DATE 07/08/23
PAGE

# Intro :- unit :- 1 megapixels = 10^6 pixels
FPS = frames Per second

09/08/2023

Computer Graphics :- Computer graphics are the graphics created using computer and the representation of image data by a computer specifically with help from specialized graphic hardware and software.

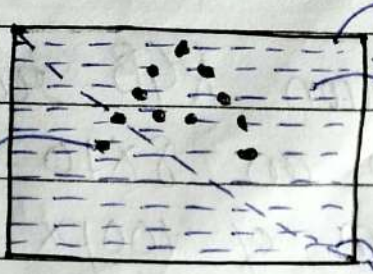
★ Uses of Computer graphics :-
↳ CAD (Computer aided design)
↳ Presentation graphics ↳ Computer art
↳ Entertainment industry ↳ Visualization
↳ Education & training ↳ GUI creation
↳ Image processing ↳ I/O devices

⇒ Cathode Ray Tube (CRT)

⇒ Aspect ratio = $\frac{\text{No. of horizontal line (Pixel)}}{\text{No. of vertical line (Pixel)}}$

• Persistance

⇒ Raster Scan :-

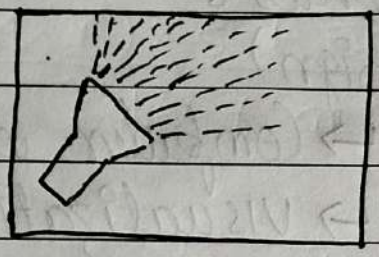


Line grow at some point when they needed

→ It also helps in refresh buffer (one to one mapping)

⇒ Interfaced Tracing System :- It traces alternatively like firstly it traces all odd line then even lines and then combine it.

⇒ Random Scan System :-



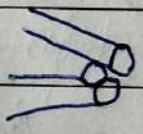
Information can store in display buffer in the form of commands.

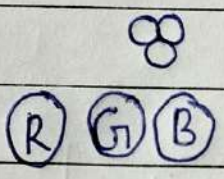
Color CRT monitor :-

→ Beam penetration (Used with Random scan monitors) → Shadow masking (used with raster scan system)

⇒ Two methods :- (in shadow masking)

↳ Delta-Delta method ↳ Inline method

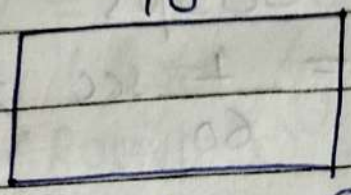




Ques) Suppose an RGB raster system is to be design on 8×10 inch screen with a resolution of 100px per inch in each direction. If we want to store 6 bits

per pixel in the frame buffer. How much storage in bytes do we need in the frame buffer.

Soln



$\Rightarrow$ 8 × 10 inches
{∵ 8 bits = 1 byte}

$\Rightarrow$ No. of pixels = 8 × 10 × 100 × 100

$\Rightarrow$ Total no. of bits = $\frac{8 \times 10 \times 100 \times 100 \times 6}{8}$

$\Rightarrow$ for one frame = 600000 ans//

Ques2) How long would it take to load a 640 × 480 frame buffer with 12 bits per px. If 10^5 bits can be transferred per s

Soln $\Rightarrow$ Total no. of bits = 640 × 480 × 12

$\Rightarrow$ 1 sec = 10^5 bits transferred.

then,

$\Rightarrow$ Time to load one frame buffer = $\frac{640 \times 480 \times}{10^5}$

$\Rightarrow$ 36.864 seconds ans//

10/08/2023

Ques3) What is the fraction of the total refresh time per frame spent in refresh of e⁻ beam for a non interlaced raster system with a resolution of 1280 × 1024. A refresh rate of 60 Hz a horizontal retrace time of 5 microseconds and a vertical

retrace time of 500 micro seconds

Sol^n ⇒ Total refresh time = Horizontal + vertical
= 500ps + 5ps
= 505ps

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

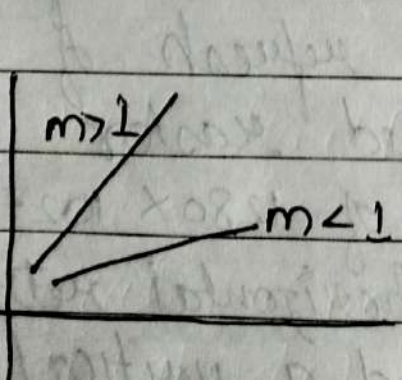
⇒ Horizontal Retrace time = (1024-1) × 5ps

⇒ Vertical Retrace time = (1024-1) × 5ps + 500ps

⇒ $\frac{(1024-1) \times 5 + 500ps}{16.7\ ms}$

Program to create a Pixel

```
#include <graphics.h>
void main()
{ int gd = DETECT, gm;
initgraph(&gd, &gm, "C:\\tc\\bg");
putpixel(x, y, RED);
   ↓10  ↓10
closegraph();
}
```



When m>1
y++
x = (y-c)/m

When m<1
x++
y = mx+c

```
#include <graphics.h>
#define ROUND(x) (int)(x+0.5)
  void main()
{ int gd = DETECT, gm;
  initgraph(&gd, &gm, "C:\\TC\\BGI");
  putpixel(ROUND(x), ROUND(Y), RED);
  closegraph()
}
```

```
#include <graphics.h>
  void main()
{ float x, y, m, c, i, x1, x2, y1, y2;
  scanf("%f %f %f %f", &x1, &x2, &y1, &y2);
  m = (y2-y1)/(x2-x1);
  c = y1 - m*x;
  x = x1; y = y1;
  if (m<=1)
{ for(i=x1; i<=x2; i++)
  {putpixel(ROUND(x), ROUND(Y), RED);
    x++;
    y= m*x + c;
} }
 else
{ for(i=y1; i<=y2; i++)
 {putpixel(ROUND(x), ROUND(Y), RED);
   y++;
  x = (y-c)/m;
```

$\Rightarrow (x_1, y_1)$ to (x_2, y_2)

$x = x_1 + (x_2 - x_1)u$

$y = y_1 + (y_2 - y_1)u$

where $0 \leq u \leq 1$

$u = u + du$

$du = \frac{1}{L}$ where $L = |dx| + |du|$

14/08/2023

Digital Differential analyser (DDA)

$\therefore y = mx + c$

$\Rightarrow$ if $m < 1$

$\Rightarrow \frac{dy}{dx} = m \Rightarrow dy = m dx$

$\Rightarrow y_{n+1} - y_n = m(x_{n+1} - x_n)$

for $m < 1$ $x_{n+1} = x_n + 1$

$y_{n+1} - y_n = m(x_{n+1} - x_n)$

$y_{n+1} = y_n + m$

$\Rightarrow$ if $m > 1$

$\Rightarrow y_{n+1} - y_n = (x_{n+1} - x_n)$

for $m > 1$ $y_{n+1} = y_n + 1$

$x_{n+1} = x_n + 1/m$

```
Void lineDDA (int xa, int xb, int ya, int yb)
{ int dx = xb - xa, dy = yb - yd, steps, k;
float xInc, yInc, x = xa, y = ya;
if (abs(dx) > abs(dy)) steps = abs(dx)
```

