

Hence, number of bits required for the screen = 640 * 480 * 8

= 2457600 bits.

7. **If a pixel is accessed from the frame buffer with an average access time of 300ns then will this rate produce an un-flickering effect for the screen size of 640 * 480.**

Solution:

Size of screen = 640*480

Total Number of pixels = 640 * 480 = 307200

Average access time of one pixel = 300ns

Therefore, total time required to access entire pixels of image in the screen = 307200 * 300

= 92160000 ns

= 92160000/10^9

= 0.09216 seconds

i.e. 1 cycle take 0.09216 second

Now, number of cycles per second i.e. Refresh Rate =?

0.09216 Seconds = I cycle

1 second = $\frac{1}{0.09216}$

= 10.86

Therefore, refresh Rate= 10.86 cycles per second

Since the minimum refresh rate for unflicker image is 60 frames per second, hence we can say the monitor produces flickering effect.

8. **Consider a raster scan system having 12-inch by 10-inch screen with a resolution of 100 pixels per inch in each direction. If display controller of this system refresh the screen at the rate of 50 frames per second. How many pixels could be accessed per second and what is the access time per pixel of the system. (TU Exam)**

Solution:

Given,

Resolution = 12 × 10 (inch)

Pixel per inch = 100 ppi

Refresh rate = 50 fps

Total access time for a pixel = ?

Total pixels accessed in a sec = ?

Now,

Total no. of pixels in 1 frame = 12 × 100 × 10 × 100

= 1200000

Total no. of pixels in 50 frame = 1200000 × 50

= 60000000

50 frame needs 1 sec.

60000000 pixels needs 1 sec

∴ 1 pixels needs $\frac{1}{60000000}$ sec

$= \frac{1.666 \times 10^{-8}}{10^{-9}}$

= 16.67 /s

9. **Consider three different raster systems with resolutions of 640 × 480, 1280 × 1024 and 2560 × 2048. What size of Nome buffer (in bytes) is needed for each of these systems to store 12 bits per pixel ? How much storage is required for each system If 24 bits per pixel are to be stored? (UPTU MCA 2003)**

Solution:

Case 1: For 640×480,

Total number of pixel required for 640 × 480 resolution

= 640 × 480 pixels

So, size of frame-buffer required = 640 × 480 pixels

because 1 pixel can store 12 bits.

Therefore, size of frame buffer (in bits)

= 640 × 480 × 12 bit

$= \frac{640 \times 480 \times 12}{8}$ bytes (1 byte = 8 bit)

= 460800 bytes

= 450 KB (1024 bytes = 1 KB)

For 1280 × 1024: Size of frame buffer (in bits)

$$= 1280 \times 1024 \times 12 \text{ bits}$$

$$= \frac{1280 \times 1024 \times 12}{8} \text{ bytes}$$

$$= 1920 \text{ KB}$$

For 2560 × 2048:

$$\text{Size of frame-buffer} = \frac{256 \times 2048 \times 12}{8} \text{ bytes} = 7.5 \text{ MB}$$

Case II: When one pixel can store 24 bits then frame-buffer size will be twice.

$$\text{For } 640 \times 480, \text{ Frame-Buffer size} = \frac{640 \times 480 \times 24}{8} \text{ byte}$$

$$= 900 \text{ KB}$$

For 1280 × 1024, Frame-buffer size = 1920 × 2 KB

= 3840 kB = 3.75 MB

For 2560 × 2048, Frame-buffer size = 7.5 × 2 MB = 15 MB.

10. Consider a raster system with a resolution of 1024 × 1024. What is the size of the raster (in bytes) needed to storestire 4 bits per pixel ? How much storage is required if 8 bits per pixel are to be stored ? **(UPTU, B-Tech, 2002)**

Solution:

Resolution = 1024 × 1024

Case-1: When 1 pixel = 4 bits

Size of raster = 1024 × 1024 × 4 bits

$$= \frac{1024 \times 1024 \times 4}{8} \text{ bytes} \quad \text{(since, 1 byte = 8 bits)}$$

$$= 515 \times 1024 = 1 \times 2^{19} \text{ bytes}$$

Case-2: When 1 pixel = 8 bits

Size of raster = 1024 × 1024 × 8 bits

$$= \frac{1024 \times 1024 \times 8}{8} \text{ bytes}$$

$$= 1 \times 2^{20} \text{ bytes}$$

11. Consider two raster systems with resolution of 640 by 480 and 1280 by 1024. How many pixels could be accessed per second in each of these systems by a display controller that refreshes the screen at a rate of 60 frames per second ? What is the access time per pixel in each system?

Solution:

Case-1: Resolution = 640 × 480

No. of pixels in one frame = 690 × 480 = 307200

Since controller can access 60 frames in one second

Therefore, total no. of pixels accessed = 60 × 307200

= 18432000 per sec

$$\text{Access time/pixel} = \frac{1}{\text{Total pixels accessed per sec}}$$

$$= 5.4 \times 10^{-8} \text{ sec/pixel.}$$

Case-2: When resolution = 1280 × 1024

Total no. of pixels accessed by the raster system

$$= 60 \times 1280 \times 1024$$

$$= 78643200 \text{ per sec.}$$

Access time per pixel = $1178643200 = 1.2 \times 10^{-8}$ sec/pixel

12. A laser printer is capable of printing two pages (size 9 × 11 inch) per second at resolution of 600 pixels per inch. How may be bits per second does such device require ?

Solution:

Storage required per page (in pixel)

$$= 9 \times 11 \text{ inch/sec}$$

$$= 9 \times 11 \times 600 \times 600 \text{ pixel sec} \quad (\because 1 \text{ inch} = 600 \text{ pixels})$$

Since laser printer is capable printing two pages per second bits per second required by the printer.

$$= 2 \times .9 \times 11 \times 600 \times 600 \text{ pixels per second}$$

$$= 7.128 \times 10^7 \text{ pixel/sec}$$

$$= 7.128 \times 10^7 \text{ n bits/sec}$$

(Assume 1 pixel = n bits)

13. How many k bytes does a frame buffer need in a 800 × 400 pixel ?

Solution:

Given resolution is 600 × 400

Suppose 1 pixel can store n bits then the size of frame buffer

$$= \text{Resolution} \times \text{bits per pixel}$$

$$= (600 \times 400) \times n \text{ bits}$$

$$= 240000 \text{ n bits}$$

$$= \frac{240000 \text{ n}}{1024 \times 8} \text{ k bits} \quad (\because 1 \text{ kb} = 1024 \text{ bits})$$

$$= 29.30 \text{ n k bytes.}$$

14. Find out the aspect ratio of the raster system using 8 × 10 inches screen and 100 pixel/inch.

Solution:

We know that,

$$\text{Aspect ratio} = \frac{\text{width}}{\text{height}} = \frac{8 \times 100}{10 \times 100} = \frac{4}{5}$$

$\therefore$ Aspect ratio = 4 : 5

15. How much time is spent scanning acres each row of pixels during screen refresh on a raster system with resolution of 1280 × 1024 and a refresh rate of 60 frames per second?

Solution:

Here, resolution, 1280 × 1024

That means system contains 1024 scan lines and each scan line contains of 1280 pixels and refresh rate = 60 frames/sec.

that means 1 frame takes $\frac{1}{60}$ sec

Since, resolution = 1280 × 1024

$\therefore$ 1 frame consists of 1024 scan lines.

$\therefore$ 1024 scan lines takes to $\frac{1}{60}$ sec

$\therefore$ 1 scan lines take $\frac{1}{60 \times 1024}$ sec = 0.058 sec or 58 m sec.

16. Suppose RGB raster system is to be designed using on 8 inch × 10 inch screen with a resolution of 100 pixels per inch in each direction. If we want to store 6 bits per pixel in the frame buffer, how much storage (in bytes) do we need for frame buffer?

Solution:

Here, resolution = 8 inch × 10 inch

First we convert it in pixel, then

resolution = 8 × 100 by 10 × 100 pixel = 800 × 1000 pixel

1 pixel can store 6 bits

so, frame buffer size required = 800 × 1000 × 6 bits

$$= \frac{800 \times 1000 \times 6}{8} \text{ bytes}$$

$$= 6 \times 10^5 \text{ bytes}$$

17. How lung would it take to loud a 640 × 480 frame buffer with 12 bits per pixel, if 10^5 bits can be transferred per second? How long would it take to loud a 24 bits per pixel frame buffer with a resolutions of 1280 by 1024 using this same transfer rate?

Solution:

Frame buffer size = 640 × 480 pixels

= 640 × 480 × 12 bits [1 pixel = 12 bits]

Since 10^5 bits takes 1 second

So, 640 × 480 × 12 bits takes $\frac{640 \times 480 \times 12}{10^5}$ = 36.864 sec

18. Consider a raster scan system having 12 inch by 10 inch screen with a resolution of 100 pixels per inch in each direction. If the display controller of this system refreshes the screen at the rate of 50 frames per second, how many pixels could be accessed per second and what is the access time per pixel of the system? [2071 Shrawan]

Solution:

Total pixels = 12 × 100 × 10 × 100

Refresh rate = 50 frames per second

Pixels accessed per second (f) = 12 × 100 × 10 × 100 × 50

= 60000000

Access time per pixel = $\frac{1}{f}$

= 1.667×10^{-8} second

Access time per pixel is 1.667×10^{-8} second

19. Calculate the frame buffer size (in KB) for a raster system recording a video for 1 min with resolution of 1280 × 1024, and storing 24 bits per pixel with a refresh rate of 25 fps. [2076 Ashwin]

Solution:

Screen resolution =1280 × 1024

Refresh rate = 25 fps

Bit required to represent a pixel = 24 bits

Memory required just for a frame = 1280 × 1024 × 24 bits

Memory required for 1 second =1280 × 1024 × 24 × 25 bits

Memory required for recording a video for 1 min is

=1280 × 1024 × 25 × 24 × 60 bits

= 5760,0000 KB

20. Calculate the total memory required to store a 10 minutes video in a SVGA system with 24 bit true color and 60 fps refresh rate [2078 Kartik]

Solution:

The resolution of SVGA system = 800 × 600

Bit required to represent a pixel = 24 bit

Refresh rate = 60 fps

Memory required for one frame = 800 × 600 × 24

Memory required for 1 second = 800 × 600 × 24 × 60 bit

Memory required for recording a video for 10 min is

= 800 × 600 × 24 × 60 × 10 × 60 bit

= 50,625,000 KB

21. Let the resolution of screen is 1024 × 512. What is the memory captured by the frame buffer that uses primary color for display? [2078 Chaitra]

Solution:

The resolution of screen = 1024 × 512

For primary color to display, let the color depth = 24 bit

Size of frame buffer for 1 pixel = 24 bit

For 1024 × 512 pixels, size of frame buffer = 1024 × 512 × 24 bit

=4 MB

The memory captured by the frame buffer is 4 MB

22. What is the fraction of the total refresh time per frame spent in retrace of the electron beam for a noninterlaced raster system with a resolution of 1280 by 1024, a refresh rate of 60 Hz, a horizontal retrace time of 5 microseconds, and a vertical retrace time of 500 microseconds?

Solution:

1 sec = 10^6 usec

Refresh rate = 60Hz = $\frac{1}{60}$ sec to scan = 16.7 msec

The time for horizontal retrace = 1024 × 5 usec

The time for vertical retrace = 500 usec

Total time spent for retrace = 5120 + 500

= 5620 usec = 5.62 msec

The fraction of the total refresh time frame spent in retrace

$$= \frac{5.62}{16.7} = 0.337$$

Unit – 3

TWO–DIMENSIONAL ALGORITHMS

Scan Conversion | Rasterization

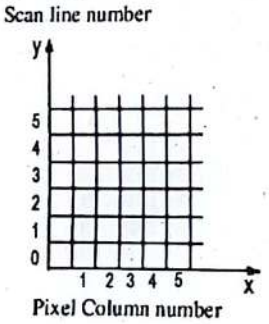
The process of representing a continuous graphical objects as a collection of discrete pixels by identifying their locations and setting them On is called scan conversion

- The process of determining which pixels will provide the best approximation to the desired line is properly known as rasterization.
- Combined with the process of rendering the picture in scan line order it is known as scan conversion.
- **Rasterization:** Process of determining which pixels give the best approximation to a desired line on the screen.
- Scan Conversion–Digitizing a picture definition given in an application program into a set of pixel intensity values for storage in the frame buffer.
- The basic building block of pictures is referred to as output primitives.
- Output primitives are the geometric structures that can be used to describe the shape and color of the objects.
- They include character, strings and geometric entities such as points, straight lines, circle, polygons etc. (Point and Lin are the most basic components of Picture)
- In a raster display system, a picture is specified by set of intensities for the pixel positions in the display.
- We generate images by turning the pixels ON or OFF, so it is necessary to represent an object (continuous) as a collection of discrete pixels.
- Hence the process of determining the appropriate pixels for representing a picture (or graphics object) is called rasterization.

- The process of conversion of the rasterized picture i.e. picture definition, stored in the frame buffer into a set of pixel intensity values for the rigid display pattern is called scan conversion. This is done by display processor.

Point Scan Conversion

- A mathematical point (x, y) where x and y are real numbers within an image area, needs to be scan converted to a pixel at location (x', y').
- So point plotting is accomplished by converting a single coordinate position into appropriate operation for the output device in use. Example: For CRT, Electron beam can be turned on for illuminate the screen phosphor or off at the selected location on the display technology.
- Now the question is how the electron beam is positioned depends upon the display technology?
- In case of raster system, a point can be plotted by setting the bit value corresponding to a specified screen position within the frame buffer to one. This is done by simply turning on the electron beam at that point.
- But in case of random system, it stores the point plotting instruction in the display list file. The coordinate values in this file are converted to deflection voltage that positions the electron beam at the screen location to be plotted.
- In raster system, line drawing is accomplished by calculating intermediate positions along the line path between two specified points. Then the output device is directed to fill in these positions between endpoints.
- But in case of random system (example: vector pen plotter), line drawing is accomplished by retrieving line drawing commands from the display list.
- Scan lines are numbered consecutively from 0, starting the bottom of the screen; the pixel columns are numbered from o, from left to right across each scan line.



Line Sean Conversion

- The Cartesian slope–intercept equation for a straight line is

  $y = m.x + c$ ..... (i)

  where,

  m = slope of line and c = y intercept

- When Use two endpoints of a line are given as (x_1, y_1) and (x_2, y_2) as shown in figure below, we can determine the value of slope m and intercept c with the following calculation

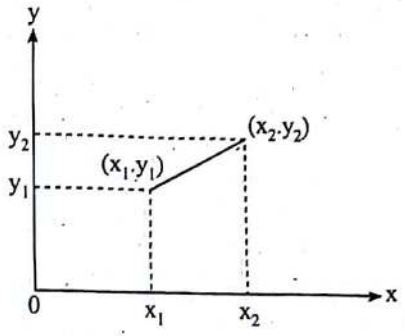


Fig: Line path between end point position $(x_1.y_1)$ and $(x_2.y_2)$

$$m = \frac{(y_2 - y_1)}{(x_2 - x_1)} \quad \text{..... (ii)}$$

Now, for any point (x_k, y_k), i becomes,

$y_k = m.x_k + c$ ..... (iii)

Again for any point (x_{k+1}, y_{k+1}), i become

$y_{k+1} = m . x_{k+1} + c$ ..... (iv)

Subtraction iii from iv we get,

$y_{k+1} - y_k = m. x_{k+1} + c - m.xk - c$

$y_{k+1} - y_k = m. (x_{k+1} - x_k)$

i.e. $\Delta y = m\Delta x$, where, Δy and Δx are small increment in x and y respectively.

Therefore, $m = \Delta y / \Delta x$ ..... (v)

- The above equation v, forms the basis for determining the intermediate pixel's position (i.e. deflection voltage in analog device like CRT).
- When the value of m is calculated using equation ii, test for three cases can be performed as:

Case I: For |m| < 1

- Δx can be set to small increment (Generally 1) and the corresponding Δy calculated from equation (v). (Δx range is high)

Case II: For |m| > 1

- Δy can be set to small increment (Generally 1) and the corresponding tot calculated from equation (v). (Δy range is high)

Case III: For |m| = 1

- Δy = Δx. x and y pixels both are equal.

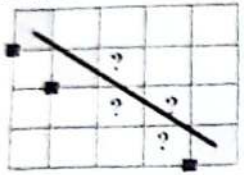
- In each case, a smooth line with slop m is generated between the specified points.

Line Drawing Algorithms

We are going to analyze how this process is achieved.

Some useful definitions

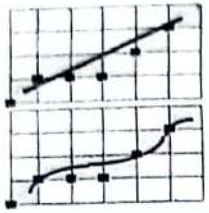
Rasterization Process of determining which pixels provide the best approximation to a desired line on the screen



Scan Conversion Combination of rasterization and generating the picture in scan line order.

General requirements

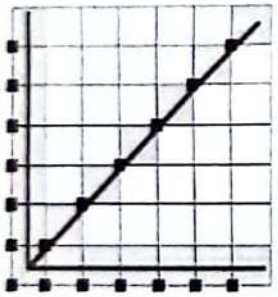
- Straight lines must appear as straight lines



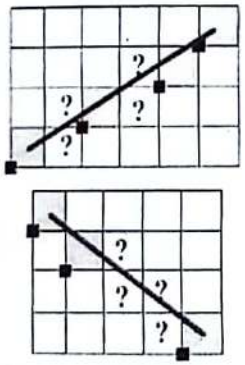
- They must start and end accurately
- Lines should have constant brightness along their length
- Lines should drawn rapidly

Line Drawing Algorithms

For horizontal, vertical and 45° lines the choice of raster elements is obvious. This lines exhibit constant brightness along the length:



For any other orientation the choice is more difficult:



Direct method of line drawing

A straight line can be represented in a slope-intercept equation as:

$y = mx + b$ ..... (1)

Where,

m is slope of the line and b is y-intercept

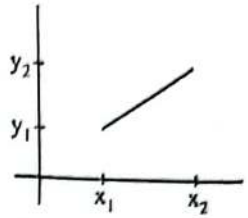


Fig.: Line path between two endpoint positions

A line is drawn by joining two end points,

For any two given points (x_1, y_1) and (x_2, y_2),

Slope can be calculated by,

$$m = \frac{y_2 - y_1}{x_2 - x_1} \quad \text{..... (2)}$$

$b = y_1 - mx_1$ ..... (3)

At any point (x_k, y_k)

$y_k = mx_k + b$

At any point $(x_{k+1} + y_{k+1})$

$y_{k+1} = mx_{k+1} + b$ ..... (4)

When the value of m is calculated using equation 2, the following test cases can be performed as,'

Case I: For lines with' slope magnitude |m| < = 1, we set small increment in x axis (Δx) equal to 1 and calculate corresponding y increment with the help of equation no. 4. i.e.,

$x_{next} = x_{previous} + 1$

$Y_{next} = MX_{next} + b$

Case II: For lines with slope magnitude |m| > 1, we set small increment in y axis (Δy) equal to .1 and calculate corresponding x increment with the help of equation no. 4.

$y_{next} = y_{previous} + 1$

$$x_{next} = \frac{y_{next} - b}{m}$$

On raster systems, we must sample a line at discrete positions and determine the nearest pixel to the line at each sampled position.

Q. Digitize the line joining the points (1, 2) and (5, 3) with direct method.

Hints;

$$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{1}{4}, \quad b = y_1 - mx_1 = \frac{7}{4}$$

Here, $m < 1$

So,

$x_{next} = x_{previous} + 1$

$y_{next} = mx_{next} + b$

x	y = mx + b	Plotpixel (x, y)
1	2	(1, 2)
2	2.25	(2, 2)
3	2.5	(3, 3)
4	2.75	(4, 3)
5	3	(5, 3)

Q. Digitize the line joining the points (3, 4) and (9, 5) with direct method.

Hints,

$$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{1}{6}, \quad b = y_1 - mx_1 = \frac{7}{2}$$

Here, $m < 1$

So,

$X_{next} = X_{previous} + 1$

$y_{next} = mx_{next} + b$

x	y = mx + b	Plotpixel (x, y)
3	4	(3, 4)
4	4.17	(4, 4)
5	4.33	(5, 4)
6	4.50	(6, 5)
7	4.66	(7, 5)
8	4.83	(8, 5)
9	5	(9, 5)

DDA line Drawing algorithm

The vector representation algorithms(and curve generation algorithms) which step along the line (or curve) to determine the pixels which should be turned on are sometimes called digital differential analyzer(DDA).The name comes from the fact that we use the same technique as a numerical method for solving differential equations.

The equation of the line is given,

$Y = m.x + b$ ..... (i)

$m = (y_2 - y_1)/(x_2 - x_1)$ ..... (ii)

for any interval Δx, corresponding interval is given by $\Delta y = m.\Delta x$

Case I:

If $|m| \leq 1$, we sample at unit x interval i.e. $\Delta x = 1$

$x_{k+1} = x_k + 1$ ..... (iii)

Then we compute each successive y-values, by setting $\Delta y = m$

(since, $m = \frac{\Delta y}{\Delta x}$ and $\Delta x = 1$)

So, $y_{k+1} = y_k + m$ .....(iv)

The calculated y value must be rounded to the nearest integer.

Case II:

If $|m| > 1$, we sample at unit y-interval i.e. $\Delta y = 1$ and compute each successive x-values.

$y_{k+1} = y_k + 1$ ..... (v)

Therefore, $1 = m.\Delta x$, and $\Delta x = \frac{1}{m}$ (since, $m = \frac{\Delta y}{\Delta x}$ and $\Delta y = 1$)

$x_{k+1} = x_k + \frac{1}{m}$ .....(vi)

The above equations are under the assumption that liens are to be processed from left endpoints (x_k, y_k) to right endpoints (x_{k+1}, y_{k+1}). If the line is to be processed from right to left i.e. starting end point is at the right then,

$\Delta x = -1$

- For $|m| \leq 1$, $x_{k+1} = x_k - 1$ and $y_{k+1} = y_k - m$ ..... (vii)

  And for slope greater than 1, $\Delta x = -1$

- For $|m| > 1$, $y_{i+1} = Y_k - 1$ and $X_{k+1} = x_k - \frac{1}{m}$ .....(viii)

The obtained equations also can be used to calculate the pixel position along a ling with negative slope as below.

- If |m| < 1 and the start end point is left, we set Δx=1 and calculate the Y value as in case i.
- If |m| < 1 and the start end point is right, we set Δx = −1 and calculate the Y value as in equation vii.
- If |m| > 1 and start end point is left, we set Δy=1 and calculate the Y value as in case ii.
- If |m| > 1 and start end point is right, we set Δy = −1 and calculate the Y value as in equation vii.

We sample the line at unit interval in one co-ordinate and determine corresponding integer values nearest the path for the other co-ordinate.

Different cases:

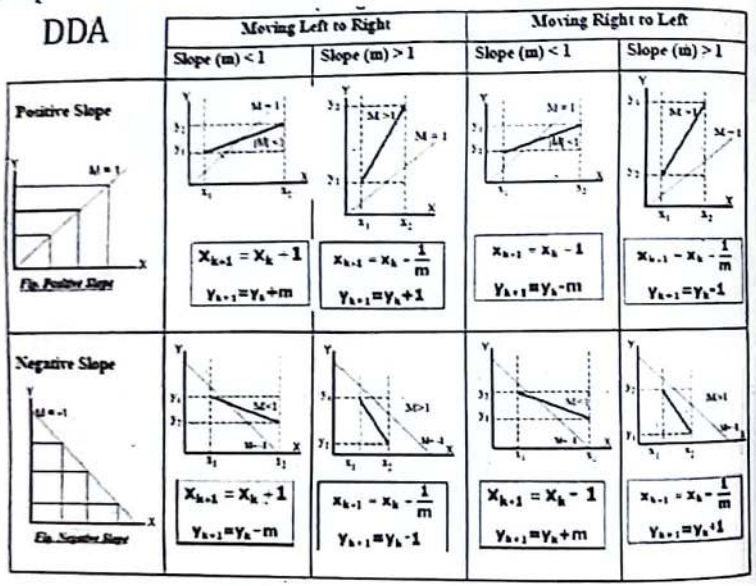
Case 1: Line with positive slope and magnitude less than or equal to 1(|m| ≤ 1)

Case 2: Line with positive slope and magnitude more than 1(|m| > 1)

Case 3: Line with negative slope and magnitude less than or equal to 1(|m| ≤ 1)

Case 4: Line with negative slope and magnitude more than 1(|m| > 1)

The possible combinations can be shown as follows:



1. Line with a positive slope

Case I:

If slope is positive with magnitude lesser than or equal to 1, |m| ≤ 1, then we sample at unit x intervals (Δx = 1) and compute each successive y value because the increment in x is more than increment in y.

So, set Δx = 1.

Now, from equation (I), we, get

Δy = m

i.e., $x_{k+1} = x_k + \Delta x = x_k + 1$ ..... (3)

$y_{k+1} = y_k + \Delta y = y_k + m$ ..... (4)

Where subscript k takes integer values starting from 1, for the first point, and increases by 1 unit until the final end point is reached. Since m can be any real number between 0 to 1, the calculated y values must be rounded to the nearest integer.

Case II:

If slope is positive with magnitude greater than 1, |m| > 1, then the increment in x (Δx) is smaller than increment in y (Δy). So, we sample at unit y intervals (Δy = 1) and calculate each succeeding x value as,

$$\Delta x = \frac{1}{m}$$

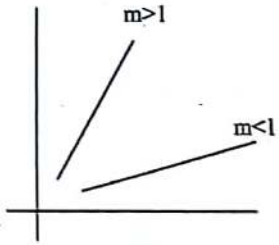


Fig.: Line with positive slope |m| < 1 and |m| > 1

That is,

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

$y_{k+1} = y_k + 1$ ..... (6)

Here, in the both case m ≤ 1 and m > 1, we consider the algorithm (i.e., equations 3, 4, 5, 6) based on the assumption that lines are to be processed from the left end point to the right end point.

If this process is reversed (that is, lines are to be processed from right to left), relation is required to be changed in both case:

Case I:

If sloped is positive with magnitude lesser than or equal to 1, |m| ≤ 1, then Δx = −1, so, Δy = −m

i.e. $x_{k+1} = x_k - 1$

$y_{k+1} = y_k - m$

Case II:

If slope is positive with magnitude greater than 1, $|m| > 1$,

then $\Delta y = -1$ and $\Delta x = -\frac{1}{m}$

i.e., $x_{k+1} = x_k - \frac{1}{m}$ ..... (7)

$y_{k+1} = y_k - 1$ ..... (8)

2. **Line with negative slope**

Case I:

If slope is negative with magnitude lesser than or equal to 1, $|m| \leq 1$, then assume start end point is the left.

$\Delta x = 1$ and $\Delta y = m$ (m is negative)

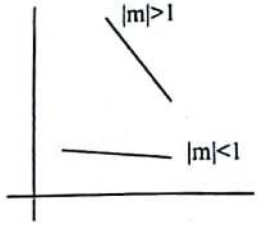


Fig.: Line with negative slope $|m| < 1$ and $|m| > 1$

That is,

$x_{k+1} = x_k + 1$ ..... (9)

$y_{k+1} = y_k + m$ ..... (10)

If algorithm is required to proceed right to left, then set on.

$\Delta x = -1$ and $\Delta y = -m$

i.e., $x_{k+1} = x_k - 1$ ..... (11)

$y_{k+1} = y_k - m$ ..... (12)

Case II:

If slope is negative with magnitude greater than 1, $|m|>1$, then assume start end is at left and set $\Delta y = -1$ and $\Delta x = -\frac{1}{m}$

i.e., $x_{k+1} = x_k - \frac{1}{m}$ ..... (13)

$y_{k+1} = y_k - 1$ ..... (14)

If the, algorithm is required to proceed right to left, then set

$\Delta y = 1$ and $\Delta x = \frac{1}{m}$

i.e., $x_{k+1} = x_k + \frac{1}{m}$ ..... (15)

$y_{k+1} = y_k + 1$ ..... (16)

DDA Algorithm

Step 1: Start Algorithm

Step 2: Declare x_1, y_1, x_2, y_2, dx, step as integer variable and x, y, x_{inc}, y_{inc} as floating point

Step 3: Read the values of two end points x_1, y_1, x_2, y_2

Step 4: Calculate $dx = x_2 - x_1$

Step 5: Calculate $dy = y_2 - y_1$.

Step 6: Calculate the value of step

If absolute (dx) > absolute (dy)

Then step = absolute (dx)

Else step = absolute (dy)

Step 7: Select the raster unit

$x_{inc} = \frac{dx}{step}$

$y_{inc} = \frac{dx}{step}$

Assign $x = x_1$

Assign $y = y_1$

Step 8: Set pixel (x, y)

Step 9: $x = x + x_{inc}$

$y = y + y_{inc}$

Plot pixels (Round (x), Round (y))

Step 10: Repeat step 9 until $x = x_2$

Step 11: End Algorithm

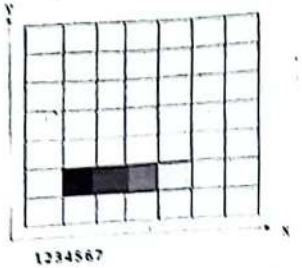
Advantage of DDA

1. It is simple to understand.
2. It is faster method than direct use of line equation $y = mx + c$. (Removes multiplication operation and only involve increments operation of x or y direction)
3. It requires no special skills for implementation

Disadvantages of DDA

1. A floating point addition is still needed in determining each successive point which is time consuming.
2. The accumulation of round off error in successive addition of the floating point increments may cause, the calculated pixel position to drift away from the true line path for long line segment.

