Chap-1 Computer graphics 15/2/24

Computer graphics - i) Data structure
ii) Algorithms
iii) language
} Combination (software part)
iv) monitor } (Hardware part)

Computer graphics = DS + graphics algo + languages
↓
(picture generation & transformation algo)
basic

- Computer graphics - 2nd

C.G. involves pictures or graphics objects which are the collection of discreate picture elements called as Pixcells (short form of picture elems). A pixcel is the smallest addressable screen element. We can control the pixcells by setting the intensity & colours of the pixcel that composes the screen.

Pixcells isn't like just a dot, rather pixcel is a region that can contain an infinite no. of points. ~~pixcel~~ 2nd

iii) Raestrization - The process of determining the appropriate pixcels for representing a picture or graphics object is called as rastrization.

iii) Framebuffer - It is a large, contiguous piece of memory into which the intensity values for all pixels are placed.

iv) Bit map & Pixel map - If a pixel has only 2 colour values (black & white), it can be encoded by 1 bit of information. If more than 1 bits are used to represent a pixel, a longer range of colours or shades of grey can be represented either by 2-bits / 4-bits. An image of 2 colours is called as a bit map. An image of more than 2 colours is called as a pixel map.

v) Resolution - It is defined as the ratio of width & height of output of the display device. There are 2 types of resolution - i) image ii) screen resolution

i) Image r - It is defined as the distance from 1 pixel to the next pixel.

ii) Screen r. - It is defined as the no. of pixels in the horizontal & vertical directions on the screen.

LW

Basic graphics system 4/3/24

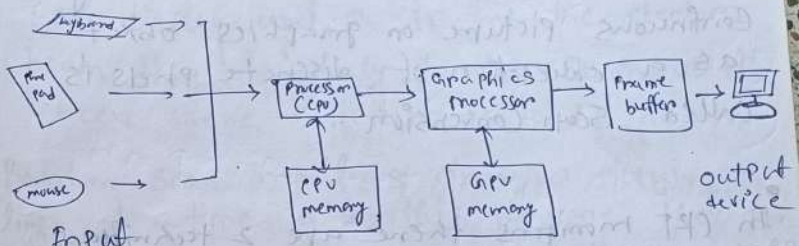
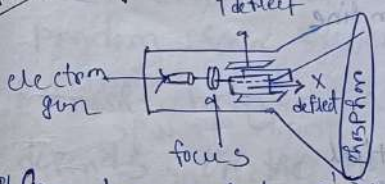


Image formed in frame buffer

CRT (Cathode Ray Tube)

Working Principle of Cathode Ray Tube



It can be used either as a line-drawing device (calligraphic) or to display contents of frame buffer (raster mode)

Working p. - i) A beam of electrons & cathode rays emmitted by an electron gun, passes through focusing and deflection system that directs the beam towards specified positions on the phosphor coated screen.

ii) The phosphor then emmits a small spot of lights at each position contacted by the

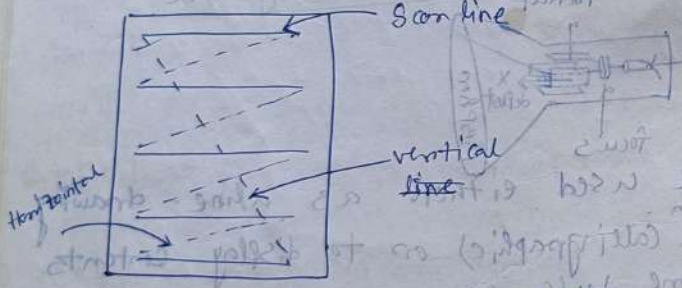
electron beam.

Scan conversion - the process of representing continuous picture or graphics object as a collection of discrete pixels is called Scan Conversion.

- In CRT monitors there are 2 techniques of displaying images.

i) Raster scan displays - ii) Random scan displays.

i) Raster scan display (CRT is based on the television technology)



In raster scan system, the electron beam is swept across the screen, 1 row, at a time from top to bottom. As the electron beam moves across each row, the beam intensity is turned on & off to create a pattern of illumina-ted spots. Pic. element stored as commands that is called frame buffer.

Random scan / vector-scan display / calligraphic displays - When operated as a random scan display unit, a CRT has the electron beam directed only to the parts of the screen where a picture is to be drawn.

Random scan monitors draw a picture, 1 line at a time & for this reasons they are also referred to as vector displays / calligraphic displays. The component lines of a

P.S.1

picture can be drawn & refreshed by a random scan system in any specified order. Refresh data on a random scan system depends on the no. of lines to be displayed. Picture definition is now stored as a set of line drawing commands in an area of memory referred to as the refresh display file / refresh buffer.

Input devices for operator interaction

(for short ing)

- Keyboards
- mouse
- Trackball & Spaceball
- Joysticks
- Data glove
- Digitizers
- Touch screens

} various i/p devices

Light Pen
voice systems
Image scanners

Track ball & space ball - A track ball is a ball that can be rotated with the fingers / Plam of the hand to produce screen cursor movements. Potentiometers, attached to the ball measure the amount & direction of rotation. Track balls are often mounted on keyboards / others devices such as the Z mouse.

While is traceball is a 2D positioning device a space ball provides 6 degrees of freedom. Unlike the track ball, spaceball doesn't actually moves. Strain gauge measures the amount of pressure applied to the spaceball to provide i/p. Spaceballs are used for 3D positioning & selection operations in virtual reality systems, modeling animation, Cad & other applications.

Joystick - A joystick consists of a small vertical lever mounted on a base that is used to steer the screen cursor around. most joysticks select screen positions with actual stick movement, others respond to pressure on the stick. Potentiometers mounted at the base of the joystick measure the amount of movement & springs returns the stick to the center position when it is released.

Data glove - It is used to grasp a virtual object. The glove is constructed with a series of sensors that detect hand & finger motions. Electromagnetic coupling blw transmitting antennas & receiving antennas is used to provide information about the position & orientation of the hand. A 2D projection of the scence can be viewed on a video monitor / a 3D projection can be viewd with a headset.

Digitizers (shortnote)

It is also known as a graphics tablets, is a computer input device that allows users to draw images & graphics by hand. It can also be used to capture data / hand written signatures.

It consists of a flat surface & a stylus / pen like instrument that you use to draw / write on the surface. It can convert analog info, like touch, light, & sound, into a digital signal. they can be used for many different purposes, including writing, drawing, painting, capturing data etc.

Digitizers inside smartphones & tablet computers can also detect how fast the device is moving & the angle it's held.

Light pen (shortnote) - It is a light-sensitive computer input device that uses a cathode ray tube (CRT) display to put info on a screen.
It works by detecting the light from the screen with photocells. The computer then uses this info to identify the pen's location on the screen. Light pens can be used to:
① Draw text & diagrams.
② Select objects
③ Edit text
④ Interact with UI elements.
They are considered as the predecessor to touchscreen technology.
Adv - i) Drawing directly on to the screen
Disadv - i) Hard to use, ii) Not accurate.

6/9/24

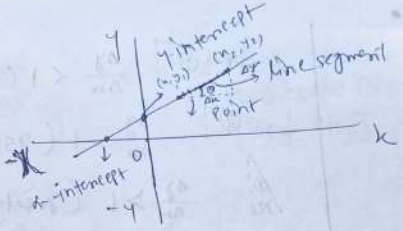
Chap-2
Line drawing algos -

Line - It is infinite in dimension.
Line segment - It has 2 confined end points. This line segment has a collection of points.

Point - It is a dimensionless since it represents a dot only.

Line equation

$y = mx + b$ — ①



We can determine the value for slope m & b intercept as,

$m = \frac{y_2 - y_1}{x_2 - x_1}$ — ⑪

i.e, $m = \frac{\Delta y}{\Delta x} = \frac{\text{(perpendicular)}}{\text{(base)}}$

$= \tan\theta$

$\therefore \boxed{m = \frac{\Delta y}{\Delta x} = \tan\theta}$

Eg - ① The endpoints of line are (0,0) & (6,18). Compute each value of y as x steps from 0 to 6 & plot the result.

$m = \frac{18-0}{6-0}$

$\boxed{m = 3}$

$\therefore y = 3x + b$

Now $(x_1, y_1) = (0,0)$

$\therefore 0 = 3 \cdot 0 + b$

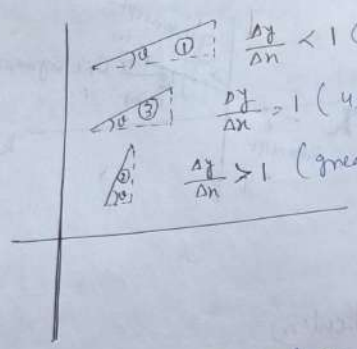
$\Rightarrow \boxed{b = 0}$

$\therefore y = 3x$ → ans (as it is for we need are told to compute each val. of y as x steps as it's a line & the val of a will be a const (i.e. $y_1 = y_2$))

Here, val of b = 0, so y = 3x
if b has some val, then y and & the y will change as per y = 3x + b, but here as per y = 3x

Starts from (0,0) & ends at (6,18)

if we have $(x_2, y_2) = (6,18)$
then, $y_2 = 3x + b$
$\Rightarrow 18 = 18 + b$
$\Rightarrow b = 0$



If angle > 45°, slope > 1 [$\because m = \frac{\Delta y}{\Delta x}$, $\tan\theta$ m...]

If angle < 45°, slope < 1

DDA (Digital differential analyser) algo

It is an incremental scan conversion method (line drawing method) based on calculating either Δx or Δy.

$$\Delta x = \frac{\Delta y}{m} \quad \text{— (III)} \qquad [\because m = \frac{\Delta y}{\Delta x}]$$

$$\Delta y = m \Delta x \quad \text{— (IV)}$$

We sample the line at unit intervals in 1 coordinate & determine corresponding integer values nearest the line path for the other coordinates.

[i.e, if we find 3.2 then we have to take 3, 2.6 have to take the value 3 etc]

Algo — (x_1, y_1) & (x_2, y_2) are the endpoints & dx, [i.e Δx] dy are the float variables.

where dx = abs $(x_2 - x_1)$ & dy = abs $(y_2 - y_1)$ (absolute value)

i) If dx > = dy then,
length = dx
else
length = dy
endif

[how long our loop will execute is length / step]

ii) $x_{increment}$ = dx / length
$y_{increment}$ = dy / length

iii) setPixel(Round(x), Round(y))

iii) for(k=0; k<length; k++){
x += $x_{increment}$
y += $y_{increment}$
setPixel(Round(x), Round(y));
}

end of algo

Consider a line with positive slope. If the slope is <= 1 (first), we sample at unit x intervals ($\Delta x = 1$) & compute each successive y value as

$$y_{k+1} = y_k + m \cdot 1 \quad \text{— (V)}$$

(y_{k+1} is successive y value, y_k is previous value, 1 is value of x) [k = length]

ii) For lines with the +ve slope > 1, we reverse the roles of x & y. that is we sample at unit y intervals ($\Delta y = 1$) & calculate each successive (succeeding) x value as,

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

this 2 eqns (vi & vii) are based on the assumption that lines are to be processed from left end to the right end.