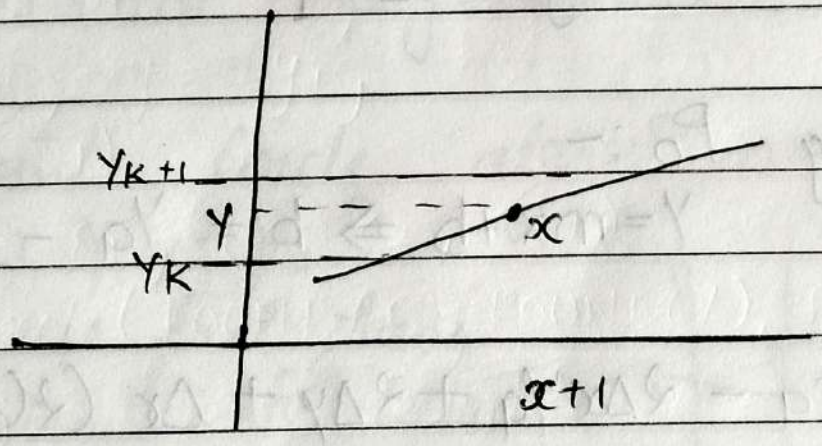
```
else steps = abs(dy),
xInc = dx/(float) steps;
yInc = dy/(float) steps;
putpixel (ROUND(x), ROUND(Y), RED);
for (K=0; K< steps, K++)
{ x+t = xInc;
  y+t = y Inc;
putpixel (ROUNID(X), ROUND(Y), RED);
}
```

16/08/2023

Bresenham's Line Drawing Algo

$x_{K+1} = x_K + 1$, $Y_{K+1} = Y_K + 1/Y_K$

★ This algo is also known as integer line draw algo.



⇒ if PK < 0

⇒ $(x_K + 1, Y_K)$

⇒ else

⇒ $(x_K + 1, Y_K + 1)$ {∵ $Y = mx + b$}

⇒ $d_1 = Y - Y_K$ ⇒ $m(X_K + 1) + b - Y_K$

⇒ $d_2 = (Y_K + 1) - Y$ ⇒ $Y_K + 1 - m(x_K + 1) - b$

$\Rightarrow P_k = \Delta x(d_1 - d_2) = (2m(x_k + 1) - 2y_k + 2b - 1) \times \Delta x$

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

$\{\because \text{where } c = 2\Delta y + \Delta x(2b-1)\}$

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

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

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

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

$\Rightarrow$ if $P_k < 0$ then $y_{k+1} = y_k$ and Points (x_{k+1}, y_k)

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

$\Rightarrow$ else then $y_{k+1} = y_k + 1$ and Points $(x_k + 1, y_k + 1)$

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

$\Rightarrow$ Calculating P_0 :-

$\Rightarrow (x_a, y_a)$ $\quad Y = mx + b \Rightarrow b = y_a - \frac{\Delta y}{\Delta x} x_a$

$\Rightarrow P_0 = 2\Delta y x_a - 2\Delta x y_a + 2\Delta y + \Delta x\left(2\left(y_a - \frac{\Delta y}{\Delta x} x_a\right)\right)$

$\Rightarrow \boxed{P_0 = 2\Delta y - \Delta x}$

28/08/23

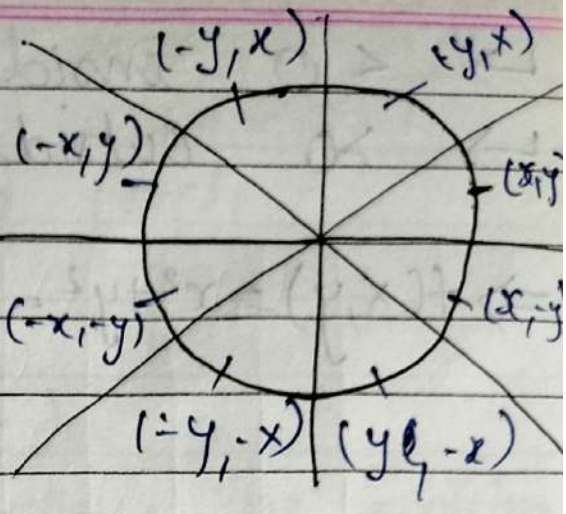
Program for drawing a circle

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

$\Rightarrow (x - x_c)^2 + (y - y_c)^2 = r^2$

$\Rightarrow x = \sqrt{r^2 - y^2}$

```
drawcircle (int x, int y,
            int xc, int yc)
{ putpixel (x+xc, y+yc, RED)
  putpixel (y+xc, x+yc, RED)
        :
}
```



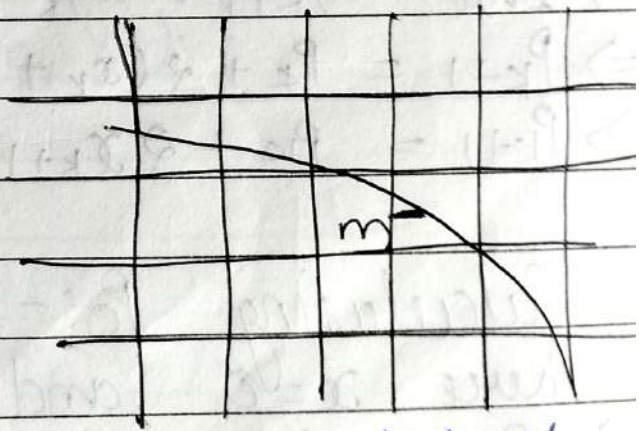

```
for (Y=0 ; Y<=x ; Y++)
{ x = sqrt(r*r - y*y)
```

⇒ another eqn by which we can create circle

$x = r \sin\theta$, $y = r \cos\theta$.

Mid point circle algorithm

$m(x_k+1, y_k - y_2)$



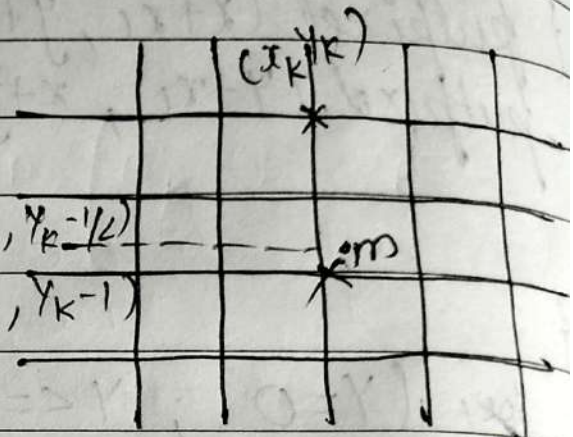
now if we put these coordinates into below eqn if $m<0$ then m will go inside and we will take the ~~value~~ previous value of y and if it is outside the we will subtract 1 from y.

eqn :- $f(x,y) = x^2 + y^2 - r^2$

↳ < 0 inside (x_k+1, y_k)

↳ > 0 outside (x_k+1, y_k-1)

$\Rightarrow f(x,y) = x^2+y^2 = r^2$



$\Rightarrow P_k = f(x_k+1, y_k-1/2) = (x_k+1)^2 + (y_k-1/2)^2 - r^2$ ——— (1)