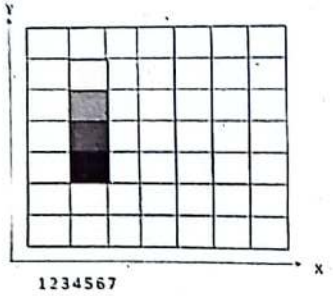
Example: Horizontal line



1. $(X_1, Y_1) = (2, 2)$
   $(X_2, Y_2) = (5, 2)$
   $\Delta x = 3$
   $\Delta y = 0$
   $m = \Delta y/\Delta x = 0$
   Steps=3
   $\Delta x = \Delta x/step = 1$
   $\Delta y = \Delta y/step = 0$

x_i	y_i	x_{i+1}	y_{i+1}
2	2	3	2
3	2	4	2
4	2	5	2

Example: Vertical line



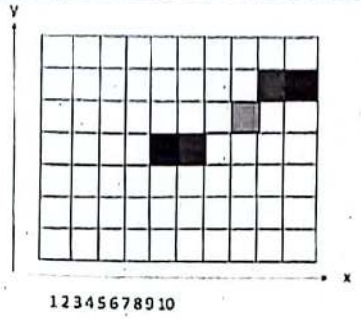
1. $(X_1, Y_1) = (2, 3)$
   $(X_2, Y_2) = (2, 6)$
   $\Delta x = 0$
   $\Delta y = 3$
   $m = \Delta y/\Delta x$ = Infinite
   Steps = 3
   $\Delta x = \Delta x/step = 0$
   $\Delta y = \Delta y/step = 1$

x_i	y_i	x_{i+1}	y_{i+1}
2	3	2	4
2	4	2	5
2	5	2	6

Example: m < 1

1. $(X_1, Y_1) = (5, 4)$
   $(X_2, Y_2) = (10, 6)$
   $\Delta x = 5$
   $\Delta y = 2$
   $m = \Delta y/\Delta x = 0.4$
   Steps = 5
   $\Delta x = \Delta x/step = 1$
   $\Delta y = \Delta y/step = 2/5 = 0.4$
   From DDA algorithm
   $X_{i+1} = X_i + 1$
   $Y_{i+1} = Y_i + m$

x	y	X_{i+1}	Y_{i+1}	Y round off	Plot (x,y)
5	4	6	4.4	4	(6,4)
6	4.4	7	4.8	5	(7,5)
7	4.8	8	5.2	5	(8,5)
8	5.2	9	5.6	6	(9,6)
9	5.6	10	6	6	(10,6)



Points

(5,4), (6,4), (7,5), (8,5), (9,6), (10,6)

Example: m > 1

1. $(X_1, Y_1) = (5,4)$

   $(X_2, Y_2) = (8,9)$

   $\Delta x = 3$

   $\Delta y = 5$

   $m = \Delta y/\Delta x = 1.67$

   Steps = 5

   $\Delta x = \Delta x/step = 0.6$

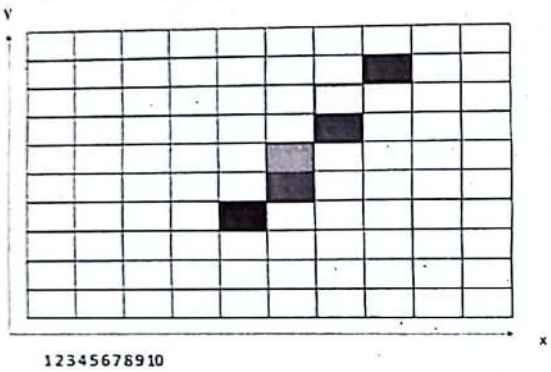
   $\Delta y = \Delta y/step = 5/5 = 1$

   From DDA algorithm

   $X_{i+1} = X_i + 1/m$

   $Y_{i+1} = Y_i + 1$

x	y	X_{i+1}	Y_{i+1}	X round off	Plot (x,y)
5	4	5.6	5	6	(6,5)
5.6	5	6.2	6	6	(6,6)
6.2	6	6.8	7	7	(7,7)
6.8	7	7.4	8	7	(7,8)
7.4	8	8	9	8	(8,9)



Points

(5,4), (6,5), (6,6), (7,7), (7,8), (8,9)

Q. Digitize a Line with end point A(2, 3) and B(6, 8) , using DDA.

Here, Slope (M) = $\frac{(8-3)}{(6-2)} = 1.25$

Here Slope is positive and greater than 1

And moving left to right and $\frac{1}{m} = 0.8$

So, we use $X_{k+1} = X_k + \frac{1}{M}$

$Y_{k+1} = Y_k + 1$

K	$X_{k+1} = X_k + \frac{1}{M}$	$Y_{k+1} = Y_k + 1$	(X, Y)
1	= 2 + 0.8 = 2.8 ~ 3	4	(3,4)
2	= 2.8 + 0.8 = 3.6 ~ 4	5	(4,5)
3	= 3.6 + 0.8 = 4.4 ~ 4	6	(4,6)
4	= 4.4 + 0.8 = 5.2 ~ 5	7	(5,7)
5	= 5.2 + 0.8 = 6	8	(6,8)

Solved Numerical Problems

1. **Consider a line from (2,1) to (8,3). Using simple DDA algorithm, rasterize this line.**

Solution:

Given Starting Point: $(x_1, y_1) = (2,1)$ and

Ending Point: $(x_2, y_2) = (8,3)$

Now Slope $m = (y_2-y_1)/(x_2-x_1) = (3-1)/(8-2) = (1/3) = 0.333$

Since $|m|<1$, From DDA algorithm we have

$X_{k+1} = X_k + 1$

$Y_{k+1} = Y_k + (M)$

X	Y	X-Plot	Y-Plot	(X,Y)
2	1	2	1	(2,1)
3	1.33	3	1	(3,1)
4	1.66	4	2	(4,2)
5	1.993	5	2	(5,2)
6	2.326	6	2	(6,2)
7	2.659	7	3	(7,3)
8	2.999	8	3	(8,3)

2. Digitized the line with end points (0,0) and (4,5) using DDA.

Solution:

Given Starting Point: $(x_1, y_1) = (0,0)$ and Ending Point $(x_2, y_2) = (4,5)$

Now Slope $m = (y_2-y_1)/(x_2-x_1) = (5-0)/(5-0) = (5/4) = 1.25$

Since $|m| > 1$, From DDA algorithm we have

$y_{k+1} = y_k + 1$

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

X	Y	X-Plot	Y-Plot	(X,Y)
0	0	0	0	(0,0)
0.8	1	1	1	(1,1)
1.6	2	2	2	(2,2)
2.4	3	2	3	(2,3)
3.2	4	3	4	(3,4)
4	5	4	5	(4,5)

3. Digitized the line with end points (3,7) and (8,3) using DDA.

Solution:

Given Starting Point: $(x_1, y_1) = (3,7)$ and Ending Point: $(x_2, y_2) = (8,3)$

Now Slope $= (y_2 - y_1)/(x_2 - x_1) = (3-7)/(8-3) = (-4/5) = -0.8$

Since $|m| < 1$, From DDA algorithm we have

$X_{k+1} = X_k + 1$

$Y_{k+1} = Y_k + (M)$

X	Y	X-Plot	Y-Plot	(X,Y)
3	7	3	7	(3,7)
4	7 – 0.8 = 6.2	4	6	(4,6)
5	5.4	5	5	(5,5)
6	4.6	6	5	(6,5)
7	3.8	7	4	(7,4)
8	3	8	3	(8,3)

4. Draw line using DDA (3,7) (8,3)

Solution:

$\Delta x = x_2 - x_1 = 5$

$\Delta y = y_2 - y_1 = -4$

Steps = 5

$m = \frac{-4}{5}$

$x_{in}\ c = \frac{\Delta x}{5} = \frac{5}{5} = 1$

$y_{in}\ c = \frac{-4}{5} = -0.8$

x_i	y_i	x_{i+1}	y_{i+1}	y_{round}	$x_{plot}y_{plot}$
3	7	4	6.2	6	(4,6)
4	6.2	5	5.4	5	(5,5)
5	5.4	6	4.6	5	(6,5)
6	4.6	7	3.8	4	(7,4)
7	3.8	8	3	3	(8,3)

5. Draw line using (5,5) (0,0) using DDA

Solution:

$\Delta x = 0 - 5 = -5$

$\Delta y = 0 - 5 = -5$

$m = \frac{\Delta y}{\Delta x} = 1$

Steps = 5

$\Delta x = \frac{\Delta x}{5} = \frac{-5}{5} = -1$

$\Delta y = \frac{\Delta y}{5} = -1$

X_i	Y_i	X_{i+1}	Y_{i+1}	Plot
5	5	4	4	4,4
4	4	3	3	3,3
3	3	2	2	2,2
2	2	1	1	1,1
1	1	0	0	0,0

6. Draw line using (3,6) – (5,9)

Solution:

$\Delta x = 2$

$\Delta y = 3$

$m = \frac{3}{2} > 1$

$X_{i+1} = x_i + \frac{1}{m}$

$Y_{i+1} = y_i + 1$

Step = 3

$X_{inc} = \frac{2}{3} = 0.67$

$Y_{inc} - \frac{3}{3} = 1$

x_i	y_i	x_{i+1}	y_{i+1}	x_{round}	Plot
3	6	3.67	7	4	4,7
3.67	7	4.34	8	4	4,8
4.34	8	5.01	9	5	5,9

7. **Consider a line from (0,0) to (5,5). Using simple DDA algorithm, rasterize this line.**

Solution:

Given starting point: $(x_1, y_1) = (0,0)$ and

Ending point: $(x_2,y_2) = (5,5)$

Now Slope $m = (y_2-y_1)/(x_2-x_1)$

$= (5-0)/(5-0) = (/55) = 1$

Since $|m| <= 1$, From DDA algorithm we have

$X_{k+1} = X_k + 1$

$Y_{k+1} = Y_k + (M)$

X	Y	X-Plot	Y-Plot	(X,Y)
0	0	0	0	(0,0)
1	1	1	1	(1,2)
2	2	2	2	(2,2)
3	3	3	3	(3,3)
4	4	4	4	(4,4)
5	5	5	5	(5,5)

Bresenhams Algorithm

The algorithm selects the best pixel co-ordinate by testing the sign of integer parameter whose value is proportional to the difference between the separations of two-pixel actual line path.

1. **Line with positive slope and magnitude $|m| < 1$**

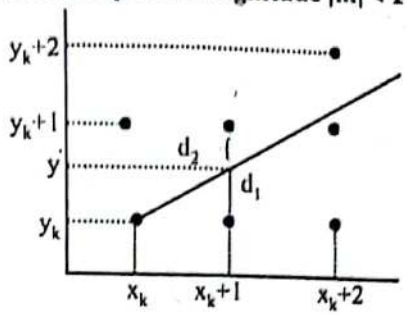


Fig.: Line with m<1

- Pixel positions are determined by sampling at unit x intervals.
- Starting from left end position (x_0, y_0) of a given line, we step to each successive column (x-position) and plot the pixel whose scan line y value is closed to the line path.

Assuming the pixel at (x_k, y_k) to be displayed is determined, we next to decide which pixel to plot in column x_{k+1}, our choices are the pixels at positions.

$(x_k + 1, y_k)$ and $(x_k + 1, y_k + 1)$

At sampling position $x_k + 1$, we label vertical pixel separations from the mathematical line path d_1 and d_2.

The y-co-ordinate the mathematical at pixel column position $x_k + 1$ is calculation.

As,

$y = m(x_k + 1) + b$ ..... (1)

Then,

$d_1 = y - y_k$

$d_1 = m(x_k + 1) + b - y_k$ ..... (2)

And,

$d_2 = (y_k + 1) - y$

$d_2 = y_k + 1 - m(x_k + 1) - b$ .....(3)

Now,

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

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

$= 2\frac{\Delta y}{\Delta x}(x_k + 1) - 2y_k + 2b - 1$

Defining decision parameter $p_k = \Delta x(d_1 - d_2)$

$p_k = \Delta x(d_1 - d_2) = 2\Delta y(x_k + 1) - 2\Delta x y_k + 2\Delta x b - \Delta x$

$= 2\Delta y x_k + 2\Delta y - 2\Delta x y_k + \Delta x(2b - 1)$

$= 2\Delta y x_k - 2\Delta x y_k + 2\Delta y + \Delta x(2b - 1)$

$= 2\Delta y x_k - 2\Delta x y_k + c$ ..... (4)

Where, $c = 2\Delta y + \Delta x(2b - 1)$

Here, sign of p_k is same, as the sign of $d_1 - d_2$ since $\Delta x > 0$.

Case I:

If $p_k < 0$ then $d_1 < d_2$ which implies pixel at y_k is nearer than pixel at $(y_k + 1)$. So, pixel at y_k is better to choose which reduce error than pixel at $(y_k + 1)$. This determines next pixel co-ordinate to plot is $(x_k + 1, y_k)$

Case II:

If $p_k \geq 0$, then $d_2 < d_1$ which implies that $y_k + 1$ is nearer than y_k. So, pixel at $(y_k + 1)$ is better to choose which reduce error than pixel at y_k. This determines next pixel co-ordinate to plot is $(x_k + 1, y_k + 1)$.

Now, similarly, pixel as $(x_k + 2)$ can be determined whether it is $(x_k + 2, y_k + 1)$ or $(x_k + 2, y_k + 2)$ by looking the sign of deciding parameter $p_k + 1$ assuming pixel as $(x_k + 1)$ is known.

$p_{k+1} = 2\Delta y x_{k+1} - 2\Delta x y_{k+1} + c$ where c is same as in p_k

Now,

$p_{k+1} - p_k = 2\Delta y x_{k+1} - 2\Delta x y_{k+1} + c - (2\Delta y x_k - 2\Delta x y_k + c)$

$= 2\Delta y(x_{k+1} - x_k) - 2\Delta x(y_{k+1} - y_k)$ where $x_{k+1} = x_k + 1$

$= 2\Delta y(x_k + 1 - x_k) - 2\Delta x(y_{k+1} - y_k)$

$= 2\Delta y - 2\Delta x(y_{k+1} - y_k)$

This implies that decision parameter for the current column can be determined if the decision parameter of the last column is known.

Here, $(y_{k+1} - y_k)$ could either 0 or 1 which depends on sign of p_k.

Case I:

If $p_k < 0$(i.e., $d_1 < d_2$),

$y_{k+1} = y_k$ which implies $(y_{k+1} - y_k = 0)$

i.e., at $p_k < 0$, then pixel to be plotted is (x_{k+1}, y_k)

$x_{k+1} = x_k + 1$

$y_{k+1} = y_k$

and,

$p_{k+1} = p_k + 2\Delta y$

Case II:

If $p_k \geq 0$ (i.e., $d_2 < d_1$), $y_{k+1} = y_k + 1$ which implies $(y_k + 1 - y_k) = 1$

That is, at $p_k \geq 0$,

the pixel to plot is $(x_k + 1, y_k + 1)$

$x_{k+1} = x_k + 1$

$y_{k+1} = y_k + 1$

and,

$p_{k+1} = p_k + 2\Delta y - 2\Delta x$

Initial decision parameter (p_0)

Here, $p_k = 2\Delta y x_k - 2\Delta x y_k + 2\Delta y + \Delta x(2b - 1)$

$p_0 = 2\Delta y x_0 - 2\Delta x y_0 + 2\Delta y + \Delta x(2b - 1)$

But $b = y_0 - mx_0 = y_0 - \frac{\Delta y}{\Delta x} x_0$

$p_0 = 2\Delta y x_0 - 2\Delta x y_0 + 2\Delta y + 2\Delta x y_0 - 2\Delta y x_0 - \Delta x = 2\Delta y - \Delta x$

2. Line with positive slope and magnitude |m| > 1

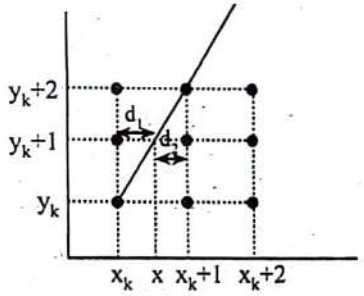


Fig.: Line with m > 1

- Pixel positions are determined by sampling at unit y intervals.
- Starting from left end position (x_0, y_0) of a given line, we step to each successive row (y-position) and plot the pixel whose scan line x value is closed to the line path.

  Assuming the pixel at (x_k, y_k) to be displayed is determined, we next to decide which pixel to plot in row y_{k+1}, our choices are the pixels at position.

  $(x_k, y_k + 1)$ and $(x_k + 1, y_k + 1)$

  At sampling position $y_k + 1$, we label horizontal pixel separations from the mathematical line path, d_1 and d_2.

  The x-co-ordinate on the mathematical at pixel row position y_k+1 is calculation.

As,

$y_k + 1 = mx + b$

$x = (y_k + 1 - b)/m$ .....(1)

Then,

$d_1 = x - x_k$ .....(2)

And,

$d_2 = (x_k + 1) - x$ .....(3)

Now,

$d_1 - d_2 = x - x_k - x_k - 1 + x$

$= 2x - 2x_k - 1$

$x = (y_k + 1 - b)/m$

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

$= 2\Delta x(y_k + b)/\Delta y - 2x_k - 1$

$\Delta y(d_1 - d_2) = 2\Delta x y_k + 2\Delta x - 2\Delta x b - 2\Delta y x_k - \Delta y$

Defining decision parameter $p_k = \Delta y\,(d_1 - d_2)$

$p_k = \Delta y(d_1 - d_2) = 2\Delta x y_k + 2\Delta x - 2\Delta x b - 2\Delta y x_k - \Delta y$

$= 2\Delta x y_k - 2\Delta y x_k + c$ .....(4)

Where; $c = 2\Delta x - 2\Delta x b - \Delta y$

Here, sign of p_k is same as the sign of $d_1 - d_2$ since $\Delta y > 0$.

Case I:

If $p_k < 0$ then $d_1 < d_2$ which implies pixel at x_k is nearer than pixel at $(x_k + 1)$. So, pixel at x_k is better to choose which reduce error than pixel at $(x_k + 1)$. This determines next pixel co-ordinate to plot is $(x_k, y_k + 1)$

Case II:

If $p_k \geq 0$, then $d_1 > d_2$, which implies that $x_k + 1$ is nearer than x_k. So, pixel at $(x_k + 1)$ is better to choose which reduce error than pixel at x_k. This determines next pixel co-ordinate to plot is $(x_k + 1, y_k + 1)$.

Now, similarly, pixel as $(y_k + 2)$ can be determined whether it is $(x_k + 1, y_k + 2)$ or $(x_k + 2, y_k + 2)$ by looking the sign of deciding parameter p_{k+1} assuming pixel as $(y_k + 1)$ is known.

$p_{k+1} = 2\Delta x\, y_{k+1} - 2\Delta y\, x_{k+1} + c$ where c is same as in p_k

Now,

$p_{k+1} - p_k = 2\Delta x\, y_{k+1} - 2\Delta y\, x_{k+1} + c - 2\Delta x\, y_k + 2\Delta y\, x_k - c$

$= 2\Delta x(y_{k+1} - y_k) - 2\Delta y(x_{k+1} - x_k)$

$= 2\Delta x(y_{k+1} - y_k) - 2\Delta y(x_{k+1} - x_k)$

$= 2\Delta x(y_k + 1 - y_k) - 2\Delta y(x_{k+1} - x_k)$

$= 2\Delta x - 2\Delta y(x_{k+1} - x_k)$

This implies that decision parameter for the current row can be determined if the decision parameter of the last row is known.

Here, $(x_{k+1} - x_k)$ could either 0 or 1 which depends on sign of p_k.

Case I:

If $p_k < 0$ (i.e., $d_1 < d_2$), then $x_{k+1} = x_k$ which implies $(x_{k+1} - x_k = 0)$

i.e., at $p_k < 0$, then pixel to be plotted is $(x_k, y_k + 1)$

$x_{k+1} = x_k$

$y_{k+1} = y_k + 1$

and,

$p_{k+1} = p_k + 2\Delta x$

Case II:

If $p_k \geq 0$ (i.e., $d_1 > d_2$), then $x_{k+1} = x_k + 1$ which implies $(x_k + 1 - x_k) = 1$

That is, at $p_k \geq 0$, the pixel to plot is $(x_k + 1, y_k + 1)$

$x_{k+1} = x_k + 1$

$y_{k+1} = y_k + 1$

and,

$p_{k+1} = p_k + 2\Delta x - 2\Delta y$

Initial decision parameter (p_0)

$p_k = 2\Delta x\, y_k + 2\Delta x - 2\Delta x\, b - 2\Delta y\, x_k - \Delta y$

$p_o = 2\Delta x\, y_o + 2\Delta x - 2\Delta x\, b - 2\Delta y\, x_o - \Delta y$

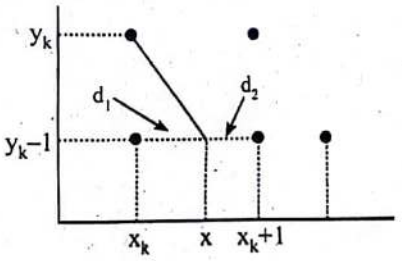
$b = y_0 - mx_0$

$p_0 = 2\Delta x y_0 + 2\Delta x - 2\Delta x(y_0 - \Delta y\, x_0/\Delta x) - 2\Delta y\, x_0 - \Delta y$

$= 2\Delta x\, y_0 + 2\Delta x - 2\Delta x y_0 - 2\Delta x \times \Delta y\, x_0/\Delta x - 2\Delta y\, x_0 - \Delta y$

$= 2\Delta x - \Delta y$

3. **Line with negative slope and magnitude |m| > 1**



- Pixel positions are determined by sampling at unity intervals.
- Starting from left end position (x_0, y_0) of a given line, we step to each successive row (y position) and plot the pixel whose x value is closest to the line path.
- Assuming the pixel at (x_k, y_k) to be displayed is determined, we next need to decide which pixel to plot in row $y_k - 1$.
- The candidate pixels are at position $(x_k, y_k - 1)$ and $(x_k + 1, y_k - 1)$
- At sampling position $y_k - 1$, we label horizontal pixel separation from the mathematical line path d_1 and d_2.
- The x-coordinate on the mathematical line at pixel row position $y_k - 1$ is calculated as

$y_k - 1 = mx + b$

$$\text{or, } x = \frac{y_k - 1 - b}{m}$$

where $m = -\frac{\Delta y}{\Delta x}$ since m is negative.

Then,

$d_1 = x - x_k$ .....(ii)

$d_2 = x_k + 1 - x$ .....(iii)

And,

$$d_1 - d_2 = x - x_k - x_k - 1 + x$$
$$= 2x - 2x_k - 1$$
$$= 2\left(\frac{y_k - 1 - b}{m}\right) - 2x_k - 1$$
$$= 2\left(-\frac{\Delta x}{\Delta y}\right)(y_k - 1 - b) - 2x_k - 1$$

$(d_1 - d_2)\Delta y = -2\Delta x(y_k - 1 - b) - (2x_k + 1)\Delta y$

Now, defining decision parameter $p_k = \Delta y\,(d_1 - d_2)$

$p_k = \Delta y(d_1 - d_2) = -2\Delta x(y_k - 1 - b) - (2x_k + 1)\Delta y$

$= -2\Delta x y_k + 2\Delta x + 2\Delta x b - 2\Delta y x_k - \Delta y$

$= -2\Delta x\, y_k - 2\Delta y x_k + 2\Delta x + 2\Delta x b - \Delta y$

$\therefore p_k = -2\Delta x y_k - 2\Delta y x_k + c$ .....(iv)

Where $C = 2\Delta x + 2\Delta x b - \Delta y$ (All are constant)

Here, the sign of p_k is same as the sign of $d_1 - d_2$, as $\Delta y > 0$.

Case I:

If $p_k < 0$, then $d_1 < d_2$ which implies that x_k is nearer than $x_k + 1$, so the next pixel to select is $(x_k, y_k - 1)$

Case II:

If $p_k \geq 0$, then $d_1 > d_2$ which implies that $x_k + 1$ is nearer than x_k, so the next pixel to choose is $(x_k + 1, y_k - 1)$

Now,

Similarly, pixel at $(y_k - 2)$ can be determined whether it is $(x_k + 1, y_k - 2)$ or $(x_k + 2, y_k - 2)$ by looking the sign of deciding parameter p_{k+1} assuming pixel at $(y_k - 1)$ is known.

$p_{k+1} = -2\Delta x y_{k+1} - 2\Delta y x_{k+1} + C$

where C is same as in p_k.

Now,

$p_{k+1} - p_k = -2\Delta x y_{k+1} + C + 2\Delta x y_k + 2\Delta y x_k - C$

$= -2\Delta x(y_{k+1} - y_k) - 2\Delta y(x_{k+1} - x_k)$

$= -2\Delta x(y_k - 1 - y_k) - 2\Delta y(x_{k+1} - x_k)$

$= 2\Delta x - 2\Delta y(x_{k+1} - x_k)$

This implies that decision parameter for the current row can be determined if the decision parameter of the last row is known.

Here $(x_k + 1 - x_k)$ could either 0 or 1 which depends on sign of p_k.

Case I:

If $p_k < 0$, then

$x_{k+1} = x_k$

$y_{k+1} = y_k - 1$

$p_{k+1} = p_k + 2\Delta x$

Case II:

If $p_k \geq 0$ (i.e., $d_1 > d_2$) then

$x_{k+1} = x_k + 1$

$y_{k+1} = y_k - 1$

$p_{k+1} = p_k + 2\Delta x - 2\Delta y$

Now, Initial decision parameter

$p_k = -2\Delta x y_k - 2\Delta y x_k + 2\Delta x b - 2\Delta x - \Delta y$

Now,

$p_0 = -2\Delta x y_0 - 2\Delta y x_0 + 2\Delta x b - 2\Delta x - \Delta y$

$$= -2\Delta x y_0 - 2\Delta y x_0 + 2\Delta x\left(y_0 + \frac{\Delta y}{\Delta x} x_0\right) + 2\Delta x - \Delta y$$

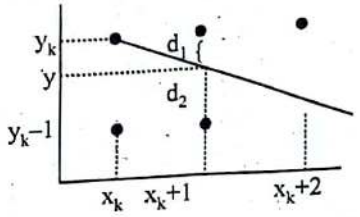
$$\left[y = mx + b,\ b = y - mx,\ m = -\frac{\Delta y}{\Delta x}\right]$$

$$= 2\Delta xy_0 - 2\Delta yx_0 + \Delta x\left(\frac{2\Delta xy_0 + 2\Delta yx_0}{\Delta x}\right) + 2\Delta x - \Delta y$$

$$= 2\Delta xy_0 - 2\Delta yx_0 + 2\Delta xy_0 + 2\Delta yx_0 + 2\Delta x - \Delta y$$

$$= 2\Delta x - \Delta y$$

4. Line with negative slope and magnitude $|m| < 1$.



$d_1 = y_k - y$

$d_2 = y - (y_k - 1) = y - y_k + 1$

$d_1 - d_2 = y_k - y - y + y_k - 1 = 2y_k - 2y - 1$

Again, $y = m(x_k + 1) + b$

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

$m = -\frac{\Delta y}{\Delta x}$ (m is negative)

$$d_1 - d_2 = 2y_k - 2\left(-\frac{\Delta y}{\Delta x}\right)(x_k + 1) - 2b - 1$$

$$= \frac{2yk + \Delta x + 2\Delta yxk + 2\Delta y - \Delta x(2b) - \Delta x}{\Delta x}$$

$p_k = (d_1 - d_2)\,\Delta x = 2\Delta xy_k + 2\Delta yx_k + 2\Delta y - \Delta x(2b) - \Delta x$

$p_k = 2\Delta xy_k + 2\Delta yx_k + c$

where $c = 2\Delta y - \Delta x(2b) - \Delta x$

For next decision parameter,

$p_{k+1} = 2\Delta xy_{k+1} + 2\Delta yx_{k+1} + c$

Now,

$p_{k+1} - p_k = 2\Delta x(y_{k+1} - y_k) + 2\Delta y(x_{k+1} - x_k)$

Case I:

If $p_k < 0$, $d_1 < d_2$ then

$x_{k+1} = x_k + 1$

$y_{k+1} = y_k$

$p_{k+1} = p_k + 2\Delta y$

Case II:

If $p_k \geq 0$, $d_1 \geq d_2$ then

$x_{k+1} = x_k + 1$

$y_{k+1} = y_k - 1$

$p_{k+1} = p_k + 2\Delta y - 2\Delta x$

Now, initial decision parameter,

$p_k = p_0, x_k = x_0, y_k = y_0$

$$p_0 = 2\Delta xy_0 + 2\Delta yx_0 + 2\Delta y - \Delta x(2\left(y_0 - \left(-\frac{\Delta y}{\Delta x}\right)x_0\right) + 1)$$

$$= 2\Delta xy_0 + 2\Delta yx_0 + 2\Delta y - \Delta x\left(\frac{2\Delta xy_0 + 2\Delta yx_0 + \Delta x}{\Delta x}\right)$$

$$= 2\Delta xy_0 + 2\Delta yx_0 + 2\Delta y - 2\Delta xy_0 - 2\Delta yx_0 - \Delta x$$

$$= 2\Delta y - \Delta x$$

BLA Algorithm

Step 1: Start

Step 2: Declare variables $x_1, y_1, x_2, y_2, l_x, l_y, \Delta x, \Delta y, p_0, p_k, p_{k+1}$

Step 3: Read values of x_1, y_1, x_2, y_2

Step 4: Calculate Δx = absolute$(x_2 - x_1)$

Δy = absolute $(y_2 - y_1)$

Step 5: If $(x_2 > x_1)$

assign $l_x = 1$

else

assign $l_x = -1$

Step 6: If $(y_2 > y_1)$

assign $l_y = 1$

else

assign $l_y = -1$

Step 7: Plot (x_1, y_1)

Step 8: If $\Delta x > \Delta y$ (i.e., $|m| < 1$)

compute $p_0 = 2\Delta y - \Delta x$

starting at k = 0 to Δx times, repeat

if $(p_k < 0)$

$x_{k+1} = x_k + l_x$

$y_{k+1} = y_k$

$p_{k+1} = p_k + 2\Delta y$

else

$x_{k+1} = x_k + l_x$

$y_{k+1} = y_k + l_y$

$p_{k+1} = p_k + 2\Delta y - 2\Delta x$

plot(x_{k+1}, y_{k+1})

else

calculate p = $2\Delta x - \Delta y$

starting at k = 0 to Δy times, repeat

If ($p_k < 0$)

$x_{k+1} = x_k$

$y_{k+1} = y_k + l_y$

$p_{k+1} = p_k + 2\Delta x$

else

$x_{k+1} = x_k + l_x$

$y_{k+1} = y_k + l_y$

$p_{k+1} = p_k + 2\Delta x - 2\Delta y$

Plot (x_{k+1}, y_{k+1})

Step 9: Stop

Advantage

1. DDA includes calculations related to m and 1/m which is still complicated. BLA improves DDA algorithm by only involving integer calculations making it simple and easy to use.
2. Using integer addition and subtraction and multiplication by 2 makes the algorithm more time efficient as addition, subtraction and multiplication operation is computed in lesser time in comparison to division as compared to DDA.
3. BLA is more accurate as compared to DDA because it avoids the ROUND OFF operation and scan converts the line using only incremental integer calculations.

DDA vs BLA

Criteria	DDA Algorithm	Bresenham's Line Algorithm
Arithmetic Operations	Involves floating-point arithmetic	Uses only integer arithmetic
Slope Calculation	Requires calculation of slopes	Does not require slope calculations
Efficiency	Less efficient	More efficient

Criteria	DDA Algorithm	Bresenham's Line Algorithm
Line Restrictions	Handles lines with any slope	Handles lines with slopes between 0 and 1
Line Smoothness	Provides smoother lines	May produce a staircase effect on diagonal lines
Rounding Errors	Prone to rounding errors	Minimizes rounding errors
Computational Steps	May require more steps	Requires fewer steps
Handling Negative Slopes	Handles negative slopes	Requires additional modifications to handle negative slopes
Line Length Limitations	No specific limitations	May have limitations due to integer calculations
Application Flexibility	Suitable for various line types	Primarily used for simple lines (e.g., horizontal, vertical)
Implementation Complexity	Relatively simpler	Requires additional calculations for decision-making
Drawing	Can't draw curves and circles as accurate as BLA	Draw Accurately

Solved Numerical Problems

1. Apply Bresenham's algorithm to draw a line from (20,15) and end point is (30,30)

Solution:

Starting Point (x_0, y_0) = (20,15) Ending Point (x_n, y_n) = (30,30) such that scanning takes place from left to right Slope $m = \frac{(30-15)}{(30-20)} = 1.5$

i.e. $|M| > 1$

Here:

$\Delta x = 30 - 20 = 10$

$\Delta y = 30 - 15 = 15$

$2\Delta x = 2 * 10 = 20$

$2\Delta y = 2 * 15 = 30$

Now from Bresenhams Algorithm for |M| > 1, and scanning from left to right

Initial Decision Parameter: $p_0 = p_0 = 2\Delta x - \Delta y$

If ($p_k < 0$) then

$X_{k+1} = X_k$ and

$Y_{k+1} = Y_k + 1$ and

$P_{k+1} = P_k + 2\Delta x$

otherwise

$X_{k+1} = X_{k+1}$ and

$Y_{k+1} = Y_k + 1$ and

$P_{k+1} = P_k + 2\Delta x - 2\Delta y$

Repeat Δy times

K	P_k	X_{k+1}, Y_{k+1}
0	5	(21,16)
1	–5	(21,17)
2	15	(22,18)
3	5	(23,19)
4	–5	(23,20)
5	15	(24,21)
6	5	(25,22)
7	–5	(25,23)
8	15	(26,24)
9	5	(27,25)
10	–5	(27,26)
11	15	(28,27)
12	5	(29,28)
13	–5	(29,29)
14	15	(30,30)

Therefore the digitized coordinates are $(x_0, y_0) = (20,15)$, $(x_1, y_1) = (21,16)$,.....,$(x_{15}, y_{15}) = (30,30)$

2. Apply Bresenham's algorithm to draw a line from (–3,0) and end point is(4,4)

Solution:

Starting Point $(x_0, y_0) = (-3, 0)$ Ending Point $(x_n, y_n) = (4,4)$ such that scanning takes place from left to right Slope $m = \frac{(4-0)}{(4+3)} = \frac{4}{5} = 0.57$

i.e. |M| < 1

Here:

$\Delta x = 4 + 3 = 7$

$\Delta y = 4 - 0 = 4$

$2\Delta x = 2 * 7 = 14$

$2\Delta y = 2 * 4 = 8$

Now from Bresenhams Algorithm for |M| < 1, and scanning from left to right

Initial Decision Parameter: $P_0 = 2\Delta y - \Delta x$

If ($p_x < 0$) then

$X_{k+1} = X_k + 1$ and

$Y_{k+1} = Y_k$ and

$P_{k+1} = P_k + 2\Delta y$

otherwise

$X_{k+1} = X_k + 1$ and

$Y_{k+1} = Y_{k+1}$ and

$P_{k+1} = P_k + 2\Delta y - 2\Delta x$

Repeat Δx times

K	P_k	X_{k+1}, Y_{k+1}
0	1	(–2,1)
1	–5	(–1,1)
2	3	(0,2)
3	–3	(1,2)
4	5	(2,3)
5	–1	(3,3)
6	7	(4,4)

Therefore the digitized coordinates are $(x_0,y_0) = (-3,0)$, $(x_1,y_1) = (-2,1)$, ... $(x_7, y_7) = (4,4)$

Note: If the initial point for a line with positive slope is right endpoint, then both X and Y decreases but for negative slope, the procedure is similar, except that now one coordinate decreases as the other increases.