If the processing is reversed, then (for line with -ve slope)

$\Delta x = -1$ & $\Delta y = -1$

$\therefore \boxed{y_{k+1} = y_k - m} \quad -(VII)$

$\therefore \boxed{x_{k+1} = x_k - \frac{1}{m}} \quad -(VIII)$

(1) Scan convert a line having end points $(3,2)$ & $(4,7)$ using DDA.

$x_1, y_1 \quad x_2, y_2$

$\therefore \Delta x = 4 - 3 = 1$

$\Delta y = 7 - 2 = 5$

$\therefore m = \frac{\Delta y}{\Delta x} = 5 > 1$ → length = Δy = 5 [as dx<dy] (dy)

applicable for both ($m>1$, $m<1$)

$\therefore y_{k+1} = y_k + \frac{1}{m}$

y increment = $\frac{1}{5} = 0.2$ [same as $x_{k+1} = x_k + \frac{1}{m}$]

y increment = $\frac{5}{5}$ [$y_{k+1} = y_k + 1$]

$= 1$

this type is mainly when $m<1$, i.e dx>dy as $x_{k+1} - x_k$ since $\approx y_{k+1} - y_k$

initialize the starting point at $(x_1, y_1) = (3,2)$

[we have to calculate the next point by this way

next point $(x_{k+1}, y_{k+1}) = (x_k + \frac{1}{m}, y_k + 1)$ as $m>1$]

x_1	y_1	y_{k_2}	y_2	L	Δx	Δy	i	x	y	Result	Plot
3	2	4	7	5	1	5	0	3.2	3	(3,3)	→
							1	3.4	4	(3,4)	→
							2	3.6	5	(4,5)	→
							3	3.8	6	(4,6)	→
							4	4	7	(4,7)	→
							5				

↑ same ↓

$\frac{1}{m} = \frac{1}{5} = 0.2$

Other way

(1) $\Delta x = 4 - 3 = 1$

$\Delta y = 7 - 2 = 5$

$m = \frac{5}{1}$

(2) Calculate no. of points (length)

$\Delta x < \Delta y$ ~~=~~ $1 < 5$

$\therefore$ length = $k = \Delta y = 5$

This case type is mainly for $m>1$, dx<dy

Applicable for both ($m>1$, $m<1$)

(3) As $m>1$ ∴ case (II) is satisfied

$x_{k+1} = x_k + \frac{1}{m}$

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

[x_{k+1} = next val. x_k = prev val]

next points

x_1	y_1	x_{k+1}	y_{k+1}	Round off (x_{k+1}, y_{k+1})	
3	2	3+0.2 = 3.2	2+1 = 3	(3,3)	k=1 (start) - step/length
		3.2+0.2 = 3.4	3+1 = 4	(3,4)	k=2
		3.6	5	(4,5)	k=3
		3.8	6	(4,6)	k=4
		4	7	(4,7) → end point	k=5

11/3/25

② (i) $(x_1, y_1) = (2, 5)$ slope = ?

(ii) $(x_2, y_2) = (2, 12)$

Soln

① $\Delta x = 2 - 2 = 0$

$\Delta y = 12 - 5 = 7$

$m = \frac{\Delta y}{\Delta x} = \frac{7}{0} = \infty > 1$

② Calculate no. of points (length)

$\Delta x < \Delta y = 0 < \infty 7$

∴ length = k = dy (or Δy) = 7

③ As $m > 1$, ∴ case (ii) satisfied

$x_{k+1} = x_k + \frac{1}{m}$

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

otherwise

$x_{inc} = \frac{0}{7} = 0$

$y_{inc} = \frac{7}{7} = 1$

x_1	y_1	x_{k+1}	y_{k+1}	Result
2	5	2	6	
		2	7	
		2	8	
		2	9	
		2	10	
		2	11	
		2	12	

which one is greater (b/w Δx, Δy), that one will be incremented by 1 (i.e. x_{k+1}/y_{k+1} or x_k/y_k)

K = bigger one

③ Soln

(iii) $(x_1, y_1) = (5, 4)$

$(x_2, y_2) = (12, 7)$

① $dx = 12 - 5 = 7$

$dy = 7 - 4 = 3$

$m = \frac{3}{7} = 0.4 < 1$

② $dx > dy = 7 > \frac{3}{7}$

∴ length = $k = dx = 7$

③ As $m < 1$, ∴ case (i) satisfied

x_{k+1} = x increment = $\frac{7}{7} = 1$

y_{k+1} = y increment = $\frac{3}{7} = 0.43 \simeq 0.4$

otherwise

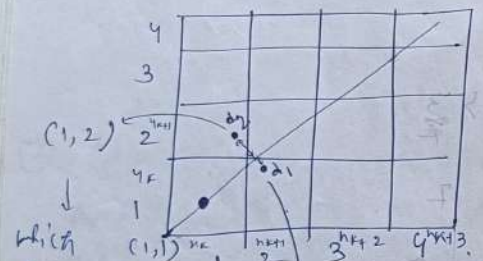
$x_{k+1} = x_k + 1$ (unit interval increment)

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

x_1	y_1	x_{k+1}	y_{k+1}	Result
5	4	5	4.4	
		5.8	6.8	
		6	7.2	
		8.4	8.6	
		9	9	
		10	9.4	
		11	9.8	

Bresenham's line drawing algo

An accurate & efficient raster line generating algo is developed by Bresenham which converts line s using only incremental integer calculations.



which Point n_{K+1} is more closer, by that we have to take decision

$y = mx + c$ at $x_{next} = x_{K+1}$

then $y_K = m(x_{K+1}) + c$

$d_1 = \text{actual } y - y_K$

$= y - y_K$

$= m(x - x_{K+1})$

$= (m(x_{K+1}) + c) - y_K$

$d_1 = m(x_{K+1}) + c - y_K$

$d_2 = y_{K+1} - y$

$d_2 = y_{K+1} - m(x_{K+1}) - c$

(Same as next follow next)

Prop to calculate P_0, P_{K+1} for both cases

i) We 1st consider the scan conversion process for one line with +ive slope ≤ 1.

ii) Starting from the left end point (x_0, y_0) of a given line, we step to each successive column (x-position) & plot the pixel whose scan line y-value is closest to the line path.

iii) Assuming we have determined that the pixel at (x_K, y_K) is to be displayed, we next need to decide which pixel is to plot in column $x_K + 1$. Here the choices are positions (x_{K+1}, y_K) & (x_{K+1}, y_{K+1}).

iv) the y-coordinate on the line at pixel position x_{K+1} is calculated as

$$y_{\text{at } x_{K+1}} = m(x_{K+1}) + c \quad \text{--- (1)}$$

v) At sampling position x_{K+1}, we label vertical pixel separations from the original line path as d_1 & d_2, then

$d_1 = y - y_K = m(x_{K+1}) + c - y_K$

$(\because y = y_{K+1})$

$\therefore d_1 = m(x_{K+1}) + c - y_K$ --- (1) (from (1))

$d_2 = y_{K+1} - y$ $(\because y_{K+1} = y)$

$d_2 = y_{K+1} - m(x_{K+1}) - c$ --- (11) (from (1))

∴ the difference b/w 2 seperation is,

$d_1 - d_2 = m(x_{k+1}) + c - y_k - y_{k+1} + m(x_{k+1}) + c$

$= 2m(x_{k+1}) + 2c - y_k - (y_k + 1)$

$= 2m(x_{k+1}) - 2y_k + 2c - 1$

$= 2y - y_k - y_{k+1}$ [$\because y_{k+1} = y_{k+1}$ (next point)]

$\therefore \Delta x(d_1 - d_2) = \Delta x \left[\frac{2\Delta y}{\Delta x}(x_{k+1}) + 2c - y_k - y_{k+1} \right]$

$\therefore \Delta x(d_1 - d_2) = 2\Delta y(x_{k+1}) + 2\Delta x c - y_k \Delta x - y_{k+1}\Delta x$

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

decision parameter = $\Delta x(d_1 - d_2)$ (P_k) (Recollection: not this, see this)

$= 2\Delta y \cdot x_k - 2\Delta x \cdot y_{k+1}$

$\therefore P_k = \Delta x(d_1 - d_2) = 2\Delta y x_k - 2\Delta x y_k + c$ (P_0)

∴ the next decision parameter

$P_{k+1} = 2\Delta y\, x_{k+1} - 2\Delta x\, y_{k+1} + c$

$\therefore P_{k+1} - P_k = 2\Delta y x_{k+1} - 2\Delta x y_{k+1} + c - 2\Delta y x_k + 2\Delta x y_{k+1}$ (recollection)

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

$\therefore P_{k+1} = 2\Delta y(x_{k+1} - x_k) + P_k + c$

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

where the term $y_{k+1} - y_k$ is either 0 or 1 depending on the sign of parameter P_k.

Bresenham's algo (pg 423)

i) I/p the 2 line endpoints & store the left end point in (x_0, y_0).

ii) Load (x_0, y_0) into the framebuffer that is plots the 1st point.

iii) Calculate constants - Δx, Δy, $2\Delta y$, $2\Delta y - 2\Delta x$ & obtain the starting value for the decision parameter as $P_0 = 2\Delta y - \Delta x$ → [1st P_k]

iv) At each x_k along the line, starting at $k=0$ perform the following tests

a) If $P_k < 0$, the next point to plot is (x_{k+1}, y_k) & $P_{k+1} = P_k + 2\Delta y$ → [calculating the next P_k]

b) If $P_k \geq 0$, the next point to plot is, (x_{k+1}, y_{k+1}) & $P_{k+1} = P_k + 2\Delta y - 2\Delta x$

v) Repeat step (iv) 'Δx' times.