$\Rightarrow P_{k+1} = f(x_{k+1}+1, y_{k+1}-1/2) = (x_{k+1}+1)^2 + (y_{k+1}-1/2)^2 - r^2$ ——— (2)

$\Rightarrow \because x_{k+1} = x_k + 1$

$\Rightarrow P_{k+1} = P_k + 2(x_k+1) + (y_{k+1}^2 - y_k^2) - (y_{k+1} - y_k) + 1$

$\Rightarrow$ If $P_k < 0$ then $y_{k+1} = y_k$, Points (x_k+1, y_k)

$\Rightarrow P_{k+1} = P_k + 2(x_k+1) + 1$

$\Rightarrow$ else $y_{k+1} = y_k - 1$, Points (x_k+1, y_k-1)

$\Rightarrow P_{k+1} = P_k + 2(x_k+1) - 2(y_k-1) + 1$

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

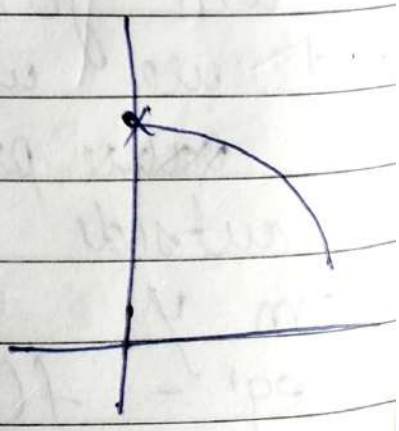
Calculating P_0:-

here $x=0$, and $y=r$

$\Rightarrow P_0 = f(0, r-1/2)$

$\Rightarrow P_0 = 1 + (r-1/2)^2 - r^2$

$\Rightarrow P_0 = \frac{5}{4} - r \simeq 1 - r$



$P_0 = 1 - r$

⇒ Algorith for drawing Circle :-

1. Read x_c, y_c, r
2. calculate, $P_0 = 1 - r$
3. draw Circle $(0, r, x_c, y_c)$
4. make a while loop where the condition will be $x != y$. or use for loop where $x = 0$; $x <= y$ and $x++$

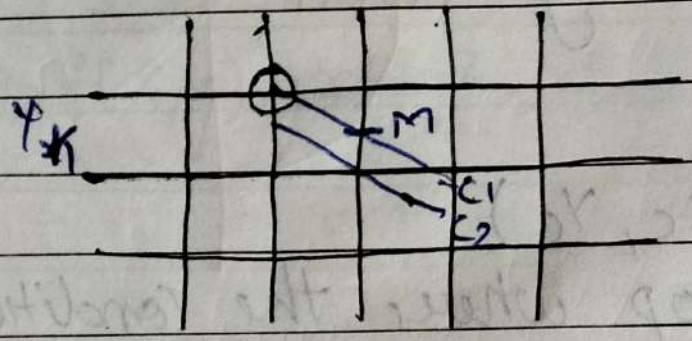
⇒ Program for drawing Circle

```
void midPoint (int xc, int yc, int r) {
int P = 1 - r , x = 0 , y = r ;
draw Circle ( x, y, xc, yc ){
for (x=0; x<=y; x++) {
if (P<0) {
P = P +2(x+1)+1 ;
x++ ;
drawCircle (x, y, xc, yc); }
else { y = y-1 ;
x++ ;
P = P+2 (x+1) -2(y-1)+1
drawCircle (x, y, xc, yc );
} }
```

05/09/2023

Putpixel (x+xc , y+yc , RED)
" (-x+xc, y+yc, ")

" $(-x + x_c, -y + y_c,$ "
" $(x + x_c, -y + y_c,$ "



$$m\left[x_k + 1, y_k - \frac{1}{2}\right]$$

x_k

$\Rightarrow f(x, y) = x^2 r_y^2 + y^2 r_x^2 - r_x^2 r_y^2$

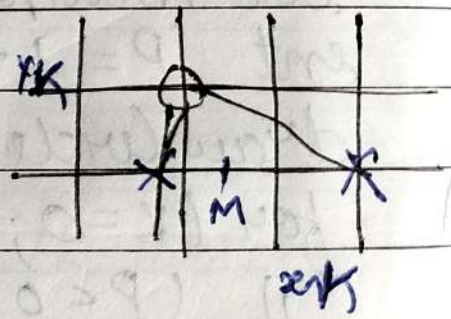
$\Rightarrow f(m) < 0 \quad (x_k + 1, y_k)$ } for R_1

$f(m) > 0 \quad (x_k + 1, y_k - 1)$ } region

for R_2 region

$f(m) < 0 \quad (x_k + 1, y_k - 1)$

$f(m) > 0 \quad (x_k, y_k - 1)$



$$m\left(x_k + \tfrac{1}{2}, y_k - 1\right)$$

Now parameter of decision

↳ decision parameters are

$$P1_k = f\left[x_k + 1, y_k - \frac{1}{2}\right] \Rightarrow (x_k + 1)^2 r_y^2 + \left(y_k - \frac{1}{2}\right)^2 r_x^2 - r_x^2 r_y^2$$

$$P'_{k+1} = f\left[x_{k+1}, y_{k+1} - \frac{1}{2}\right]$$

$$P1_{k+1} - P1_k$$

If $P1_k < 0$ then

$\Rightarrow P1_{k+1} = P1_k + 2r_y^2 x_{k+1} + r_y^2$

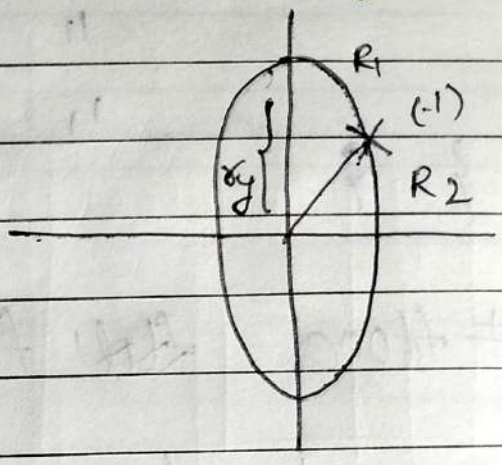
$= P1_k + 2r_y^2 (x_k + 1) + r_y^2$

$\because [x_{k+1} = x_k + 1]$

else

$P1_{k+1} = P1_k + 2r_y^2 x_{k+1} + r_y^2 - 2r_x^2 y_{k+1}$

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



Now,

Since $x = 0, Y = r_y$

then

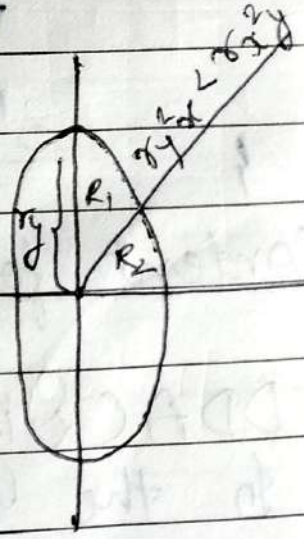
$f(0+1, r_y - 1/2)$

Now same for finding P_2 (decision parameter two)