3. Digitize the line with end points (20,10) and (30,18) using BLA.

Solution:

Here, Starting point of line = $(x_1, y_1) = (20,10)$ and

Ending point of line = $(x_2, y_2) = (30,18)$

Thus, slope of line, $m = \frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1}$

$= \frac{(18 - 10)}{(30 - 20)}$

$= \frac{8}{10}$

As the given points, it is clear that the line is moving left to right with the positive slop

$|m| = 0.8 < 1$

The initial decision parameter $(P_0) = 2\Delta y - \Delta x = 2*8 - 10 = 6$

Since, for the Bresenham's line drawing Algorithm of slope, $|m| \leq 1$, we have

If $P_k < 0$ (i.e. $d_1 - d_2$ is Negative)

Then,

$X_{k+1} = X_k + 1$

$Y_{k+1} = Y_k$

$P_k = P_k + 2\Delta y$

If $P_k \geq 0$

Then,

$X_{k+1} = X_k + 1$

$Y_{k+1} = Y_k + 1$

$P_k = P_k + 2\Delta y - 2\Delta x$

(20,10)

k	P_k	X_{k+1}	Y_{k+1}	(X_{k+1}, Y_{k+1})
0	6	20 + 1 = 21	11	(21, 11)
1	6 + 2 × 8 – 2 × 10 = 2	21 + 1=22	12	(22, 12)
2	2 + 2 × 8 – 2 × 10 = –2	22 + 1 = 23	12	(23, 12)
3	–2 + 2 × 8 = 14	23 + 1 = 24	13	(24, 13)
4	14 + 2 × 8 – 2 × 10 = 10	24 + 1 = 25	14	(25, 14)
5	10 + 2 × 8 – 2 × 10 = 6	25 + 1 = 26	15	(26, 15)
6	6 + 2 × 8 – 2 × 10 = 2	26 + 1 = 27	16	(27, 16)
7	2 + 2 × 8 – 2 × 10 = –2	27 + 1 = 28	16	(28, 16)
8	–2 + 2 × 8 = 14	28 + 1 = 29	17	(29, 17)
9	14 + 2 × 8 – 2 × 10 = 10	29 + 1 = 30	18	(30, 18)

4. Digitize the line with end points (1,0) and (3,3) using BLA.

Solution:

Here, Starting point of line = $(x_1, y_1) = (1,0)$ and

Ending point of line = $(x_2, y_2) = (3,3)$

Thus,

Slope of line, $m = \frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1}$

$= \frac{(3 - 0)}{(3 - 1)}$

$= \frac{3}{2}$

As the given points, it is clear that the line is moving left to right with the positive slope,

$|m| = 1.5 > 1$

Thus, the initial decision parameter

$(P_0) = 2\Delta x - \Delta y = 2*2 - 3 = 1$

$\Delta x = x_2 - x_1 = 3 - 1 = 2$

$\Delta y = y_2 - y_1 = 3 - 0 = 3$

we have

If $P_k < 0$

Then,

$X_{k+1} = X_k$

$Y_{k+1} = Y_k + 1$

$P_k = P_k + 2\Delta x$

If $P_k \geq 0$

Then,

$X_{k+1} = X_k + 1$

$Y_{k+1} = Y_k + 1$

$P_k = P_k + 2\Delta x - 2\Delta y$

k	P_k	X_{k+1}	Y_{k+1}	(X_{k+1}, Y_{k+1})
0	1	2	1	(2,1)
1	1+2*2–2*3 = –1	2	1+1 = 2	(2,2)
2	–1+2*2 = 3	3	2+1 = 3	(3,3)

5. Draw a line for (10,7) and (5,10) using BLA

Solution:

Here, $(X_1, Y_1) = (10, 7)$ & $(X_2, Y_2) = (5, 10)$

$M = \frac{(10-7)}{(5-10)} = \frac{-3}{5}$ -ve slope and $|M| < 1$

$\Delta x = |5 - 10| = 5$

$\Delta y = |10 - 7| = 3$

Here,

Initial (P_0) $= 2\Delta y - \Delta x$

$= 2 \times 3 - 5$

$= 1$

If $P_k <$

$X_{k+1} = X_k - 1$

$Y_{k+1} = Y_k$

$P_k = P_k + 2\Delta y$

If $P_k \geq 0$

$X_{k+1} = X_k - 1$

$Y_{k+1} = Y_k + 1$

$P_k = P_k + 2\Delta y - 2\Delta x$

(10, 7)

k	P_k	X_{k+1}	Y_{k+1}	(X_{k+1}, Y_{k+1})
0	1	9	8	(9,8)
1	$= 1 + 2 \times 3 - 2 \times 5 = -3$	8	8	(8,8)
2	$= -3 + 2 \times 3 = 3$	7	9	(7,9)
3	$= 3 + 2 \times 3 - 2 \times 5 = -1$	6	9	(6,9)
4	$= -1 + 2 \times 3 = 5$	5	10	(5,10)

6. Draw a line for (6,2) and (3,10) using BLA

Solution:

Here, $(X_1, Y_1) = (6,2)$ & $(X_2, Y_2) = (3,10)$

$M = \frac{(10-2)}{(3-6)} = \frac{8}{-3}$ -ve slope and $|M| > 1$

$\Delta x = |3 - 6| = 3$

$\Delta y = |10 - 2| = 8$

Here, Initial $(P_0) = 2\Delta x - \Delta y$

$= 2 \times 3 - 8$

$= -2$

If $P_k < 0$

$X_{k+1} = X_k$

$Y_{k+1} = Y_k + 1$

$P_k = P_k + 2\Delta x$

If $P_k \geq 0$

$X_{k+1} = X_k - 1$

$Y_{k+1} = Y_k + 1$

$P_k = P_k + 2\Delta x - 2\Delta y$

(6,2)

k	P_k	X_{k+1}	Y_{k+1}	(X_{k+1}, Y_{k+1})
0	-2	6	3	(6,3)
1	$= -2 + 2 \times 3 = 4$	5	4	(5,4)
2	$= 4 + 6 - 16 = -6$	5	5	(5,5)
3	$= -6 + 6 = 0$	4	7	(4,7)
4	$= 0 + 6 - 16 = -10$	4	8	(4,8)
5	$= -10 + 6 = -4$	4	9	(4,9)
6	$= -4 + 6 = 2$	3	10	(3,10)

7. Draw a line for (15,15) and (10,18) using BLA

Solution:

Here, $(X_1, Y_1) = (15,15)$ & $(X_2, Y_2) = (10,18)$

$M = \frac{(18-15)}{(10-15)} = \frac{3}{-5}$ -ve slope and $|M| < 1$

$\Delta x = |10 - 15| = 5$

$\Delta y = |18 - 15| = 3$

Here,

Initial (P_0) $= 2\Delta y - \Delta x$

$= 2 \times 3 - 5$

$= 1$

If $P_k <$

$X_{k+1} = X_k - 1$

$Y_{k+1} = Y_k$

$P_k = P_k + 2\Delta y$

If $P_k \geq 0$

$X_{k+1} = X_k - 1$

$Y_{k+1} = Y_k + 1$

$P_k = P_k + 2\Delta y - 2\Delta x$

(15, 15)

k	P_k	X_{k+1}	Y_{k+1}	(X_{k+1}, Y_{k+1})
0	1	14	16	(14,16)
1	$= 1 + 2 \times 3 - 2 \times 5 = -3$	13	16	(13,16)
2	$= -3 + 2 \times 3 = 3$	12	17	(12,17)
3	$= 3 + 2 \times 3 - 2 \times 5 = -1$	11	17	(11,17)
4	$= -1 + 2 \times 3 = 5$	10	18	(10,18)

8. Draw a line for (20,5) and (10,10) using BLA

Solution:

Here, $(X_1, Y_1) = (20,5)$ & $(X_2, Y_2) = (10,10)$

$M = \frac{(10-5)}{(10-20)} = \frac{5}{-10}$ -ve slope and $|M| < 1$

$\Delta x = |10 - 20| = 10$

$\Delta y = |10 - 5| = 5$

Here,

Initial (P_0) $= 2\Delta y - \Delta x$

$= 2 \times 5 - 10$

$= 0$

If $P_k <$

$X_{k+1} = X_k - 1$

$Y_{k+1} = Y_k$

$P_k = P_k + 2\Delta y$

If $P_k \geq 0$

$X_{k+1} = X_k - 1$

$Y_{k+1} = Y_k + 1$

$P_k = P_k + 2\Delta y - 2\Delta x$

(20, 5)

k	P_k	X_{k+1}	Y_{k+1}	(X_{k+1}, Y_{k+1})
0	0	19	6	(19,6)
1	$= 0 + 2 \times 5 - 2 \times 10 = -10$	18	6	(18,6)
2	$= -10 + 2 \times 5 = 0$	17	7	(17,7)
3	$= 0 + 2 \times 5 - 2 \times 10 = -10$	16	7	(16,7)
4	$= -10 + 2 \times 5 = 0$	15	8	(15,8)
5	$= 0 + 2 \times 5 - 2 \times 10 = -10$	14	8	(14,8)
6	$= -10 + 2 \times 5 = 0$	13	9	(13,9)
7	$= 0 + 2 \times 5 - 2 \times 10 = -10$	12	9	(12,9)
8	$= -10 + 2 \times 5 = 0$	11	10	(11,10)
9	$= 0 + 2 \times 5 - 2 \times 10 = -10$	10	10	(10,10)

9. Draw a line for (6,3) and (5,6) using BLA

Solution:

Here, $(X_1, Y_1) = (6,3)$ & $(X_2, Y_2) = (5,6)$

$M = \frac{(6-3)}{(5-6)} = \frac{3}{-1}$ -ve slope and $|M| > 1$

$\Delta x = |5 - 6| = 1$

$\Delta y = |6 - 3| = 3$

Here,

Initial (P_0) $= 2\Delta x - \Delta y$

$= 2 \times 1 - 3$

$= -1$

If $P_k < 0$

$X_{k+1} = X_k$

$Y_{k+1} = Y_k + 1$

$P_k = P_k + 2\Delta x$

If $P_k \geq 0$

$X_{k+1} = X_k - 1$

$Y_{k+1} = Y_k + 1$

$P_k = P_k + 2\Delta x - 2\Delta y$

(6,3)

k	P_k	X_{k+1}	Y_{k+1}	(X_{k+1}, Y_{k+1})
0	-1	6	4	(6,4)
1	$= -1 + 2 \times 1 = 1$	5	5	(5,5)
2	$= 1 + 2 \times 1 - 2 \times 3 = -3$	5	6	(5,6)

10. Draw a line for (10,10) and (14,13) using BLA line drawing algorithm.

Solution:

Slope: $m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{13 - 10}{14 - 10} = \frac{3}{4} < 1$

Tips: m < 1 conditions (Δx > Δy)

$d_1 - d_2 > 0$

$y_{k+1} = y_k + 1$

$x_{k+1} = x_k + 1$

$p_{k+1} = p_k + 2\Delta y - 2\Delta x$

$d_1 - d_2 < 0$

$x_{k+1} = x_k + 1$

$y_{k+1} = y_k$

$p_{k+1} = p_k + 2\Delta y$

Initial decision parameter

$p_0 = 2\Delta y - \Delta x$

$\Delta x = 14 - 10 = 4 \qquad 2\Delta x = 8$

$\Delta y = 13 - 10 = 3 \qquad 2\Delta y = 6$

Initial decision parameter

$p_0 = 2\Delta y - \Delta x$

$= 6 - 4 = 2$

n	P_k	X_{k+1}	Y_{k+1}
–	–	10	10
0	2	11	11
1	0	12	12
2	–2	13	12
3	4	14	13

11. Use BLA to draw line which end points are A(–3,0), B(4,4)

Solution:

$x_1 = -3 \qquad x_2 = 4$

$y_1 = 0 \qquad y_2 = 4$

$\Delta x = 4 - (-3) = 7 \qquad 2\Delta x = 14$

$\Delta y = 4 - 0 = 4 \qquad 2\Delta y = 8$

$m = \frac{4}{7} < 1 \ [\Delta x > \Delta y]$

$p_0 = 2\Delta y - \Delta x = 8 - 7 = 1$

n	P_k	X_{k+1}	Y_{k+1}
–	–	–3	0
0	1	–2	1
1	–5	–1	1
2	3	0	2

n	P_k	X_{k+1}	Y_{k+1}
3	–3	1	2
4	5	2	3
5	–1	3	3
6	7	4	4

Circle Generation Algorithm

A circle is a geometric shape that is defined as a set of points in a plane that are equidistant from a fixed center point. In simpler terms, it is a perfectly round shape with no corners or edges. The distance from the center of a circle to any point on its boundary is called the radius, and the longest distance across the circle passing through the center is called the diameter. Circle has 8 symmetry (i.e. 8 octants each of 45°)

$(x-h)^2 + (y-k)^2 = r^2$

Problems in drawing Circle

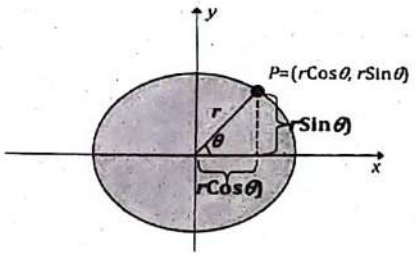
- **Uneven curvature:** One of the most common issues is creating a circle with uneven or wobbly curves. This usually happens due to shaky hands or inconsistent pressure while drawing.
- **Off-center circle:** It can be difficult to accurately position the center point of the circle, resulting in a circle that is not centered properly on the page or object.
- **Jagged edges:** When drawing with certain tools, such as pencils or markers with a blunt tip, the edges of the circle may appear jagged instead of smooth.

Circle equations

- Polar form

  $x = r\cos\theta$

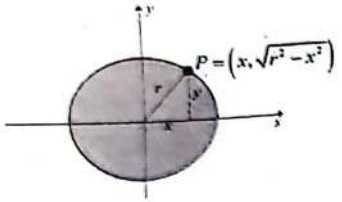
  $y = r\sin\theta$ (r = radius or circle)

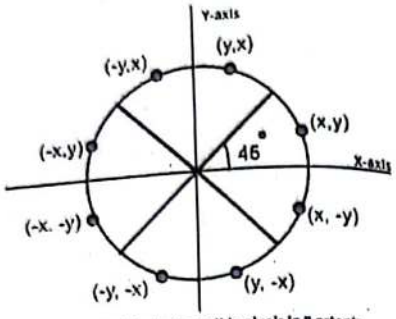


Cartesian form

- Use Pythagoras theorem

$x^2 + y^2 = r^2$





For each pixel (x,y) all possible pixels in 8 octants

Bresenham's circle algorithm

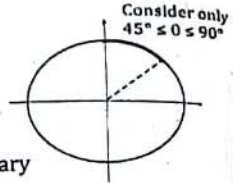
General principle

- **The circle function:**

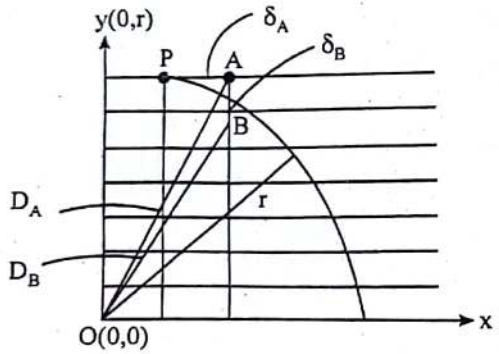
$f_{circle}(x, y) = x^2 + y^2 - r^2$

and

$$f_{circle}(x, y) = \begin{cases} < 0 \text{ if } (x,y) \text{ is inside the circle boundary} \\ = 0 \text{ if } (x, y) \text{ is on the circle boundary} \\ > 0 \text{ if } (x, y) \text{ is outside the circle boundary} \end{cases}$$



BLA circle drawing algorithm



Here,

$P(x_i, y_i)$

$A(x_i + 1, y_i)$

$B(x_i + 1, y_i - 1)$

Here, x will surely

Increase but y ..... or ..... not decrease.

Let D_A be the distance between origin and A and D_B be the distance between origin and B.

δA and δB are the distance between A & circle surface and B & circle surface.

So, $D_A = \sqrt{(x_i + 1)^2 + y_i^2}$, $D_B = \sqrt{(x_i + 1)^2 + (y_i - 1)^2}$

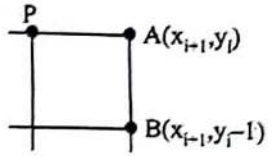
$\delta A = D_A - r(+ve)$ $\delta B = D_B - r(-ve)$

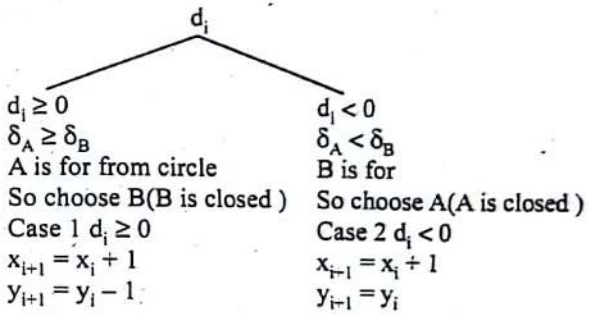
To avoid square root in derivation of decision variable we use.

$\delta A = D_A^2 - r^2$, $\delta B = D_B^2 - r^2$

$\therefore$ $\delta A = (x_i + 1)^2 + y_i^2 - r^2$, $\delta B = (x_i + 1)^2 + (y_i - 1)^2 - r^2$

δA is always +ve and δB is always -ve. So we define decision variable di as $d_i = \delta A + \delta B$





Starting point is always (0,r)

So, x = 0, y = r then

$d_i = \delta A + \delta B$

$= (x_i + 1)^2 + y_i^2 - r^2 + (x_i + 1)^2 + (y_i - 1)^2 - r^2$

$= 1^2 + r^2 - r^2 + 1^2 - r^2 - 2r + 1 - r^2$

$= 3 - 2r$ (which is initial decision parameter)

Again,

$d_i = (x_i + 1)^2 + y_i^2 - r^1 + (x_i + 1)^2 + (y_i - 1)^2 - r^2$

$= 2(x_i + 1)^2 + 2y_i^2 - 2r^2 - 2y_i + 1$

At i + 1 pixel

$d_{i+1} = 2(x_{i+1} + 1) + 2y_{i+1}^2 - 2r^2 - 2y_{i+1} + 1$

We know that x will surely increase so

$x_{i+1} = x_i + 1$

$\therefore \quad d_{i+1} = 2(x_i + 1 + 1)^2 + 2y_{i+1}^2 - 2r^2 - 2y_{i+1} + 1$

$= 2(x_i + 2)^2 + 2y_{i+1}^2 - 2r^2 - 2y_{i+1} + 1$

Now,

$d_{i+1} - d_i = 2(x_i + 2)^2 - 2y_{i+1}^2 - 2r^2 - 2y_{i+1} + 1 - 2(x_i + 1)^2 - 2y_i^2 + 2r^2 + 2y_i - 1$

On solving:

$d_{i+1} = d_i + 4x_i + 6 + 2y_{i+1}^2 - 2y_{i+1} - 2y_i^2 + 2y_i$

Case 1: $d_i > 0$

$x_{i+1} = x_i + 1$

$y_{i+1} = y_i - 1$

$d_{i+1} = d_i + 4x_i + 6 + 2(y_i - 1)^2 - 2(y_i - 1) - 2y_i^2 + 2y_i$

$= d_i + 4x_i + 6 + 2(y_i^2 - 2y_i + 1) - 2y_i + 2 - 2y_i^2 + 2y_i$

On solving

$= d_i + 4(x_i - y_i) + 10$

Case 2: $d_i < 0$

$x_{i+1} = x_i + 1$

$y_{i+1} = y_i$

$d_{i+1} = d_i + yx_i + 6$

Initial decision parameter

$x_0 = 0$

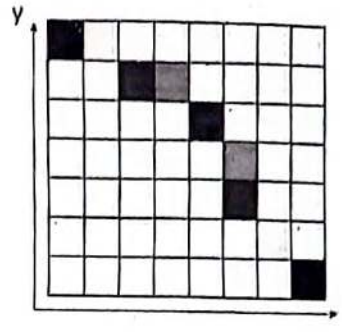
$y_0 = r$

$p_0 = [1^2 + r^2 - r^2] + [1^2 + (r - 1)^2 + r^2] = 3 - 2r$

if $p_i < 0$ then
$y_{i+1} = y_i$
$p_{i+1} = p_i + 4x_i + 6$
else if $p_i \geq 0$ then
$y_{i+1} = y_i - 1$
$p_{i+1} = p_i + 4(x_i - y_i) + 10$
} $x_{i+1} = x_i + 1$

Stop when $x_i \geq y_i$ and determine symmetry points in the other octants

Draw the circle of radius 7 pixel with center (0,0) using Bresenhams circle drawing algorithm.



- We start from a point (0,r) = (0,7)
- We know initial decision parameter = 3 – 2r = 3 – 2 × 7 = –11
- We know $d_i \geq 0$ then we use $d_{i+1} = d_i + 4(X_i - Y_i) + 10$
- And if $d_i < 0$ then we use $d_{i+1} = d_i + 4X_i + 6$

n	X_i	Y_i	X_{i+1}	Y_{i+1}	Decision parameter	Plot
0	0	7			–11	(0,7)
1	0	7	1	7	–5	(1,7)
2	1	7	2	7	5	(2,7)
3	2	7	3	6	–5	(3,6)
4	3	6	4	6	13	(4,6)
5	4	6	5	5	End	(5,5)

Other points of 1st quadrant:

(6,4), (6,3), (7,2), (7,1), (7,0)

We can find other points as well using 8 way symmetry.

Classwork

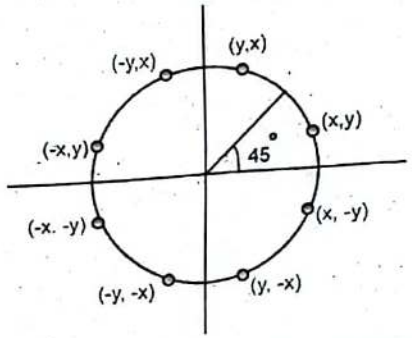
Draw the circle of radius 7 pixel with center (5,6) using Bresenhams circle drawing algorithm.

- Hint start same as (0,0) and then add (5,6)

n	X_i	Y_i	X_{i+1}	Y_{i+1}	Decision parameter	Plot
0	0	7			–11	(0,7) + (5,6) = (5,13)
1	0	7	1	7	–5	(1,7) + (5,6) = (6,13)
2	1	7	2	7	5	(2,7) + (5,6) = (7,13)
3	2	7	3	6	–5	(3,6) + (5,6) = (8,12)
4	3	6	4	6	13	(4,6) + (5,6) = (8,12)
5	4	6	5	5	End	(5,5) + (5,6) = (10,11)

Midpoint circle drawing algorithm

- A circle is a symmetrical figure since the shape of the circle is similar in each quadrant as shown in the figure.

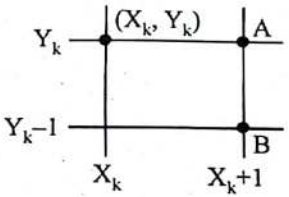


- The calculation of a point (X , Y) in one of octants yields the circle points shown for the other seven octants just by reflection.
  - For example if octant (0 to 45) is generated ,the second octant can be obtained by reflection through the line Y=X to yield the first quadrant.
  - The result in first quadrant are reflected through the line X=0, to obtain those in the second quadrant.
  - The combined results on the upper semicircle are reflected through the line Y=0 to complete the circle.
- The equation of the circle centered at (0,0) is

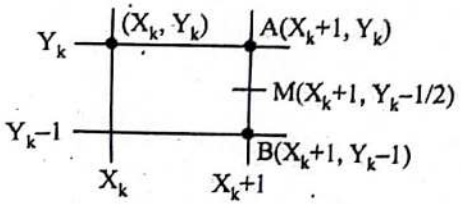
  $F_{circle}(X, Y) = X^2 + Y^2 - r^2$ .....(1)
- The relative position of any point (X,Y) can be determined by checking the sign of the circle function.

$$F_{circle}(X,Y)\begin{cases} < 0 \text{ if } (X,Y) \text{ is inside the circle boundary} \\ = 0 \text{ if } (X,Y) \text{ is on the circle boundary} \\ > 0 \text{ is outside the circle boundary} \end{cases}$$

- Here, the circle function is the decision parameter and we can set up incremental calculations for the function as we did in the algorithm.
- Let us assume that the pixel location is initially at position of (X_k, Y_k) as shown in figure.



- Now, we determine whether the next plotted pixel location lies at $A(X_k + 1, Y_k)$ or $B(X_k + 1, Y_k - 1)$.
- We take midpoint M between these two candid pixels at sampling position $X_k + 1$ as shown in figure below:



- The decision parameter is given by

  $P_k = f_{circle}(X_k + 1, Y_k - 1/2)$

  $= (X_k + 1)^2 + (Y_k - 12)^2 - r^2$ ..... equation (2)
- At $(k + 1)^{th}$ position,

  $P_{k+1} = (X_{k+1} + 1)^2 + (Y_{k+1} - 1/2)^2 - r^2$ ..... equation (3)
- Subtracting equation (2) from (3)

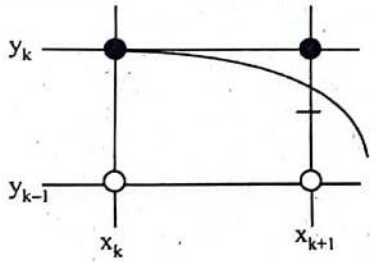
  $P_{k+1} = P_k + (X_{k+1} + 1)^2 - (X_k + 1)^2 + (Y_{k+1} - 1/2)^2 - (Y_k - 1/2)^2$ .....(4)

Case 1: $P_k < 0$

- The point M lies inside the circle i.e. next pixel position will be at A.
- Then $X_{k+1} = X_k + 1$ and $Y_{k+1} = Y_k$
- Equation (4) becomes as;

$$\begin{aligned} P_{k+1} &= P_k + (X_{k+1} + 1)^2 - (X_k + 1)^2 + (Y_{k+1} - 1/2)^2 - (Y_k - 1/2)^2 \\ &= P_k + (X_k + 1 + 1)^2 - (X_k + 1)^2 + (Y_k - 1/2)^2 - (Y_k - 1/2)^2 \\ &= P_k + (X_k + 2)^2 - (X_k + 1)^2 \\ &= P_k + X_k^2 + 4X_k + 4 - X_k^2 - 2X_k - 1 \\ &= P_k + 2X_k + 3 \\ &= P_k + 2(X_{k+1} - 1) + 3 \\ &= P_k + 2X_{k+1} - 2 + 3 \end{aligned}$$

$\therefore P_{k+1} = P_k + 2X_{k+1} + 1$

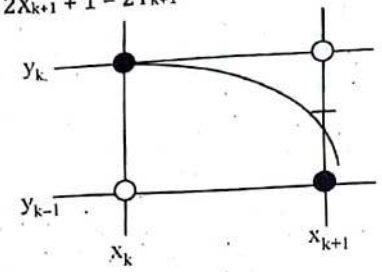


Case 2: $P_k >= 0$

- The point M lies outside the circle i.e. next pixel position will be at B.
- Then $X_{k+1} = X_k + 1$ and $Y_{k+1} = Y_k - 1$
- Equation becomes as;

$$P_{k+1} = P_k + (X_{k+1} + 1)^2 - (X_k + 1)^2 + (Y_{k+1} - 1/2)^2 - (Y_k - 1/2)^2$$
$$= P_k + (X_k + 1 + 1)^2 - (X_k + 1)^2 + (Y_k - 1 - 1/2)^2 - (Y_k - 1/2)^2$$
$$= P_k + (X_k + 2)^2 - (X_k + 1)^2 + (Y_k - 3/2)^2 - (Y_k - 1/2)^2$$
$$= P_k + X_k^2 + 4X_k + 4 - X_k^2 - 2X_k - 1 + Y_k^2 + 9/4 - 3Y_k - Y_k^2 + Y_k - 1/4$$
$$= P_k + 2X_k + 3 + 2 - 2Y_k$$
$$= P_k + 2(X_{k+1} - 1) + 3 + 2 - 2(Y_{k+1} + 1)$$
$$= P_k + 2X_{k+1} - 2 + 3 + 2 - 2Y_{k+1} - 2$$

$\therefore P_{k+1} = P_k + 2X_{k+1} + 1 - 2Y_{k+1}$



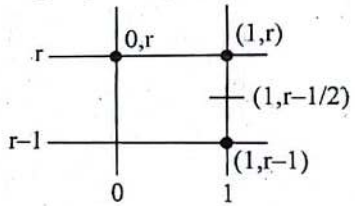
Initial decision parameter

- The initial decision parameter is obtained by evaluating the circle function at start position $(X_0,Y_0) = (0,r)$.

i.e. $P_0 = f_{circle}(1, r - 1/2)$

$$= 1 + r^2 + 1/4 - r - r^2$$
$$= \frac{5}{4 - r}$$
$$= 1.25 - r$$

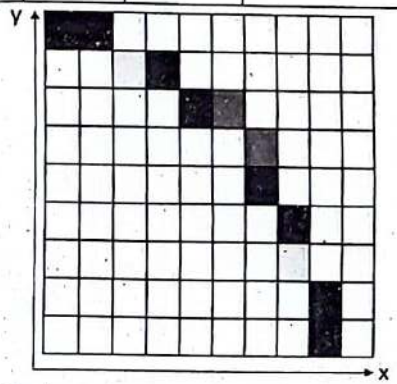
$\therefore P_0 = 1 - r$ [taking only integer part]



- Draw the circle of radius 9 pixel with center (0,0) using midpoint circle drawing algorithm.

- We start from a point (0,r) = (0,7)
- We know Initial decision parameter = 1 – r = 1 – 9 = –8
- We know $d_i \geq 0$ then we use $d_{i+1} = d_i + 2(x_i - y_i) + 5$ or $d_{i+1} = d_i + 2(x_{i+1} - y_{i+1}) + 1$
- And if $d_i < 0$ then we use $d_{i+1} = d_i + 2x_i + 3$ or $d_i + 2x_{i+1} + 1$

n	X_i	Y_i	X_{i+1}	Y_{i+1}	Decision parameter	Plot
0	0	9			–8	(0,9)
1	0	9	1	9	–5	(1,9)
2	1	9	2	9	0	(2,9)
3	2	9	3	8	–9	(3,8)
4	3	8	4	8	0	(4,8)
5	4	8	5	7	–3	(5,7)
6	5	7	6	7	10	(6,7)
7	6	7	7	6	End	(7,6)



Other points of 1st Quadrant

(7,5), (8,4), (8,3), (9,2), (9,1), (9,0)

We can find other points as well using 8 way symmetry.

Solved Numerical Problems

1. Digitized the circle with radius 10

Solution:

Initial point = $(X_0, Y_0) = (0,r) = (0,10)$ and origin = (0,0)

Initial decision parameter $(p_0) = 1 - r = 1 - 10 = -9$

From midpoint circle algorithm we have

If $P < 0$

Plot $(x_k + 1, y_k)$

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

Else $(P \geq 0)$

Plot $(x_k + 1, y_k - 1)$

$P_{k+1} = P_k + 2x_{k+1} - 2y_{k+1} + 1$

Thus,

Now, the successive decision parameter values and position along the circle path are determined as follow.

k	P_k	X_{k+1}	Y_{k+1}	(X_{k+1}, Y_{k+1})
0	-9	1	10	(1,10)
1	$-9 \times 2 \times 1 + 1 = 6$	2	10	(2,10)
2	$-6 + 2 \times 2 + 1 = -1$	3	10	(3,10)
3	$-1 + 2 \times 3 + 1 = 6$	4	9	(4,9)
4	$6 + 2 \times 4 - 2 \times 9 + 1 = -3$	5	9	(5,9)
5	$-3 + 2 \times 5 + 1 = 8$	6	8	(6,8)
6	$8 + 2 \times 6 - 2 \times 8 + 1 = 6$	7	7	(7,7)
7	$5 + 2 \times 7 - 2 \times 7 + 1 = 5$	8	6	[stop and discard this point since x > y]

Hence, the digitized coordinates for circle in first octant are (x_0, y_0) = (0,10), (x_1, y_1) = (1,10) .....(x_7, y_7) = (7,7)

Note: For each calculated point we need to apply symmetry principle to obtain eight different coordinates one for each of the eight different octants.

2. Digitized a circle with radius r = 12 and centered at (250,300)

Solution:

Note: When centre is not origin, we first calculate the octants points of the circle in the same way as the centre at origin then odd the given circle center on each calculated pixels.