Eg - ① The line end points are - (20, 10) & (30, 18). This line has a slope of 0.8 with $\Delta x = 10$, $\Delta y = 8$. Calculate the (x_k, y_k values) / Successive line positions. (pg 423) (imp)

End points - $(x_0, y_0) = (20, 10)$ & $(30, 18)$ end points

i) $\Delta x = 10$, $\Delta y = 8$

slope (m) = 0.8 (given)

ii) $P_0 = 2\Delta y - \Delta x = 2 \times 8 - 10 = 6$ (or) P_k $P_0 = 6$

111% case (b) satisfied

for 1st it $\begin{cases} P_{k+1} = P_k + 2\Delta y - 2\Delta x = 6 + 2*8 - 2*10 = 2 \\ x_{k+1} = x_k + 1 = 20+1 = 21 \\ y_{k+1} = y_k + 1 = 10+1 = 11 \end{cases}$ [$\because$ k starting from 0 & $(x_0, y_0) = (20, 10)$]

P_k (-ve)	P_{k+1} (+ve)	x_{k+1}	y_{k+1}
6	.	20	10 → starting points
6 >0	2	21	11
1 >0	-2	22	12
-2 <0	14	23	12
14 >0	10	24	13
10 >0	6	25	14
6 >0	2	26	15
2 >0	-2	27	16
-2 <0	14	28	16
14 >0	10	29	17
10 >0	6	30	18 → end points

$\Delta x = 10$ ∴ 10 times

[$\because P_{k+1} = P_k + 2\Delta y$, $x_{k+1} = x_k + 1$, $y_{k+1} = y_k$]

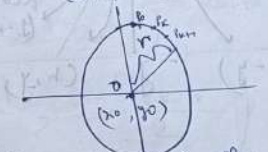
Imp
- Various output devices - (Remote) imp
- Diff b/w raestrom scan & random scan - imp

Imp
Aspect ratio - Ratio of width & height of the o/p of display device, or we can say it is a proportional relationship b/w an image's width & height.

14/3/24

LW for mid Point circle drawing algo

1) Input radius r & circle center (x_c, y_c) & obtain the first point on the circumference of a circle centered on the origin as $(x_0, y_0) = (0, r)$



2) Calculate the initial value of the decision parameter as $P_0 = \frac{5}{4} - r$

3) At each x_k position, starting at $k = 0$, Performed the following test -

i) If $P_k < 0$ then, the next point along the circle centered on (0,0) is (x_{k+1}, y_k) & (next point)

next $P_{k+1} = P_k + 2x_{k+1} + 1$ ← (inside the circle boundary, $P_k < 0$)

Otherwise, $(P_k \geq 0)$

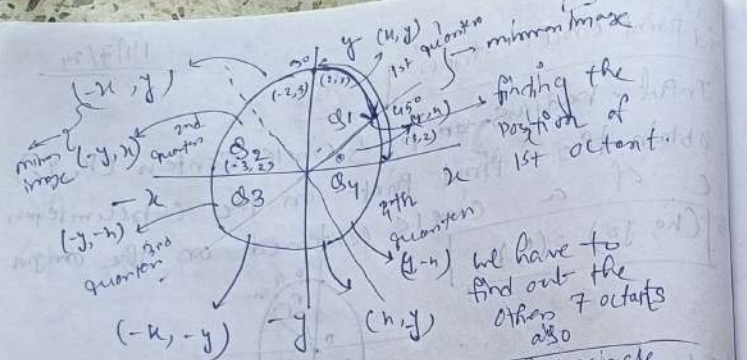
(ii) the next point along with the circle is (x_{k+1}, y_{k-1})

& $P_{k+1} = P_k + 2x_{k+1} + 1 - 2y_{k+1}$ where ← (outside the circle boundary) $P_k > 0$

$2x_{k+1} = 2x_k + 2$ & $2y_{k+1} = 2y_k - 2$

4) Determine symmetric points in the other 7 octants

(on the circle boundary $P_k = 0$)



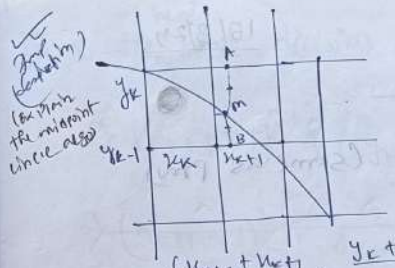
If field a circle, then we can see - only it's 1/8 i.e mirror image

5) move each calculated pixel position (x,y) onto the circular path centered on (x_c, y_c) & plot the ~~coordinate~~ values:

co-ordinate

$$x = x + x_c$$
$$y = y + y_c$$

6) Repeat steps 3 to 5 until $x \geq y$.



[Depending on the disision parameter we have choose A or B]

$$\text{Mid} = \left(\frac{x_k + 1 + x_{k+1}}{2}, \frac{y_k + y_k - 1}{2}\right) \quad \text{Prefid}$$
$$= \left(\frac{x_{k+1} + x_{k+1}}{2}, \frac{y_k + y_k - 1}{2}\right)$$
$$= \left(x_k + 1, y_k - \frac{1}{2}\right)$$

circle formula = $x^2 + y^2 = r^2$ → r = radious.

$x^2 + y^2 - r^2 = 0$

Circle eqn of mid point (m)

$$F_{circle}\left(x_k + 1, y_k - \frac{1}{2}\right) = (x_k + 1)^2 + \left(y_k - \frac{1}{2}\right)^2 - r^2 = 0$$

function of circle

$$\Rightarrow \boxed{P_k = (x_k + 1)^2 + \left(y_k - \frac{1}{2}\right)^2 - r^2} \quad -(1)$$

$P_{k(0,r)} = (0+1)^2 + \left(r - \frac{1}{2}\right)^2 - r^2$
$= 1 + r^2 - r + \frac{1}{4} - r^2$
$= \frac{5}{4} - r$

Proof disicion Parameter

$$= x_k^2 + 2x_k + 1 + y_k^2 - y_k + \frac{1}{4} - r^2$$
$$= (x_k + 1)^2 + y_k^2 - y_k + \frac{1}{4} - r^2$$

put P_k at $(0, r) = (0+1)^2 + r^2 - r + \frac{1}{4} - r^2$

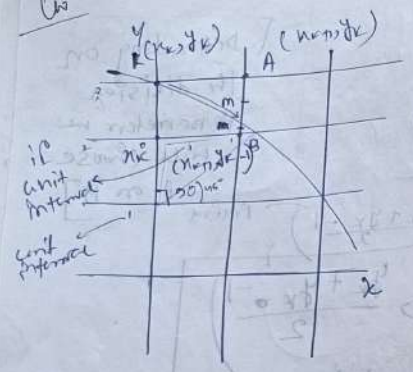
$$= 1 - r + \frac{1}{4}$$

P_k at $(0,r) = \frac{5}{4} - r$

P_0 or P_k at $(0,r)$ → Initial decision Parameter

$1 + 2s - r$

15/3/24



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

$m = (x_k + 1, y_k - \frac{1}{2})$

- for numerical calculation we will take consider the fractional part for p_0 or

d_k (initial decision parameter) : p_k or p_0 or $d_k = 1 - r$

Eg ① $d_k = 1.25 - r$

$d_k < 0$ → It means A is far from true circle, m is inside of the circle, then B is selected

$(x_{k+1}, y_{k+1}) = (x_k + 1, y_k - 1)$ (next point)

$d_k > 0$ → It means B is far from the circle, m is inside the circle, then A is selected

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

$y_{k+1} = y_k$ → next points

(same as next following next)

Then the new next decision variable $d_{k+1} = F_{circle}(x_{k+1} + 1, y_{k+1} - \frac{1}{2}) = r^2$

$\Rightarrow (x_{k+1} + 1)^2 + (y_{k+1} - \frac{1}{2})^2 - r^2 = 0$ [since this is a point] [∵ the circle formula $x^2 + y^2 = r^2$]

$\Rightarrow \{(x_k + 1) + 1\}^2 + (y_{k+1} - \frac{1}{2})^2 - r^2$ [∵ $x_{k+1} = x_k + 1$]

$\Rightarrow (x_k + 1)^2 + 1 + 2(x_k + 1) + y_{k+1}^2 + \frac{1}{4} - y_{k+1} - r^2$

$$d_{k+1} = (x_k + 1)^2 + 2(x_k + 1) + y_{k+1}^2 - y_{k+1} - r^2 + 1 + \frac{1}{4}$$

$\Delta d_k = d_{k+1} - d_k$ — (ii)

$= (x_k + 1)^2 + 2(x_k + 1) + y_{k+1}^2 - y_{k+1} - r^2 + 1 + \frac{1}{4} - (x_k + 1)^2 - (y_k - \frac{1}{2})^2 + r^2$

$= 2(x_k + 1) + y_{k+1}^2 - y_{k+1} + 1 + \frac{1}{4} - y_k^2 + y_k$

$= 2(x_k + 1) + (y_{k+1}^2 - y_k^2) - (y_{k+1} - y_k) + 1$

$\therefore d_{k+1} - d_k = 2(x_k + 1) + (y_{k+1}^2 - y_k^2) - (y_{k+1} - y_k) + 1$

$$d_{k+1} = d_k + 2(x_k + 1) + (y_{k+1}^2 - y_k^2) - (y_{k+1} - y_k) + 1$$

$d_{k+1} = d_k + 2x_{k+1} + (y_{k+1}^2 - y_k^2) + (-y_{k+1} + y_k) + 3$ kmmm

— (iii)

Now 2 cases may occur,

if $d_k < 0$ then, it means A ... circle then B is selected (next point (x_{k+1}, y_{k+1}) will be $= (x_k + 1, y_k - 1)$

then from (iii) we get,

$$d_{K+1} = d_K + 2(x_K + 1) + (y_K - 1)^2 - y_K^2 - \{(y_K - 1) - y_K\} + 1$$

$$= d_K + 2(x_K + 1) + y_K^2 - 2y_K + 1 - y_K^2 + 1 + 1$$

$$= d_K + 2(x_K + 1) - 2y_K + 2 + 1$$

$$= d_K + 2(x_K + 1) - 2(y_K - 1) + 1$$

as we know, $x_{K+1} = x_K + 1$ &

$y_K - 1 = y_{K+1}$ (next point)

$$\therefore \boxed{d_{K+1} = d_K + 2x_{K+1} - 2y_{K+1} + 1}$$

$d_{K+1} = d_K + 2x_K - 2y_K + 5$ ← maam — (iv)

If $d_K < 0$, then it means B — — — Circle, then A is selected & next point (x_{K+1}, y_{K+1}) will be $= (x_K + 1, y_K)$

then from (ii) we get

$$d_{K+1} = d_K + 2(x_K + 1) + y_K^2 - y_K^2 - \{y_K - y_K\} + 1$$

$$= d_K + 2(x_K + 1) + 1 + 1$$

as we know, $x_K + 1 = x_{K+1}$ &

$$\therefore \boxed{d_{K+1} = d_K + 2x_{K+1} + 1}$$

$d_{K+1} = d_K + 2x_K + 3$ ← maam — (v)

Eg —

$d_K = 1.25 - r$

$\simeq 1 - r$

$d_K < 0$.