$\Rightarrow$ for finding slope i.e m

$\frac{dy}{dx} = \frac{d(x^2 r_y^2 + y^2 r_x^2 - r_x^2 r_y^2)}{dx}$

$= \frac{2r_y^2 x}{2r_x^2 y} = -1$



DATE 06/9/23
PAGE

Boundary fill Algo

```
void fullBoundary (int x, int y, int fullC, int
                                    boundaryC)
{ int currentC = getPixel (x,y);
  if (currentC != boundaryC && currentC != fullC)
  { putpixel (x, y, fullC);
    fullBoundary (x-1, y, fullC, boundaryC);
                 (x+1, y,   "     "   );
                 (x, y+1,   "     "   );
                 (x, y-1,   "     "   );
  }}
```

flood fill Algo

```
void fillflood (int x, int y, int oldcolor, int fillcolor
{ if (getPixel (x,y) == oldcolor)
  { putpixel (x,y, fillcolor)
    fillflood (x-1, x oldcolor, fillcolor);
               (x+1, y,   "     "   );
               (x, y+1,   "     "   );
               (x, y-1,   "     "   );
}}
```

13/09/23

Content for Internal :-

① DDA (Digital Differential Analyser) :-
In the question we are given with the

Points of line

ex:- (1,1) and (3,3)

step 1:- calculate slope $(m) = \frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1}$

$\Rightarrow m = \frac{3-1}{3-1} = \frac{2}{2} \Rightarrow 1$ //

step 2:- find Δx and Δy

$\Rightarrow \Delta x = \frac{\Delta y}{m} = \frac{\Delta y \; (x_2 - x_1)}{(y_2 - y_1)}$ $\left\{ \because m = \frac{y_2 - y_1}{x_2 - x_1} \right\}$

$\Rightarrow \Delta x = x_2 - x_1 \Rightarrow 2$ //

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

$\Rightarrow \Delta y = 2$ //

step 3:- $|\Delta x| \geq |\Delta y|$, so assign $\Delta x = 1$

there are 2 cases

case 1:- if $|\Delta x| \geq |\Delta y|$ then assign $\Delta x = 1$

$\Rightarrow x_{i+1} = x_i + \Delta x$

$\Rightarrow \boxed{x_{i+1} = x_i + 1}$

$\Rightarrow y_{i+1} = y_i + \Delta y = y_i + m\Delta x$

$\Rightarrow \boxed{y_{i+1} = y_i + m}$

case 2:- if $|\Delta x| < |\Delta y|$ then assign $\Delta y = 1$

$\Rightarrow x_{i+1} = x_i + \Delta x = x_i + \frac{\Delta y}{m}$

$\Rightarrow \boxed{x_{i+1} = x_i + 1/m}$

$\Rightarrow y_{i+1} = y_i + \Delta y$

$\Rightarrow \boxed{y_{i+1} = y_i + 1}$

In this question we are having case 1 that why we have assigned $\Delta x = 1$

$\Rightarrow x_{i+1} = x_i + 1$ | $\Rightarrow y_{i+1} = y_i + m$

$\Rightarrow x_{i+1} = 1 + 1 = 2$// | $\Rightarrow y_{i+1} = 1 + 1 = 2$//

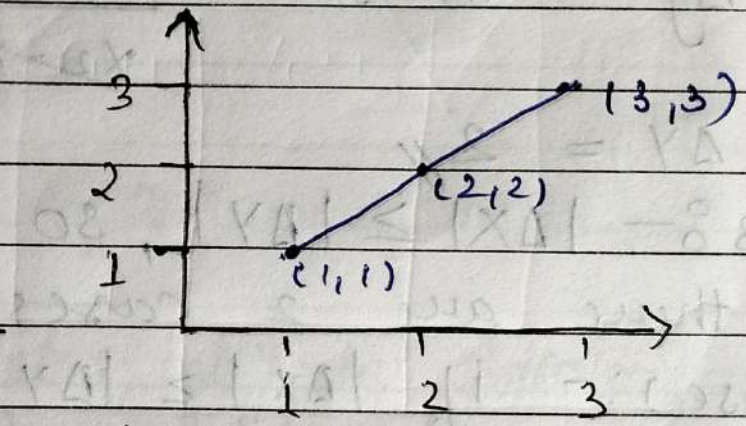
again calculating with the updated values of x_i and y_i

$\Rightarrow x_{i+1} = x_i + 1 = 2 + 1$ | $\Rightarrow y_{i+1} = y_i + m = 2 + 1$

$\Rightarrow x_{i+1} = 3$ | $\Rightarrow y_{i+1} = 3$

Now the points are (1,1), (2,2) and (3,3)

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



(12) Bresenham's line drawing algorithm

In DDA in most of the cases the points are in floating point which cause problem that why we had Bresenham's algorithm

ex:- (1, 1) and (5, 3)

Step 1:- calculate slope (m) $= \frac{\Delta Y}{\Delta X} = \frac{Y_2 - Y_1}{X_2 - X_1}$

$\Rightarrow m = \frac{3-1}{5-1} = \frac{2}{4} = \frac{1}{2} = 0.5 \Rightarrow \boxed{m = 0.5}$

Step 2:- Calculate Decision Parameter (P)

$P = 2\Delta Y - \Delta X$

Now there are 2 cases:-

case 1:- If $m < 1$	case 2:- If $m \geq 1$
(a) $P < 0$	(a) $P < 0$
$x_{i+1} = x_i + 1$	$x_{i+1} = x_i$
$y_{i+1} = y_i$	$y_{i+1} = y_i + 1$
$P_{k+1} = P_k + 2\Delta y$	$P_{k+1} = P_k + 2\Delta x$
(b) $P \geq 0$	(b) $P \geq 0$
$x_{i+1} = x_i + 1$	$x_{i+1} = x_i + 1$
$y_{i+1} = y_i + 1$	$y_{i+1} = y_i + 1$
$P_{k+1} = P_k + 2\Delta y - 2\Delta x$	$P_{k+1} = P_k + 2\Delta x - \Delta y$

$\Rightarrow$ Our $m = 0.5 < 1$ so we'll follow case 1

so now let us check the value of P

$\Rightarrow P = 2\Delta y - \Delta x = 2(Y_2 - Y_1) - (X_2 - X_1)$

$\Rightarrow P = 2(3-1) - (5-1) = 2(2) - 4 = 0$

$\Rightarrow \boxed{P = 0}$

now we will follow the (b) part of case 1 as our $P = 0$

$\Rightarrow$ Now we will calculate $x_{i+1}, y_{i+1}, P_{k+1}$

$\Rightarrow x_{i+1} = x_i + 1 = 1 + 1 \Rightarrow \boxed{x_{i+1} = 2}$

$\Rightarrow y_{i+1} = y_i + 1 = 1 + 1 \Rightarrow \boxed{y_{i+1} = 2}$

$\Rightarrow P_{k+1} = P_k + 2\Delta y - 2x = 0 + 2(2) - 2(4)$

$\Rightarrow P_{k+1} = 4 - 8 \Rightarrow \boxed{P_{k+1} = -4}$

Now $P < 0$ so (a) part will be followed

$\Rightarrow x_{i+1} = x_i + 1 = 2 + 1 \Rightarrow \boxed{x_{i+1} = 3}$

$\Rightarrow y_{i+1} = y_i \Rightarrow \boxed{y_{i+1} = 2}$

$\Rightarrow P_{k+1} = P_k + 2\Delta y \Rightarrow P_{k+1} = -4 + 2(2) = 0$ //

now $P=0$ so following parts (b)