Here, Initial point (x_0, y_0) (0,r) = (0,12) and origin = (0,0)

Initial decision parameter $(p_0) = 1 - 12 = 1 - 12 = -11$

From midpoint circle algorithm we have

If $P < 0$

Plot $(x_k + 1, y_k)$

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

Else $(P \geq 0)$

Plot $(x_k + 1, y_k - 1)$

$P_{k+1} = P_k + 2x_{k+1} - 2y_{k+1} + 1$

Thus,

Now, the successive decision parameter values and position along the circle path are determined as follow.

k	P_k	X_{k+1}	Y_{k+1}	(X_{k+1}, Y_{k+1}) at (0,0)	(X_{k+1}, Y_{k+1}) at (250,300)
0	-11	1	12	(1,12)	(250+1, 300+12) = (251,312)
1	$-11 + 2 \times 1 + 1 = -8$	2	12	(2,12)	(250+2, 300+12) = (252,312)
2	$-8 + 2 \times 1 + 1 = -3$	3	12	(3,12)	(250+3, 300+12) = (253,312)
3	$-3 + 2 \times 3 + 1 = 4$	4	11	(4,11)	(250+4, 300+11) = (254,311)
4	$4 + 2 \times 3 - 2 \times 11 + 1 = -9$	5	11	(5,11)	(250+5, 300+11) = (255,311)
5	$-9 + 2 \times 5 + 1 = 2$	6	10	(6,10)	(250+6, 300+10) = (256,310)
6	$2 + 2 \times 6 - 2 \times 10 + 1 = -5$	7	10	(7,10)	(250+7, 300+10) = (257,310)
7	$-5 + 2 \times 7 + 1 = 10$	8	9	(8,9)	(250+8, 300+10) = (258,309)
8	$10 + 16 - 18 + 1 = 9$	9	8	(Discard and stop since X>Y)	

Therefore the digitized coordinates are (x_0, y_0) = (250+0, 300+12) = (250,312), (x_1, y_1) = (251, 312)..... (x_8, y_8) = (258,312)

Digitize a circle with a radius 9 and center at(6,7)

Here, the initial decision parameter (P_0)

$P_0 = 1 - r = 1 - 9 = -8$

Since, for the midpoint circle algorithm of starting point (0,r) & centre at origin (0,0) rotating at clockwise direction, we have

If $P < 0$

Plot $(x_k + 1, y_k)$

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

Else ($P \geq 0$)

Plot $(x_k + 1, y_k - 1)$

$P_{k+1} = P_k + 2x_{k+1} - 2y_{k+1} + 1$

k	P_k	X_{k+1}	Y_{k+1}	(X_{k+1}, Y_{k+1}) at (0,0)	(X_{k+1}, Y_{k+1}) at (6,7)
0	−8	0+1=1	9	(1,9)	(1+6,9+7) = (7,16)
1	= −8 + 2 × 1 + 1 = −5	1+1=2	9	(2,9)	(2+6,9+7) = (8,16)
2	= −5 + 2 × 2 + 1 = 0	2+1=3	8	(3,8)	(3+6,8+7) = (9,15)
3	= 0 + 2 × 3 − 2 × 8 + 1 = −9	3+1=4	8	(4,8)	(4+6,8+7)=(10,15)
4	= −9 + 2 × 4 + 1 = 0	4+1=5	7	(5,7)	(5+6,8+7)=(11,15)
5	= 0 + 2 × 5 − 2 × 7 + 1 = −3	5+1=6	7	(6,7)	(6+6,7+7)=(12,14)
6	= −3 + 2 × 6 + 1 = 10	6+1=7	6	(7,6)	(7+6,6+7)=(13,13)

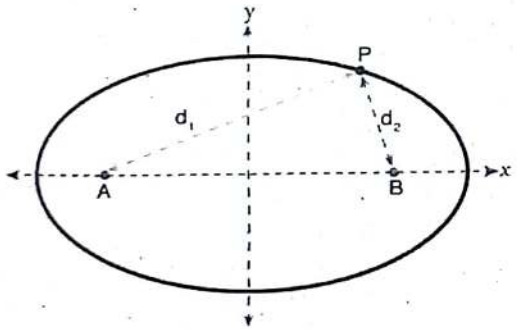
Ellipse Generation Algorithm

An ellipse is the locus of all those points in a plane such that the sum of their distances from two fixed points in the plane, is constant. The fixed points are known as the foci (singular focus), which are surrounded by the curve. The fixed line is directrix and the constant ratio is eccentricity of ellipse. Eccentricity is a factor of the ellipse, which demonstrates the elongation of it and is denoted by 'e' elongation of it and is denoted by 'e'.

Major and Minor Axis

Ellipse is defined by its two-axis along x and y-axis:

- Major axis
- Minor Axis



Here,

P is any point on ellipse,

A & B are the foci,

$PA = d_1$, $PB = d_2$, & $d_1 + d_2 = \text{constant}$

The major axis is the longest diameter of the ellipse (usually denoted by 'a'), going through the center from one end to the other, at the broad part of the ellipse. Whereas the minor axis is the shortest diameter of ellipse (denoted by 'b'), crossing through the center at the narrowest part. Half of major axis is called the semi-major axis and half of minor axis is called semi-minor axis.

Midpoint Ellipse Algorithm

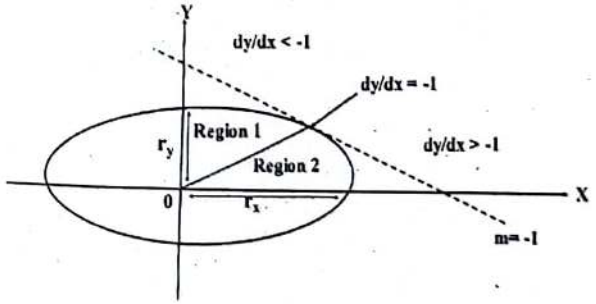
- Ellipse is an elongated circle. Therefore, elliptical curves can be generated by modifying circle drawing procedure to consider the different dimensions of ellipse along the major and minor axes.
- The equation of an ellipse whose centre at origin in standard form is given by

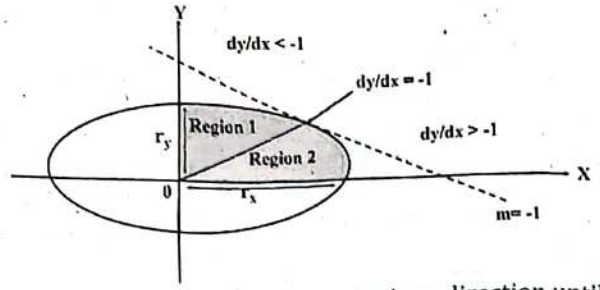
$$\frac{x^2}{r_x^2} + \frac{y^2}{r_y^2} = 1 \quad \text{.....equation (1)}$$

$$\therefore \; y = \pm r_y \sqrt{1 - \frac{x^2}{r_x^2}}$$

- Where r_x and r_y are the radii of the ellipse along the x- and y-direction.
- Midpoint ellipse algorithm is a method for drawing ellipses in computer graphics.
- The method is applied throughout the first quadrant in two parts i.e. region 1 and region 2.
- We are forming these two regions by considering the slope of the curve.

  If the slope of the curve is less than −1 then we are in region 1 and when the slope of the curve becomes greater than −1 then we are in region 2 as shown in figure below.





- Starting at $(0,r_y)$, we take unit steps in the x-direction until we reach the boundary of the region 1 and region 2. Then we switch to unit steps in y-direction over the remainder of the curve in the first quadrant.
- The slope of the ellipse is given by

$$\frac{dy}{dx} = -\frac{2ry^2x}{2rx^2y}$$ .....equation (2) [From equation 1]

- Ellipse function can be defined as:

$f_{ellipse}(x, y) = r_y^2x^2 + r_x^2y^2 - r_x^2r_y^2$ ..... equation (3)

- From co-ordinate geometry,
  - $f_{ellipse}(x, y) < 0$ which means (x, y) is inside the ellipse boundary.
  - $f_{ellipse}(x, y) > 0$ which means (x, y) is outside the ellipse boundary.
  - $f_{ellipse}(x, y) = 0$ which means (x, y) is on the ellipse boundary.
- At the boundary of region 1 and region 2,

slope = −1

or, $\frac{dy}{dx} = -1$

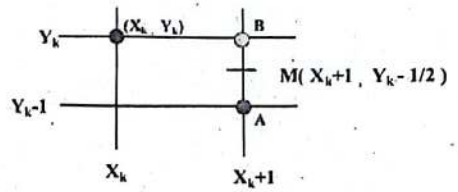
or, $\frac{-2ry^2x}{2rx^2y} = -1$

$\therefore\ 2ry^2x = 2rx^2y$

- Therefore we move out from region 1 whenever

$2ry^2x \geq 2r_x^2y$

- Assuming initial position at (X_k, Y_k), we determine the next pixel position along the ellipse path.



- Our choices are the pixel positions at $A(X_k+1, Y_k-1)$ and $B(X_k+1, Y_k)$.
- The decision parameter of the midpoint at k^{th} step,
- $P1_k = f_{ellipse}(X_k + 1, Y_k - 1/2)$

$= r_y^2(X_k + 1)^2 + r_x^2(Y_k - 1/2)^2 - r_x^2r_y^2$ .....equation (4)

- At $(k + 1)^{th}$ step, equation (4) becomes as ;
- $P1_{k+1} = f_{ellipse}(X_{k+1} + 1, Y_{k+1} - 1/2)$

$= r_y^2(X_{k+1} + 1)^2 + r_x^2(Y_{k+1} - 1/2)^2 - r_x^2r_y^2$ .....equation (5)

- Solving equation (4) and equation (5);
- $P1_{k+1} = P1_k + r_y^2[(X_{k+1} + 1)^2 - (X_k + 1)^2] + r_x^2[(Y_{k+1} - 1/2)^2 - (Y_k - 1/2)^2]$

.....equation (6)

Case 1: $P1_k < 0$

The midpoint M lies inside the ellipse boundary and the pixel on the scan line Y_k is closer to the ellipse boundary.

So, $X_{k+1} = X_k + 1$

And $Y_{k+1} = Y_k$

The equation (6) becomes as;

$P1_{k+1} = P1_k + r_y^2[(X_k + 1 + 1)^2 - (X_k + 1)^2] + r_x^2[(Y_k - 1/2)^2 - (Y_k - 1/2)^2]$

or, $P1_{k+1} = P1_k + 2r_y^2X_k + 3r_y^2$

or, $P1_{k+1} = P1_k + 2r_y^2(X_{k+1} - 1) + 3r_y^2$

or, $P1_{k+1} = P1_k + 2r_y^2X_{k+1} - 2r_y^2 + 3r_y^2$

$\therefore\ P1_{k+1} = P1_k + 2r_y^2X_{k+1} + r_y^2$

Case 2: $P1_k \geq 0$

The midpoint M lies outside the ellipse boundary and the pixel on the scan line $Y_k - 1$ is closer to the ellipse boundary.

So, $X_{k+1} = X_k + 1$

And $Y_{k+1} = Y_k - 1$

Then equation (6) becomes as;

$P1_{k+1} = P1_k + r_y^2[(X_k+1+1)^2 - (X_k+1)^2 + r_x^2[(Y_k-1-1/2)^2 - (Y_k - 1/2)^2]$

or, $P1_{k+1} = P1_k + 2r_y^2X_{k+1} + r_y^2 + r_x^2[(Y_k - 3/2)^2 - (Y_k - 1/2)^2]$

or, $P1_{k+1} = P1_k + 2r_y^2X_{k+1} + r_y^2 + r_x^2(2 - 2Y_k)$

or, $P1_{k+1} = P1_k + 2r_y^2X_{k+1} + r_y^2 + 2r_x^2 - 2r_x^2Y_k$

or, $P1_{k+1} = P1_k + 2r_y^2X_{k+1} + r_y^2 + 2r_x^2 - 2r_x^2(Y_{k+1} + 1)$

$\therefore\ P1_{k+1} = P1_k + 2r_y^2X_{k+1} - 2r_x^2Y_{k+1} + r_y^2$

- In region 1, the initial value of decision parameter can be obtained by evaluating the ellipse function at the start position $(X_0, Y_0) = (0,r_y)$.

- Initial decision parameter

$P1_0 = f_{ellipse}(1, r_y - 1/2)$

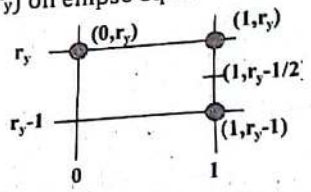
$= r_y^2 1^2 + r_x^2(r_y - 1/2)^2 - r_x^2 r_y^2$

$= r_y^2 + r_x^2(r_y^2 + 1/4 - r_y) - r_x^2 r_y^2$

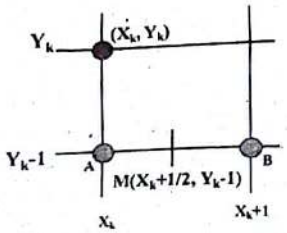
$= r_y^2 + r_x^2 r_y^2 + r_x^2/4 - r_y r_x^2 - r_x^2 r_y^2$

$\therefore \quad P1_0 = r_y^2 - r_x^2 r_y + r_x^2/4$

Or simply put $(0, r_y)$ on ellipse equation we will get



- Over region 2, we sample at unit steps along y-direction as shown in figure.



- The decision parameter of the midpoint at k^{th} step,

$P2_k = f_{ellipse}(X_k + 1/2, Y_k - 1)$

$= r_y^2(X_k + 1/2)^2 + r_x^2(Y_k - 1)^2 - r_x^2 r_y^2$ .....equation (7)

- At $(k + 1)^{th}$ step, equation (7) becomes as;

$P2_{k+1} = f_{ellipse}(X_{k+1} + 1/2, Y_{k+1} - 1)$

$= r_y^2(X_{k+1} + 1/2)^2 + r_x^2(Y_{k+1} - 1)^2 - r_x^2 r_y^2$ ..... equation (8)

- Solving equation (7) and (8);

$P2_{k+1} = P2_k + r_y^2[(X_{k+1}+1/2)^2 - (X_k+1/2)^2] + r_x^2[(Y_{k+1}-1)^2 - (Y_k-1)^2]$

.....equation (9)

$P2_{k+1} = P2_k + r_y^2[(X_{k+1} + 1/2)^2 - (X_k + 1/2)^2] + r_x^2[(Y_{k+1}-1)^2 - (Y_k-1)^2]$

$= P2_k + r_y^2[(X_{k+1}+1/2)^2 - (X_k + 1/2)^2] + r_x^2[(Y_{k+1} - 1)^2 - (Y_k-1)^2]$

Case 1: $P2_k \le 0$

The midpoint M lies inside the ellipse boundary and the pixel on the scan line $X_k + 1$ is closer to the ellipse boundary.

So, $X_{k+1} = X_k + 1$

And $Y_{k+1} = Y_k - 1$

The equation (9) becomes as;

$P2_{k+1} = P2_k + r_y^2[(X_k+1/2)^2 - (X_k+1/2)^2] + r_x^2[(Y_k-1-1)^2 - (Y_k-.1)^2]$

or, $P2_{k+1} = P2_k + r_y^2[(X_k+3/2)^2 - (X_k+1/2)^2] + r_x^2[(Y_k - 2)^2 - (Y_k - 1)^2]$

$\therefore \quad P2_{k+1} = P2_k + 2r_y^2 X_{k+1} - 2r_x^2 Y_{k+1} + r_x^2$

Solution portion skip if u want

$P2_{k+1} = P2_k + r_y^2[(X_k+1+1/2)^2 - (X_k+1/2)^2] + r_x^2[(Y_k-1-1)^2 - (Y_k-1)^2]$

or, $P2_{k+1} = P2_k + r_y^2[(X_k+3/2)^2-(X_k+1/2)^2]+r_x^2[(Y_k-2)^2-(Y_k-1)^2]$

or, $P2_{k+1} = P2_k+r_y^2[X_k^2+9/4+3X_k-X_k^2-1/4-X_k]+r_x^2[Y_k^2+4-4Y_k-Y_k^2-1+2Y_k]$

or, $P2_{k+1} = P2_k + r_y^2(2 + 2X_k) + r_x^2(3 - 2Y_k)$

or, $P2_{k+1} = P2_k + 2r_y^2 + 2r_y^2 X_k + 3r_x^2 - 2r_x^2(Y_{k+1} + 1)$

or, $P2_{k+1} = P2_k + 2r_y^2 + 2r_y^2 X_{k+1} - 2r_y^2 + 3r_x^2 - 2r_x^2 Y_{k+1} - 2r_x^2$

$\therefore \quad P2_{k+1} = P2_k + 2r_y^2 X_{k+1} - 2r_x^2 Y_{k+1} + r_x^2$

Case 2: $P2_k > 0$

The midpoint M lies outside the ellipse boundary and the pixel on the scan line X_k is closer to the ellipse boundary.

So, $X_{k+1} = X_k$

and, $Y_{k+1} = Y_k - 1$

Then equation (9) becomes as;

$P2_{k+1} = P2_k + r_y^2[(X_{k+1/2})^2 - (X_k+1/2)^2] + r_x^2[(Y_k-1-1)^2 - (Y_k - 1)^2]$

or, $P2_{k+1} = P2_k + r_x^2[(Y_k - 2)^2 - (Y_k - 1)^2]$

or, $P2_{k+1} = P2_k + r_x^2[Y_k^2 + 4 - 4Y_k - Y_k^2 - 1 + 2Y_k]$

or, $P2_{k+1} = P2_k + r_x^2(3 - 2Y_k)$

or, $P2_{k+1} = P2_k + 3r_x^2 - 2r_x^2 Y_k$

or, $P2_{k+1} = P2_k + 3r_x^2 - 2r_x^2(Y_{k+1} + 1$

or, $P2_{k+1} = P2_k + 3r_x^2 - 2r_x^2 Y_{k+1} - 2r_x^2$

$\therefore P2_{k+1} = P2_k - 2r_x^2 Y_{k+1} + r_x^2$

- In region 2, the initial value of the decision parameter can be obtained by evaluating the ellipse function at the last position in region 1.

$P2_0 = f_{ellipse}(X_0 + 1/2, Y_0 - 1)$

$\therefore \quad P2_0 = r_y^2(X_0 + 1/2)^2 + r_x^2(Y_0 - 1)^2 - r_x^2 r_y^2$

Where (X_0, Y_0) is the last position in the region 1.

Algorithm

1. Take input radius along x axis and y axis and obtain center of ellipse.
2. Initially, we assume ellipse to be centered at origin and the first point as: $(x_0, y_0) = (0, r_y)$.
3. Obtain the initial decision parameter for region 1 as:
   $P1_0 = r_y^2 - r_x^2 r_y + r_x^2/4$
4. For every X_k position in region 1 until $(2r_y^2X_{k+1} > 2r_x^2Y_{k+1})$
   - Check if $P1_k < 0$ then the next point is (X_{k+1}, Y_k) and $P1_{k+1} = P1_k + 2r_y^2X_{k+1} + r_y^2$
   - Otherwise, the next point is (X_{k+1}, Y_{k-1}) and $P1_{k+1} = P1_k + 2r_y^2X_{k+1} - 2r_x^2Y_{k+1} + r_y^2$
5. Obtain the initial decision parameter for region 2 using the last point (X_0, Y_0) of region 1 as:
   $P2_0 = r_y^2(X_0 + 1/2)^2 + r_x^2(Y_0 - 1)^2 - r_x^2r_y^2$
6. At each Y_k in region 2 starting at k = 0 perform the following tasks.
   - Check if $P2_k < 0$ the next point is $(X_k + 1, Y_k - 1)$ and $P2_{k+1} = P2_k + 2r_y^2X_{k+1} - 2r_x^2Y_{k+1} + r_x^2$
   - Otherwise, the next point is $(X_k, Y_k - 1)$ and $P2_{k+1} = P2_k - 2r_x^2Y_{k+1} + r_x^2$
7. Repeat step 6 until we get one complete quadrant of ellipse i.e. the pixel point $(r_x, 0)$.
8. Now obtain the symmetric points in the three quadrants and plot the coordinate value as: $X = X + X_c$, $Y = Y + Y_c$

Draw and Ellipse with radii $r_x = 8$ and $r_y = 6$ with center (0,0)

Quick hints

- We have to find all the points of ellipse (i.e. region 1 and region 2)
- We have to start with $(0,r_y)$ [point of region 1] and ended with $(r_x,0)$ [point of region 2]
- Find $dy = 2Y_{k+1} \times r_x^2$ and $dx = 2X_{k+1} \times r_y^2$ and compare every time. If dy > dx then we are in region 1 so use all the formula related to region 1

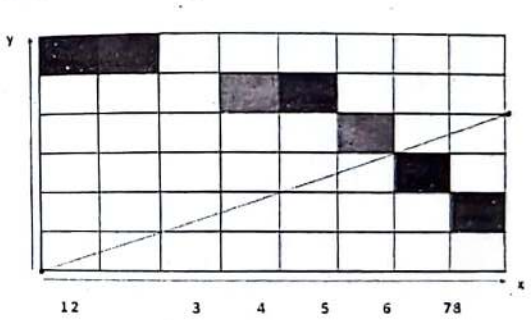
  Initial decision parameter of region 1 : $P1_0 = r_y^2 - r_x^2r_y + r_x^2/4$
- If $P1_k < 0$ then the next point is (X_{k+1}, Y_k) and $P1_{k+1} = P1_k + 2r_y^2X_{k+1} + r_y^2$
- Otherwise, the next point is (X_{k+1}, Y_{k-1}) and $P1_{k+1} = P1_k + 2r_y^2X_{k+1} - 2r_x^2Y_{k+1} + r_y^2$

When dx > dy] we shift from region 1 to region 2 and use the formula related to region 2

Initial decision parameter of region 2: $P2_0=r_y^2(X_0+1/2)^2+r_x^2(Y_0-1)^2-r_x^2r_y^2$

- Check if $P2_k <= 0$ the next point is $(X_k + 1, Y_k - 1)$ and $P2_{k+1} = P2_k + 2r_y^2X_{k+1} - 2r_x^2Y_{k+1} + r_x^2$.
- Otherwise, the next point is $(X_k, Y_k - 1)$ and $P2_{k+1} = P2_k - 2r_x^2Y_{k+1}+r_x^2$

n	X_k	Y_k	Decision parameter (p)	X_{k+1}	Y_{k+1}	$dx=2X_{k+1}xr_y^2$	$dy=2Y_{k+1}xr_x^2$	Plot (x,y)	
0	0	6	$P1_0 = -332$	1	6	72	768	(1,6)	Region 1
1	1	6	$P1_1 = -224$	2	6	144	768	(2,6)	
2	2	6	$P1_2 = -44$	3	6	216	768	(3,6)	
3	3	6	$P1_3 = 208$	4	5	288	640	(4,5)	
4	4	5	$P1_4 = -108$	5	5	360	640	(5,5)	
5	5	5	$P1_5 = 288$	6	4	432	512	(6,4)	
6	6	4	$P1_6 = 244$	7	3	504	384	(7,3)	Region 2
7	7	3	$P2_0 = -23$	8	2	576	256	(8,2)	
8	8	2	$P2_1 = 361$	8	1	576	128	(8,1)	
9	8	1	$P2_2 = 297$	8	0	End	End		



Region 1 points

(0,6)	Starting point
(1,6)	
(2,6)	
(3,6)	
(4,5)	
(5,5)	
(6,4)	

Region 2 points

(7,3)	
(8,1)	
(8,0)	ending point

Area Filling Scan –line Algorithm

Filling is the process of "coloring in" a fixed area or region. A standard output primitive in general graphics is a solid color or patterned polygon area.

There are two basic approaches to area filling on a raster system.

a. One way to fill an area is to determine the overlap intervals for scanlines across the area, i.e., the scan line fill algorithm.

b. Another method for area filling is to start from a given interior position and point outward from this point until we encounter the specified boundary editions.

- **Scanline Fill:** The scanline fill algorithm scans each horizontal line of a polygon and determines the intersections with the polygon edges. It then fills the pixels between the intersections. This technique works well for convex and concave polygons and can handle gradients or patterns.
- **Boundary Fill:** The boundary fill algorithm starts at a given seed point inside a shape and recursively fills the area until it encounters a boundary or a different color. This technique is useful for filling closed shapes with solid colors or patterns.
- **Flood Fill:** The flood fill algorithm is similar to boundary fill, but it fills an entire connected region based on a starting point and a target color. It continues to fill adjacent pixels until it reaches the boundary or encounters a different color.
- **Seed Fill:** Seed fill is a variation of the flood fill algorithm where multiple seed points are specified, and each seed point fills its connected region. This technique is useful for filling complex shapes or regions with different colors.
- **Texture Mapping:** Texture mapping involves mapping a 2D image or texture onto a 3D surface. This technique applies the texture to the surface's polygons, filling the areas with the texture's color or pattern. It is commonly used in 3D rendering and computer games.

Scan line polygon fill algorithm

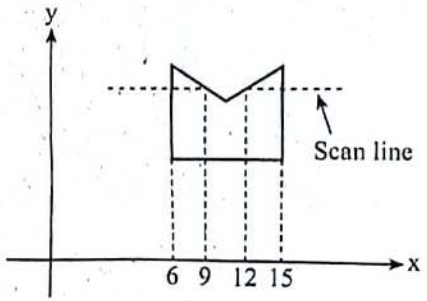


Fig.: (a)

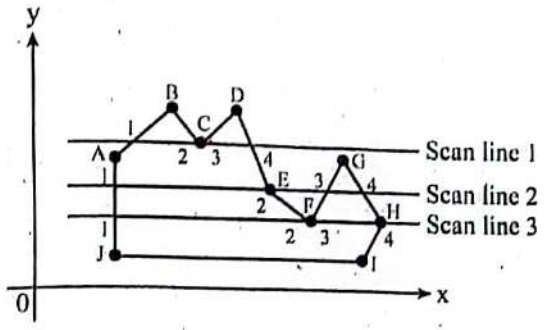


Fig.: (b) Intersection points along the scan line that intersect polygon vertices

Figure (a) illustrates the scan line algorithm for filling of the polygon. For each scan line crossing a polygon, this algorithm locates the intersection points of the scan line with the polygon edges. These intersection points are then sorted from left to right, and the corresponding positions between each intersection pair are set to the specified fill color.

The important task in the scan line algorithm is to find the intersection points of the scan line with the polygon boundary. When intersection points are even, they are sorted from left to right, paired and pixels between paired points are set to the fill color. But in some cases, the intersection point is a vertex. When a scan line intersects a polygon vertex special handling is required to find the exact intersection points To handle such cases, we must look at the other endpoints of the two line segments of the polygon which meet at this vertex. If these points lie on the same (up or down) side of the scan line, then the point in question counts as an even number of intersections. If they lie on opposite sides of the scan line, then the point is counted as a single intersection. This is illustrated in Figure (b).

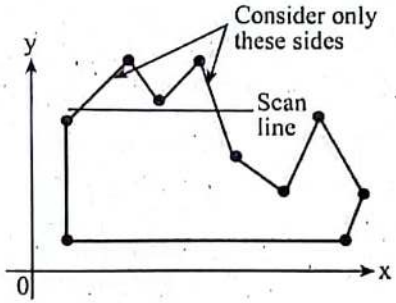


Fig.: (c) Consider only the sides which intersect the scan line

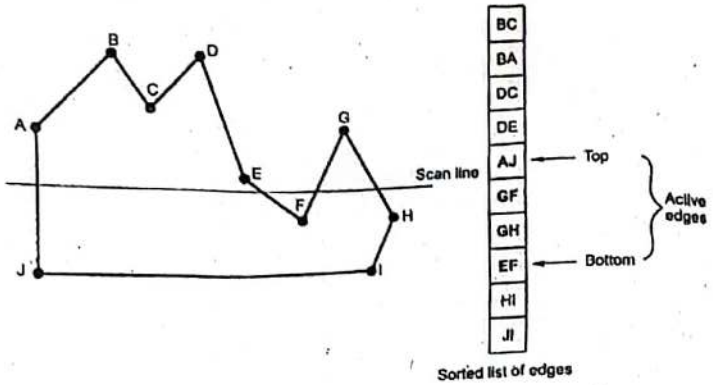
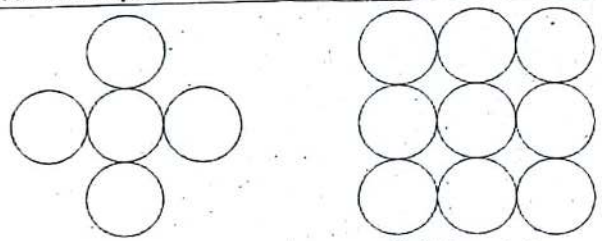


Fig.: (d) Sorted edges of the polygon with active edges

Many times it is not necessary to compute the x intersections for scan line with every polygon side. We need to consider only the polygon sides with endpoints straddling the current scan line. See figure (c).

It will be easier to identify which polygon sides should be tested for x-intersection. if we first sort the sides in order of their maximum y value. Once the sides are sorted we can process the scan lines from the top of the polygon to its bottom producing an active edge list for each scan line crossing the polygon boundaries. The active edge list for a scan line contains all edges crossed by that scan line. The figure (d) shows sorted edges of the polygon with active edges.

Boundary Fill Techniques



4–Connected:

In a 4–connected system, a pixel is considered to be connected to its four immediate neighbors: the pixel to the north, south, east, and west. These are the four cardinal directions on a grid. The concept of 4–connectedness implies that diagonal neighbors are not considered connected.

For example, in a 2D grid, if we have a pixel at coordinates (x, y), its 4–connected neighbors would be (x, y+1) to the north, (x, y–1) to the south, (x+1, y) to the east, and (x–1, y) to the west.

8–Connected:

In an 8–connected system, a pixel is considered to be connected to its eight neighboring pixels, which include both cardinal and diagonal directions. In addition to the north, south, east, and west neighbors, it also includes the four diagonal neighbors.

Continuing from the previous example, in a 2D grid, if we have a pixel at coordinates (x, y), its 8–connected neighbors would be (x, y+1) to the north, (x, y–1) to the south, (x+1, y) to the east, (x–1, y) to the west, (x+1, y+1) to the northeast, (x+1, y–1) to the southeast, (x–1, y–1) to the southwest, and (x–1, y+1) to the northwest.

Boundary Fill Algorithm

This is a recursive algorithm that begins with a starting pixel called a seed, inside a region and continuous painting toward the boundary. The algorithm checks to see if this pixel is a boundary pixel or has already filled. If answer is No, it fills the pixel and makes a recursive call to itself using each and every neighbouring pixel as a new seed. If the answer is yes, the algorithm simply returns to its caller.

A boundary - fill procedure accepts as input the coordinates of an interior point (x, y), a fill colour and a boundary colour. Starting from (x, y), the procedure tests neighbouring positions to determine whether they are of the boundary colour. If not, they are painted with the fill colour and their neighbours are tested. This process continues until all pixels up to the boundary colour for the area have been tested. There may be two methods for preceding to neighbouring pixels from the seed point.

1. 4-connected
2. 8-connected

The following procedure illustrates a recursive method for filling a 4-connected area. The boundary fill method requires the coordinates of a starting point, a fill colour and background colour as arguments.

```
Void boundary fill (int. x, int. y, int fill, int boundary)
{
int current = get pixel (x, y);
if ((current ! = boundary) & current ! = fill)
{
SetColor (fill);
```

```
SetPixel (x, y);
boundary Fill (x + 1, y, fill, boundary);
boundary Fill (x – 1, y, fill, boundary);
boundary Fill (x, y + 1, fill, boundary);
boundary Fill (x, y – 1, fill, boundary);
}
}
```

Recursive boundary-fill algorithms may not fill regions correctly if some interior pixels are already displayed in the fill colour. This occurs because the algorithm checks next pixels both for boundary colour and for fill colour. Encountering a pixel with the fill colour can cause a recursive branch to terminate, leaving other interior pixels unfilled. To avoid this, we can first change the colour of any interior pixels that are initially set to the fill colour before applying the boundary-fill procedure.

Also, since this procedure requires considerable stacking of neighbouring points, more efficient methods are generally employed. These methods fill horizontal pixel spans across scan line, instead of procedding to 4-connected or 8-connected neighbouring points. Then we need only stack a beginning position for each horizontal pixel span, instead of stacking all unprocessed neighbouring positions around the current position. Starting from the initial interior point with this method, we first fill in the contiguous span of pixels n this starting scan line. Then we locate and stack starting positions for spans on the adjacent scan lines, where spans are defined as the contiguous horizontal string of positions bounded by pixels displayed in the area border colour. At each subsequent step, we unstack the next start position and repeat the process.

The boundary fill algorithm, is having limitation. It fills the polygon having unique boundary colour. If the polygon is having boundaries with different colours then this algorithm fails.

Flood fill Algorithm

This algorithm also begins with a seed (starting pixel) inside the region. It checks to see if the pixel has the regions original color. If the answer is yes, it fills the pixel with a new colour and uses each of the pixel's neighbour as a new seed in a recessive call. If answer is no, it returns to the caller.

This method shares great similarities in its operating principle with the boundary fill algo. It is particularly useful when the region to be filled has no uniformly coloured boundary i.e., fill in an area that is not defined within a single colour boundary. So in this we can point such areas by

replacing a specified interior colour instead of searching for a boundary colour value. If the rea we want to paint has more than one interior colour, we can first reassign pixel values, so that all interior points have the some colour. Using either a 4-connected or 8-connected approach, we than step through pixel positions until all interior points have been repainted.