(A)

$x_{K+1} = x_K + 1$

$y_{K+1} = y_K$

$d_{K+1} = d_K + 2x$

(1) original $(x_K, y_K) =$ the circ

	x_K	
1	0	
2	1	
3	2	
4	3	

(nor

Con

(for +ve slope $m \leq 1$ (i.e., slope is +ve but less than 1))

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

$\Rightarrow d_1 - d_2 = 2\frac{dy}{dx}(x_k+1) - 2y_k + 2c - 1$ [$\because m = \frac{dy}{dx} = \frac{\Delta y}{\Delta x}$]

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

$\Rightarrow P_k = 2\Delta y x_k + 2\Delta y - 2\Delta x y_k + 2\Delta x c - \Delta x$

[$\because$ decision parameter P_k or $P_0 = \Delta x(d_1 - d_2)$]

$\Rightarrow P_k = 2\Delta y x_k - 2\Delta x y_k + (2\Delta y + 2\Delta x c - \Delta x)$

$\Rightarrow \boxed{P_k \text{ or } P_0 = 2\Delta y x_k - 2\Delta x y_k + c}$ — (IV)

[$\because$ let us consider $2\Delta y + 2\Delta x - \Delta x = c$] [$\because$ by constant y_k & x_k are the only variable here rest $\Delta x, \Delta y$ or m is fixed for a line]

$\Rightarrow P_k$

Now,

$\boxed{P_{k+1} \text{ (next point of } P_k \text{ or } P_0) = 2\Delta y(x_{k+1}) - 2\Delta x(y_{k+1}) + c}$ — (V)

$\therefore P_{k+1} - P_k = 2\Delta y(x_k+1) - 2\Delta x(y_k+1) + c - 2\Delta y x_k + 2\Delta x y_k - c$

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

here x is always incrementing so we can write x_{k+1} as x_k+1

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

$\Rightarrow \boxed{P_{k+1} = P_k + 2\Delta y - 2\Delta x(y_{k+1} - y_k)}$ — (VI)

Now, there are 2 cases,

when $P_k < 0$ & $d_1 - d_2 < 0$; $d_1 < d_2$ [$\because \Delta x \neq 0$]

18/3/24

then we'll write y_{k+1} as y_k (as current pixel's distance from intersection point y is smaller, so we'll have to choose current pixel)

[∵ we have to choose closest point to the original line path & $d_1 < d_2$, ∴ d_1 is closest, for d_2 point need to be considered (x_{k+1}, y_k)]

∴ then from (VI) we get,

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

$$\Rightarrow \boxed{p_{k+1} = p_k + 2\Delta y} \quad - (VII)$$

when, $p_k > 0$; $d_1 + d_2 > 0$; $d_1 > d_2$ then we'll write y_{k+1} as y_{k+1} (as current pixel's distance from intersection point y is longer, ——— pixel)

∴ then from (VI) we get

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

(reverse reason to select d_2)

$$\Rightarrow \boxed{p_{k+1} = p_k + 2\Delta y - 2\Delta x} \quad - (VIII)$$

18/3/24

Eg =

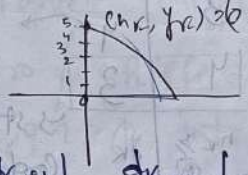
(1) $d_k = 1.25 - r$ ← Assume
$\simeq 1 - r$

$d_k < 0$ (A)
$x_{k+1} = x_k + 1$
$y_{k+1} = y_k$
$d_{k+1} = d_k + 2x_k + 3$

$d_k \geq 0$ (B)
$x_{k+1} = x_k + 1$
$y_{k+1} = y_k - 1$
$d_{k+1} = d_k + 2(x_k - y_k) + 5$

(prev) not this

① original
Start $(x_k, y_k) = (0, 5)$. find out the 1/8 th part of the circle,

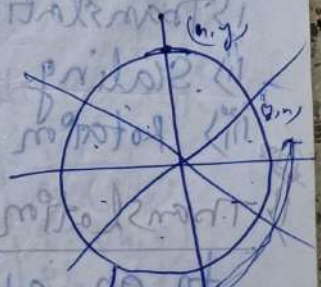


	x_k	y_k	x_{k+1}	y_{k+1}	d_{k+1}	$d_k = 1 - r$
1.	0	5	1	5	$-4+2+3$ $= -1 < 0$	$1 - 5 = -4 < 0$
2.	1	5	2	5	$-1+2+3$ $= 4 \geq 0$	
3.	2	5	3	4	$4 + 2(2-5) + 5$ $= 4 - 6 + 5$ $= 3 > 0$	⇒ pt 0 next way
4	3	4	4	3	$3 + 2(-1) + 5$ $= 6$ Stop	

(moved)

$x_k \geq$ ↓ Stopping Condition
So Stop

y_k



can get the values by doing like this

Chap-5

another way (like bresenham line drawing algo)

2D - transfor if we know that k starts from 0 & given $(x_k, y_k) = (0, 5)$

$\therefore (x_0, y_0) = (0, 5) \therefore r = 5$

ii) $d_{k+1} - r_0 = 1 - 5 = -4 < 0$

iii) $d_k < 0$ ∴ case (i) is satisfied

for if it

$\therefore \begin{cases} x_{k+1} = x_k + 1 = x_0 + 1 = 0 + 1 = 1 \\ y_{k+1} = y_k = y_0 = 5 \\ d_{k+1} = d_k + 2x_{k+1} + 1 \\ \quad = -4 + 2*1 + 1 \\ \quad = -1 \end{cases}$ (k starts from 0)

d_k	d_{k+1}	x_{k+1}	y_{k+1}
		0	5
1. $-4<0$	-1	1	5
2. $-1<0$	4	2	5
3. $4>0$	3	3	4
4. $3>0$	6	4	3

$[\because d_{k+1} = d_k + 2x_{k+1} - 2y_{k+1} + 1$
$x_{k+1} = x_{k+1}$
$y_{k+1} = y_k - 1]$

$x_k \geq y_k$ stopping condition

Chap 5

2D - transformation

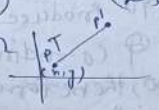
Transformation - changes in orientation, size & shape are accomplished with geometric transformation that alter the coordinates of the objects. there are 3 basic transformation -

i) Translation
ii) Scaling
iii) Rotation

i) Translation

- A translation is applied to an object by repositioning it along a straight line path from one co-ordinate location to another.

In eg - we translate a 2D point by adding translation distance $T(t_x, t_y)$, to the original coordinate position (x, y) to move the point to a new position (x', y').



$$\left.\begin{aligned} x' &= x + t_x \\ y' &= y + t_y \end{aligned}\right\} \quad \text{(translation eqn)} \quad \text{— (1)}$$

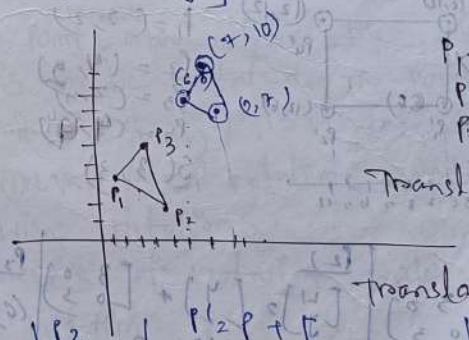
the translation distance pair (t_x, t_y) is called a translation vector / shifted vector. we can also express the eqn (1) into matrix form

$$P' = P + T \quad \text{— (11)}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

$P = (x, y)$
$T = (t_x, t_y)$
$P' = (x', y')$
$P' = P + T$

Eg - (1)



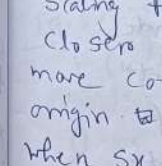
$P_1 = (2, 3)$
$P_2 = (5, 2)$
$P_3 = (3, 5)$

Translation vector = $T(2, 5)$ i.e. t_x t_y.

Translate the obj.

P_1	P_2	$P'_2 = P + T$	P_3
$P = (2, 3)$	$P = (5, 2)$	$\Rightarrow \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \end{bmatrix} + \begin{bmatrix} 4 \\ 5 \end{bmatrix}$	$P_2 = (3, 5)$
$T = (4, 5)$	$T = (4, 5)$	$= \begin{bmatrix} 6 \\ 8 \end{bmatrix}$	$T = (4, 5)$
$P'_1 = (x', y')$	$P'_2 = (x', y')$	$P' = (6, 8)$	$P'_{32} = (7, 10)$
	$= (9, 7)$		

Scaling — A scaling transformation alters the size of an obj. This operation can be carried out for polygons by multiplying the co-ordinate values (x, y) of each vertex by scaling factors $S(s_x, s_y)$ to produce the transformed co-ordinates (x', y').



So, therefore (scaling eqn)

$$x' = x \cdot s_x$$
$$y' = y \cdot s_y \quad \text{— (1)}$$

If scaling factors $= 1$ the size will not change
$S \cdot F > 1$ → increase
$S \cdot F < 1$ → decrease

Scaling factors s_x scales in x-direction, while s_y scales in y-direction. The transformation eqn (1) can also be written in matrix form as

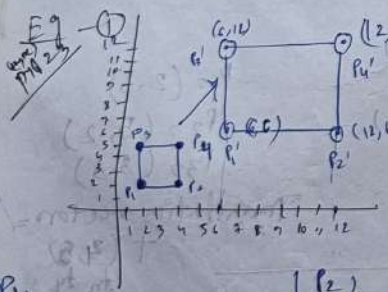
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} \cdot \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$$

[If identity matrix $s_x = s_y = 1$ then there will be no change]

$$P' = P \cdot S \quad \text{— (11)}$$

↓ Original point

Eg (page no. 24)



$P_1 = (2, 2)$
$P_2 = (4, 2)$
$P_3 = (2, 4)$
$P_4 = (4, 4)$
$S(3, 3)$

P_1

$$\begin{bmatrix} x_1' \\ y_1' \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \end{bmatrix} \cdot \begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix}$$
$$= \begin{bmatrix} 6+0 \\ 0+6 \end{bmatrix}$$
$$P_1'(6, 6)$$

P_2

$$\begin{bmatrix} x_2' \\ y_2' \end{bmatrix} = \begin{bmatrix} 4 \\ 2 \end{bmatrix} \cdot \begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix}$$
$$= \begin{bmatrix} 12+0 \\ 0+6 \end{bmatrix}$$
$$P_2'(12, 16)$$

$P_3 \rightarrow (6, 12) \rightarrow P_3'$

$P_4 \rightarrow (12, 12) \rightarrow P_4'$

21/3/24

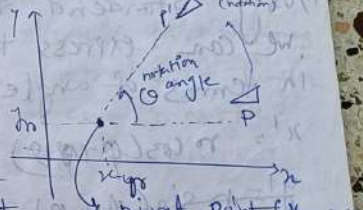
(Scaling)

Objects transforming with the scaling eqn $(P' = P \cdot S)$ are used for both scaled & repositioned. Scaling factors with values < 1, move objects closer to the co-ordinate origin, while values > 1 move co-ordinate positions farther from the origin.