$\Rightarrow x_{i+1} = x_i + 1 = 3+1 \Rightarrow \boxed{x_{i+1} = 4}$

$\Rightarrow y_{i+1} = y_i + 1 = 2+1 \Rightarrow \boxed{y_{i+1} = 3}$

$\Rightarrow P_{k+1} = P_k + 2\Delta y - 2\Delta x \Rightarrow 0 + 4 - 8$

$\Rightarrow \boxed{P_{k+1} = -4}$

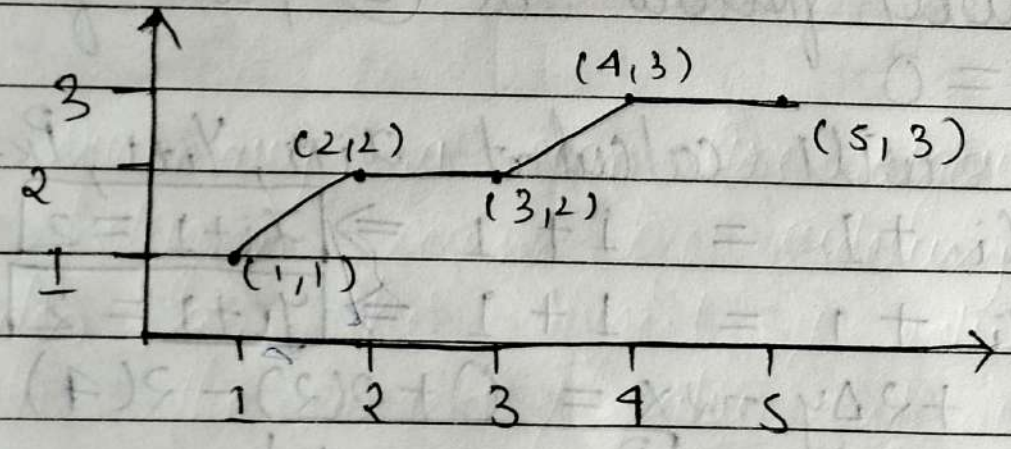
$P_K < 0$ so following part (a)

$\Rightarrow x_{i+1} = x_i + 1 = 4+1 = 5 \Rightarrow \boxed{x_{i+1} = 5}$

$\Rightarrow y_{i+1} = y_i = 3 \Rightarrow \boxed{y_{i+1} = 3}$

Our points are (1,1), (2,2), (3,2), (4,3) and (5,3)

P	x_i	y_i	x_{i+1}	y_{i+1}
0	1	1	2	2
-4	2	2	3	2
0	3	2	4	3
-4	4	3	5	3



(3) Mid point circle drawing algorithm

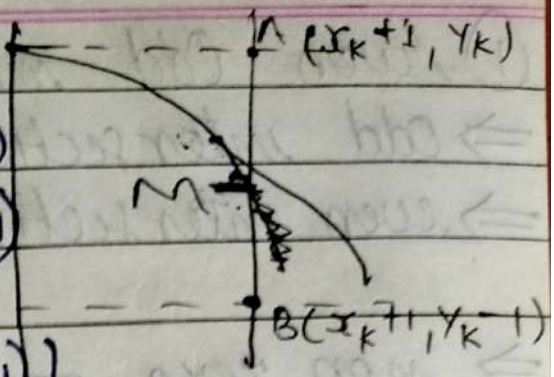
$\Rightarrow x^2 + y^2 = r^2 \rightarrow (1) \Rightarrow x^2 + y^2 - r^2 = 0 \rightarrow (2)$

point of $M = (x_m, y_m)$

$\Rightarrow (x_m)^2 + (y_m)^2 - r^2 \rightarrow (3)$

If the result of eqn (3) are

↳ 0 : Point lies on a circle
↳ <0 : Point lies inside (x_k+1, y_k)
↳ >0 : Point lies outside (x_k+1, y_k-1)



$\Rightarrow$ Midpoint coordinate $= \left[\frac{(x_k+1, x_k+1)}{2}, \frac{(y_k+y_k-1)}{2}\right]$

$= (x_k+1, y_k-1/2)$ ($\because$ dk = decision paramater)

$\Rightarrow d_k = (x_k+1)^2 + (y_k-1/2)^2 - r^2$ — (4)

$\Rightarrow d_{k+1} = (x_{k+1}+1)^2 + (y_{k+1}-1/2)^2 - r^2$

$\Rightarrow d_{k+1} - d_k$

after solving this we got

$\Rightarrow d_{k+1} = d_k + 2x_k + 3 + (y_k+1)^2 - y_{k+1} - y_k^2 + y_k$

- if $d_k < 0$ then $y_{k+1} = y_k$

$\Rightarrow d_{k+1} = d_k + 2x_k + 3$

- if $d_k > 0$ then $y_{k+1} = y_k - 1$

$\Rightarrow d_{k+1} = d_k + 2x_k - 2y_k + 5$

$\Rightarrow (0, r)$ IDP (d_0) {IDP = Initial decision Parameter

Put $(0, r)$ in eqn (4)

