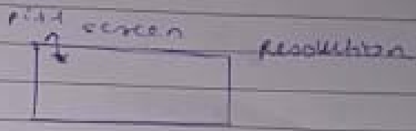
CEC12 : COMPUTER GRAPHICS



2D on a screen.

resolution

↳ no of pixel on X, no of pixel on Y.

pixel → smallest part as a graphic unit on a 2-D unit.

pixel

picture element.

C++

#include <graphics.h>

Primitives

x_1, y_1 x_2, y_2

10, 10   20, 18.

line → $y = mx + c$

$y_i = mx_i + c$

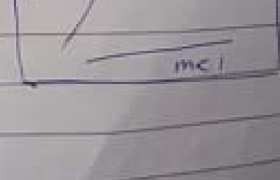
$m = \frac{8}{10} = <1$

$m<1$

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

$(x = \frac{y}{m} - \frac{c}{m})$

$y_{i+1} = m(x_{i+1}) + c$



$y_{i+1} = m(x_i) + m + c$

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

Coherence Property

$m > 1$

$\frac{\Delta x}{m}$ is constant.

$x_i = \frac{1}{m} y_i - c$

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

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

$x_{i+1} = \frac{1}{m} y_{i+1} - c$

$x_{i+1} = \frac{1}{m} y_i + \frac{1}{m} - c$

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

* line drawn from left to right / right to left

algorithm differ

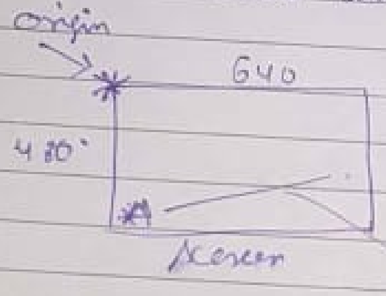
*

```
mline (x1, y1, x2, y2)
{
  dy = y2-y1 ;
  dx = x2-x1 ;
  if ( abs(dy) > abs(dx)          m > 1
   {
     y++ ;  x+= 1/m ;
   }
  else if ( abs(dy) < abs (dx)    m < 1
   {
     x++ ; y+= m ;
   }
```

```
else if ( dy==0)                          m= 0
  {
      x++;
  }
else if ( dx= 0 )                         m= ∞
  {
      y++;
  }
else                                      m= 1
  {
      x++; y++;
  }
}.

*
mline (x1, y1, x2, y2)
{
   dy = y2 - y1 ;
   dx = x2 - x1 ;
    m = dy/dx ;
  if ( abs (dy) < abs (dx) )
   {
       x = x1 ; y = y1 ;
       display ( x, y) ;
       while ( x <= x2 )
        {
           x++ ;
           y+= m ;
           display ( x, y) ;
        }.
}
```

```
if (abs(dy) > abs(dx))
{
    x = x1, y = y1;
    display (x, y);
    while (y <= y2)
    {
        y++;
        x += 1/m;
        display (x, y);
    }.
}.
```

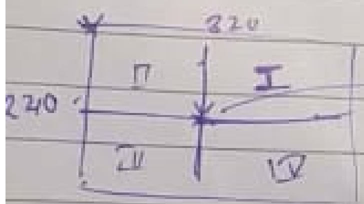
vertical, horizontal line, or 45° → smooth
but other → zig zag.
but if $m = 0.4$.
but locks are at integer so 3.4 etc are rounded off.



To draw this

display (x, 480-y)   at 0

$\underbrace{0, 480}$ → point A



→ display (320+x, 240-y)

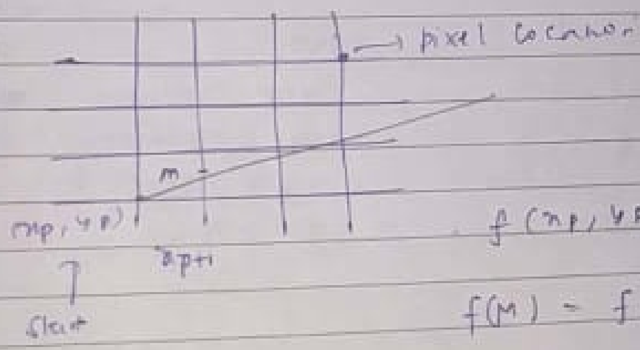
primitives → line, circle, ellipse, parabola, Hyperbola

floating point computation takes more time than integer computation

drawbacks in line-algo.
↳ $y + m$
floating point computation ⇒ slow.