The following procedure flood fills a 4-connected region, recursively, starting from the input position.

```
Void floodfill (int x, int y, int fill, int oldcolor)
{
if (getpixel (x, y) = oldcolor)
{
SetColor (fill);
SetPixel (x, y);
Floodfill (x + 1, y, fill, oldcolor);
Floodfill (x – 1, y, fill, oldcolor);
Floodfill (x, y + 1, fill, oldcolor);
Floodfill (x, y – 1, fill, oldcolor);
}
}
```

We can modify procedure flood fill to reduce the storage requirements of the stack by filling horizontal pixel spans as boundary-fill algorithm. In this approach, we stack only the beginning positions for those pixel spans having the value oldcolor. The steps in this modified algorithm are similar to those for boundary fill. Starting at the first position of each span, the pixel values are replaced until a value other than oldcolor is encountered.

Unit – 4

TWO DIMENSIONAL GEOMETRIC TRANSFORMATION AND VIEWING

Introduction

2D Transformation in computer graphics is a process of modifying and re-positioning the existing graphics in 2 dimensions. Transformations help change the object's position, size, orientation, shape, etc.; there are three basic rigid transformations: reflections, rotations, and translations. 2D Transformations in computer graphics take place in a two-dimensional plane.

There are two complementary points of view for describing object transformation.

1. **Geometric Transformation:** The object itself is transformed relative to the coordinate system or background. The mathematical statement of this viewpoint is defined by geometric transformations applied to each point of the object.
2. **Coordinate Transformation:** The object is held stationary while the coordinate system is transformed relative to the object. This effect is attained through the application of coordinate transformations.

The movement of an automobile against a scenic background we can simulate this by

- Moving the automobile while keeping the background fixed- (Geometric Transformation)
- We can keep the car fixed while moving the background scenery- (Coordinate Transformation)

In 2D co-ordinate system, any point is represented in terms of x and y co-ordinates. The point (x, y) can be converted into matrix in the following two ways:

a. Row-major matrix: $[x \quad y]_{1\times2}$

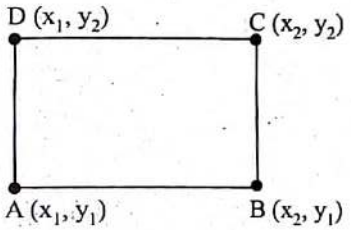
b. Column-major matrix: $\begin{bmatrix} x \\ y \end{bmatrix}_{2\times1}$

The above two matrices are frequently called position vectors. A series of points, each of which is a position vector relative to some co-ordinate system, is stored in a computer as a matrix or array of numbers. The position of these points is controlled by manipulating the matrix which defines the points. Lines are drawn between the points to generate lines, curves or pictures.

Suppose, we represent a rectangle in matrix form. Let (x_1, y_1) and (x_2, y_2) be the opposite vertices of a rectangle. Then, the four vertices of rectangle will be: $(x_1\ y_1)$, $(x_2\ y_1)$, $(x_2\ y_2)$, (x_1, y_2).

In order to represent this rectangle in matrix form as:

$$\begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} = \begin{bmatrix} x_1 & y_1 \\ x_2 & y_1 \\ x_2 & y_2 \\ x_1 & y_2 \end{bmatrix}$$



Basic Transformation

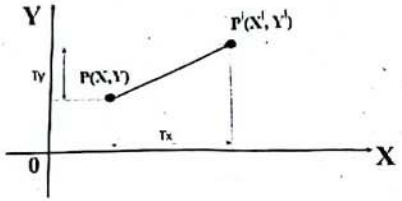
1. Translation

- A process of moving an object from one position to another in a two-dimensional plane without deformation (without changing shape and size) is called translation.
- Consider a point object P that has to be moved from one position to another in a 2D plane.
- **Let**
  - Initial coordinates of the object P = (X, Y)
  - New coordinates of the object P after translation = (X', Y')

- Translation vector or Shift vector = (T_x, T_y)
  - T_x defines the distance the X coordinate of A has to be moved.
  - T_y defines the distance the Y coordinate of A has to be moved.
- This translation is achieved by adding the translation coordinates to the old coordinates of the object as–
- **Equation form:**

  $X' = X + T_x$ (This denotes translation towards X axis)

  $Y' = Y + T_y$ (This denotes translation towards Y axis)
- **Matrix form:**



$P' = P + T$

$$\text{Or,} \begin{bmatrix} X' \\ Y' \end{bmatrix} = \begin{bmatrix} X \\ Y \end{bmatrix} + \begin{bmatrix} T_x \\ T_y \end{bmatrix}$$

- **Homogenous Co-ordinates:**

$$\begin{bmatrix} X' \\ Y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

2. Scaling

- Alters the size and shape of an object.
- Either expand or compress the dimensions of an object.
- Achieved by multiplying the original coordinates of the object with the scaling factor to get the desired result.
- Consider a point object P that is to be scaled in a 2D plane.
- **Let:**
  - Initial coordinates of the object P = (X, Y)
  - New coordinates of the object P after scaling = (X', Y')
  - Scaling factor = (S_x, S_y)
    - S_x and S_y scales the object in X and Y direction respectively.

- This scaling is achieved by multiplying the scaling factor coordinates to the old coordinates of the object as–
- **Equation form:**

  $X' = X * S_x$ (This denotes Scaling towards X axis)

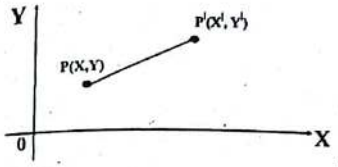
  $Y' = Y * S_y$ (This denotes Scaling towards Y axis)
- **Matrix form:**

  $P' = P * S$ where S is scaling matrix

$$\text{or,} \begin{bmatrix} X' \\ Y' \end{bmatrix} = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} + \begin{bmatrix} X \\ Y \end{bmatrix}$$

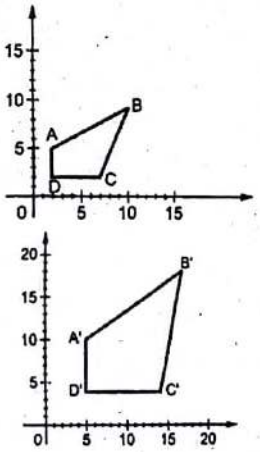
- **Homogenous Co-ordinates:**

$$\begin{bmatrix} X' \\ Y' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$



Cases:

1. **If $(S_x, S_y) < 1$:** Size decreases, objects moves closer to origin.
2. **If $(S_x, S_y) = 1$:** Size unchangeable, objects remains same.
3. **If $(S_x, S_y) > 1$:** Size increases, objects moves away from the origin
4. **If $(S_x = S_y)$:** Uniform Scaling
5. **If $(S_x$ not $= S_y)$:** Non-uniform Scaling



Fixed Point Scaling (Find scaling transformation matrix to scale S_x, S_y units with respect to a fixed point P(x, y).

- The scaling matrix that we discussed performs scaling about the origin.
- The location of a scaled object can be controlled by a position known as the fixed point that is to remain unchanged after the scaling transformation.

Steps:

1. Translation of an object so that the fixed point about which an object is to be scaled coincides with origin.
2. Scaling of an object about the origin.
3. Inverse Translation of an object so that the fixed point reaches to its original position.

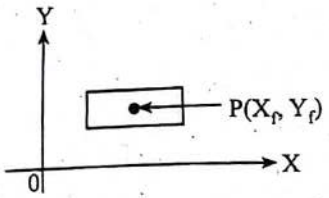


Fig.: (a) Object which is to be scaled about the fixed point P

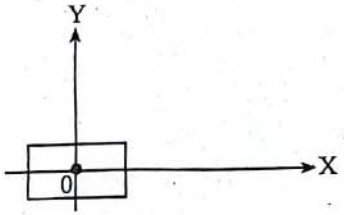


Fig.: (b) Translation of an object so that P reaches to origin

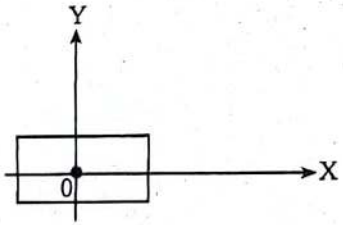


Fig.: (c) Scaling of an object about origin

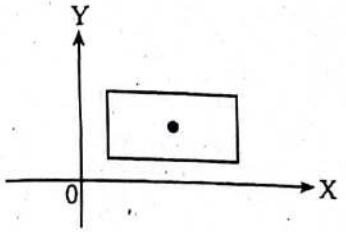


Fig.: (d) Translation of an object so that P reaches to original position

Composite matrix (C_m) = $T^{-1}ST$

$$= \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -t_x \\ 0 & 1 & -t_y \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} S_x & 0 & -S_x t_x \\ 0 & S_y & -S_y t_y \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} S_x & 0 & -S_x t_x + t_x \\ 0 & S_y & -S_y t_y + t_y \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} S_x & 0 & (1 - S_x) t_x \\ 0 & S_y & (1 - S_y) t_y \\ 0 & 0 & 1 \end{bmatrix}$$

In matrix form:

$P' = C_m.P$

$$\text{or, } \begin{bmatrix} X' \\ Y' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & (1 - S_x) t_x \\ 0 & S_y & (1 - S_y) t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

$$\therefore \begin{bmatrix} X' \\ Y' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & (1 - S_x) X_f \\ 0 & S_y & (1 - S_y) X_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

- **In equation form:**

$X' = X * S_x + X_f(1 - S_x)$

$Y' = Y * S_y + Y_f(1 - S_y)$

3. Rotation

- Changing the co-ordinate position along a circular path is called rotation.
- Points can be rotated through an angle θ about an origin or about some point.
- The sign of angle determines the direction of rotation.
- Positive value for rotation angle defines counterclockwise rotations (anti-clockwise).
- Negative value for rotation angle defines clockwise rotations.

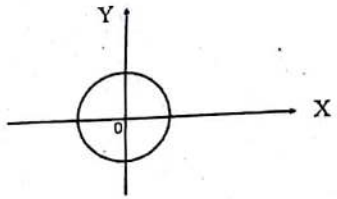


Fig.: Anti clockwise (θ is positive)

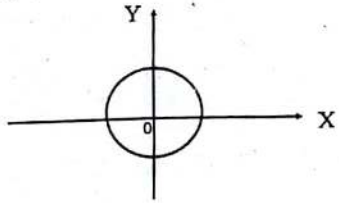


Fig.: Clockwise (θ is negative)

Types of rotation

I. Rotation about an origin
II. Rotation about any point

I. Rotation about an origin

- If P(X, Y) is rotated to a new position p'(X', Y') in anti-clockwise direction by an angle θ, then (X', Y') can be calculated as following.
- Let the angle made by line OP with x-axis is ϕ and the radius of circulation path is r then,

In ΔPOB

$X = r\cos\phi$ [$\cos\phi = b/h$]

$Y = r\sin\phi$ [$\sin\phi = p/h$]

Also, In ΔP'OA

$X' = r\cos(\theta + \phi)$

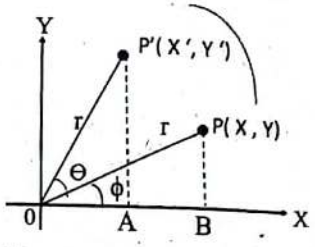
$= r(\cos\theta.\cos\phi - \sin\theta.\sin\phi)$

$\therefore \quad X' = X\cos\theta - Y\sin\theta$

$Y' = r\sin(\theta + \phi)$

$= r(\sin\theta.\cos\phi + \cos\theta.\sin\phi)$

$\therefore \quad Y' = X\sin\theta + Y\cos\theta$



- **In matrix form:**

$P' = R(\theta) \cdot P$

$$\begin{bmatrix} X' \\ Y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix}$$ Anti-clockwise

$$\begin{bmatrix} X' \\ Y' \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix}$$ clockwise

For anti clock θ = –ve. We use column vector representation.

- **In homogenous co-ordinate form**

$$\begin{bmatrix} X' \\ Y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$ Anti-clockwise

$$\begin{bmatrix} X' \\ Y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$ clockwise

II. Rotation about an arbitrary

- If P(X, Y) is rotated to a new position P'(X', Y') in anti-clockwise direction by an angle θ, about a fixed-point A(X_r, Y_r) then (X', Y') can be calculated as following.
- Let the radius of circulation path is r then,

In ΔP'AB

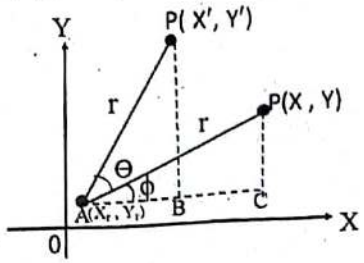
$$\cos(\theta + \phi) = \frac{AB}{P'A} = \frac{X' - X_r}{r}$$

or, $r\cos(\theta + \phi) = X' - X_r$

or, $r\cos\theta\cos\phi - r\sin\theta\sin\phi + X_r = X'$

or, $r\cos\theta\frac{(X - X_r)}{r} - r\sin\theta\frac{(Y - Y_r)}{r} + X_r = X'$

$\therefore X' = X_r + (X - X_r)\cos\theta - (Y - Y_r)\sin\theta$



Similarly,

$$\sin(\theta + \phi) = \frac{P'B}{P'A} = \frac{Y - Y_r}{r}$$

or, $r\sin(\theta + \phi) = Y' - Y_r$

or, $r\sin\theta\cos\phi + r\cos\theta\sin\phi + Y_r = Y'$

or, $r\sin\theta\frac{(X - X_r)}{r} + r\cos\theta\frac{(Y - Y_r)}{r} + Y_r = Y'$

$\therefore Y' = Y_r + (X - X_r)\sin\theta + (Y - Y_r)\cos\theta$

- When an object is rotated through some angle about a fixed point other than origin then following major THREE tasks take place:
  1. Translation of an object to the origin.
  2. Rotation of an object by given angle (clockwise or anticlockwise).
  3. Inverse Translation of an object back to its original location.

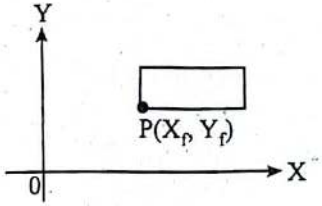


Fig.: (a) Object which is to be rotated about the fixed point P

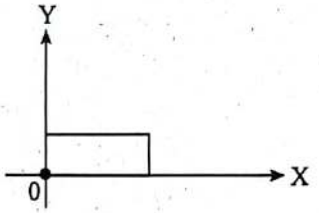


Fig.: (b) Translation of an object so that P reaches to origin

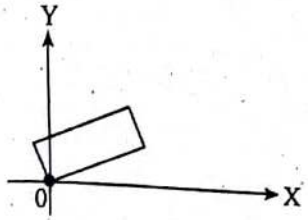


Fig.: (c) Rotation of an object about origin

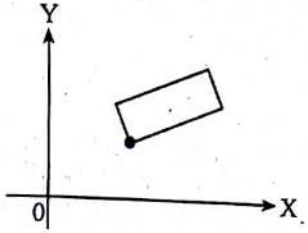


Fig.: (d) Translation of an object so that P reaches to original position

Composite matrix (C_m) = $T^{-1}RT$

$$= \begin{bmatrix} 1 & 0 & X_r \\ 0 & 1 & Y_r \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -X_r \\ 0 & 1 & -Y_r \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & X_r \\ 0 & 1 & Y_r \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & Y_r\sin\theta - X_r\cos\theta \\ \sin\theta & \cos\theta & -Y_r\cos\theta - X_r\sin\theta \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos\theta & -\sin\theta & Y_r\sin\theta - X_r\cos\theta + X_r \\ \sin\theta & \cos\theta & -Y_r\cos\theta - X_r\sin\theta + Y_r \\ 0 & 0 & 1 \end{bmatrix}$$

In matrix form:

$P' = C_m.P$

$$\therefore \begin{bmatrix} X' \\ Y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & Y_r\sin\theta - X_r\cos\theta + X_r \\ \sin\theta & \cos\theta & -Y_r\cos\theta - X_r\sin\theta + Y_r \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$ For Anticlockwise

$$\therefore \begin{bmatrix} X' \\ Y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & -Y_r\sin\theta - X_r\cos\theta + X_r \\ -\sin\theta & \cos\theta & -Y_r\cos\theta + X_r\sin\theta + Y_r \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$ For clockwise

4. Shear

- Distorts the shape of an object.
- The transformation results as if an object is composed of internal layers that had been caused to slide over each other.
- Shearing can be done either in X-direction or Y-direction.

I. X-Shear: (Shearing along X-direction)

- Y-coordinate remains unchanged, but X-coordinate is changed.

$X' = X + Sh_x \times Y$

$Y' = Y$

In matrix form:

$$\begin{bmatrix} X' \\ Y' \end{bmatrix} = \begin{bmatrix} 1 & Sh_x \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix}$$

- **In homogenous co-ordinate form:**

$$\begin{bmatrix} X' \\ Y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & Sh_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

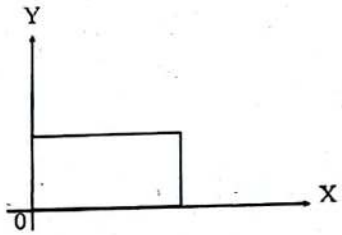


Fig.: Original object

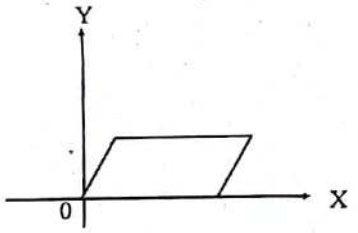


Fig.: Object after X-shearing

II. Y-Shear: (Shearing along Y-direction)

- X-coordinate remains unchanged but Y-coordinate is changed.

$X' = X$

$Y' = Y + Sh_y \times X$

- **In matrix form:**

$$\begin{bmatrix} X' \\ Y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ Sh_y & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix}$$

- **In homogenous co-ordinate form:**

$$\begin{bmatrix} X' \\ Y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ Sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

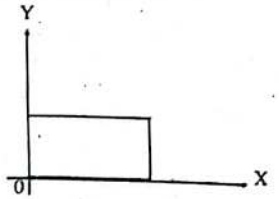


Fig.: Original object

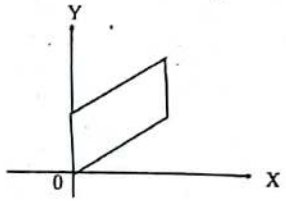


Fig.: Object after Y-Shearing

III. Shearing in both directions

- Both X- and Y-coordinates are changed.

$X' = X + Sh_x \times Y$

$Y' = Y + Sh_y \times X$

- **In matrix form:**

$$\begin{bmatrix} X' \\ Y' \end{bmatrix} = \begin{bmatrix} 1 & Sh_x \\ Sh_y & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix}$$

- **In homogenous co-ordinate form:**

$$\begin{bmatrix} X' \\ Y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & Sh_x & 0 \\ Sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

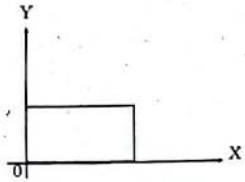


Fig.: Original object

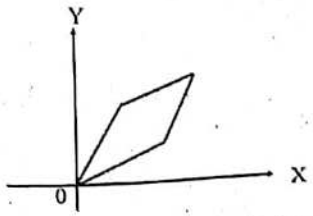


Fig.: Object after shearing in both the directions

5. Reflection

- Produces mirror image of an object.
- In 2D reflection, we consider any line in 2D plane as mirror.
- The reflection is also referred as rotation by 180^0.

a. Reflection about X-axis

- Also referred as Reflection about the line Y = 0.
- The transformation flips Y-coordinates i.e. X-coordinate is not changed, and sign of Y-coordinate is changed.

  $X' = X$

  $Y' = -Y$
- **In matrix form:**

$$\begin{bmatrix} X' \\ Y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix}$$

- **In homogenous co-ordinate form:**

$$\begin{bmatrix} X' \\ Y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

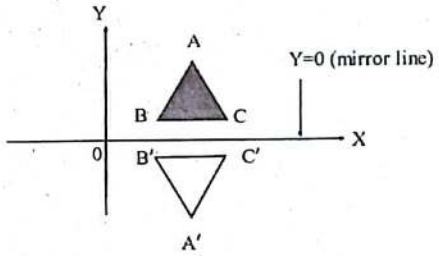


Fig.: Reflection of an object ABC along X-axis

b. Reflection about Y-axis

- Also referred as Reflection about the line X = 0.
- The transformation flips X-coordinates i.e. Y-coordinate is not changed and sign of X-coordinate is changed.

  $X' = -X$

  $Y' = Y$
- **In matrix form:**

$$\begin{bmatrix} X' \\ Y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix}$$

- **In homogenous co-ordinate form:**

$$\begin{bmatrix} X' \\ Y' \\ 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

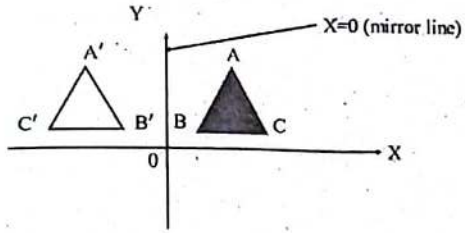


Fig.: Reflection of an object ABC along Y-axis

c. Reflection about origin

- The transformation flips both X- and Y-coordinates.

  $X' = -X$

  $Y' = -Y$
- In matrix form:

$$\begin{bmatrix} X' \\ Y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix}$$

- In homogenous co-ordinate form:

$$\begin{bmatrix} X' \\ Y' \\ 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

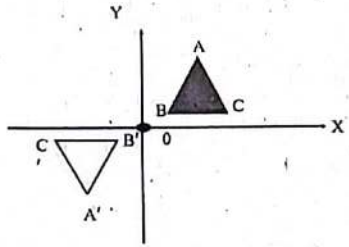


Fig.: Reflection of an object ABC about origin

d. Reflection about the line Y = X

- Transforms the point (X,Y) as follows:

  $X' = Y$

  $Y' = X$

- **In matrix form:**

$$\begin{bmatrix} X' \\ Y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix}$$

- **In homogenous co-ordinate form:**

$$\begin{bmatrix} X' \\ Y' \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

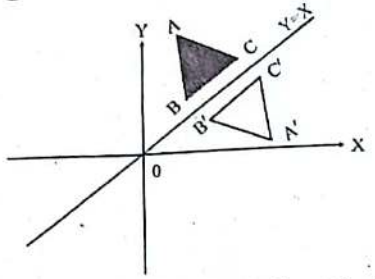
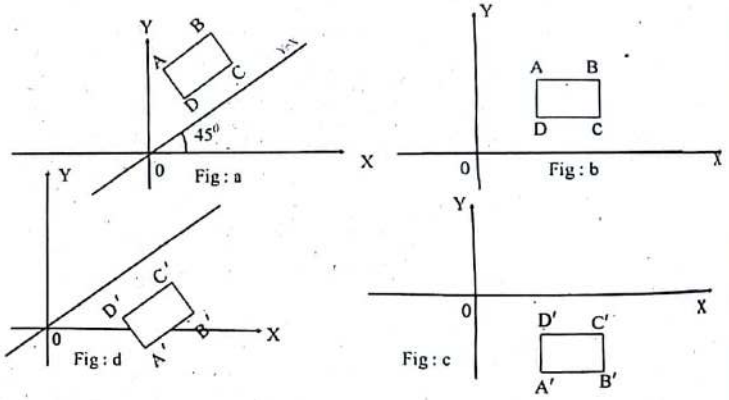


Fig.: Reflection of an object ABC about the line Y=X

This reflection involves following steps:

1. Rotate the object so that the mirror line coincides either to X– or Y–axis as shown in fig b.
2. Reflect the object about X– or Y–axis as step 1 as shown in fig c.
3. Rotate the object in the opposite direction to step 1 so that the mirror line reaches to original position as shown in fig d.



Composite matrix $(C_m) = R^{-1}_{\theta=45(\text{anti-clockwise})} R_{x\text{-axis}} R_{\theta=45(\text{clockwise})}$

$$= \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ \sin\theta & -\cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

In matrix form:

$P^I = C_m . P$

$$\therefore \begin{bmatrix} X' \\ Y' \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

e. Reflection about the line

$Y = -X$

- **Transforms the point (X,Y) as follows:**

  $X' = -Y$

  $Y' = -X$

- **In matrix form:**

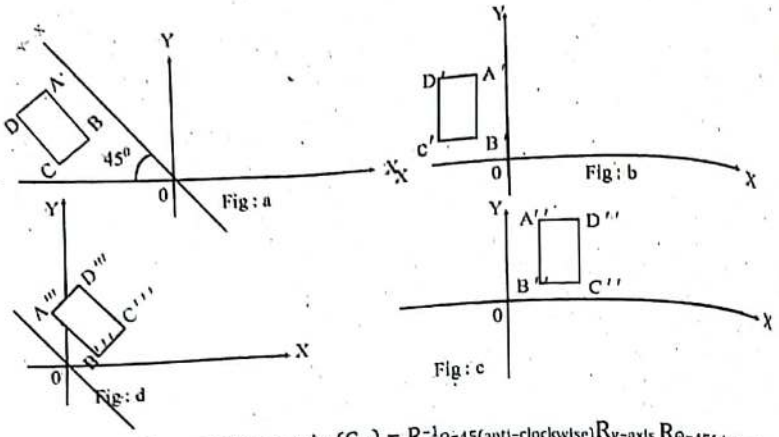
$$\begin{bmatrix} X' \\ Y' \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix}$$

- In homogenous co-ordinate form:

$$\begin{bmatrix} X' \\ Y' \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

This reflection involves following steps:

1. Rotate the object so that the mirror line coincides either to X– or Y–axis as shown in fig b.
2. Reflect the object about X– or Y–axis as step 1 as shown in fig c.
3. Rotate the object in opposite direction to step 1 so that the mirror line reaches to original position as shown in fig d.



Composite matrix $(C_m) = R^{-1}_{\theta=45(\text{anti-clockwise})} R_{y\text{-axis}} R_{\theta=45(\text{clockwise})}$

$$= \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -\cos\theta & -\sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

In matrix form:

$P' = C_m . P$

$$\therefore \begin{bmatrix} X' \\ Y' \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

f. Reflection through an arbitrary line

y = mx + c

(***Q. Derive the composite transformation matrix that reflects an object about line 'L' with necessary figures***)

When reflection is to be performed about a line that neither passes through the origin nor is parallel through the co-ordinate axes can be solved using following steps.

1. Translate the object so that the line y = mx + c passes through the origin as shown in figure (b).
2. Rotate the object about the origin so that the line y = mx + c coincides with one of the co-ordinate axes as shown in figure (c).
3. Reflect the object through the co-ordinate axis (either X- or Y- depending upon step 2 as shown in figure (d).
4. Apply inverse rotation to step 2 so that the line shifted to its translated position as shown in figure (e).
5. Apply inverse translation to step 1 so that the line reaches to its original position as shown in figure (f).

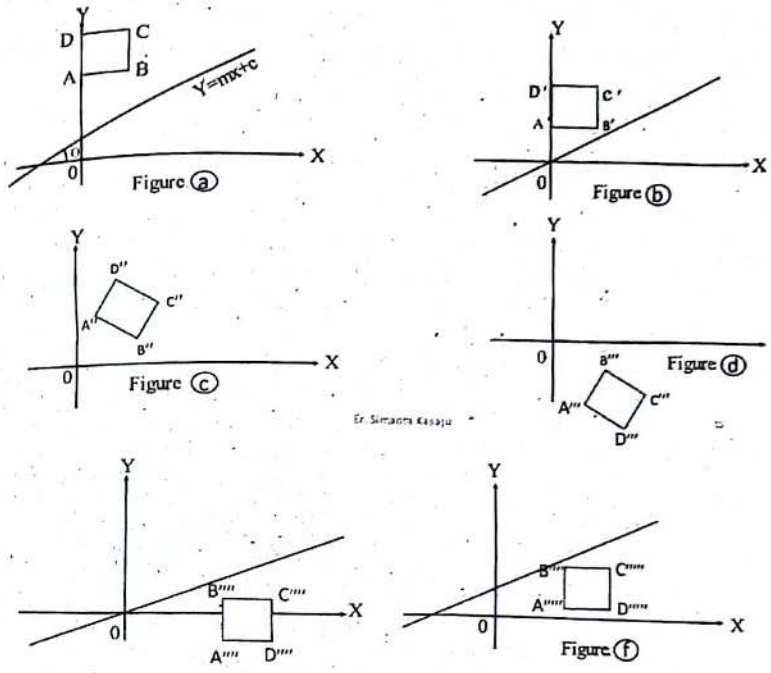


Figure (e)

CM: = T^{-1} (0,c) . Rccw . Rfx . Rcw . T(0, −c)

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -c \\ 0 & 0 & 1 \end{bmatrix}$$

Homogenous co–ordinates

- Virtual coordinate system associated with 2D representation in order to maintain uniformity in the transformation.
- Among all the transformations, translation is in additive form and others are in multiplicative form. This creates problems in the calculation.
- Thus using homogenous co-ordinates, the additive translation can be converted into multiplicative form so that it helps in easy and efficient calculation.
- Hence, homogenous co-ordinates are created to treat all the transformation in the same manner.
- We represent each Cartesian co-ordinate position **(X , Y)** with homogenous triple co-ordinate **(X_h , Y_h , h)** where

$X = \frac{Xh}{h}$ $X_h = X_h$ and $Y = \frac{Yh}{h}$ $Y_h = Y_h$

Why homogeneous coordinate system?

- To represent all the transformation equations as matrix multiplications.
- To perform more than one transformation at a time.
- To reduce unwanted calculations of intermediate steps, to save time and memory and produce a sequence of transformations

1. **Translation**

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix} \text{ or } \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

2. **Scaling**

$$\begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

3. **Rotation (clockwise)**

$$\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

4. **Rotation (anti-clock)**

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

5. **Reflection against X-axis**

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

6. **Reflection against Y-axis**

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

7. **Reflection against origin**

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

8. **Reflection against line Y = X**

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

9. **Reflection against Y = –X**

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

10. **Shearing in X direction**

$$\begin{bmatrix} 1 & 0 & 0 \\ Sh_x & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & Sh_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

11. **Shearing in Y direction**

$$\begin{bmatrix} 1 & Sh_y & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ Sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

12. Shearing in both x and y direction

$$\begin{bmatrix} 1 & Sh_y & 0 \\ Sh_x & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & Sh_x & 0 \\ Sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Composite matrix

- Composite transformation matrix is formed by calculating the product of individual transformations

  $P' = M_2.M_1.P = M.P.$

- **Successive translations (additive)**

  $P' = T(t_{2x}, t_{2y}) . \{T(t_{1x}, t_{1y}) . P\}$

  $= \{T(t_{2x}, t_{2y}) . T(t_{1x}, t_{1y})\} . P$

  $$\begin{bmatrix} 1 & 0 & t_{2x} \\ 0 & 1 & t_{2y} \\ 0 & 0 & 1 \end{bmatrix}.\begin{bmatrix} 1 & 0 & t_{1x} \\ 0 & 1 & t_{1y} \\ 0 & 0 & 1 \end{bmatrix}=\begin{bmatrix} 1 & 0 & t_{1x}+t_{2x} \\ 0 & 1 & t_{1y}+t_{2y} \\ 0 & 0 & 1 \end{bmatrix}$$

  $T(t_{2x}, t_{2y}) . T(t_{1x}, t_{1y}) = T(t_{1x} + t_{2x}, t_{1y} + t_{2y})$

- **Successive rotations (additive)**

  $P' = R(\theta_2) . \{R(\theta_1) . P\}$

  $= \{R(\theta_2) . R(\theta_1)\} . P$

  $R(\theta_2) . R(\theta_1) = R(\theta_1 + \theta_2)$

  $P' = R(\theta_1 + \theta_2) . P$

- **Successive scaling (multiplicative)**

  $$\begin{bmatrix} s_{2x} & 0 & 0 \\ 0 & s_{2y} & 0 \\ 0 & 0 & 1 \end{bmatrix}.\begin{bmatrix} s_{1x} & 0 & 0 \\ 0 & s_{1y} & 0 \\ 0 & 0 & 1 \end{bmatrix}=\begin{bmatrix} s_{1x}.s_{2x} & 0 & 0 \\ 0 & s_{1y}.s_{2y} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

  $S(s_{2x}, s_{2y}) . S(s_{1x}, s_{1y}) = S(s_{1x} . s_{2x}, s_{1y} . s_{2y})$

Prove that two successive rotations are additive.

To show that the composition of two successive rotations are additive i.e $R(\theta_1). R(\theta_2) = R(\theta_1 + \theta_2)$.