$\Rightarrow d_0 = (0+1)^2 + (r - 1/2)^2 - r^2$

$\Rightarrow d_0 = 5/4 - r$

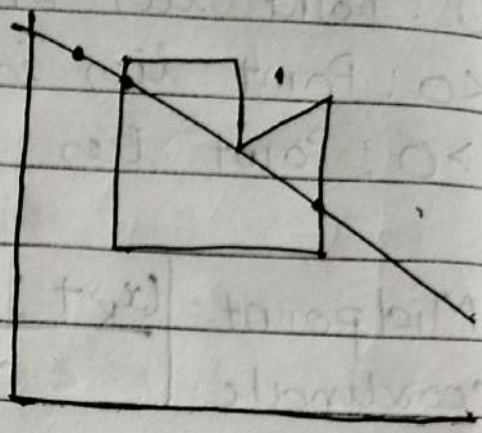
25/09/2023

How to find a point inside a polygon or outside a polygon

① Even Odd method

⇒ odd intersection = In

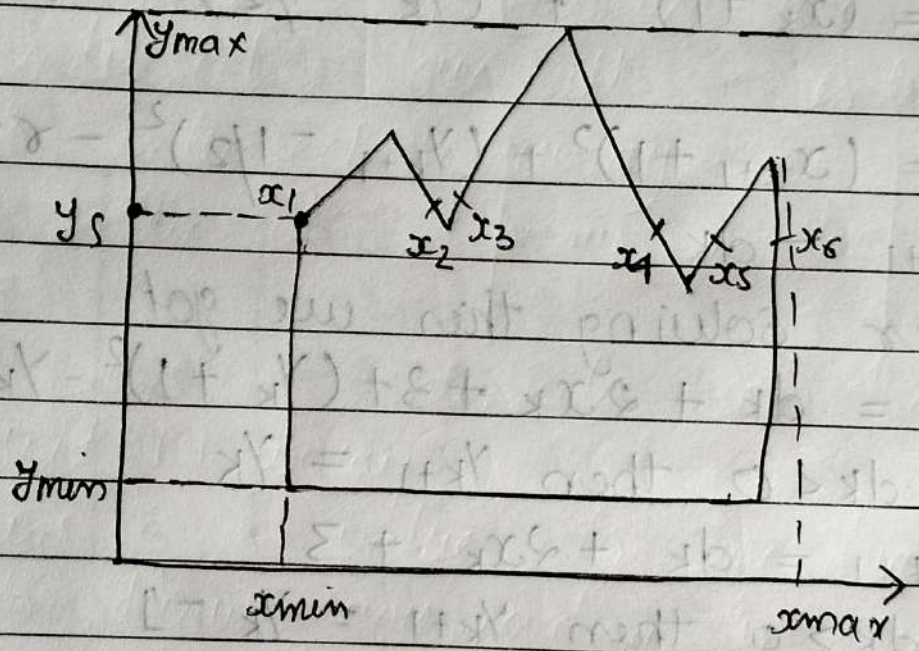
⇒ even intersection = Out



⇒ non zero = inside

⇒ zero = outside

Scale line Algorithm:-



In summary, a scan line algorithm for filling a polygon begins by ordering the polygon sides on the largest y value. It begins with the largest y value and scans down the polygon. For each y, it determines which sides can be intersected and finds the x values of these intersection points. It then sorts, pairs and passes these x values to a line drawing routine.

⇒ Scan line Conversion Algorithm for polygon filling :-

1. Read n, the number of vertices of polygon
2. Read x and y coordinates of all vertices in array x[n] and y[n]
3. find y_{min} and y_{max}
4. Store the initial x values (x_1) y values y_1 and y_2 for two endpoints and x increment Δx from scan line for each edge in the array edges [n] [4]

   while doing this check that $y_1 > y_2$ if not interchange y_1 and y_2 and corresponding x_1 and x_2 so that for each edge, y_1 represents its maximum y coordinate and y_2 represents it minimum y coordinate
5. Sort the rows of array, edges [n] [4] in descending order of y_1, descending order of y_2 and ascending order of x_2
6. Set $y = y_{max}$
7. find the active edges and update active edge list

   if ($y > y_2$ and $y \leq y_1$)

   edge is active

   else

   edge is not active
8. compute the x intersects for all active edges for current y values initially x intersects

is x_i and x intersect for successive y values can be given as

$x_{i+1} = x_i + \Delta x$

where $\Delta x = \frac{-1}{m}$ and $m = \frac{y_2 - y_1}{x_2 - x_1}$ i.e slop of a line segment

9. If x intersect is vertex i.e x-intersect $= x_1$ and $y = y_1$ then apply vertex list to check whether to consider one intersect or two intersect. Store all x intersect in the x-intersect [] array
10. Sort x-intersect [] array in the ascending order
11. Extract pairs of intersects from the sorted x-intersect [] array
12. Pass pairs of x value to line drawing routine to draw corresponding line segments
13. Set $y = y - 1$
14. Repeat steps 7 through 13 until $y \geq y_{min}$
15. Stop.

03/10/2023

2D Transformation

$x' = r\cos(\phi + \theta) \Rightarrow r\cos\phi\cos\theta - r\sin\phi\sin\theta$

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

$x = r\cos\phi$

$y = r\sin\phi$

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

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

$P' = P \times R$

↳ Scaling an Object :-

$x' = x \times Sx$

$y' = y \times Sy$

$[x'\ y'] = [x\ y] \cdot \begin{bmatrix} Sx & 0 \\ 0 & Sy \end{bmatrix}$

$P' = P \times S$

Sx and Sy are scaling factors.

Ques :- Scale the triangle with coordinati A (2,5) ; B (7,10) ; C (10,2) by 2 unit in x direction and 2 unit in y direction

Sol^n $P' = P \times S$

$$\begin{matrix} A' \\ B' \\ C' \end{matrix}\begin{bmatrix} x_1' & y_1' \\ x_2' & y_2' \\ x_3' & y_3' \end{bmatrix} = \begin{bmatrix} 2 & 5 \\ 7 & 10 \\ 10 & 2 \end{bmatrix} \times \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} = \begin{bmatrix} 4 & 10 \\ 14 & 20 \\ 20 & 4 \end{bmatrix}\begin{matrix} A \\ B \\ C \end{matrix}$$

New coordinates are :-

A (4, 10) ; B (14, 20) ; C (20, 4)

⇒ Homogeneous Coordinate

$[x_w \; y_w \; w]$

$x = \frac{x_w}{w}, \quad y = \frac{y_w}{w}$

$[x, y, 1]$

$$T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix}$$

$$R = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

P' = P x Transformable Matrix

$\Rightarrow P' = P \times T$

$\Rightarrow P' = P \times R$

$\Rightarrow P' = P \times S$

$$S = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$P' = P \cdot M_1 + M_2$

M_1 = Identity Matrix

M_2 = Translation Vector

$$P' = P\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$ translation

$$P' = P\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$ Rotation

$$P' = P\begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$$

Q refer previous Ques. But here we have to do ① $t_x = 5$, $t_y = 3$ and ② $S_x = 2$, $S_y = 1/2$

Sol° First calculate $CM = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 5 & 3 & 1 \end{bmatrix} \times \begin{bmatrix} 2 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

$\Rightarrow P' = P \times CM$

$$\begin{matrix} A' \\ B' \\ C' \end{matrix}\begin{bmatrix} x_1' & x_1' \\ x_2' & y_2' \\ x_3' & y_3' \end{bmatrix} = \begin{bmatrix} 2 & 5 & 1 \\ 7 & 10 & 1 \\ 10 & 2 & 1 \end{bmatrix} \times \begin{bmatrix} CM \end{bmatrix}$$

04/10/2023

Q find the transformation matrix that transform the square ABCD to half its size with centre still remaining at the same position. The coordinates of square are A(1,1); B(3,1); C(3,3); D(1,3) and center at (2,2) also find the resultant coordinat of square.