When s_x & s_y are assigned the same value, a uniform scaling is produced that maintains relative object proportions.

Rotation pg 23

1. A 2D rotation is applied to an object by repositioning it along a circular path in the (x, y) plane.



Rotation angle is θ & the position of the rotation point (pivot point) about which the object is to be rotated. [The position of the pivot point (x_r, y_r) is (x_r, y_r)]

3) The values of rotation angle means counter clockwise rotations.

4) $-$ values of rotation angle means clockwise rotation.

5) Ist we determine the transformation eqns for rotation of a point P

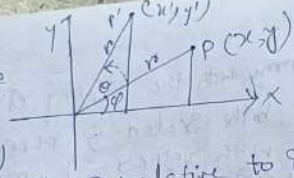


Fig- Rotation of a point from (x,y) to (x',y') through an angle θ relative to coordinate origin. The original angular displacement of the point from the x, axis is φ.

Here, Pivot point is at the co-ordinate origin.

$\sin\theta = \frac{\text{Perpendicular}}{\text{Hypotenuse}}$

$\cos\theta = \frac{\text{Base}}{\text{Hypotenuse}}$

Using Standard trigonometric identities we can express the transformed co-ordinates in terms of angle θ & φ as -

$x' = r\cos(\phi+\theta)$ $\quad \left[\begin{matrix} x = r\cos\phi \\ y = r\sin\phi \end{matrix}\right]$ -(1)

$y' = r\sin(\phi+\theta)$

$x' = r\cos\phi\cos\theta - r\sin\phi\sin\theta$

$y' = r\sin(\theta+\phi)$

$= r\cos\phi\sin\theta + r\sin\phi\cos\theta$

$y' = r\cos\phi\sin\theta + r\sin\phi\cos\theta$

-(1)

Substituting eqn (1) into eqns we get (rotation eqn)

$x' = x\cos\theta - y\sin\theta$

$y' = x\sin\theta + y\cos\theta$ } -(11)

Its Rotation matrix will be $P' = R \cdot P$ where,

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

Eg-1

P (4, 3)

angle = 45°

θ = 45°

$x' = 4\cos 45 - 3\sin 45 = 0.707$

$y' = 4\sin 45 + 3\cos 45 = 4.95$ ≈5

dw ch-4 10 marks

Attributes of output Primitives 15/4/24

Any parameter that affects the way a primitive is to be displayed is referred to as an attribute parameter. Some attribute parameters are color & size which determine the fundamental characteristics of a primitive.

Line attributes 2023

~~The attri parameters are -~~

It have type, width, color, pen & brush - this properties.

Type - 3 types of lines - solid, dotted, dashed. ① A dashed line could be displayed by generating an inter dash spacing, equal to the length of the solid sections. That is

② A dotted line can be displayed by generating very short dash with spacing equal to or greater than dash itself (dash size).

③ A solid line can be displayed by generating a line without any breaks / spacings.

Solid dotted Dashed

3rd Width - Implementation of line width options depends on the capabilities of the output device.
Line width parameter - 1w is assigned a positive no. to indicate the relative width of the line to be displayed. A value of 1 specifies a standard width line. For eg, on a pen plotter, for instance, a user could set 1w (i.e. you have to put any +ive no.) a value of 0.5 to plot a line whose width is $\frac{1}{2}$ of the standard line.
Values greater than (>)1 produce lines thicker than the standard.
For raster implementation, a standard width line is generated with single pixels at each sample position as in the Bresenham algo.

2nd Line endcaps - the problem with implementing width options using horizontal/vertical pixel spans is that, the method produces lines whose ends are horizontal/vertical regardless of the slope of the line. This effect is more noticeable with very thick lines. We can adjust the shape of the line ends to give them a better appearance by adding

Line caps.
Types are -
1) Butt cap -
2) Round cap -
3) Projecting square cap -

The butt cap is obtained by adjusting the end positions of the component parallel lines so that the thick line is displayed with square ends that are perpendicular to the line path.

Another line cap is the round cap. This is obtained by adding a filled semi-circle to each butt cap. The circular arcs are centered on the line endpoints & have a diameter equal to the line thickness.

The 3rd type of line cap is the projecting square cap. Here we simply extend the line & add butt caps that are positioned 1/2 of the line width beyond the specified end points.

2nd Line joins - There are 3 possible methods for smoothly joining 2 line segments. These are -

1) miter join -

A miter join is accomplished by extending the outer boundaries of each of the 2 lines until they meet.

2) Round join

A round join is produced by capping the connection b/w the 2 segments with a circular boundary whose diameter is equal to the line width.

3) Bevel join

A bevel join is generated by displaying the line segments with butt caps & filling in the triangular gap where the segments meet. If the angle b/w the 2 connected line segments is very small, a miter join can generate a long spike that distorts the appearence of poly lines.

(In that case, we can ~~avoid it~~ switching to avoid this,
it from miter join to bevel / round join

Pen & brush - i) the selected pen & brush determine the way a line will be drawn.
ii) pens & brushes have size, shape, color & pattern attribute.
iii) pixel mask is applied in both of them.

Curve attribute

We can display curves with varrying colors, widths dot-dash patterns & available pen / brush options.
Raster curves of various widths can be displayed using the method of horizontal / vertical pixel spans. If the magnitude of the curve slope is < 1, we plot vertical spans if it is > 1, we plot horizontal spans.

Color & gray scale level

① Colors are represented by color codes which are positive integers.
② Color information is stored in frame buffer or in seperate table & uses pixel values that as index to the color table.
③ Two ways to store the color information - indirect & direct

22/4/24

matrix representation homogeneous co-ordinate system

CW

generalized eqn

$P' = M_1 P + M_2$ (multiplicative, additive)

[$P' = P + T$ (scaling)
$P' = P \cdot S$ (2D transformation)
$P = P' P_1$]

M_1 = multiplicative for translation (as M_2 isn't needed)

M_2 = Scaling (as M_1 isn't needed)

Why we need the homogenous co-ordinate system? (Imp)

reason

[Homogenous co. are used in C.G to represent points & hyperplanes in projective space. they are used to rep. 3D positions & transformations. we need them, because they allow common vector operations such as translation, rotation, scaling etc to be rep. as a matrix by which the vector is multiplied.

Matrix rep. & homogeneous coordinates

$P' = M_1 \cdot P + M_2$

- P & P' are column vectors.
- M_1 is a 2 * 2 array containing multiplicative factors.
- M_2 is a 2 element column matrix containing translational terms
- for translation M_1 is the identity matrix
- For rotation / scaling, M_2 contains the translational terms associated with the pivot point (scaling) fixed point.

- to produce a sequence of operations, such as scaling followed by rotation then translation we could calculate the transformed coordinates one step at a time.
- A more efficient approach is to combine transformations, without calculating intermediate co-ordinate values.

- multiplicative & translational terms for a 2D geometric transformation can be combined into a single matrix if we expand the rep.s to 3 by 3 matrices.
- we can use the 3rd col. for translation terms, & all transformation eqns can be expressed as matrix muls.
- Expand each 2D coordinate (x, y) to 3 element rep (x_h, y_h, h) called homogeneous co-ordinates.
- h is the homogeneous parameter such that $x = x_h/h$, $y = y_h/h$
- Infinite homogeneous rep. for a point
- A convenient choice is to choose $h = 1$
- 2D translation matrix

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

or, $P' = T(t_x, t_y) \cdot P$ (multiplicative - by ignoring all intermediate calc.s)

Cas, there are infinite vals for h, so for simplification $h = 1$ (taken)

- 2D Rotation matrix

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

or, $\boxed{P' = R(\theta)P}$

- 2D scaling matrix

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

or, $\boxed{P' = S(S_x, S_y) \cdot P}$

2D Composite transformation

we can setup a sequence of transformations as a composite transformation matrix by calculating the product of the individual transformations

- $\boxed{P' = M_2 M_1 P = MP}$

Composite 2D translation (without pivot point)

If 2 successive translation are applied to a point P, then the final transformed location P' is calculated as,

$$\boxed{P' = T(t_{x_2}, t_{y_2}) \cdot T(t_{x_1}, t_{y_1}) \cdot P = T(t_{x_1}+t_{x_2}, t_{y_1}+t_{y_2}) \cdot P}$$

$$\begin{bmatrix} 1 & 0 & t_{x_2} \\ 0 & 1 & t_{y_2} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & t_{x_1} \\ 0 & 1 & t_{y_1} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_{x_1}+t_{2x} \\ 0 & 1 & t_{y_1}+t_{2y} \\ 0 & 0 & 1 \end{bmatrix}$$

Composite 2D rotations (with pivot point)

$\boxed{P' = R(\theta_1 + \theta_2) \cdot P}$

$$\begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 \\ \sin\theta_2 & \cos\theta_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} =$$

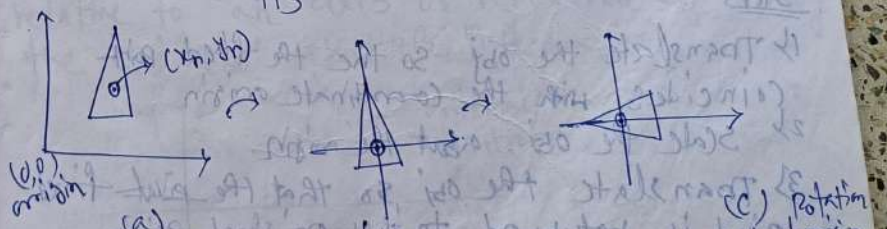
$$\begin{bmatrix} \cos(\theta_1+\theta_2) & -\sin(\theta_1+\theta_2) & 0 \\ \sin(\theta_1+\theta_2) & \cos(\theta_1+\theta_2) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

2D

General pivot point rotation

Steps:

1. Translate the obj. so that the pivot point is moved to the coordinate origin.
2. Rotate the obj about the origin
3. Translate the obj. so that the pivot point is returned to its original position.