Solution:

The rotation matrix R is given as,

$$R = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

we can write rotation matrix $R(\theta_1)$ as

$$R(\theta_1) = \begin{bmatrix} \cos\theta_1 & \sin\theta_1 \\ -\sin\theta_1 & \cos\theta_1 \end{bmatrix} \text{ and } R(\theta_2) = \begin{bmatrix} \cos\theta_2 & \sin\theta_2 \\ -\sin\theta_2 & \cos\theta_2 \end{bmatrix}$$

$$R(\theta_1) . R(\theta_2) = \begin{bmatrix} \cos\theta_1 & \sin\theta_1 \\ -\sin\theta_1 & \cos\theta_1 \end{bmatrix} \times \begin{bmatrix} \cos\theta_2 & \sin\theta_2 \\ -\sin\theta_2 & \cos\theta_2 \end{bmatrix}$$

$$= \begin{bmatrix} \cos\theta_1 . \cos\theta_2 + \sin\theta_1 . (-\sin\theta_2) & \cos\theta_1 . \sin\theta_2 + \sin\theta_1 . \cos\theta_2 \\ -\sin\theta_1 . \cos\theta_2 + \cos\theta_1 . (-\sin\theta_2) & -\sin\theta_1 . \sin\theta_2 + \cos\theta_1 . \cos\theta_2 \end{bmatrix}$$

$$= \begin{bmatrix} \cos(\theta_1 + \theta_2) & \sin(\theta_1 + \theta_2) \\ -\sin(\theta_1 + \theta_2) & \cos(\theta_1 + \theta_2) \end{bmatrix}$$

since,

$\cos(\theta_1 + \theta_2) = \cos\theta_1 \cos\theta_2 - \sin\theta_1 \sin\theta_2$

$\sin(\theta_1 + \theta_2) = \cos\theta_1 \sin\theta_2 + \sin\theta_1 \cos\theta_2$

Prove that two scaling transformation commute

The two scaling transformation commute, i.e. $S_1S_2 = S_2S_1$.

Solution:

The scaling matrix S is given as,

$$S = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix}$$

we have

$$S_1S_2 = \begin{bmatrix} S_{x1} & 0 \\ 0 & S_{y1} \end{bmatrix}\begin{bmatrix} S_{x2} & 0 \\ 0 & S_{y2} \end{bmatrix} \text{ and}$$

$$S_2S_1 = \begin{bmatrix} S_{x2} & 0 \\ 0 & S_{y2} \end{bmatrix}\begin{bmatrix} S_{x1} & 0 \\ 0 & S_{y1} \end{bmatrix}$$

$$\therefore\ S_1S_2 = \begin{bmatrix} S_{x1}S_{x2} & 0 \\ 0 & S_{y1}S_{y2} \end{bmatrix} \text{ and } S_2S_1 = \begin{bmatrix} S_{x2}S_{x1} & 0 \\ 0 & S_{y2}S_{y1} \end{bmatrix}$$

Since multiplication is commutative

$S_{x1}S_{x2} = S_{x2}S_{x1}$ and $S_{y1}S_{y2} = S_{y2}S_{y1}$. Therefore, $S_1S_2 = S_2S_1$

Solved Numerical Problems

1. Translate the given points (2,5) by the translating value (3,3).

Solution:

Given point P(2,5) and translation distance $T_x = 3$, and $T_y = 3$

We have from translation matrix

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

i.e. $\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 2 \\ 5 \end{bmatrix} + \begin{bmatrix} 3 \\ 3 \end{bmatrix} = \begin{bmatrix} 5 \\ 8 \end{bmatrix}$

Therefore, P(2,5) is translated to new point P'(5,8).

2. Consider a triangle A(0,0), B(1,1), C(5,2). The triangle has to be rotated by an angle 45° about the point P(−1,−1). What will be the co-ordinate of new triangle?

Solution:

$CM = T_{(-1,-1)}\, R_{45}\, T_{(1,1)}$

$$= \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos 45° & -\sin 45° & 0 \\ \sin 45° & \cos 45° & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & -1 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \sqrt{2}-1 \\ 0 & 0 & 1 \end{bmatrix}$$

$P' = C.M. . P$

$$= \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & -1 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \sqrt{2}-1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 5 \\ 0 & 1 & 2 \\ 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} -1 & -1 & \frac{3}{\sqrt{2}}-1 \\ \sqrt{2}-1 & 2\sqrt{2}-1 & \frac{9}{\sqrt{2}}-1 \\ 1 & 1 & 1 \end{bmatrix}$$

$A' = (-1, \sqrt{2} - 1)$

$B' = (-1, 2\sqrt{2} - 1)$

$C' = \left(\frac{3}{\sqrt{2}} - 1, \frac{9}{\sqrt{2}} - 1\right)$

The co-ordinates of new triangle are $A'(-1, \sqrt{2} - 1)$, $B'(-1, 2\sqrt{2} - 1)$, and $C'\left(\frac{3}{\sqrt{2}} - 1, \frac{9}{\sqrt{2}} - 1\right)$

3. Scale an object (4,4), (3,2), (5,2) about a fixed point (4,3) with scaling factor 2.

Solution:

$P' = C.M. . P$

$$C.M. = \begin{bmatrix} 1 & 0 & 4 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -4 \\ 0 & 1 & -3 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 4 \\ 0 & 1 & 3 \\ 0 & 0 & 3 \end{bmatrix} \begin{bmatrix} 2 & 0 & -8 \\ 0 & 2 & -6 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 2 & 0 & -8+4 \\ 0 & 2 & -6+3 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 & -4 \\ 0 & 2 & -3 \\ 0 & 0 & 1 \end{bmatrix}$$

$$P = \begin{bmatrix} 4 & 3 & 5 \\ 4 & 2 & 2 \\ 1 & 1 & 1 \end{bmatrix}$$

$P' = C.M. \times P$

$$= \begin{bmatrix} 2 & 0 & -4 \\ 0 & 2 & -3 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 4 & 3 & 5 \\ 4 & 2 & 2 \\ 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 8-4 & 6-4 & 10-4 \\ 8-3 & 4-3 & 4-3 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 4 & 2 & 6 \\ 5 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

So, scaled points are (4,5), (2,1) and (6,1)

4. Rotate the triangle (5,5), (7,3), (3,3) about fixed points (5,4) in counter clockwise by 90°.

Solution:

$C.M. = T_{(5,4)} . R_{90} T_{(-5,-4)}$

$$= \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 4 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos 90° & -\sin 90° & 0 \\ \sin 90° & \cos 90° & 0 \\ 0 & 01 & \end{bmatrix} \begin{bmatrix} 1 & 0 & -5 \\ 0 & 1 & -4 \\ 0 & 0 & 1 \end{bmatrix}$$

5. **Rotate a triangle at A(5,7), B(2,5) and C(8,5) about any fixed point (5,6) in anti-clockwise direction.**

Solution:

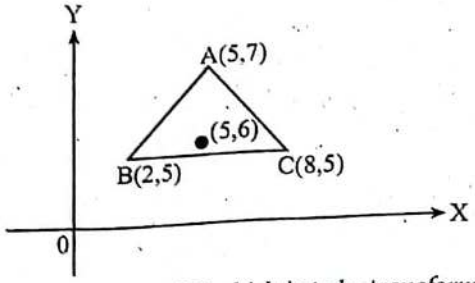


Fig.: A triangle ABC which is to be transformed

Step 1: Translation of a triangle so that fixed point reaches to zero.

Here, $t_x = -5$ and $t_y = -6$

A(5,7) = (5 – 5, 7 – 6) = (0, 1)

B(2,5) = (2 – 5, 5 – 6) = (–3, –1)

C(8, 5) = (8 – 5, 5 – 6) = (3, –1)

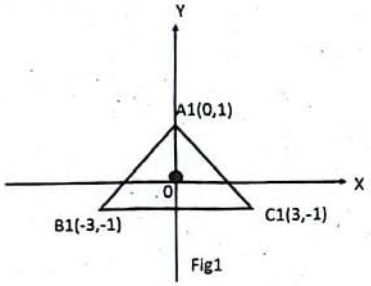


Fig1

Step 2: Rotation of triangle in anti-clockwise direction through an angle of 45°.

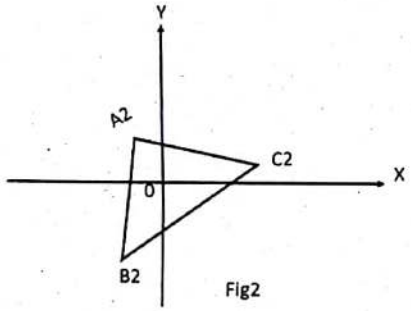


Fig2

Step 3: Inverse translation to step 1 so that the fixed point reaches to its original place.

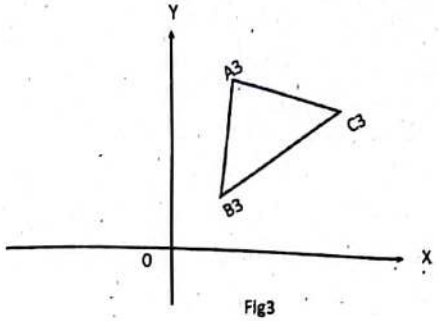


Fig3

Composite matrix $(C_m) = T^{-1}_{(5,6)} R_{\theta=45(\text{anti-clockwise})} T_{(-5,-6)}$

$$= \begin{bmatrix} 1 & 0 & 6 \\ 0 & 1 & 6 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -5 \\ 0 & 1 & -6 \\ 0 & 0 & 1 \end{bmatrix}$$

Calculate yourself

$P' = C_m.P$

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} 0.7071 & -0.7071 & 5.7071 \\ 0.7071 & -0.7071 & -1.7781 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 5 & 2 & 8 \\ 7 & 5 & 5 \\ 1 & 1 & 1 \end{bmatrix}$$

6. **Rotate a triangle at A(0,0), B(6,0) and C(3,3) by 90° about origin in anticlockwise direction.**

Solution:

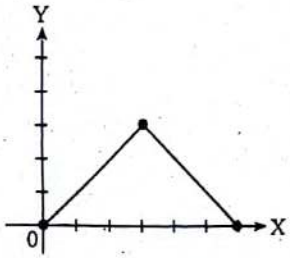


Fig.: Original image

Given point: A(0,0), B(6,0), C(3,3)

θ = 90°

Direction = Anticlockwise

Now from the rotation matrix we have

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\therefore A' = \begin{bmatrix} \cos 90 & -\sin 90 \\ \sin 90 & \cos 90 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\therefore B' = \begin{bmatrix} \cos 90 & -\sin 90 \\ \sin 90 & \cos 90 \end{bmatrix} \begin{bmatrix} 6 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 6 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0 \\ 6 \end{bmatrix}$$

$$\text{And } C' = \begin{bmatrix} \cos 90 & -\sin 90 \\ \sin 90 & \cos 90 \end{bmatrix} \begin{bmatrix} 3 \\ 3 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 3 \\ 3 \end{bmatrix}$$

$$= \begin{bmatrix} -3 \\ 3 \end{bmatrix}$$

$\therefore$ A(0,0) → A'(0,0),
B(6,0) → B'(0,6),
C(3,3) → C'(−3,3)

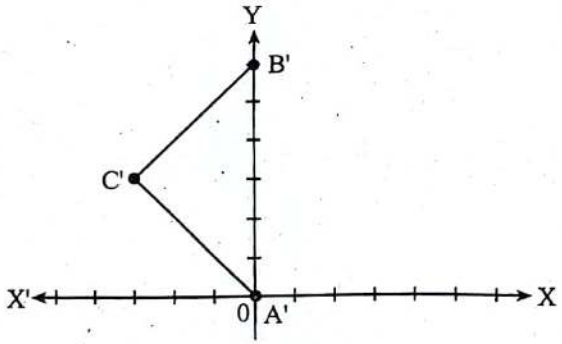


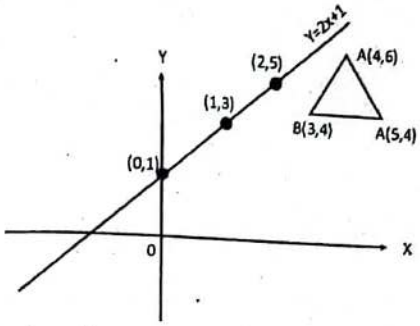
Fig.: Transformed image

7. For the triangle ABC such that A(4,6), B(3,4) and C(5,4), find the mirror-reflection about an arbitrary line y = 2x + 1.

Solution:

For a given line y = 2x + 1,

X	0	1	2
Y	1	3	5



Step 1: Translate the object so that line y = 2x + 1 passes through the origin

Here, $t_x = 0$ and $t_y = 1$

i.e. $X' = X - t_x$

and $Y' = Y - t_y$

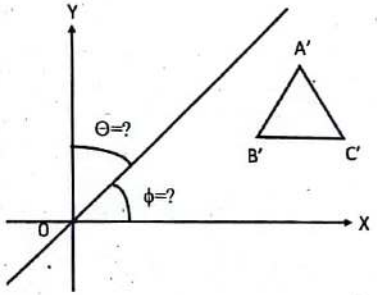
For ΔABC;

X	0	1	2
Y	0	2	4

A(4,6) = A'(4,5)

B(3,4) = B'(3,3)

C(5,4) = C'(5,3)



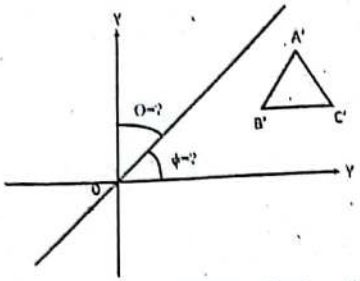
Step 2: In order to rotate the object, we need to find the angle of rotation about the origin.

Here, slope of the line (m) = $\frac{\Delta y}{\Delta x}$

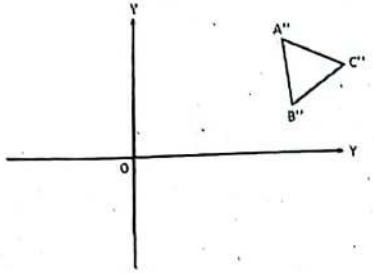
or, $\tan\phi = \frac{2-0}{1-0} = \frac{2}{1}$

$\therefore \phi = \tan^{-1}(2) = 63.43°$

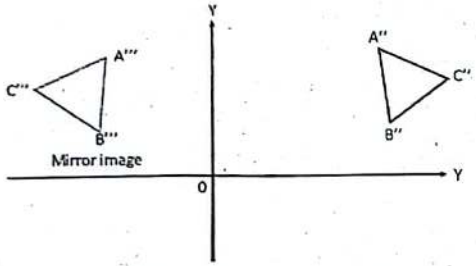
Here angle of rotation = 63.43°, if you want to rotate the object clockwise.



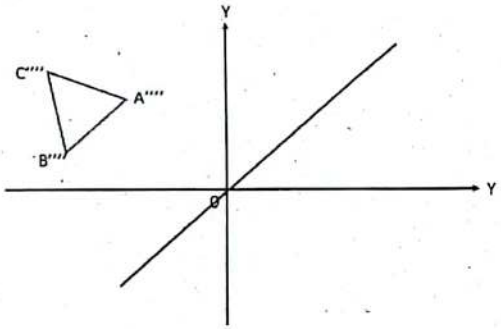
Step 3: Rotate the object in anti-clockwise direction through an angle of 63.43° about an origin.



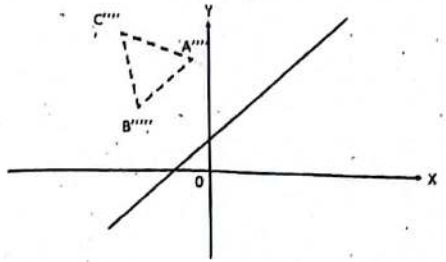
Step 4: Reflect the object about Y-axis.



Step 5: Rotate the object in clockwise direction through an angle of 63.43° about an origin. (just reverse to step 3).



Step 6: Translate the object so that the line reaches to the original place (Inverse translation to step 1),



$$C_m = T^{-1}_{(0,1)} R^{-1}_{\theta=63.43(\text{clockwise})} Rf_{y\text{-axis}} R_{\theta=63.43(\text{C clockwise})} T_{(0,-1)}$$

$$= \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix}$$

8. Create a triangle A(4,5), B(2,1), C(6,1) about any arbitrary point (4,3) in a counterclockwise direction by 90°. What will be the final coordinates of

Solution:

A(4,5), B(2,1), C(6,3)

Point (4,3)

θ = 90° (cw)

$$C_m = T^{-1}_{(4,3)} R_{\theta=90°(cw)} T_{(-4,-3)}$$

$$= \begin{bmatrix} 1 & 0 & 4 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos 90° & -\sin 90° & 0 \\ \sin 90° & \cos 90° & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -4 \\ 0 & 1 & -3 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 4 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -4 \\ 0 & 1 & -3 \\ 0 & 0 & 1 \end{bmatrix}$$

Calculate yourself:

$$P' = C_m \times P = C_m \times \begin{bmatrix} 4 & 2 & 6 \\ 5 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

9. Find the mirror reflection of a line segment formed by points A(−1,−1) and B(2,1) around the axis described by X= −2.

Solution:

A(−1,−1), B(2,1)

X = −2

$C_m = T^{-1}_{(-2,0)} \text{Ref-y-axis } T_{(2,0)}$

$$= \begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Calculate y-axis

$\therefore$ $P' = C_m \times P$

$$= C_m \times \begin{bmatrix} -1 & 2 \\ -1 & 1 \\ 1 & 1 \end{bmatrix}$$

10. Scale an object (4,4) (3,2) (5,2) about a fixed point (4,3) by Scaling Factor 2 on both directions.

Solution:

Given points

A(4,4), B(3,2) and C(5,2)

Scaling factors,

$S_x = S_y = 2$

Pivot point $(x_f, y_f) = (4,3)$

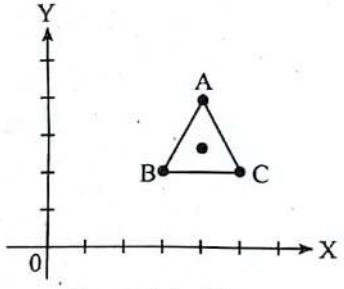


Fig.: Original image

Now first translate the object such that the pivot point is shifted to origin.

For this translation distance (−4,−3) since $\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$

$\therefore$ A is translated to (4 − 4, 4 − 3) = (0,1)

B is translated to (3 − 4, 2 − 3) = (−1,−1)

C is translated to (5 − 4, 2 − 3) = (1,−1)

Then apply scaling about origin.

Since $\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$\therefore$ A is scaled to (2 × 0 + 0 × 1, 0 × 0 + 2 × 1) = (0,2)

B is scaled to (2 × −1, 2 × −1) = (−2,−2)

C is scaled to (2 × 1, 2 × −1) = (2,−2)

Finally apply inverse translation to shift to its original position.

For this translation distance = (4,3) since, $\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$

Hence A is translated to (0 + 4, 2 + 3) = (4, 5)

B is translated to (−2 + 4, −2 + 3) = (2, 1)

Cis translated to (2 + 4, −2 + 3) = (6,1)

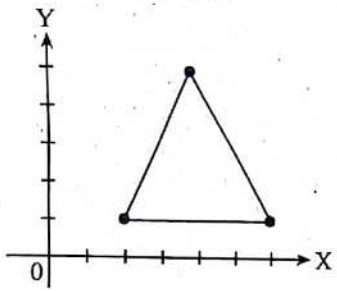


Fig.: Transformed image

$\therefore$ A(4,4) → A'(4,5)

B(3,2) → B'(2,1)

C(5,2) → C'(6,1)

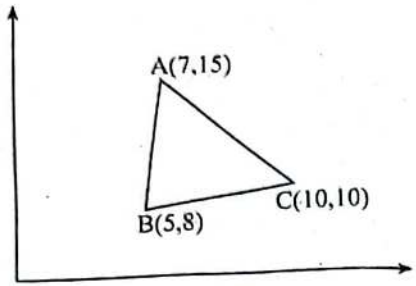
Note: Can be directly done by using following transformation equation

$x' = x_f + S_x(x - x_f)$

$y' = y_f + S_y(y - y_f)$

11. Rotate triangle ABC by 45° clockwise about origin and scale it by (2,3) about origin.

Solution:



Step I: Rotation by 45° (clock wise)

Step II: Scaling by (2,3) S(2,3)

Net transformation: = S(2,3), R(−45)

$$= \begin{bmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos 45 & \sin 45 & 0 \\ -\sin 45 & \cos 45 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{2}{\sqrt{2}} & \frac{2}{\sqrt{2}} & 0 \\ -\frac{3}{\sqrt{2}} & \frac{3}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Finally the transformed co-ordinates are

$A' = TA$

$$\begin{bmatrix} \frac{2}{\sqrt{2}} & \frac{2}{\sqrt{2}} & 0 \\ -\frac{3}{\sqrt{2}} & \frac{3}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 7 \\ 15 \\ 1 \end{bmatrix} =$$

$B' = TB$

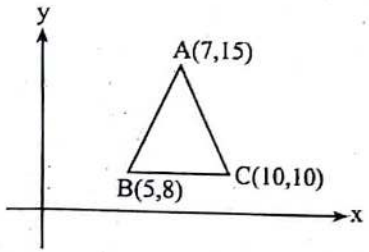
$$\begin{bmatrix} \frac{2}{\sqrt{2}} & \frac{2}{\sqrt{2}} & 0 \\ -\frac{3}{\sqrt{2}} & \frac{3}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 5 \\ 8 \\ 1 \end{bmatrix} =$$

$C' = TC$

$$\begin{bmatrix} \frac{2}{\sqrt{2}} & \frac{2}{\sqrt{2}} & 0 \\ -\frac{3}{\sqrt{2}} & \frac{3}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 10 \\ 10 \\ 1 \end{bmatrix} =$$

12. Rotate the ΔABC by 90° anti clockwise about (5,8) and scale it by (2,2) about (10,10)

Solution:



The composite matrix for the above transformation can be defined as

$T = T_{(10,10)} S_{(2,2)} T_{(-10,-10)} T_{(5,8)} R_{(\theta=90°)} T_{(-5,-8)}$

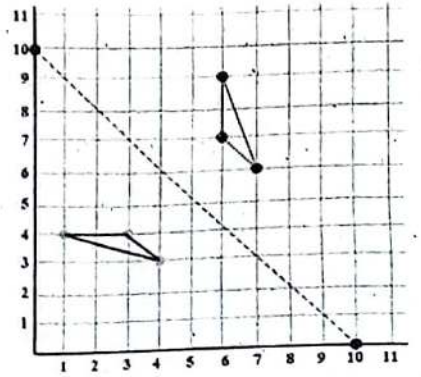
$$= \begin{bmatrix} 1 & 0 & 10 \\ 0 & 1 & 10 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -10 \\ 0 & 1 & -10 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 8 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} \cos 90 & -\sin 90 & 0 \\ \sin 90 & \cos 90 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -5 \\ 0 & 1 & -8 \\ 0 & 0 & 1 \end{bmatrix}$$

Complete yourself .....

13. A mirror is placed such that it passes through (0,10) (10,0). Find the mirror image of an object (6,7), (7,6), (6,9).

Solution:



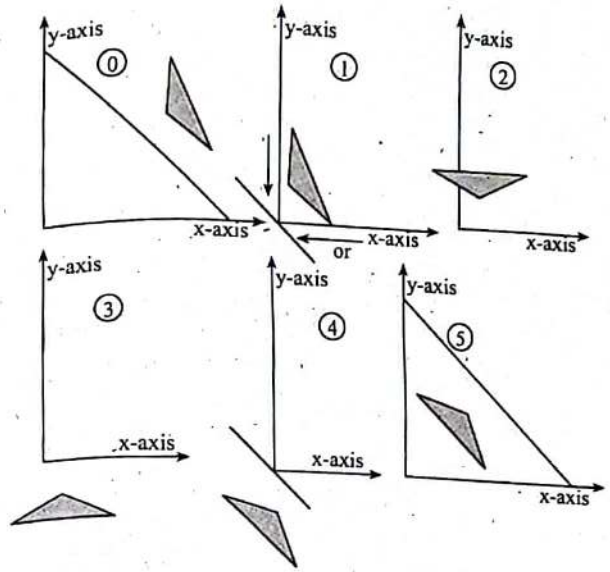
Now, the slope of the line $(m) = \frac{(y_2 - y_1)}{(x_2 - x_1)} = \frac{(0 - 10)}{(10 - 0)} = -1$

Thus, the rotation angle $(\theta) = \tan^{-1}(m) = \tan^{-1}(-1) = -45°$

The composite matrix for the above transformation can be defined as

$T = T_{(-10,0) \text{ or } (0,-10)} R_{(\theta=45)} R_x R_{(\theta=45)} T_{(-10,0) \text{ or } (0,-10)}$

Addition x-intercept	CW rotation	Reflection about x-axis	CCW rotation	Reduce x-intercept
$= \begin{pmatrix} 1 & 0 & 10 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} \cos45 & \sin45 & 0 \\ -\sin45 & \cos45 & 0 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} \cos45 & -\sin45 & 0 \\ \sin45 & \cos45 & 0 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & -10 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$
$= \begin{pmatrix} 1 & 0 & 10 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & -10 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$
$= \begin{pmatrix} 1 & 0 & 10 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & -\frac{10}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & \frac{10}{\sqrt{2}} \\ 0 & 0 & 1 \end{pmatrix}$	
$= \begin{pmatrix} 1 & 0 & 10 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 0 & 10 \\ 0 & 0 & 1 \end{pmatrix}$			
$= \begin{pmatrix} 0 & -1 & 10 \\ -1 & 0 & 10 \\ 0 & 0 & 1 \end{pmatrix}$				



Now, the required co-ordinate can be calculated as:

$$P' = Com \times P = \begin{pmatrix} 0 & -1 & 10 \\ -1 & 0 & 10 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 6 & 7 & 6 \\ 7 & 6 & 9 \\ 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 3 & 4 & 1 \\ 4 & 3 & 4 \\ 1 & 1 & 1 \end{pmatrix}$$

Hence, the final co-ordinates are (3,4), (4,3) & (1,4).

14. Magnify the triangle with vertices A(0,0), B(1,1) and C(5,2) to twice its size while keeping C(5,2) fixed.

Steps:

1. First, translate the triangle by $T_x = -5$ and $T_y = -2$
2. Then magnify the triangle by twice its size i.e. Scaling factor S_x and S_y is equal to 2.
3. Again translate the triangle by $T_x = 5$ and $T_y = 2$.

15. Reflect the subject (2,3), (4,3), (4,5) about lin y = x + 1.

Solution:

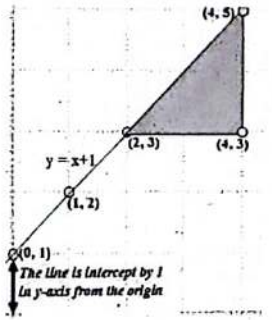
The given line is y = x + 1.

Thus,

When x = 0, y = 1

When x = 1, y = 2

When x = 2, y = 3



Also,

The slope of the line (m) = 1

Thus, the rotation angle (θ) = $\tan^{-1}(m) = \tan^{-1}(1) = 45°$

Here, the required steps are:

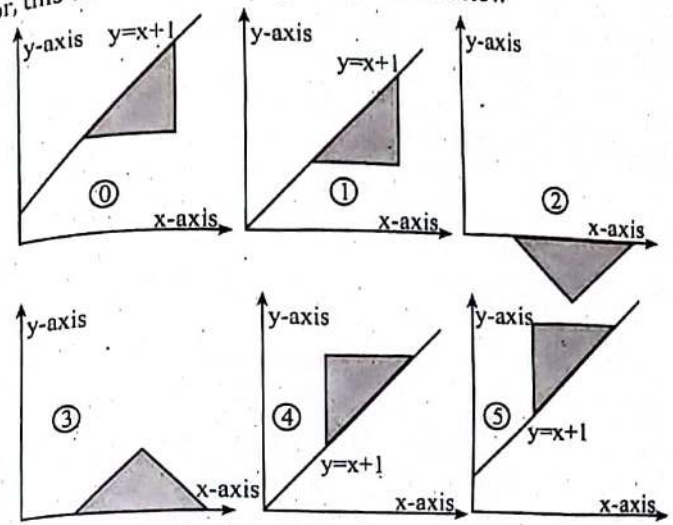
- Translate the line to origin by decreasing the y-intercept with one.
- Rotate the line by angle 45° in clockwise direction so that the given line must overlap x-axis.
- Reflect the object about the x-axis.
- Reverse rotate the line by angle –45° in counter-clockwise direction.
- Reverse translate the line to original position by adding the y-intercept with one.

Thus, the composite matrix is given by: $Com = T'R_0'R_{fx}R_0T$

Thus, composite matrix is given by

Addition y-intercept	CCW rotation	Reflection about x-axis	CW rotation	Reduce y-intercept
$= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} \cos 45 & -\sin 45 & 0 \\ \sin 45 & \cos 45 & 0 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} \cos 45 & \sin 45 & 0 \\ -\sin 45 & \cos 45 & 0 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{pmatrix}$
$= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{pmatrix}$
$= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & -\frac{10}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & -\frac{10}{\sqrt{2}} \\ 0 & 0 & 1 \end{pmatrix}$	
$= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & -\frac{10}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & \frac{10}{\sqrt{2}} \\ 0 & 0 & 1 \end{pmatrix}$		
$= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 & -1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$			
$= \begin{pmatrix} 0 & 1 & -1 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix}$				

or, this can also be directly computed as below



Also, the composite matrix can be calculated as:

$$= \begin{pmatrix} \frac{-(m^2-1)}{(m^2+1)} & \frac{2m}{(m^2+1)} & \frac{2mc}{(m^2+1)} \\ \frac{2m}{(m^2+1)} & \frac{(m^2-1)}{(m^2+1)} & \frac{2c}{(m^2+1)} \\ 0 & 0 & 1 \end{pmatrix}$$

Where, m = 1, c = 1

$$= \begin{pmatrix} 0 & 1 & -1 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

Now, the required co-ordinate can be calculated as:

$P' = Com \times P$

$$= \begin{pmatrix} 0 & 1 & -1 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 2 & 4 & 4 \\ 3 & 3 & 5 \\ 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 2 & 4 \\ 3 & 5 & 5 \\ 1 & 1 & 1 \end{pmatrix}$$

Hence, the final co-ordinates are (2,3), (2,5) & (4,5).

16. Find the transformation matrix that transforms the square ABCD whose center is at (2,2) is reduced to half of its size, with center still remaining at (2,2). The co-ordinates of the square ABCD are A(0,0), B(0,4), C(4,4) and D(4,0). Find the coordinates of the new square.

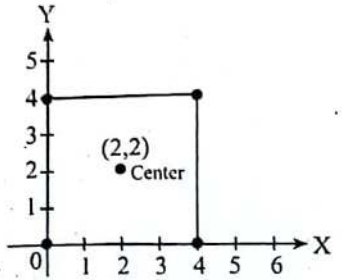
Solution:

Given points of square

A(0,0), B(4,4), C(4,4), D(4,0)

Now, we have to reduce it to half, so scaling factor

$S_x = S_y = \frac{1}{2}$



We have scaling matrix of

$$S = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

So applying the transformation, we get

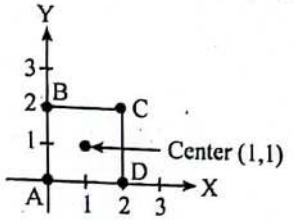
$$A' = SA = \begin{bmatrix} \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$B' = SB = \begin{bmatrix} \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 4 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix}$$

$$C' = SC = \begin{bmatrix} \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 4 \\ 4 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \\ 1 \end{bmatrix}$$

$$D' = SD = \begin{bmatrix} \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 4 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ 1 \end{bmatrix}$$

∴ New points after scaling are A(0,0), B(0,2), C(2,2), D(2,0)



So finally translate so as to shift the center (1,1) to (2,2)

i.e. $t_x = 1$ and $t_y = 1$

The translation matrix is given by

$$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

So applying this transformation we get

$$A' = TA = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = (1,1)$$

$$B' = TA = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \\ 1 \end{bmatrix} = (1,3)$$

$$C' = TC = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 3 \\ 1 \end{bmatrix} = (3,3)$$

$$D' = TD = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \\ 1 \end{bmatrix} = (3,1)$$

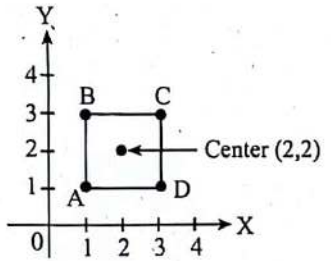


Fig.: Final transformed image

17. The reflection along the line y = x is equivalent to reflection along x-axis followed by counter clockwise rotation by θ degree. Find the θ.