Sol° $CM = T^{-1}_{(-2,-2)} \times S_{(1/2\ 1/2)} \times T_{(2,2)}$

$$\Rightarrow CM = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -2 & -2 & 1 \end{bmatrix} \times \begin{bmatrix} 1/2 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 2 & 0 & 1 \end{bmatrix}$$

$$\Rightarrow CM = \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

$P' = P \times CM$

$$\begin{bmatrix} A' \\ B' \\ C' \\ D' \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 3 & 1 & 1 \\ 3 & 3 & 1 \\ 1 & 3 & 1 \end{bmatrix} \times \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1.5 & 1.5 & 1 \\ 2.5 & 1.5 & 1 \\ 2.5 & 2.5 & 1 \\ 1.5 & 2.5 & 1 \end{bmatrix}$$

A(1.5, 1.5) B(2.5, 1.5)
C(2.5, 2.5) D(1.5, 2.5) ans

Now rotate that square by 45°

$CM = T^{-1}_{(-2,-2)} \times R_{(45°)} \times T_{(2,2)}$

$$\Rightarrow CM = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -2 & -2 & 1 \end{bmatrix} \times \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ -1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 2 & 2 & 1 \end{bmatrix}$$

then Calculate CM and P'

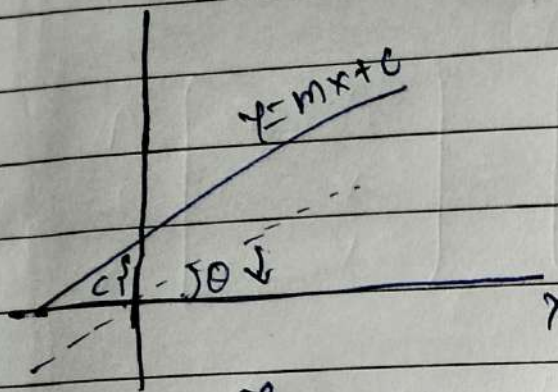
Q) find a transformation of △ A1,0 B0,1 C1,1

(a) Rotating 45° about origin then translating

1 Unit in x direction and y direction
(b) translate in 1 unit in x and y direction and then rotate 45° about the origin

05/10/2023

Reflection



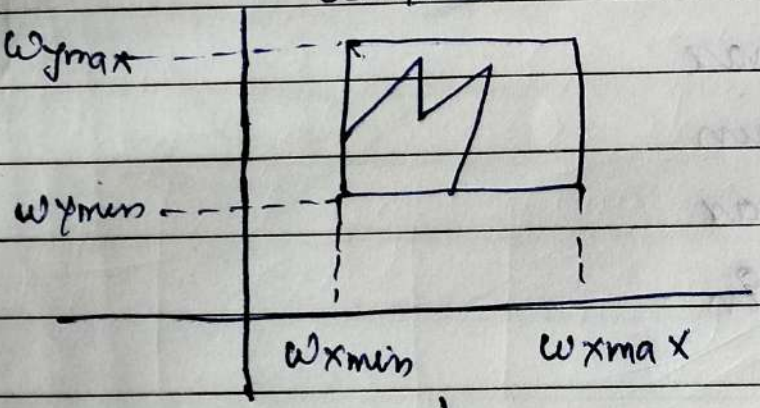
$m = \tan\theta$

$\theta = \tan^{-1} m$

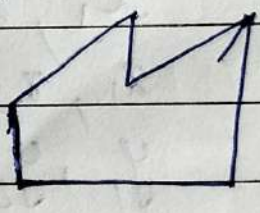
$CM = T(0, -c) \times R(-\theta) \times Ref_x \times R(\theta) \times T(0, c)$

Windowing and ~~Trimming~~ clipping

window

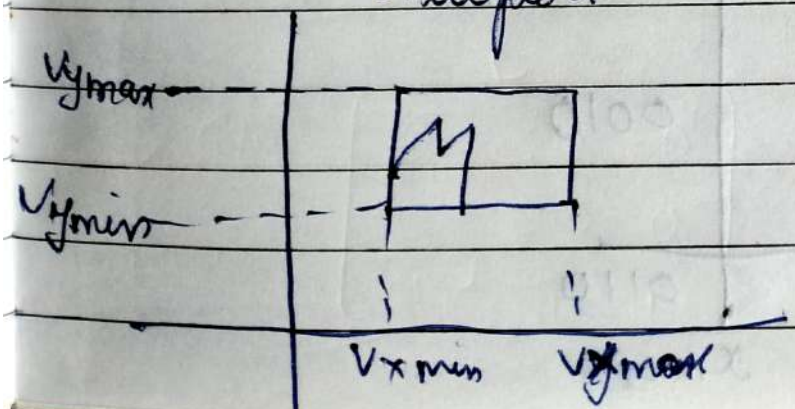


original image :-



↓

viewport



$$CM = T_{(-wx_{min} - wy_{min})} \times S_{(sx, sy)} \times T_{(vx_{min} - vy_{min})}$$

$$\Rightarrow sx = \frac{vx_{max} - vx_{min}}{wx_{max} - wx_{min}}$$

$$\Rightarrow sy = \frac{vy_{max} - vy_{min}}{wy_{max} - wy_{min}}$$

⇒ Point clipping :-

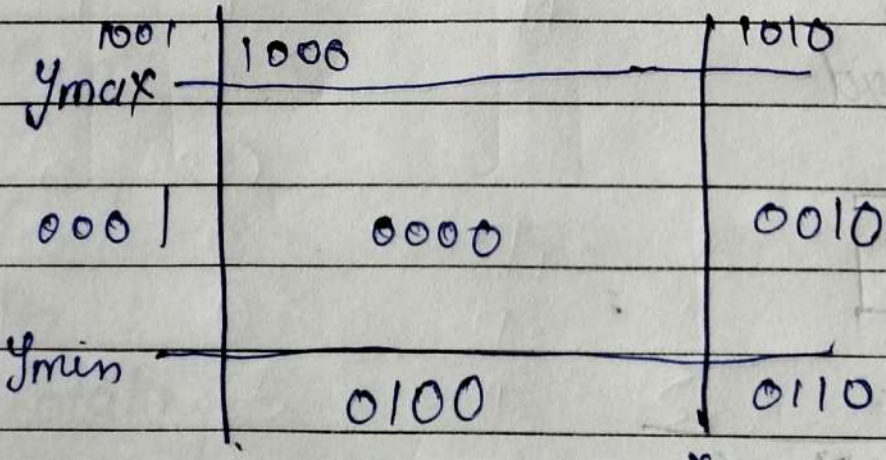
$$wx_{min} \leq x \leq wx_{max}$$
$$wy_{min} \leq y \leq wy_{max}$$

09/10/2023

Cohen Sutherland line clipping Algo

$x_1, x_2 > x_{max}$
$x_1, x_2 < x_{min}$
$y_1, y_2 > y_{max}$
$y_1, y_2 < y_{min}$