original pos of obj & pivot point

translation pivot point (obj so that (x_h, y_h) is at origin

(d) translation of obj so that the pivot point is returned to its position (x_p, y_p)

$$\text{matrix} = \begin{bmatrix} 1 & 0 & x_p \\ 0 & 1 & y_p \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -x_p \\ 0 & 1 & -y_p \\ 0 & 0 & 1 \end{bmatrix}$$

(translation) (Rotation) (inverse translation)

$$= \begin{bmatrix} \cos\theta & -\sin\theta & x_p(1-\cos\theta) + y_p \sin\theta \\ \sin\theta & \cos\theta & y_p(1-\cos\theta) - x_p \sin\theta \\ 0 & 0 & 1 \end{bmatrix}$$

(final matrix)

Composite 2D scaling (without pivot point)

$$S(S_{x_2}, S_{y_2}) \cdot S(S_{x_1}, S_{y_1}) = S(S_{x_1} \cdot S_{x_2}, S_{y_1} \cdot S_{y_2})$$

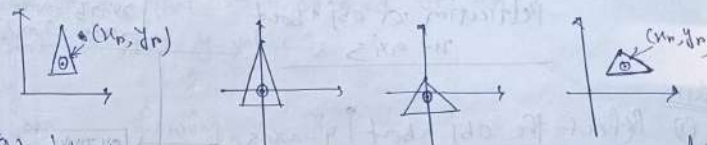
$$\begin{bmatrix} S_{x_2} & 0 & 0 \\ 0 & S_{y_2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} S_{x_1} & 0 & 0 \\ 0 & S_{y_1} & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} S_{x_1} S_{x_2} & 0 & 0 \\ 0 & S_{y_1} S_{y_2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

[$x' = x_n$] (same matrix if)

General fixed point scaling

Steps

1) Translate the obj so the the fixed point coincides with the coordinate origin
2) Scale the obj about the origin.
3) Translate the obj so that the fixed point is returned to its original position.



(a) original position of obj & fixed point

(b) Translate obj so that the fixed point (x_p, y_p) is at origin

(c) scale the obj with respect to origin

(d) Translate obj, so that the fixed point (x_p, y_p) is returned to it's original position

$$\text{matrix} = \begin{bmatrix} 1 & 0 & x_f \\ 0 & 1 & y_f \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -x_f \\ 0 & 1 & -y_f \\ 0 & 0 & 1 \end{bmatrix} =$$

(translation) (scaling) (reverse / inverse translation)

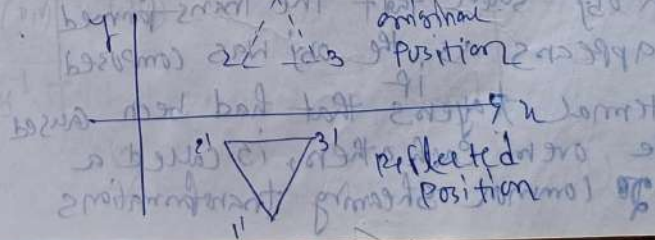
$$\begin{bmatrix} S_x & 0 & x_f(1-S_x) \\ 0 & S_y & y_f(1-S_y) \\ 0 & 0 & 1 \end{bmatrix}$$

(final mat)

$$T(x_f, y_f) \cdot S(S_x, S_y) \cdot T(-x_f, -y_f) = S(x_f, y_f, S_x, S_y)$$

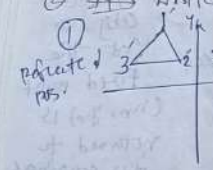
Other 2D transformation

Reflection - A reflection is a transformation that produces a mirror image of an obj. The mirror image for a 2D reflection is generated relative to an axis of reflection by rotating the obj 180° about the reflection axis



Reflection of obj about x-axis

Q.1N
(1) Reflect the obj about y-axis
(2) ~~This~~ write the matrix for it (it).



Reflection of obj about y-axis / about the line x=0 (180° rotation)

(2) matrix = $\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

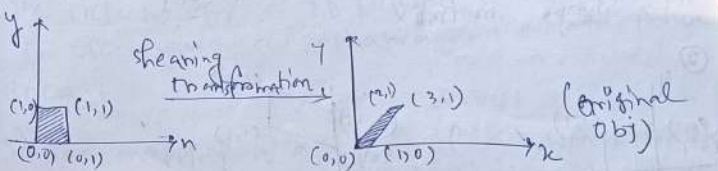
matrix = $\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Reflection about the line y=0 (the x axis)
(Previous slide)

- Image is generated relative to an axis of reflection by rotating the object 180° about the reflection axis.

Shear

A transformation that distorts the shape of an obj such that the transformed shape appears as if the obj was composed of internal layers that had been caused to slide over each other, is called a shear. Two common shearing transformations are those that shift co-ordinate x values & those that shift y-values.



$Sh_x = 2$ (Shearing parameter)

also Sh_y = Shearing parameters

Shearing in direction x-direction

$\begin{bmatrix} 1 & Sh_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

$x' = x + Sh_x \cdot y$

$y' = y$

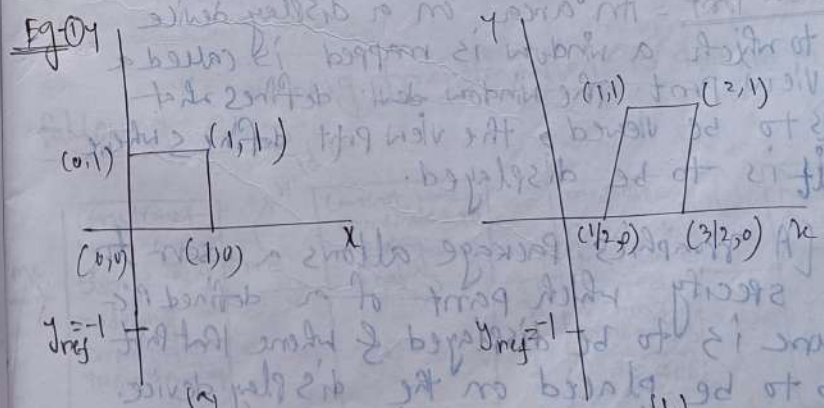
Shearing in direction y-direction

$\begin{bmatrix} 1 & 0 & 0 \\ Sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

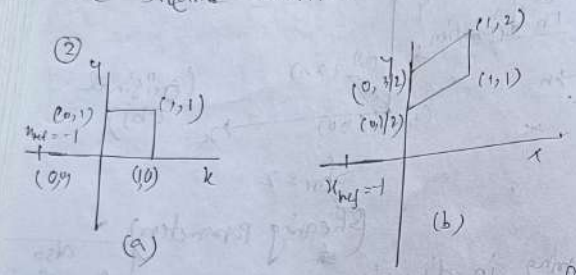
$x' = x$

$y' = y + Sh_y \cdot x$

Eg-04



A unit square (a) is transformed to a shifted Parallelogram (b) with $sh_x = 0.5$ & $y_{ref} = -1$ in the shear matrix.



A unit square (a) is turned into a shifted Parallelogram (b) with parameter values $sh_y = 0.5$ & $y_{ref} = -1$ in the y direction shearing transformation in the shear matrix

(2)

Chap 6 — 2D- viewing

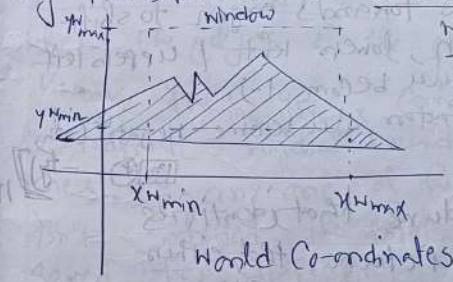
Window - A world co-ordinate area selected for displaying display is called a window.

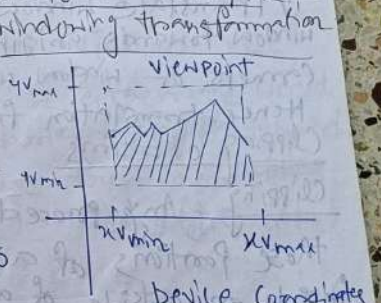
View Port - An area on a display device to which a window is mapped is called a view port. The window defines what is to be viewed, the view port defines where it is to be displayed.

[A graphics package allows a user to specify which part of a defined picture is to be displayed & where that part is to be placed on the display device. Any conventional cartesian co-ordinate system, referred to as the world-coordinate reference frame, can be used to define the picture. For a 2D picture, a view is selected by specifying a subarea of the total picture area.]

The mapping of a part of a world coordinate scene to device co-ordinates is referred to as a viewing transformation - viewing transformation

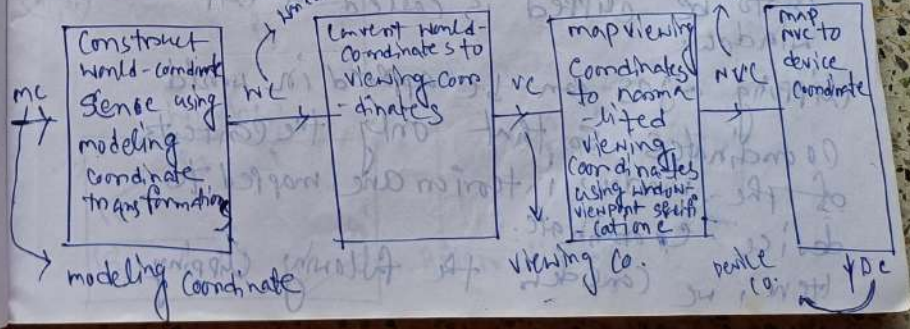
Sometimes this transformation is also called window-to-viewport transformation / window-to-viewport transformation - the window-ing transformation





Windowing transformation

the viewing transformation is done in the following steps:



The 2D viewing transformation pipeline

2/5/24

Window to viewport Co-ordinate transformation.

It is the process of transforming 2D world co-ordinate objects to device coordinates. objects inside the world or clipping window are mapped to the viewport which is the area on the screen where world coordinates are mapped to be displayed.

Mathematical calculation of window to viewport

1) Translate window towards origin to shift window towards origin, lower left / upper left corners of window will become (-).

Hence translation factor will become negative (-tx, -ty)

Clipping operations

Clipping - Any procedure that identifies those portions of a picture that either inside or outside of a specified area, region, of space is referred to as a clipping algorithm / simply clipping.

The region against which an obj is to be clipped is called a clip window.

Clipping algos can be applied in world coordinates so that only the contents of the window interior are mapped to device coordinate.

Here, we consider the following clipping algos of the following primitive types.