Solution:

The transformation matrix for y = x is $\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Reflection about x-axis is $\begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$

Counter clockwise rotation about θ

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Therefore the composite transformation matrix is given by

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ \sin\theta & -\cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

it is given that,

$$\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ \sin\theta & -\cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

i.e. $\sin\theta = 1$ i.e. $\theta = 90$

18. Rotate a triangle A(5,6), B(6,2) and C(4,1) by 45 degree about an arbitrary point (3,3).

Solution:

Composite matrix

$= T_{(3,3)}R_{(45)}T_{(-3,-3)}$

$$= \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} \cos45° & \sin45° & 0 \\ \sin45° & \cos45° & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & -3 \\ 0 & 1 & -3 \\ 0 & 0 & 1 \end{bmatrix}$$

Now, P' = C.M. × P

$$= \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 3 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{-6}{\sqrt{2}}+1 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 5 & 6 & 4 \\ 6 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} -\frac{1}{\sqrt{2}}+3 & -\frac{4}{\sqrt{2}}+3 & 3 \\ \frac{5}{\sqrt{2}}+1 & \frac{2}{\sqrt{2}}+1 & -\frac{1}{\sqrt{2}}+1 \\ 1 & 1 & 1 \end{bmatrix}$$

19. Scale the triangle with vertices A(1,1), B(4,4) and C(2,3) to double along horizontal direction and triple of vertical direction about point (2,3).

Solution:

Here, the vertices of the triangle and A(1,1), B(4,4) and C(2,3).

Scaling factor along horizontal direction $(S_x) = 2$

Scaling factor along vertical direction $(S_y) = 3$

Fixed point about while scaling is done is (2,3)

Now, P' = C.M. × P

Composite matrix (C.M.) $= T_{(2,3)}S_{(2,3)}T_{(-2,-3)}$

$$= \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & -3 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 2 & 0 & -4 \\ 0 & 3 & -9 \\ 0 & 0 & 1 \end{bmatrix}$$

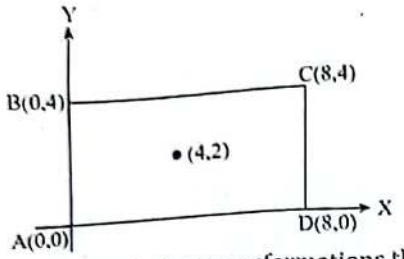
$$= \begin{bmatrix} 2 & 0 & -2 \\ 0 & 3 & -6 \\ 0 & 0 & 1 \end{bmatrix}$$

$$P' = \begin{bmatrix} 2 & 0 & -2 \\ 0 & 3 & -6 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 4 & 2 \\ 1 & 4 & 3 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 6 & 4 \\ -3 & 6 & 3 \\ 1 & 1 & 1 \end{bmatrix}$$

So, the new scaled vertices are A'(0,−3), B'(6,6) and C'(4,3).

20. Find the transformation matrix that transforms the rectangle ABCD whose center is at (4,2) is reduced to half of its size, the center will remain same. The co-ordinate of ABCD are A(0,0), B(0,4), C(8,4) and D(8,0). Find co-ordinate of new square. Also derive the transformation matrix to convert this rectangle to square.

Solution:



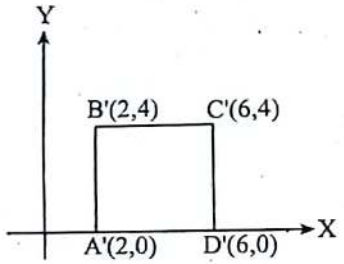
The transformation matrix that transformations the rectangle ABCD whose center is (4,2) is reduced to half of its size keeping the center same is,

$P' = C.M. \times P$

$$C.M. = \begin{bmatrix} 1 & 0 & 4 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1/2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -4 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix}$$

$$C.M. = \begin{bmatrix} 1 & 0 & 4 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1/2 & 0 & -2 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1/2 & 0 & 2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Now to find new co-ordinate of new square,



$A' = C.M. \times A$

$$\begin{bmatrix} 1/2 & 0 & 2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ 1 \end{bmatrix}$$

$B' = C.M. \times B$

$$\begin{bmatrix} 1/2 & 0 & 2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 4 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \\ 1 \end{bmatrix}$$

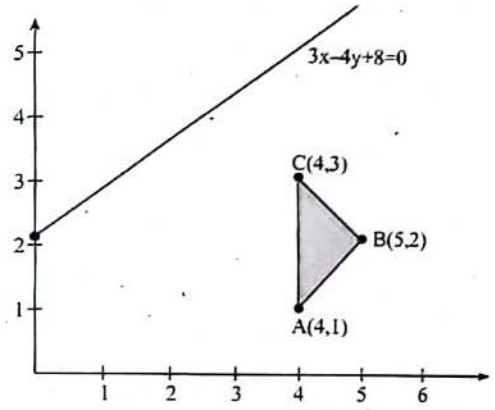
$C' = C.M. \times C$

$$\begin{bmatrix} 1/2 & 0 & 2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 8 \\ 4 \\ 1 \end{bmatrix} = \begin{bmatrix} 6 \\ 4 \\ 1 \end{bmatrix}$$

$$D' = C.M. \times D = \begin{bmatrix} 1/2 & 0 & 2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 8 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 6 \\ 0 \\ 1 \end{bmatrix}$$

The co-ordinates of new square are (2,0), (2,4), (6,4), and (6,0).

21. Reflect the triangle ABC about the line 3x-4y+8=0. The position vector of the coordinate ABC is given as A=(4,1), B=(5,2),C=(4,3). Find the composite transformation matrix and its new coordinates.



Equation of line is $3x - 4y + 8 = 0$, i.e.; $4y = 3x + 8$

i.e.; $y = \left(\frac{3}{4}\right)x + 2$

Comparing with

$y = mx + c$, $m = \frac{3}{4}$ and $c = 2$

Also

$\theta = \tan^{-1} m$

$\theta = -26.87°$

Step 1: Transformation of Point P(0,2) to origin O(0,0)

$$[T_r] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -2 & 1 \end{bmatrix} \{t_x = 0, t_y = -2\}$$

(–ve as direction is downward.)

Step 2: Rotation of line by an angle of $\theta = -36.87°$

(–ve as it is moving in clockwise direction)

$$[R] = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos(-36.87) & \sin(-36.87) & 0 \\ -\sin(-36.87) & \cos(-36.87) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\therefore [R] = \begin{bmatrix} 0.8 & -0.6 & 0 \\ 0.6 & 0.8 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Step 3: Reflection of triangle about x-axis

Matrix for reflection about x-axis is given as,

$$[M]_{@x\text{-axis}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Step 4: Reverse rotation of line to its original angle

$$[R_x]^{-1}_{CCW} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos(36.87) & \sin(36.87) & 0 \\ -\sin(36.87) & \cos(36.87) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0.8 & 0.6 & 0 \\ -0.6 & 0.8 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Step 5: Inverse translation of point P to its original position

$$[T_r] - 1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 2 & 1 \end{bmatrix} \{t_x = 0, t_y = +2\}$$

(+ve as direction is upwards)

Now,

The composite transformation matrix,

$[T] = [T_r][R][M][R]^{-1}[Tr]^{-1}$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -2 & 1 \end{bmatrix} \begin{bmatrix} 0.8 & -0.6 & 0 \\ 0.6 & 0.8 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0.8 & 0.6 & 0 \\ -0.6 & 0.8 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 2 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0.28 & 0.96 & 0 \\ 0.96 & -0.28 & 0 \\ -1.92 & 0.56 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 2 & 1 \end{bmatrix}$$

$$[T] = \begin{bmatrix} 0.28 & 0.96 & 0 \\ 0.96 & -0.28 & 0 \\ -1.92 & 2.56 & 1 \end{bmatrix}$$

Now, new co-ordinate of triangle ABC are

$[X'] = [X]\,[T]$

$$= \begin{bmatrix} 4 & 1 & 1 \\ 5 & 2 & 1 \\ 4 & 3 & 1 \end{bmatrix} \begin{bmatrix} 0.28 & 0.96 & 0 \\ 0.96 & -0.28 & 0 \\ -1.92 & 2.56 & 1 \end{bmatrix} = \begin{bmatrix} 0.16 & 6.12 & 1 \\ 1.4 & 6.8 & 1 \\ 2.08 & 5.56 & 1 \end{bmatrix}$$

A' = (0.16, 6.12)

B' = (1.4, 6.8)

C' = (2.08, 5.56)

22. A Triangle with vertices A(0,10), B(10,0), C(–10,0) is required to be shifted down by 5 units, then rotate in anticlockwise direction by 30 degrees and scaled by twice its original size. What will be the final location?

Solution:

$CM = T(t_x, t_y) R_{\theta CCW} T(-t_x, t_y)$

$$= \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & -t_x \\ 0 & 1 & -t_y \\ 0 & 1 & \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 5 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos 30° & -\sin 30° & 0 \\ \sin 30° & \cos 30° & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -5 \\ 6 & 0 & 1 \end{bmatrix}$$

$$P' = CM \begin{pmatrix} 0 & 10 & -10 \\ 10 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$$

Now on scaling about S(2,2)

$CM = T(t_x, t_y)\ S(2,2)\ T(-t_x, -t_y)$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 5 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -5 \\ 0 & 0 & 1 \end{bmatrix}$$

$\therefore\ R' = CM'.P'$

23. Reflect a Triangle A(1,0) B(3,1) C(1,2) about the line $y = -x + 5$ then scale it about a fixed point P(10,10) by 2.

Solution:

$CM = T^{-1}_{(0,5)} R^{-1}_{0(45° CCW)}\ Ref_{x\text{-axis}}\ R_{0(45° CW)}\ T_{(0,-5)}$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 5 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(-45°) & -\sin(-45°) & 0 \\ \sin(-45°) & \cos(-45°) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(-45°) & \sin(-45°) & 0 \\ -\sin(-45°) & \cos(-45°) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -5 \\ 0 & 0 & 1 \end{bmatrix}$$

= --- *Calculate yourself.*

$P' = CM \times P$

$$P' = CM \times \begin{bmatrix} 1 & 3 & 1 \\ 0 & 1 & 2 \\ 1 & 1 & 1 \end{bmatrix} = P_1$$

Again we have to scale it

$S_x = S_y = 2$

Now again

$CM' = T^{-1}_{(10,10)}\ S_{Sx=Sy=2}\ T_{(-10,-10)}$

$$= \begin{bmatrix} 1 & 0 & 10 \\ 0 & 1 & 10 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -10 \\ 0 & 1 & -10 \\ 0 & 0 & 1 \end{bmatrix}$$

Final Point

$P_2 = CM' \times P_1$

... *Calculate yourself*

24. A triangle with vertices A(5,2), B(4,1), C(6,1) is required to be rotated by 45 degrees in counter clockwise direction about i. origin and ii. line y = 5

Solution:

A(5, 2), B(4, 1), C(6, 1)

i. About origin by 45° CCW

$$= \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 5 & 4 & 6 \\ 2 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos 45° & -\sin 45° & 0 \\ \sin 45° & \cos 45° & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 5 & 4 & 6 \\ 2 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

... *Calculate yourself*

ii. About line y = 5 by 45° CCW

$CM = T^{-1}_{(0,5)}\ R_{0=45° CCW}\ T_{(0,-5)}$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 5 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos 45° & -\sin 45° & 0 \\ \sin 45° & \cos 45° & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

= ... *calculate yourself*

$\therefore\ P' = CM \times P$

$$= CM \times \begin{bmatrix} 5 & 4 & 6 \\ 2 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Two-Dimensional Viewing

2D viewing is the process of displaying and visualizing two-dimensional objects or scenes on a two-dimensional display surface. It involves transforming and projecting the objects from their original representation onto a 2D space, such as a computer screen or a piece of paper. This transformation includes operations like translation, scaling, rotation, and projection to position and size the objects appropriately within the viewing window or viewport. The visible portion of the scene is determined by clipping, which removes objects or parts of objects that fall outside the viewing window. The result is a rendered image of the 2D scene, allowing users to interact with and perceive the objects on the display surface.

Viewing Transformation

Displaying an image of a picture involves mapping the co-ordinates of the points and lines that form the picture into the appropriate co-ordinates on the device or workstation where the image is to be displayed. This is done through the use of co-ordinate transformations known as viewing transformation. To perform a viewing transformation we deal with finite region in the WCS(World coordinate system) called **window.** The window can be mapped directly on the display area of the device or on to a sub region of the display called **viewport.**

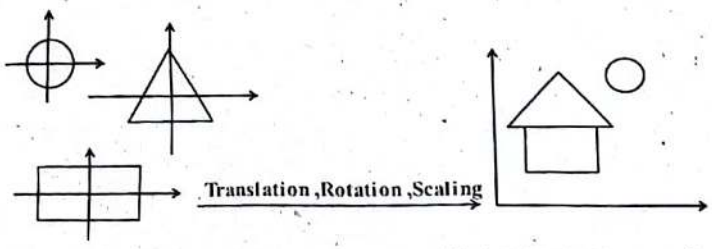


Fig.: (a) Local co-ordinate *Fig. (b) World co-ordinate*

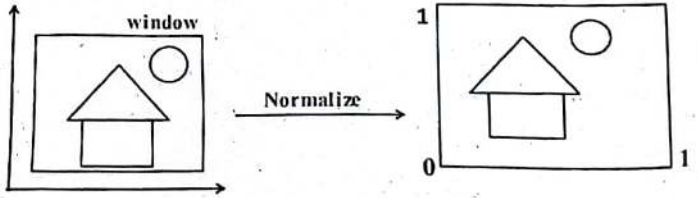


Fig.: (c) Viewing co-ordinate *Fig.: (d) Normalized co-ordinate*

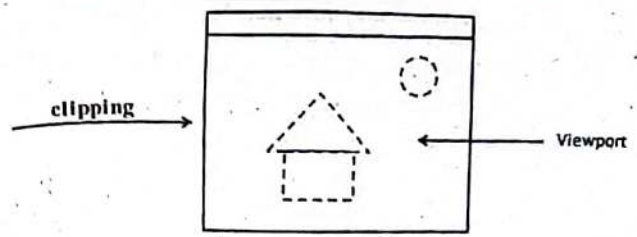


Fig.: (e) Device co-ordinate

Local Co-ordinate:

The shape of the individual object can be constructed in a scene within the separate co-ordinate reference frame is called local co-ordinate.

Also called as master co-ordinate or modeling co-ordinate.

World Co-ordinate:

Once the individual object is identified, those objects are placed into appropriate position within the scene using frame and this reference frame is called world co-ordinate.

Device Co-ordinate:

When the world co-ordinate description of the scene is transformed to one or more output device reference frame for display, then display co-ordinate system is referred to as device co-ordinate.

Also called as screen co-ordinate in case of video monitor.

Viewing Co-ordinate:

Used to define window in the world co-ordinate plane with any possible orientation.

Normalized Device Coordinate

- The co-ordinates are in the range of 0 to 1.
- Generally , a graphical system first converts world co-ordinate position to normalized device co-ordinate before final conversion to specified device co-ordinate.
- This makes the system independent of the various device co-ordinates that might be used at a workstations.

2D-Viewing Pipeline:

Steps to be followed for window to viewport transformation or 2D viewing pipeline are:

1. Construct world co-ordinate scene using modeling co-ordinate transformation.

2. Convert world co-ordinate to view co-ordinate.
3. Map the view co-ordinate to normalized viewing co-ordinate.
4. Map normalized viewing co-ordinate to device co-ordinate system.

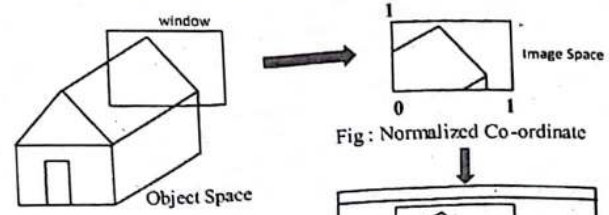


Fig : World Co-ordinate

Fig : Normalized Co-ordinate

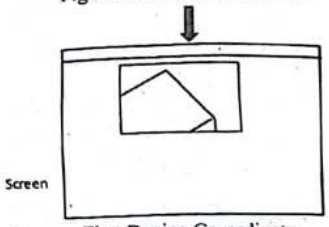


Fig : Device Co-ordinate

Fig: A 2D viewing Transformation .

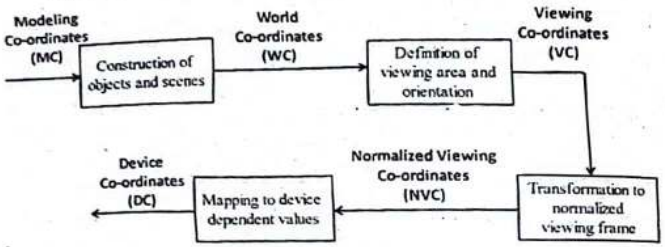


Fig.: A 2D viewing transformation pipeline

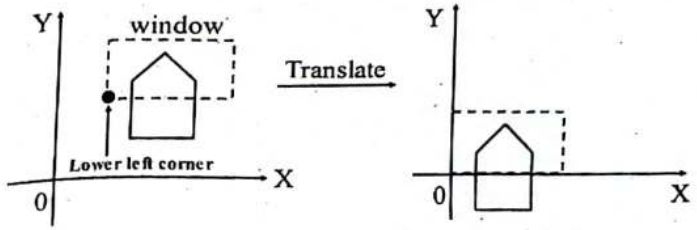
- For a fixed size viewport, zooming in effect is obtained if the window size is decreased and zooming out effect is obtained if the window size is increased.
- The NVC maps the WC to a viewport of maximum size equals to unit square.

Window to Viewport Transformation

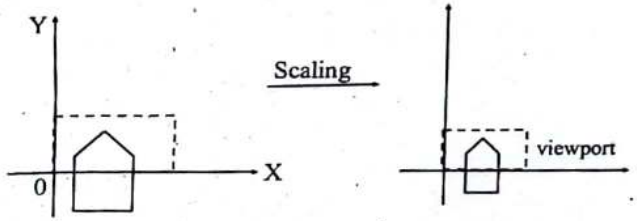
- **Window:** This is the rectangular surrounding the object as a part of it that we wish to draw on the screen. It is associated with the object rather than image.
- **Viewport:** This is the rectangular region of the screen which is selected for displaying the object or a part of it described in a window
- The mapping of a part of world co-ordinate scene to a device co-ordinate is referred to as **viewing transformation.**
- Sometimes, the 2D viewing transformation is simply referred to as the **window-to-viewport transformation** or **windowing transformation.**

- **A window to viewport co-ordinate transformation can be explained as follows:**

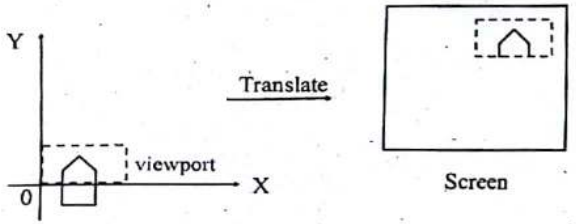
Step 1: The object together with its window is translated until the lower-left corner of the window is at origin.



Step 2: The object and the window are now scaled until the window has the dimension same as viewport. In this step, we are converting the object into image and the window in viewport.



Step 3: The final transformation step is another translation to move the viewport to its correct position on the screen.



- **We can summarize the above steps as below:**

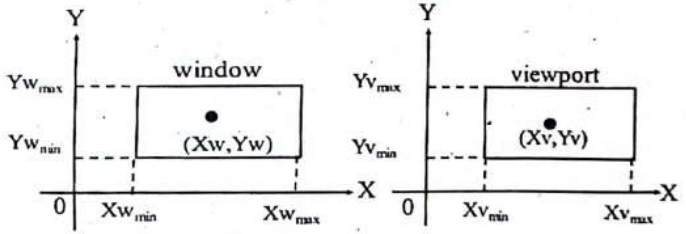


Fig: World Co-ordinate

Fig: Device Co-ordinate

- The composite matrix for above steps is;

$T_{wv} = T_{(Xv_{min}, Yv_{min})} S_{(S_x, S_y)} T_{(-Xw_{min}, -Yw_{min})}$

- Here, the value of S_x and S_y is found by calculating the position of (X_w, Y_w) in the window to the corresponding position of point (X_v, Y_v) in the viewport.

$$\text{i.e. } \frac{X_v - X_{v_{min}}}{X_{v_{max}} - X_{v_{min}}} = \frac{X_w - X_{w_{min}}}{X_{w_{max}} - X_{w_{min}}}$$

$$\text{or, } X_v = X_{v_{min}} + \frac{X_w - X_{w_{min}}}{X_{w_{max}} - X_{w_{min}}} \times (X_{v_{max}} - X_{v_{min}})$$

$$\therefore \quad X_v = X_{v_{min}} + (X_w - X_{w_{min}}) \times S_x$$

$$\text{Also, } \frac{Y_v - Y_{v_{min}}}{Y_{v_{max}} - Y_{v_{min}}} = \frac{Y_w - Y_{w_{min}}}{Y_{w_{max}} - Y_{v_{min}}}$$

$$\text{or, } Y_v = Y_{v_{min}} + \frac{Y_w - Y_{w_{min}}}{Y_{w_{max}} - Y_{v_{min}}} \times (Y_{v_{max}} - Y_{v_{min}})$$

$$\therefore \quad Y_v = Y_{v_{min}} + (Y_w - Y_{w_{min}}) \times S_y$$

Where S_x and S_y are scale factor along X- and Y-direction, respectively.

$$S_x = \frac{\text{width of viewport}}{\text{width of window}} \text{ and } S_y = \frac{\text{height of viewport}}{\text{height of window}}$$

$$= \frac{X_{v_{max}} - X_{v_{min}}}{X_{w_{max}} - X_{w_{min}}} \qquad = \frac{Y_{v_{max}} - Y_{v_{min}}}{Y_{w_{max}} - Y_{w_{min}}}$$

$$T_{wv} = \begin{bmatrix} 1 & 0 & X_{v_{min}} \\ 0 & 1 & Y_{v_{min}} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{X_{v_{max}} - X_{v_{min}}}{X_{w_{max}} - X_{w_{min}}} & 0 & 0 \\ 0 & \frac{Y_{v_{max}} - Y_{v_{min}}}{Y_{w_{max}} - Y_{w_{min}}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -X_{w_{min}} \\ 0 & 1 & -Y_{w_{min}} \\ 0 & 0 & 1 \end{bmatrix}$$

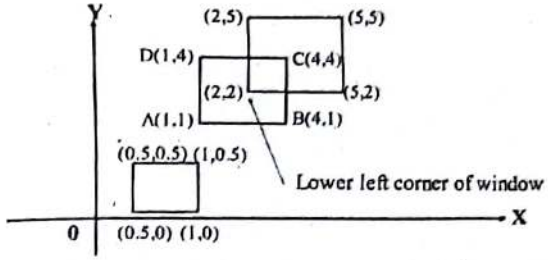
It can be combined to form

$$\begin{bmatrix} \frac{X_{v_{max}} - X_{v_{min}}}{X_{w_{max}} - X_{w_{min}}} & 0 & X_{v_{min}} - X_{w_{min}} \times S_x \\ 0 & \frac{Y_{v_{max}} - Y_{v_{min}}}{Y_{w_{max}} - Y_{w_{min}}} & Y_{v_{min}} - Y_{w_{min}} \times S_y \\ 0 & 0 & 1 \end{bmatrix}$$

$$T_{wv} = T_{(Xv_{min}, Yv_{min})} S_{(S_x, S_y)} T_{(-Xv_{min}, -Yw_{min})}$$

1. **Suppose there is a rectangle ABCD whose co-ordinates are A(1,1), B(4,1), C(4,4), D(1,4) and the window co-ordinates are (2,2), (5,2), (5,5), (2,5) and the given viewport location is (0.5,0) (1,0), (1,0.5), (0.5,0.5). Calculate the viewing transformation matrix.**

Solution:



Step 1: The object together with window is translated in such a way that left lower corner of window is shifted to origin.

Here, $T_x = -2$ and $T_y = -2$

$$\text{Then, translation matrix} = \begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix}$$

Step 2: Scaling of window to viewport is performed and for this we need scaling factor.

$$S_x = \frac{\text{width of viewport}}{\text{width of window}} \text{ and } S_y = \frac{\text{height of viewport}}{\text{height of window}}$$

$$= \frac{X_{v_{max}} - X_{v_{min}}}{X_{w_{max}} - X_{w_{min}}} \qquad = \frac{Y_{v_{max}} - Y_{v_{min}}}{Y_{w_{max}} - Y_{w_{min}}}$$

$$= \frac{1 - 0.5}{5 - 2} \qquad = \frac{0.5 - 0}{5 - 2}$$

$$= 0.16 \qquad = 0.16$$

$$\text{Then, the scaling matrix} = \begin{bmatrix} 0.16 & 0 & 0 \\ 0 & 0.16 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Step3: We translate the viewport to desired location on the screen.

$$\text{Then, the translation matrix} = \begin{bmatrix} 1 & 0 & X_{v_{min}} \\ 0 & 1 & Y_{v_{min}} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0.5 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- **Hence the viewing transformation matrix will be:**

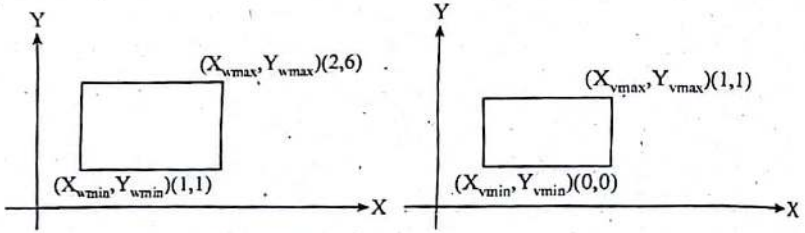
$$T_{wv} = \begin{bmatrix} 1 & 0 & 0.5 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.16 & 0 & 0 \\ 0 & 0.16 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0.5 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.16 & 0 & -0.32 \\ 0 & 0.16 & -0.32 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0.16 & 0 & 0.18 \\ 0 & 0.16 & -0.32 \\ 0 & 0 & 1 \end{bmatrix}$$

2. Find the normalization transformation window to viewport with window lower left corer at (1,1)and upper right corner (2,6) onto a viewport for entire normalized device screen.

Solution:



Since view port for entire normalized device Screen so, viewport can be

$X_{vmin} = 0 \quad X_{wmin} = 1$

$Y_{vmin} = 0 \quad Y_{wmin} = 1$

$X_{vmax} = 1 \quad X_{wmin} = 2$

$Y_{vmax} = 1 \quad Y_{wmax} = 6$

Scaling factor

$$S_x = \frac{\text{width of viewport}}{\text{width of window}} \quad S_y = \frac{\text{height of viewport}}{\text{height of window}}$$

$$= \frac{1-0}{2-1} = 1 \qquad = \frac{1-0}{6-1} = \frac{1}{5}$$

Step 1: Translate using X_{wmin}, Y_{wmin}

$$T_w = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix}$$

Step 2: Translate using X_{vmin}, Y_{vmin}

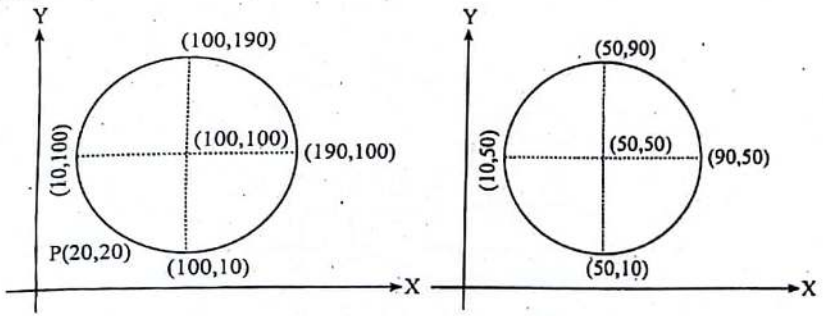
$$T_v = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$T_{wv} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1/5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix}$$

= Calculate yourself

3. A point P(20,20) in a circular window with its center at (100,100) and a radius of 90 pixels is required to be mapped to a point in circular viewport having its at (50,50) and a radius of 40 pixels. Where will the point P be placed after the transformation?

Solution:



1. Translating window to origin T(−100,−100).
2. Scale the window to match view port.

$$S_x = \frac{X_{vmax} - X_{vmin}}{X_{wmax} - X_{wmin}} = \frac{90-10}{190-10} = \frac{80}{180} = 0.44$$

$$S_y = \frac{Y_{vmax} - Y_{vmin}}{Y_{wmax} - Y_{wmin}} = \frac{90-10}{190-10} = \frac{80}{180} = 0.44$$

3. Translate it by $t_x = X_{vmin}$ & $t_y = Y_{vmin}$

$$\therefore \quad CM = T_{(50,50)}.S_{(0.44,0.44)}T_{(-100,-100)}$$

$$= \begin{pmatrix} 1 & 0 & 50 \\ 0 & 1 & 50 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0.44 & 0 & 0 \\ 0 & 0.44 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & -100 \\ 0 & 1 & -100 \\ 0 & 0 & 1 \end{pmatrix}$$

Scaling about P(10,10)

$$CM' = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 5 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 10 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & -5 \\ 0 & 0 & 1 \end{pmatrix}$$

$\therefore$ R' = CM'.P'

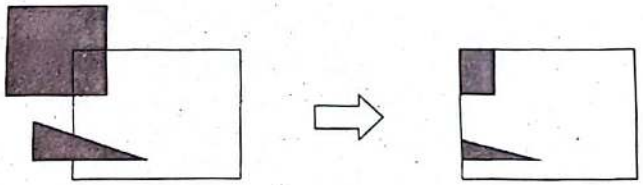
Clipping

- Any procedure that identifies those portions of a picture that are either inside or outside of a specified region of a space is referred to as a Clipping algorithm or simply Clipping.
- The region against which an object is to be clipped is called a clip window.
- Clipping algorithm can be applied in word co-ordinates so that only the contents of the window interior are mapped to device co-ordinates.
- The purpose of a clipping algorithm is to determine which points, lines or portions of lines lay within the clipping window.
- For the viewing transformation, we want to display only those picture parts that are within the clipping window area.
- Everything outside the window is discarded.

Applications of clipping

We can use clipping for drawing operations.

- It is used for separating the important part of an image.
- The solid modeling technique in clipping helps to build three dimensional objects.
- We can perform various operations that correlate with the pointing of an object. For example, delete, copy, insert, and moving selected parts of an object, etc.
- Clipping helps us to describe the visible and invisible parts of 3D objects.



Some of Clipping algorithms are

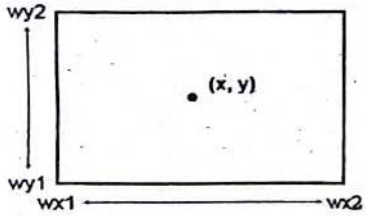
1. Point clipping
2. Line clipping
3. Polygon clipping (Area)
4. Curve clipping
5. Text clipping
6. Exterior clipping

1. Point clipping:

- Clipping a point from a given window is very easy. Assuming that the clip window is a rectangle in standard position, we save a point P(x,y) for display if the following inequalities are satisfied:

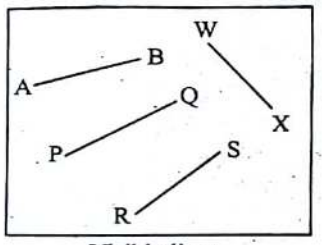
  $X_{wmin} <= X <= X_{wmax}$ and $Y_{wmin} <= Y <= Y_{wmax}$

  Where the X_{wmin}, Y_{wmin}, X_{wmax}, Y_{wmax} are the edge of the clip window.
- If any of these four inequalities is not satisfied, then the point is clipped i.e. not saved for display.

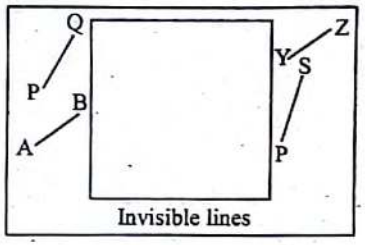


2. Line clipping:

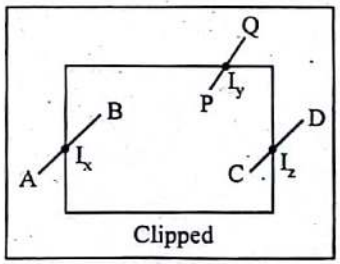
- The concept of line clipping is same as point clipping.
- In line clipping, we will cut the portion of line which is outside of window and keep only the portion that is inside the window.
- **All line segments fall into one of the following categories:**
  1. **Completely visible:** Both the end points of the line segment fall within the clipping window.
  2. **Not visible:** When line completely lies outside the window.
  3. **Partially visible (or clipping candidate):** a line is partially visible when a part of its lies within the window.



Visible lines



Invisible lines

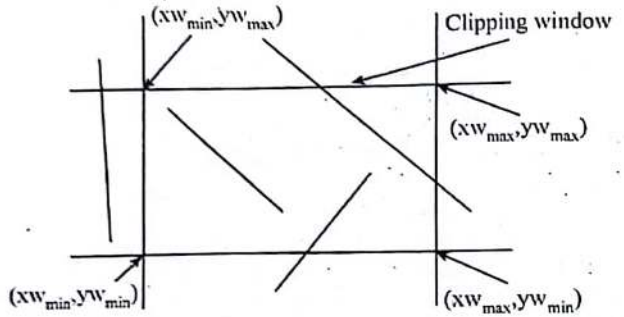


Clipped

Cohen-Sutherland line clipping algorithm:

This algorithm was named by "Danny Cohen" and "Ivan Sutherland."