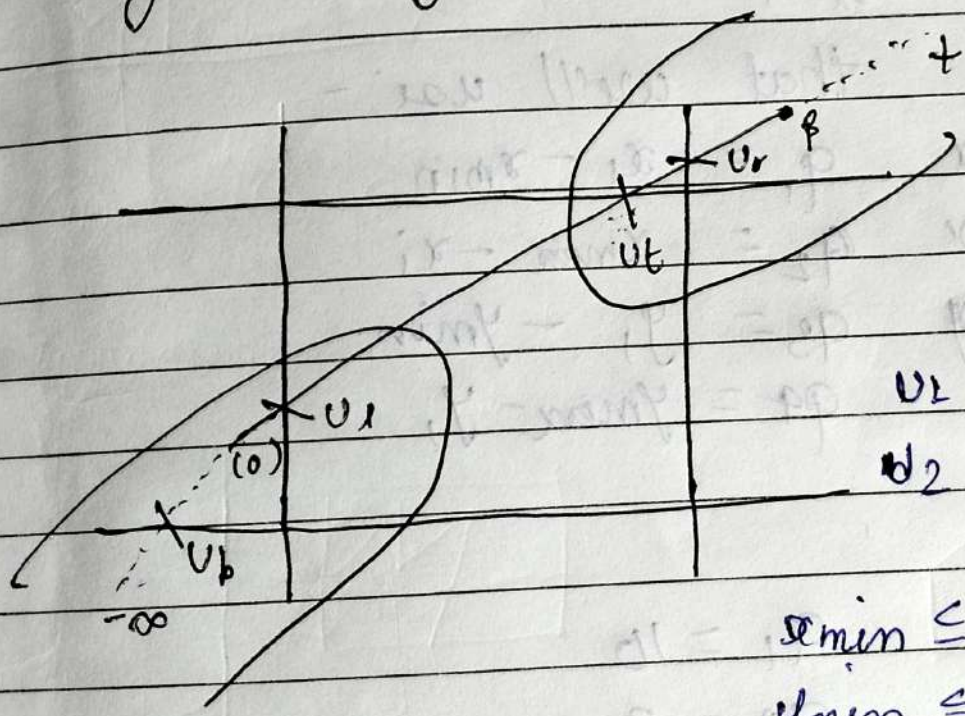
TBRL



Visible ⇒ 0000
Notvisible ⇒ Bitwise AND Non zero

$x_i = x_{min}$ or x_{max}
$y_i = y_1 + m(x_i - x_1)$
and
$x_i = x_1 + (y_i - y_1)/m$
$y_i = y_{min}$ or y_{max}

Liang Barsky Line clipping equation



$x = x_1 + \Delta x u$
$y = y_1 + \Delta y u$

v_1 max $(v_b, v_l, 0)$
v_2 min $(1, v_t, v_r)$

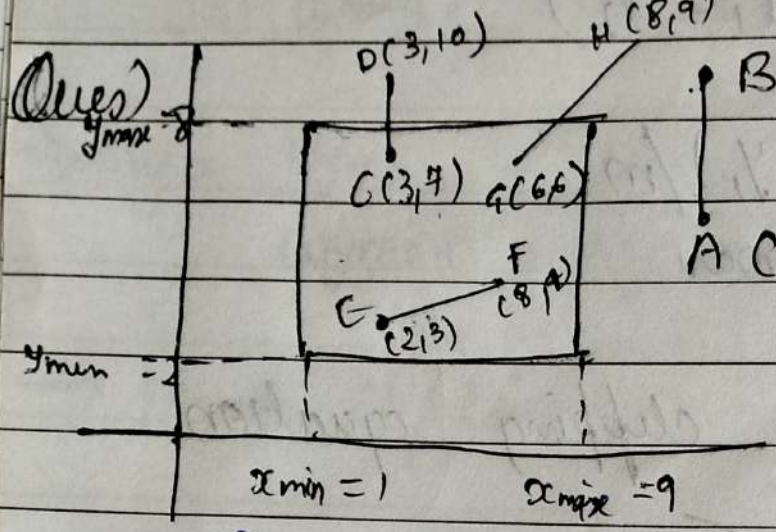
$x_{min} \le x_1 + \Delta x u \le x_{max}$
$y_{min} \le y_1 + \Delta y u \le y_{max}$

$p_k v \le q_k$ $\quad k = 1, 2, 3, 4$

$\Rightarrow p_1 = -\Delta x \quad q_1 = x_1 - x_{min}$ (left)
$\Rightarrow p_2 = \Delta x \quad q_2 = x_{max} - x_1$ (Right)
$\Rightarrow p_3 = -\Delta y \quad q_3 = y_1 - y_{min}$ (Bottom)
$\Rightarrow p_4 = \Delta y \quad q_4 = y_{max} - y_1$ (top)

$\Rightarrow -\Delta x \; 0 \leq x_1 - x_{min}$

$\Rightarrow \boxed{x_{min} \leq x_1 + \Delta x_4}$



Soln formula that we'll use:-

$p_1 = -\Delta x \quad q_1 = x_1 - x_{min}$

$p_2 = -\Delta x \quad q_2 = x_{max} - x_1$

$p_3 = -\Delta y \quad q_3 = y_1 - y_{min}$

$p_4 = \Delta y \quad q_4 = y_{max} - y_1$

$\Rightarrow$ for AB

$p_1 = 0 \quad q_1 = 10$

$p_2 = 0 \quad q_2 = -2$

$p_3 = -4 \quad q_3 = 4$

$p_4 = 4 \quad q_4 = 2$

$\Rightarrow$ for CD

$p_1 = 0 \quad q_1 = ?$

$p_2 = 0 \quad q_2 = 6$

$p_3 = -3 \quad q_3 = 5$

$p_4 = 3 \quad q_4 = 1$

Calculating r:- for P_3 because P_3 = -ve

$\Rightarrow r = \frac{s}{3} \Rightarrow u_1 = \max(0 \text{ and } -s/3) = 0$

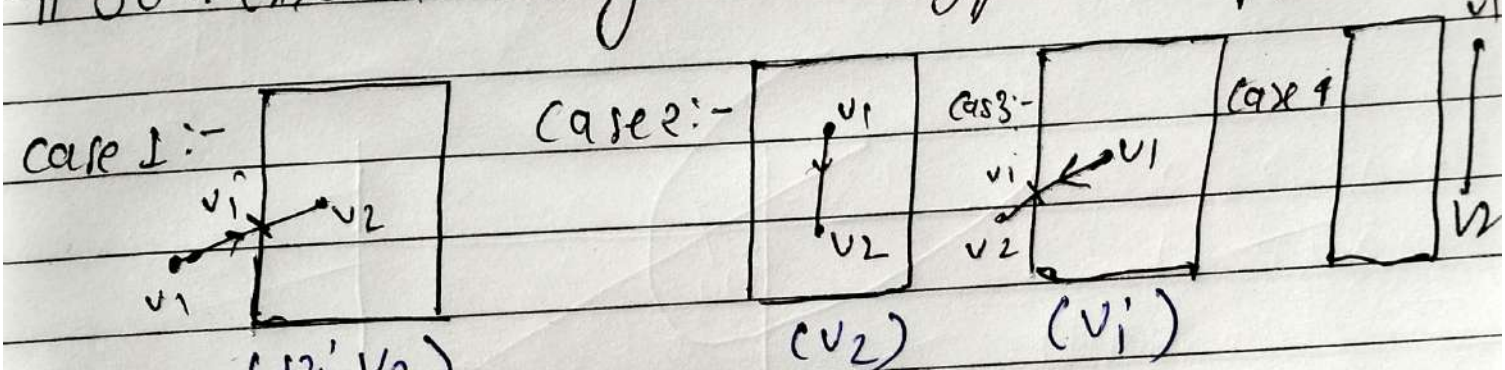
$\Rightarrow r = \frac{1}{3} \Rightarrow u_2 = \min(1 \text{ and } 1/3) = 1/3$

$\Rightarrow x = x_1 + \Delta x u \qquad y = y = y_1 + \Delta y u$

$\Rightarrow x = 3 + (3-3) \times 1/3, \quad y = 7 + 1/3 \times 3$

$\Rightarrow x = 3 \quad , y = 8$

Sutherland Hodgeman Polygon clipping



store

This is how we can create vertex list