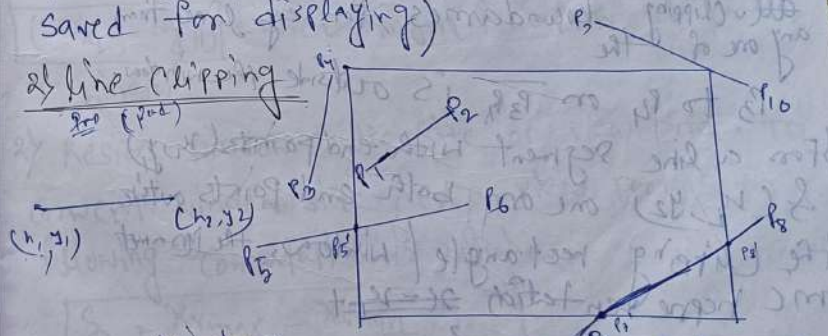
1. Point clipping
2. Line clipping (straight line segments)
3. Area clipping (polygons)

1) Point clipping. 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.

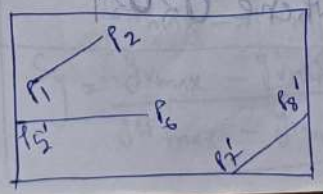
$$x_{wmin} \le x \le x_{wmax}$$
$$y_{wmin} \le y \le y_{wmax}$$

where the edges of the clip window x_{wmin}, x_{wmax}, y_{wmin}, y_{wmax} can be either the world co-ordinate window boundaries / viewport boundaries. If any one of these four inequalities isn't satisfied, the point is clipped (i.e not saved for displaying)

2) Line clipping



Window



Before clipping

After clipping

A line clipping procedure involves several parts. 1) 1st, we can test a given line segment to determine whether it lies completely inside the clipping window. 2) If it doesn't, we try to determine whether it lies completely outside the window.

3) Finally, if we can't identify a line as completely inside / completely outside we must perform intersection calculations with one or more clipping boundaries. We process lines through the "inside-outside" test by checking the line end points.

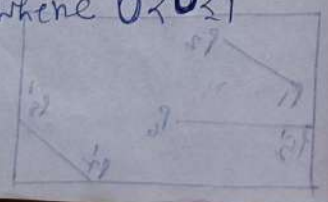
4) A line with both end points inside all clipping boundaries, such as the line from P_1 to P_2 is saved.

5) A line with both endpoints outside any one of the clipping boundaries, for eg line from P_3 to P_4 or $\overline{P_3 P_4}$ is outside the window.

6) For a line segment with end points (x_1, y_1) & (x_2, y_2) one or both end points outside the clipping rectangle / window, the parametric representation

$$x = x_1 + u(x_2 - x_1)$$
$$y = y_1 + u(y_2 - y_1)$$

where $0 \le u \le 1$



the parametric values could be used to determine the values of parameter u for intersections with the clipping boundary coordinates. If the val. of u for an intersection with a rectangle boundary edge is outside the range 0 to 1, the line doesn't enter the interior of the window. If the value of u is within the range from 0 to 1, the line segment does cross into the clipping area. This

This method can be applied to each clipping boundary edge in order to determine whether any part of the line segment is to be displayed.

7) (Window to viewpoint transformation)

$(-W_{x_L}, -W_{y_L})$ - when origin is lower left corner of the screen.

$(-W_{x_L}, -W_{y_H})$ - when origin is upper left corner of window.

2) Resize window to the size of view port. To convert window size into to view port size following computation is required.

$$S_x = \frac{x_{vmax} - x_{vmin}}{x_{Wmax} - x_{Wmin}}$$

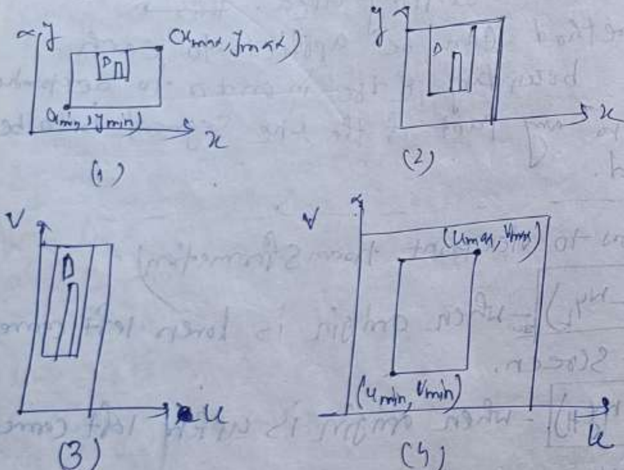
$$S_y = \frac{y_{vmax} - y_{vmin}}{y_{Wmax} - y_{Wmin}}$$

(pre 2 pictures)

3) Translate window (position of window must be same as position of view port).

If lower left corner of viewport is (0,0) we don't need to take step 3 because window lower left corner is already shifted on origin after taking first step.

If lower left corner is not (0,0) we have to take translation factor (t)



(1) window in world coordinates
(2) window translated to origin
(3) window scaled to size of viewport
(4) Translated by (u_{min}, v_{min}) to final position.

$$T(-x_{min}, -y_{min}) \; S\left(\frac{u_{max}-u_{min}}{x_{max}-x_{min}}, \frac{v_{max}-v_{min}}{y_{max}-y_{min}}\right) \; T(u_{min}, v_{min})$$

$$M_{WV} = T(u_{min}, v_{min}) \cdot S\left(\frac{u_{max}-u_{min}}{x_{max}-x_{min}}, \frac{v_{max}-v_{min}}{y_{max}-y_{min}}\right) \cdot T(-x_{min}, -y_{min})$$

$$= \begin{bmatrix} 1 & 0 & u_{min} \\ 0 & 1 & v_{min} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \frac{u_{max}-u_{min}}{x_{max}-x_{min}} & 0 & 0 \\ 0 & \frac{v_{max}-v_{min}}{y_{max}-y_{min}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -x_{min} \\ 0 & 1 & -y_{min} \\ 0 & 0 & 1 \end{bmatrix}$$

matrix form (no need)

6/5/24

Chapter: Line clipping – Cohen-Sutherland in c.s.l. algo

Every line end points in a picture is assigned a 4-digit binary code, called a region code, that identifies the location of the point relative to the boundaries of the clipping rectangle.

Bit 1 : left (If bit1 = 1)
Bit 2 : right (If bit2 = 1)
Bit 3 : below (If bit3 = 1)
Bit 4 : above (If bit4 = 1)

1001	1000	1010
0001	0000 window	0010
0101	0100	0110

Conditions

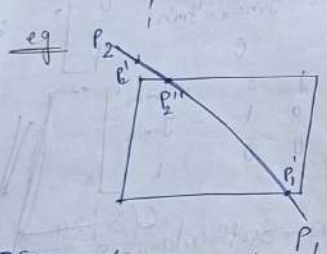
Trivially accept – both ends are code of 0000 for both end points. → inside region

Trivially reject: Code(Point1) & Code(Point2) != 0. (note the SINGLE &). ← both outside of region

pointcull - any one is outside of region

Otherwise,
start from an outside

eg



i) Starting with the bottom end point of the line from P_1 to P_2 we have to check P_1 against the left - right & bottom boundaries in

- segment P_1P_2 becomes $P_1'P_2$ after intersection with the bottom side.
- segment $P_1'P_2$ becomes $P_1'P_2'$ after intersection with the left side.
- Segment $P_1'P_2'$ becomes $P_1'P_2''$ after intersection with the top side.
- Continue until a trivial accept or trivial reject. //

~~The intersection with a~~

turn & find that this point is below the clipping rectangle

ii) Next, we find the intersection point P_1' with bottom boundary & discard the line section from P_1 to P_1'.

So, now the line segment become P_1' to P_2.

(iii) Since P_2 is outside the clipping window, we check these endpoints against the boundaries & find that it is to the left of the window.

Intersection point P_2' is calculated & this point is above the window.

(iv) So, the final intersection calculation yields P_2'' & the line from P_1' to P_2'' is saved.

~~(v) So, calculations~~ In intersection points with a clipping boundary can be used as the slope-intercept form of the line eqn.

for eg, a line with end point coordinates (x_1, y_1) & (x_2, y_2), the y-coordinate of the intersection point with a vertical boundary can be obtained with the calculation:

$$y = y_1 + m(x - x_1)$$

(x is either x_{wmin} or x_{wmax})

where, the x-value is set either to x_{wmin} or x_{wmax} & the slope of the line is calculated as $m = \frac{y_2 - y_1}{x_2 - x_1}$.

Similarly, the intersection with a horizontal boundary the x-coordinate can be calculated as:

$$x = x_1 + (y - y_1)/m$$

where, y is set to either y_{wmin} or y_{wmax}.

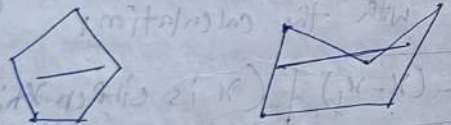
Liang like clipping have to read

Polygon clipping / Area clipping

A polygon boundary processed with a line clipper may be displayed as a series of unconnected line segments, depending on the orientation of the polygon to the clipping window.

Simple line crossing Hole

n-gon - An n-gon is a polygon with more than 4 vertices & edges.



Convex Polygon Non-Convex Polygon

Sutherland-Hodgman polygon clipping

i) Beginning with the initial set of polygon vertices, we could first clip the polygon against the left rectangle boundary to produce a new sequence of vertices.

ii) The new set of vertices could then be successively passed to a right boundary clipper, a bottom boundary clipper & a top boundary clipper.

iii) At each step, a new sequence of output vertices is generated & passed to the next window boundary clipper.

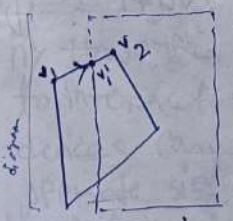
iv) We have to make the following test, when the polygon vertices is passed to the window boundary clippers:

1) If the first vertex is outside the window boundary & the 2nd vertex is inside, both the intersection point of the polygon edge with the window boundary & the 2nd vertex are added to the output vertex list.

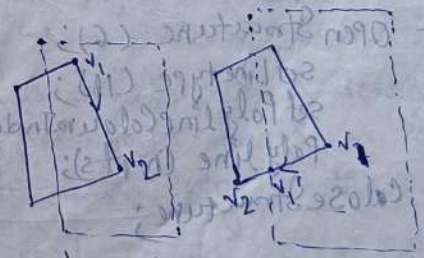
2) If both input vertices are inside the window boundary, only the 2nd vertex is added to the output vertex list.

3) If the 1st vertex is inside the window boundary & the 2nd vertex is outside only the intersection with the window boundary is added to the window vertex list.