- One of the oldest and most popular algorithm for line clipping.
- This algorithm divides a 2D space into 9 parts, using the infinite extensions of the four linear boundaries of the window.
- Uses bit operation to perform this test.
- For each end point, a 4-bit binary code called region code is assigned to identify the location relative to the boundaries of the clipping rectangle.



We will use 4-bits to divide the entire region. These 4 bits represent the top, bottom, Right, and left of the region as shown in the following figure. Here, the top and left bit is set to 1 because it is the top-left corner.

	Top			
	1001	1000	1010	
Left	0001	0000	0010	*Right*
	0101	0100	0110	
	Bottom			

There are 3 possibilities for the line

- Line can be completely inside the window. This line should be accepted.
- Line can be completely outside of the window. This line will be completely removed from the region.
- Line can be partially inside the window. We will find intersection point and draw only that portion of line that is inside region.
- The bits in the region code represents: Top, Bottom, Right, Left.

- **Each bit is set to either 1(True) or 0(False).**
  - If the end point is at **Left** side of the window, the First bit is set to 1 else to 0.
  - If the end point is at **Right** side of the window, the Second bit is set to 1 else to 0.
  - If the end point is at **Bottom** side of the window, the Third bit is set to 1 else to 0.
  - If the end point is at **Top** side of the window, the first bit is set to 1 else to 0.
- **Example:**

4th Bit	3rd Bit	2nd Bit	1st Bit
1	0	1	0
T	B	R	L

⇓

Represents an endpoint of line that is top-right of the window.

- P(X,Y)

Window

Advantages of Cohen–Sutherland algorithm

- It calculates end-points very quickly and rejects and accepts lines quickly.
- It can clip pictures much large than screen size.

Algorithm

Step1: Establish Region code for each endpoints of a given line.

- First bit set to '1' if $X<Xw_{min}$ else set to '0'.
- Second bit set to '1' if $X>Xw_{max}$ else set to '0'.
- Third bit set to '1' if $Y<Yw_{min}$ else set to '0'.
- Fourth bit set to '1' if $Y>Yw_{max}$ else set to '0'.

Step2: Determine whether the line is completely inside/outside of the window using following test.

- If both end points of the line have region code '0000' → completely inside.
- If logical ANDing of endpoints of a line is not '0000' → completely outside.

Step3: If both condition of step 2 fails i.e. logical ANDing gives '0000', we need to find the intersection with window boundary.

Here, $m = \frac{Y_1 - Y_1}{X_2 - X_1}$

a. If first bit is 1, line intersects with left boundary of the window. So,

$Y_I = Y_1 + m(X - X_1)$ where $X = Xw_{min}$

b. If second bit is 1, line intersects with right boundary of the window. So,

$Y_I = Y_1 + m(X - X_1)$ where $X = Xw_{max}$

c. If third bit is 1, line intersects with lower boundary of the window. So,

$$X_I = X_1 + \frac{1}{m}(Y - Y_1) \text{ where } Y = Yw_{min}$$

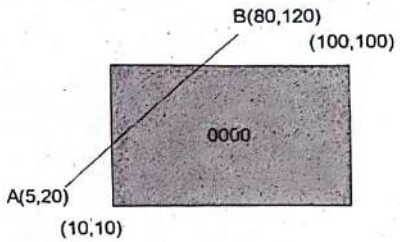
d. If fourth bit is 1, line intersects with upper boundary of the window. So,

$$X_I = X_1 + \frac{1}{m}(Y - Y_1) \text{ where } Y = Yw_{max}$$

Step 4: Repeat step 1 and 3 until line is completely inside the window.

Solved Numerical Examples

1. Use Cohen Sutherland line clipping algorithm to clip a line with end point co-ordinates A(5,20), B(80,120) against a clip window with its lower left corner at (10,10) and upper right corner at (100,100).



Solution:

$A(5,20) = A(X_1, Y_1)$

$B(80,120) = B(X_2, Y_2)$

Here,

$(X_{wmin}, Y_{wmin}) = (10,10)$

$(X_{wmax}, Y_{wmax}) = (100,100)$

Also, slope $(m) = \frac{Y_2 - Y_2}{X_2 - X_1}$

$$= \frac{120 - 20}{80 - 5} = \frac{4}{3}$$

i. Find the region code for each end point of the line

ii. Find if it is above, below, left or right of the clip window

iii. Determine the intersection point of the line segment with the boundary

Step 1

- Obtaining 4 bit binary region code for point A and B

For A

- $5 < 10 = T = 1$
- $5 > 100 = F = 0$
- $20 < 10 = F = 0$
- $20 > 100 = F = 0$

For B

- $80 < 10 = F = 0$
- $80 > 100 = F = 0$
- $120 < 10 = F = 0$
- $120 > 100 = T = 1$

Step 2

- Logical ANDing of A and B

  A = 0 0 0 1 [Note: TBRL]

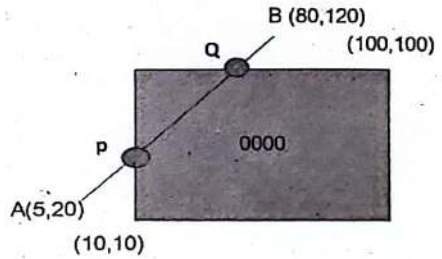
  B = 1 0 0 0

  Logical ANDing = 0 0 0 0

  This shows that the line is not completely outside (Partially visible), it might have intersected the window to some point.

Step 3

- We have to find the intersection point of line AB.



For P

- Since P cuts the window at left side so its x_{min} remains constant and y changes.
- $x_{min} = 10$

- y to find
- So A(5,20) & P(10,y)

$m = \frac{y-20}{10-5}$ since $m = \frac{4}{3}$ on solving we get $y = 26.67$

Therefore P(10,26.67)

For Q

- Since Q cuts the window at left side so its y_{max} remains constant and x changes.
- $y_{max} = 10$
- x to find
- So A(5,20) and Q(x,100)

$m = \frac{100-20}{x-5}$ since $m = \frac{4}{3}$ on solving we get $x = 65$

Therefore Q(65,100).

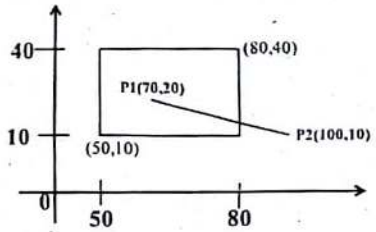
Hence, the line PQ is visible and the line AP and QB were clipped away or cutoff.

2. **Use the cohen-sutherland algorithm to clip line $P_1(70,20)$ and $P_2(100,10)$ against window lower left corner (50,10) and upper right hand corner (80,40).**

Solution:

$P_1(70,20) = P_1(X_1,Y_1)$

$P_2(100,10) = P_2(X_2,Y_2)$



Here,

$(X_{wmin}, Y_{wmin}) = (50,10)$

$(X_{wmax}, Y_{wmax}) = (80,40)$

Also, slope $(m) = \frac{Y_2 - Y_2}{X_2 - X_1} = \frac{10-20}{100-70} = \frac{-1}{3}$

Step 1: Region code for end point $P_1(70,20)$.

$P_1 = 0\ 0\ 0\ 0$

This shows P_1 lies completely inside the window.

Here, $L = b_1 = 0$

$R = b_2 = 0$

$B = b_3 = 0$

$T = b_4 = 0$

Region code for end point $P_2(100,10)$.

$P_2 = 0\ 0\ 1\ 0$

This shows P_1 lies completely inside the window.

Here, $L = b_1 = 0$

$R = b_2 = 1$

$B = b_3 = 0$

$T = b_4 = 0$

Step 2: Logical anding between the region codes of P_1 and P_2.

$P_1 = 0\ 0\ 1\ 0$

$P_2 = 0\ 0\ 1\ 0$

Logical ANDing = 0 0 0 0

This shows that the line is not completely outside, it might have intersected the window to some point.

Step 3: Here second bit i.e. b_2 of the point P_2 is set to 1.

So, $Y_1 = Y_1 + m(X - X_1)$

where $X = X_{wmax} = 80$

and $X_1 = 100$ and $Y_1 = 10$

or, $Y_1 = 10 + \frac{(-1)}{3} \times (80 - 100)$

$\therefore\ Y_1 = 16.67$

New point becomes as $P_1' = (70,20)$

and $P_2' = (80,16.67)$

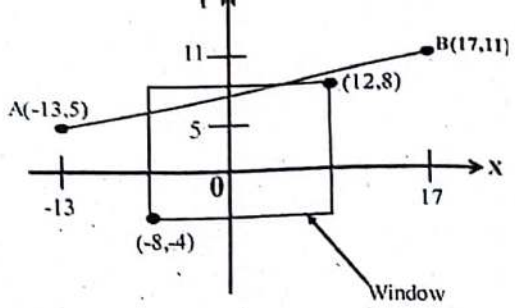
Hence, the line P_1P_2' is visible and the line P2!P2 is clipped away or cutoff.

3. **Use cohen–sutherland algorithm to clip a line starting from A(–13,5) and ending at B(17,11) against the window having its boundary lower corner at(–8,–4) and upper right corner at (12,8).**

Solution:

$A(-13,5) = A(X_1,Y_1)$

$B(17,11) = B(X_2,Y_2)$



Here,

$(X_{wmin}, Y_{wmin}) = (-8,-4)$

$(X_{wmax}, Y_{wmax}) = (12,8)$

Also,

$$\text{Slope (m)} = \frac{Y_2 - X_2}{X_2 - X_1} = \frac{11 - 5}{17 + 13} = \frac{1}{5} = 0.2$$

Step 1: Region code for end point A(–13,5).

$P_1 = 0\ 0\ 0\ 1$

Here, $L = b_1 = 1$

$R = b_2 = 0$

$B = b_3 = 0$

$T = b_4 = 0$

Region code for end point B(17,11).

$P_2 = 1\ 0\ 1\ 0$

This shows P_1 lies completely inside the window.

Here, $L = b_1 = 0$

$R = b_2 = 1$

$B = b_3 = 0$

$T = b_4 = 1$

Step 2: Logical Anding between the region codes of A and B.

$A = 0\ 0\ 0\ 1$

$B = 1\ 0\ 1\ 0$

Logical ANDing = 0 0 0 0

This shows that the line is not completely outside, it might have intersected the window to some point.

Step 3: For point A, b_1 is set to 1, so, line intersects the left boundary of the window.

i.e. $Y_I = Y_1 + m(X - X_1)$ where $X = X_{wmin} = -8$

and $X_1 = -13$ and $Y_1 = 5$

or, $Y_I = 5 + 0.2 \times (-8 + 13)$

$\therefore\ Y_I = 6$

Hence, intersecting point A' = (–8,6).

For point B, b_2 and b_4 are set to 1, so, line intersects at the right top boundary of the window.

For $b_2 = 1$;

i.e. $Y_I = Y_1 + m(X - X_1)$ where $X = X_{wmax} = 12$

and $X_1 = 17$ and $Y_1 = 11$

or, $Y_I = 11 + 0.2 \times (12 - 17)$

$\therefore\ Y_I = 10$

For $b_4 = 1$;

i.e. $X_I = X_1 + \frac{1}{m} \times (Y - Y_1)$ where $Y = Y_{wmax} = 8$

and $X_1 = 17$ and $Y_1 = 11$

or, $X_I = 17 + \frac{1}{0.2} \times (8 - 11)$

$\therefore\ X_I = 2$

Hence, another intersecting point B'(2,8).

Hence, the visible line is A'B'.

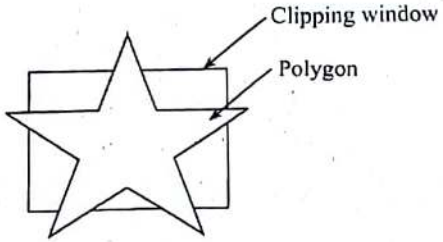
Disadvantage of Line clipping Algorithm

- **Overhead for non-clipped lines:** The algorithm requires extra calculations for all lines, even those that don't need clipping, which can slow down performance.
- **Limited to 2D clipping:** These algorithms are primarily designed for two-dimensional clipping and may require modifications for higher-dimensional spaces.
- **Complexity for complex polygons:** Dealing with complex polygons adds complexity and may necessitate additional steps or adaptations.
- **Floating-point precision issues:** Floating-point arithmetic can introduce precision errors, affecting the accuracy of clipped lines.
- **Inefficiency for overlapping regions:** Overlapping lines within the clipping region can result in redundant calculations and slower performance.

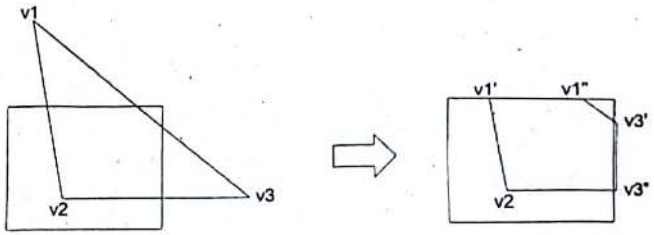
- **Lack of support for curved lines:** Line clipping algorithms assume straight line segments, making them less accurate for curved or spline-based lines.

3. Polygon clipping

- A polygon can also be clipped by specifying the clipping window, Sutherland Hodgeman polygon clipping algorithm is used for polygon clipping. In this algorithm, all the vertices of the polygon are clipped against each edge of the clipping window.
- A polygon may be represented as a number of line segments connected end to end to form a closed figure.



- **Polygon is of TWO types:**
  a. Convex polygon
  b. Concave polygon



- **Note:** Need to consider each of 4 edge boundaries.

a. Convex polygon

- For any two points on the line segment connecting them are also inside the polygon.
- **Example:**

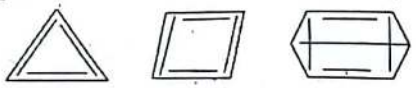


Fig.: Convex polygons

b. Concave polygon

- The one which is not convex.

- **Example:**

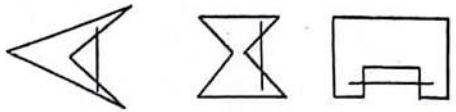
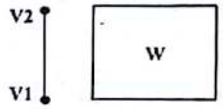


Fig.: Concave polygons

The Sutherland Hodgeman polygon clipping algorithm

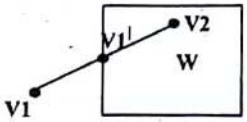
- The algorithm uses divide and conquer strategy.
- It solves simple and identical problems that when combined solve the overall problems.
- The algorithm clips against 4 edges in succession whereas the line clippers test the region codes to see which edge is crossed and clips when necessary.
- The algorithm has following FOUR different cases when we are clipping a polygon with respect to any particular edge of the window.

Case 1: If both the vertices of polygon lies outside the window boundary then no vertex is stored as output.



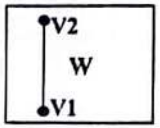
- Here, V_1 and V_2 both are outside the window W.
- We save nothing on the edge V_1V_2.

Case 2: If first vertex is outside and the second one is inside, the window boundary then in that case, we save the intersection point between the vertices on the window and the second vertex on the new edge.



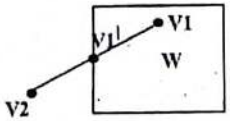
- Here, V_1 is outside V_2 is inside the window W.
- We save V_1' and V_2 on the edge V_1V_2.

Case 3: If both the vertices of polygon lies inside the window boundary then we save only second vertex on the edge.



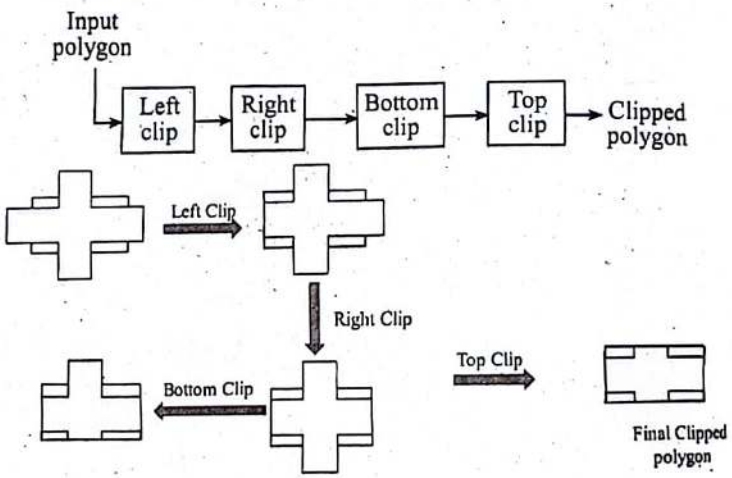
- Here, V_1 and V_2 both are inside the window W.
- We save only V_2 on the edge V_1V_2.

Case 4: When the first vertex is inside and the second one is outside the window boundary then in that case, we save the intersection point between the vertices on the boundary on the edge V_1V_2.

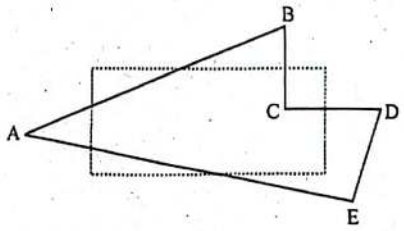


- Here, V_1 is inside and V_2 is outside the window W.
- We save only V_1' on the edge V_1V_2.

- Once all the vertices have considered for one clip window boundary, the output list of the vertices is clipped against the next window boundary.



Example:



Let us take an example to understand the polygon clipping by Sutherland and Hodgman algorithm. Suppose we are having polygon ABCDE. Which we want to clip against a rectangular Window.

Here vertex list will be A, B, C , D, E

Edges will be AB, CD, DE, and EA

Step 1: Clip left

- We will consider all edges of polygon with respect to left boundary of window.

Left clip

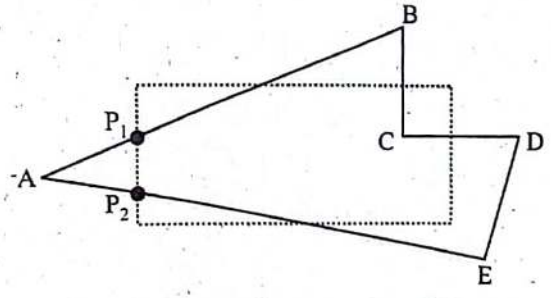
AB – P_1b

BC – C

CD – D

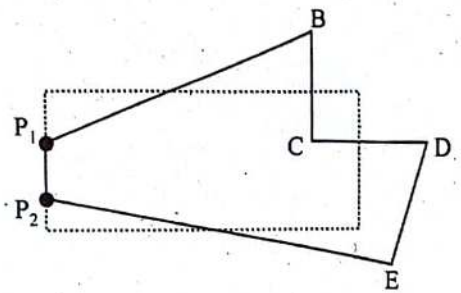
DE – E

EA – P_2



- So after left clipping output vertex list will become

{P_1, B, C, D, E, P_2}



Step 2: Clip right

- We will consider all edges of polygon with respect to right boundary of window.

Right clip

P_1B – B

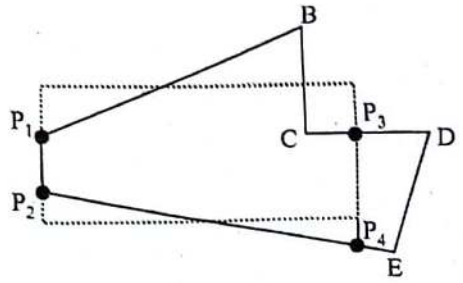
BC – C

CD – P_3

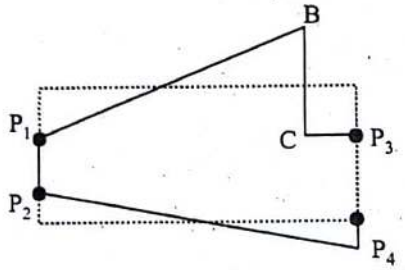
DE – No vertex

$EP_2 - P_4P_2$

$P_2P_1 - P_1$



- So after right clipping output vertex list will become {B, C, P_3, P_4, P_2, P_1}



Step 3: Clip bottom

- We will consider all edges of polygon with respect to bottom boundary of window.

Right clip

$P_2P_1 - P_1$

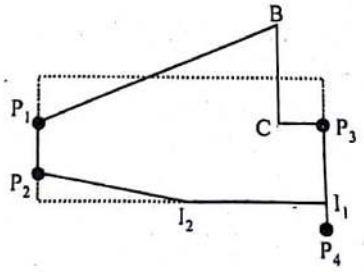
$P_1B - B$

$BC - C$

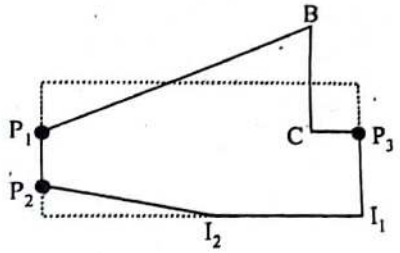
$CP_3 - P_3$

$P_3P_4 - I_1$

$P_4P_2 - I_2P_2$



- After bottom clipping set of vertices will be {P_1, B, C, P_3, I_1, I_2, P_2}



Step 4: Clip top

- We will consider all edges of polygon with respect to bottom boundary of window.

Right clip

$P_1B - I_3$

$BC - I_4C$

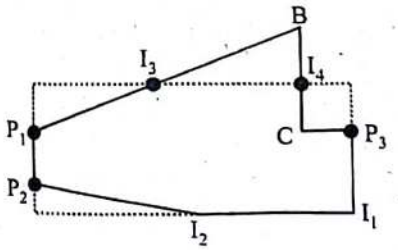
$CP_3 - P_3$

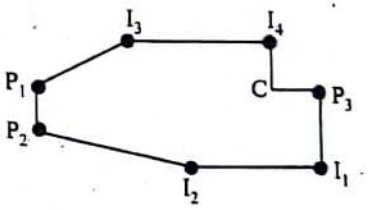
$P_3I_1 - I_1$

$I_1I_2 - I_2$

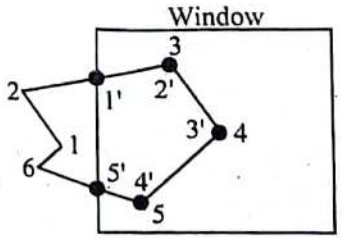
$I_2P_2 - P_2$

$P_2P_1 - P_1$

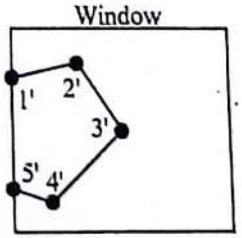




Example:



Before clipping



After clipping

I. Process area in figure against left window boundary.

II. a. Vertices 1, 2 are outside of boundary

b. Moving along to vertex 3 which is inside calculate the intersection. Save both intersection point and vertex 3

c. Vertices 4 and 5 are determined to be inside save them both

d. Vertex 6 is outside so find intersection point so save the intersection point.

Required setting up storage for an output list vertices as a polygon is clipped against each window boundary

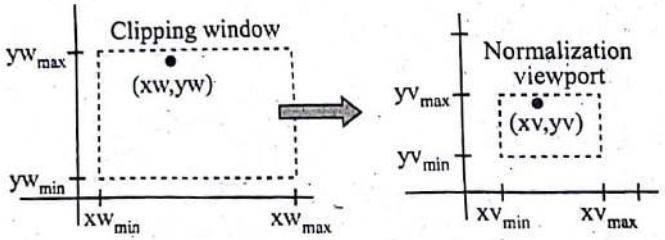
The final set of vertices of the clipped polygon are 1' 2' 3' 4' 5'.

Solved Numerical Examples

1. Window port is given by (100, 100, 300, 300) and viewport is given by (50, 50, 150, 150). Convert the window port co-ordinate (200, 200) to the viewport co-ordinate.

Solution:

Using second technique



Here $(xw_{min}, yw_{min}) = (100, 100)$ and $(xw_{max}, xw_{max}) = (300,300)$

$(xv_{min}, yv_{min}) = (50,50)$ and $(xv_{max}, xv_{max}) = (150,150)$

$(xw, yw) = (200,200)$

Then, we have

$$S_x = \frac{xv_{max} - xv_{min}}{xw_{max} - sw_{min}}$$

$$S_y = \frac{yv_{max} - yv_{min}}{yw_{max} - yw_{min}}$$

$$\text{i.e. } S_x = \frac{(150 - 50)}{(300 - 100)} = 0.5$$

$$S_y = \frac{(150 - 50)}{(300 - 100)} = 0.5$$

The equations for mapping window co-ordinate to viewport co-ordinate is given by

$xv = xv_{min} + (xw - xw_{min})S_x$

$yv = yv_{min} + (yw - yw_{min})S_y$

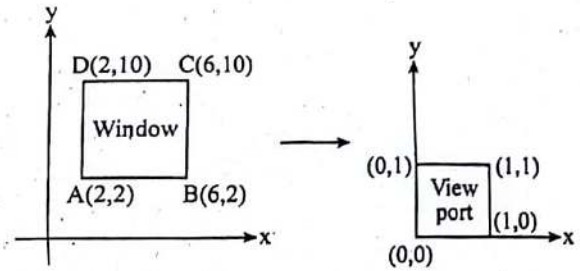
therefore, $xv = 50 + (200 - 100) \times 0.5 = 100$

and $yv = 50 + (200 - 100) \times 0.5 = 100$

hence, the transformed viewport co-ordinate is (100,100).

2. Find the normalization transformation for window to viewport which uses the rectangle whose lower left corner is at (2,2) and upper right corner is at (6,10) as a window and the viewport that has lower left corner at (0,0) and upper right corner at (1,1).

Solution:



The composite transformation matrix for transforming the window co-ordinate to viewport co-ordinate is given as

$T = S_{(s_x, s_y)} T_{(-2,-2)}$

Now we know,

$$S_x = \frac{x_{vmax} - x_{vmin}}{x_{wmax} - x_{wmin}}$$

$$S_y = \frac{y_{vmax} - y_{vmin}}{y_{wmax} - y_{wmin}}$$

$$\text{i.e. } S_x = \frac{(1 - 0)}{(6 - 2)} = 0.25$$

$$S_y = \frac{(1 - 0)}{(10 - 2)} = 0.125$$

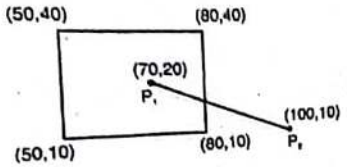
Then from above transformation matrix,

$$T = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0.25 & 0 & 0 \\ 0 & 0.125 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.25 & 0 & -0.5 \\ 0 & 0.125 & -0.25 \\ 0 & 0 & 1 \end{bmatrix}$$

3. **Use the Cohen-Sutherland algorithm to clip the line $P_1(70,20)$ and $P_2(100,10)$ against a window lower left hand corner (50,10) and upper right hand corner (80,40).**

Solution:



Assigning 4 bit binary outcome code to the two end point

$P_1 = 0000$ and $P_2 = 0010$

$P_1 \vee P_2 = 0000 \vee 0010 = 0010$, since $P_1 \vee P_2! = 0000$, hence the two point doesn't lie completely inside the window.

$P_1 \wedge P_2 = 0000 \wedge 0010 = 0000$, since $P_1 \wedge P_2 = 0000$, hence line is partially visible.

Now, for, finding the intersection of P_1 and P_2 with the boundary of Window,

Starting from point p_2 (because. p_2 lie outside the window), we get

Slope $M = \frac{(10-20)}{(100-70)} = \frac{-1}{3}$

We have to find the intersection with right edge of window, here x=80, y=?

We have

(p_2 selected)

$y = y_2 + m(x - x_2)$

$= 10 + (-1/3)(80 - 100)$

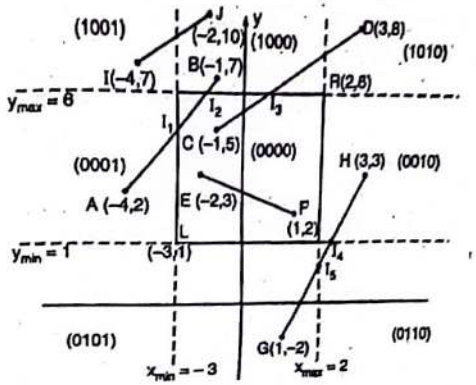
$= 10 + 6.67$

$= 16.67$

Thus the intersection point $p_3 = (80,16.66)$

So discarding the line segment that lie outside the boundary i.e. p_3p_2 we get new line P_1P_3 with co-ordinate $P_1(70,20)$ and $P_3(80,16.67)$

4. **Let R be the rectangle window whose lower right hand corner is (2,2) and upper left hand corner is (–2,5).**
   a. **Find out the region code for end points in the following figure.**
   b. **Find the clipping categories for the line segments.**
   c. **Use Cohen Sutherland algorithm to clip the line segment.**



Solution:

i. **The region code is set according to following rule**

Any endpoint (x,y) of a line segment, the code can be determined ad follows:

- If $x < x_{min}$, first bit is 1, (Point lies to left of window (Left)) (0^{th} bit) otherwise 0.
- If $x > x_{max}$, second bit is 1, (Point lies to right of window (Right)) (1^{st} bit), otherwise 0.
- If $y < y_{min}$, third bit is 1, (Point lies to below window (Bottom)) 2^{nd} bit), otherwise 0.
- If $y > y_{max}$, fourth bit is 1, (Point lies to above window (Top)) (3^{rd} bit), otherwise 0.

Region code:

A = (0001)	B = (1000)	C = (0000)
D = (1010)	E = (0000)	F = (0000)
G = (0100)	H = (0010)	I = (1001)
J = (1000)		

ii. **To get appropriate categories, we test region code**

Category 1: Inside window i.e. visible

Line segment EF because the region code for both end point is 0000 i.e. $E \vee F = 0000$ holds so is selected for display

Category 2: Outside window i.e. Not visible

Line segment IJ because the region code of I and J have 1 at same bit position i.e. $I \wedge J! = 0000$, so is selected for clipping i.e. discarded

Category 3: Partial visible

Line segment AB, CD and GH since $A \wedge B = 0000$, similarly $C \wedge D = 0000$ and similarly $G \wedge H = 0000$. So become candidate for clipping.

iii. So line AB, CD and GH need to be clipped

For line AB

We can start from A or B, let start from A.

We have to find the intersection with left edge of window, here $x=-3, y = ?$

We have (A selected)

Slope $m = \frac{(7-2)}{(-1+4)} = \left(\frac{5}{3}\right)$

$y = y_1 + m(x - x_1)$

$= 2 + \left(\frac{5}{3}\right)(-3+4)$

$= 3.667$

Thus the intersection point $I_1 = (-3,3.66)$ and region code is 0000

Hence we clip AI_1.

Now we work on I_1B

Since $I_1 \wedge B = 0000 \times 1000 = 0000$ hence is again partially visible.

Now we have to find the intersection with top edge of window, here $y = 6, x = ?$

And

$x = x_1 + \frac{1}{m}(y - y_1)$

$= -1 = \frac{3}{5}(6-7)$

$= 1.6$

Thus the intersection point $I_2 = (-1.6,6)$ with region code 0000

Hence we clip I_2B

Now we work on I_1I_2

Since the region code of I_1 and I_2 both is 0000 i.e. $I_1 \vee I_2 = 0000$, hence both point lie inside the window.

For line GH

We can start with G or H, for now, let start from G. (see case 3: opcode of G = 0100 i.e. bottom edge)

We have to find the intersection with the bottom edge of window, here $y = 1, x = ?$

Slope $M = \frac{(3+2)}{(3-1)} = \frac{5}{2}$

And

$x = x_1 + \frac{1}{m}(y - y_1)$

$= 3 + \frac{2}{5}(1-3)$

$= 2,2$

Thus the intersection point $I_4 = (2.2,1)$ and region code is 0010

Hence GI_4 is clipped

Now, we work on I_4H

Since $I_4 \wedge H = 0010 \wedge 0010 = 0010$ i.e. ! = 0000 hence both point lie outside the window hence clipped i.e. not displayed.

For line CD

Starting from point D (because D lie outside the window)

We have to find the intersection with the top edge of window, here $y = 6, x = ?$

Slope $M = \frac{(8-5)}{(3+1)} = \frac{3}{4}$

And

$x = x_1 + \frac{1}{m}(y - y_1)$

$= 3 + \frac{4}{3}(6-8)$

$= 0.33$

Thus the intersection point $I_3 = (0.33,6)$ and region code is 0000

Hence I_{3D} is clipped.

Now we work on CI_3

Since both the region code of C and I_3 is 0000 i.e. $C \vee I_3 = 0000$, hence the segment CI_3 is displayed.