4) If both input vertices are outside the window boundary nothing is added to the output list.

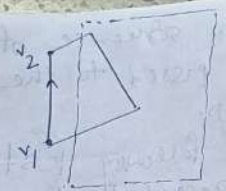


Out → in
Save v_1', v_2



in → in
Save v_2

in → out
Save v_1'



ctrl → out
same name

9/5/24

Structures of a hierarchical modeling -

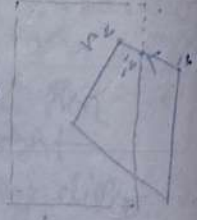
A labeled set of output primitives in PHIGS (programmer's hierarchical interactive graphic system) [set of tools] is called a structure. Other commonly used labeled collection of primitives are segments & objects

Basic structure functions -

i) OpenStructure (id)
ii) CloseStructure

Any no. of structures can be created for a picture but only 1 structure can be opened at a time. Any open structure must be closed before a new structure can be created

Eg - OpenStructure (6);
setLinetype (1);
setPolyLineColourIndex (1);
PolyLine (n, pts);
CloseStructure;



Interactive input methods - ~~Logical classification~~

i) Logical classification of i/p devices - the various kinds of input datas (text, audio, video etc) are summerised in the following 6 logical device classification used by PHIGS & GKS (Graphical Kernel System / Graphics Kernal System) -

a) LOCATOR: A device for specifying a coordinate position (x, y)
b) STROKE: A device for specifying a series of coordinate positions
c) STRING: A device for specifying text input
d) VALUATOR: A device for specifying scalar values.
e) CHOICE: A device for selecting menu option
f) PICK: A device to pick a picture component for selecting

(3-4 lines for each devices) (shortnote)

Input functions - Graphical i/p functions can be setup to allow users to specify the following options -

i) Which physical devices are used to provide i/p within a particular logical classification. For eg, a tablet used as a stroke device.
ii) How the graphics programmes & devices are to interact. For eg, either the progs or the devices can initiate data entry / both can operate simultaneously.
iii) When the data is added to the i/p & which device is to be used at that time to deliver

a particular i/p type to the specified data variables.

Input modes – the i/p modes specify how the programe & i/p devices interact. O/p could be initiated by the prog, as the prog & i/p devices both could be operational simultaniously, as data i/p could be ~~done~~ with ~~si~~ multaniously, as data i/p could be done with devices. THE 3 i/p modes are ated by the devices. THE 3 i/p modes are referred to as request mode, sample mode & event mode respectively.

In 'request mode', the application Prog initiates data entry. I/p values are requested & processing is suspended until the request values are received.

In 'sample mode' the application Prog & i/p devices operate independently.

In 'event mode' the i/p devices initiate data into the application prog. the progs & the i/p devices again operate concurrently, but now the i/p devices deliver data to an the i/p queue. (diff b/w sample & event mode)

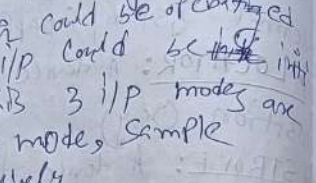
Color Concept model – (RGB, CMY color model)

Intuitive

RGB color model RGB C.m

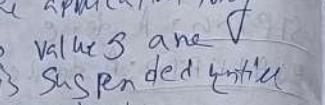
Why there are 3 colors always?

Acc to the properties of light (tri-stimulus theory of vision) the principle is that, our eyes possess only 3 receptors for colors. The 3 receptors are there for capturing the 3 colors & all others colors are the mixture of those (these 3 colors. (Red's λ = above 600 nm) [Red < Green < Blue] (wavelength)



(*)

(my C.m (cyan, magenta, yellow colors model)



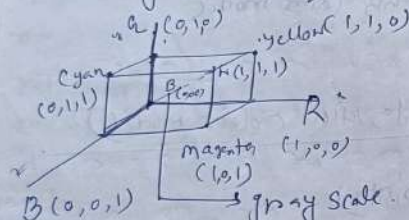
The cmy colors model, that stands for cyan, magenta, & yellow is a subtractive color model used primarily for printing. In the cmy model, cyan, magenta, & yellow are the primary colors, & red, green & blue are secondary colors. [Here colors are obtained by subtracting / masking colors from a white background.]

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} C \\ M \\ Y \end{bmatrix}$$

RGB color model pg 123

The RGB color model is one of the most widely used color representation method in CG. It uses a color coordinate system with 3 primary colors- R, G, B. [Each color can take an intensity value ranging from 0 to 1. Mixing this colors at diff intensity levels produces a variety of colors.] ||



Gray scale (pm/some) pg 123

It is a color model that uses diff shades of gray within an image. In digital formats, each pixel in a grayscale image has a value from 0 (black) to 255 (white).

At origin (0,0,0) we have white & diagonally opposite vertex at (1,1,1) we have black. This imaginary line connecting the white & black vertices forms the grey line.

[Locator device - It is an i/p device that helps users position objs on a screen go move, cursor in a text editor. Eg- mouse, a trackball, joystick etc.

Stroke device - this is used in interactive graphical techniques to enter coordinate infos. It is the outline/path that defines the shape of a graphic element such as line, curve etc.

String device - It is a logical i/p device that return strings to a prog. It provides ASCII values for i/p chars to the user prog.

Valuator device - It's an i/p device that returns a single value, such as a real value, to control an analog device. They are similar to locators, but only return 1 value. Eg- slide bars etc.

Choice device - It is a device that allows the user to select from a discrete no. of options. Eg- menus, widgets, scroll bars etc.

Pick device - It is an i/p device/method that allows the user to select an obj on a computer screen. It includes touch screens, light pens etc] ||

③

Sugg
Chap-1 13/5/24

① definations (what are the i/p, o/p primitives used for cg etc)
② Display devices & mechanism (imp)
③ CRT, Random scan, Raster scan
④ I/P devices (defination)
⑤ Line drawing algo numerical (Bresenham, DDA) [also - if to 9n not completed]
⑥ mid point circle drawing algo = algo (no derivation)

10m = FM internal exams (defination)
Q marks
30 = any 5
6 = Divided
40 = any 4
10 = Division

steps, eqn's, numericals (any octants: 2nd, 3rd etc)

chap-4 (attribute for o/p primitives)

(7) line, line attributes, joins, brush, curve attributes

(8) RGB, grey (just defination) side & s what is color grey

chap-5 (Imp) (max qns) [2D transformation]

(9) defination (transformation, func, uses)

(10) translation, rotation, their eqns, scaling (without homogeneous derivation, point translation co. system

(11) their numericals

(12) Homogeneous system = matrixs transformation with

(13) composite trans. - why it is needed?

(14) with respect to pivot point, without pivot point - how to do

(15) general pivot point rotation, general fixed point scaling - steps & egs.

(16) other trans. - numericals, reflection, sher (30°, 45°)

chap-6 (2D viewing)

(17) window / view port (defi)

(18) window - view port coordinate trans.

(19) clipping algos

(19) cohen Sutherland line clipping algo - line clipping

(20) polygon / area clipping - Sutherland hodgman polygon clipping.

structure line

(21) 9/5/24 - all (modes etc)

P1 & 2 3

1) Gr-A

1) ~~dt we~~

g) which is not an eg of emissive display?

LCD (liquid crystal display)

h) on raster system, lines are plotted with

Pixels

i) $d_1 < d_2$, then P_k is negative $\left[\begin{matrix} d_1 - d_2 < 0 \\ P_k < 0 \end{matrix} \right]$

j) Rotation is a rigid body transformation that moves obj's without deformation.

Gr-C

17) Perform 45 degree rotation of triangle A(0, 0), B(1, 1), C(5, 2) about P(-1, -1).

we know

$x' = x\cos\theta - y\sin\theta$
$y' = x\sin\theta + y\cos\theta$ } — ①

∴ from ①

$A' = (0, 0)$ [∵ A (0,0)]

B', $x' = 1 \cdot \cos 45 - 1 \cdot \sin 45 = 0$

$y' = \cos 45 + \sin 45 = 2 \times \frac{1}{\sqrt{2}} = 1 \cdot 4$ ∴ $B' = (0, 1 \cdot 4)$

C',

$x' = 5\cos 45 - 2\sin 45 = 2 \cdot 1$

$y' = 5\sin 45 + 2\cos 45 = 4 \cdot 9$

∴ $C' = (2 \cdot 1, 4 \cdot 9)$

Q8.B

i) magnify the triangle with vertices A(0,0), B(1,1) & C(5,2) to thrice its size while keeping the point C(5,2) fixed.

1st

translate the triangle by T(-5,-2) so that C(5,2) becomes (0,0).

∴ A(0,0) → (0-5, 0-2) = A'(-5,-2)

B(1,1) → (1-5, 1-2) = B'(-4,-1)

C(5,2) → (5-5, 2-2) = C'(0,0)

2nd

Scale the triangle by scaling factor = 2 [S(2,2)]

∴ A'(-5,-2) → A''(-10,-4) (left)

B'(-4,-1) → B''(-8,-2) (left)

C'(0,0) → C''(0,0)

3rd

translate back by T(5,2)

∴ A''(-10,-4) → A'''(-5,-2)

B''(-8,-2) → B'''(-3,0)

C'(0,0) → C(5,2)

End

translate

Q8.C

i) calculate the pixel positions along a line b/w A(10,10) & B(18,16) using Bresenham algo.

ii) $(x_0, y_0) = (10,10)$

iii) P_k or $P_0 = 2\Delta y - \Delta x$
$= 2(16-10) - (18-10)$
$= 2*6 - 8$
$= 12 - 8$
$= 4$

(from 1st it)

iv) $P_k > 0$, ∴ case (b) satisfied

∴ $x_{k+1} = x_k + 1 = x_0 + 1 = 11$

$y_{k+1} = y_k + 1 = y_0 + 1 = 11$

$P_{k+1} = P_k + 2\Delta y - 2\Delta x$
$= 4 + 2*6 - 2*8$
$= 4 + 12 - 16$
$= 0$

(∵ k starts from 0)

Δx = 8 times

	P_k	P_{k+1}	x_{k+1}	y_{k+1}	
			10	10	
1.	$4 > 0$	0	11	11	
2.	$0 = 0$	-4	12	12	
3.	$-4 < 0$	8	13	12	→ $P_{k+1} = P_k + 2\Delta y$, $x_{k+1} = x_k + 1$, $y_{k+1} = y_k$
4.	$8 > 0$	4	14	13	
5.	$4 > 0$	0	15	14	
6.	$0 = 0$	-4	16	15	
7.	$-4 < 0$	8	17	15	
8.	$8 > 0$	4	18	16	→ end point