$f(x,y) = ax + by + c$

value: < 0, $= 0$, > 0



m → midpoint

$f(x_p, y_p) = ax_p + by_p + c$

$f(M) = f(x_p + 1, y_p + 1/2)$

$f(M) = a(x_p + 1) + b(y_p + 1/2) + c$

- $= 0$ line from midpoint
- < 1 → ~~less than~~ below midpoint so lower coordin.
- > 1 → above mid point so above coordinate

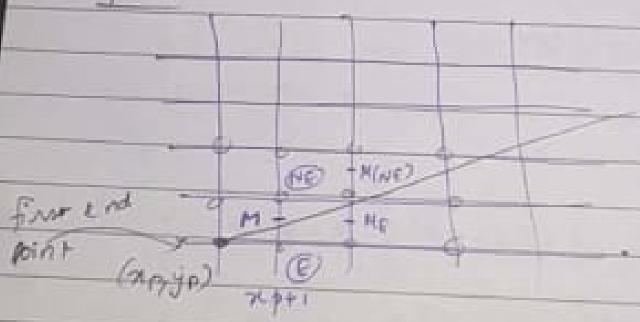
So no floating point computation is not used.

Input: x_1, y_1, x_2, y_2

$dx = x_2 - x_1$
$dy = y_2 - y_1$
$x = x_1$; $y = y_1$
putpixel (x, y, WHITE)
while ($x <= x_2$)
{
  $y += m$; → floating
  x++ ;
  putpixel (x, y, WHITE)
}

Mid point approach. (avoid floating point Computation)

Slope<1



Slope<1
major movement in x
as x increment
y is computed.
$y = mx + c$

$f(x, y) = ax + by + c$

$>0 \quad 0 \quad <0$

$f(M) = f(x_{p+1}, y_p + \frac{1}{2})$

$d = a(x_{p+1}) + b(y_p + \frac{1}{2}) + c$

(decision)
updated at every column

if $d < 0$

selection is E

$f(M_E) = f(x_p+2, y_p+\frac{1}{2})$

$d_{new} = a(x_p+2) + b(y_p+\frac{1}{2}) + c$

$\Delta E = f(M_E) - f(M)$

$= d_{new} - d$

$= a(x_p+2 - (x_p+1))$

$= a$

$y = mx + c$

$y = \frac{dy}{dx} x + c$

$dx y = dy.x + dx.c$

$dy.x - dx.y + dx.c = 0 \quad f(x,y)$

Company $f(x,y) = ax + by + c$

$a = dy$

$\Delta E = a = dy$

else

if $d > 0$.

selection is NE.

$f(M_{NE}) = f(x_p+2, y_p+\frac{3}{2})$

$= a(x_p+2) + b(y_p+\frac{3}{2}) + c$

$\Delta_{NE} = d_{new} - d$
$= a + b.$
$= dy - dx$

#

$d = f(x_p+1, y_p+1/2)$
$= a x_p + a + b y_p + b/2 + c$
$= \underbrace{a x_p + b y_p + c}_{0} + (a + b/2)$

$d_{in} \leftarrow a + b/2$
(initial value of d)

to remove b/2 (floats)
line has a property if it is multiplied by scale scaler quantity → same line

So we can start with $f(x,y) = 2(ax + by + c)$

$d_i = 2d \quad 2a + b$
$= 2dy - dx$

if $(d < 0)$
$\Delta E = 2dy$
else
$\Delta NE = 2 \times (dy - dx)$

Input : x_1, y_1, x_2, y_2

```
dx = x2 - x1
dy = y2 - y1
d = 2*dy - dx;
x = x1; y = y1

putpixel (x, y, WHITE);

while (x <= x2)
{
    if (d < 0)
        d + = 2*dy;
    else
    {
        d + = 2*(dy - dx);
        y++;
    }
    x++;
}
putpixel (x, y, WHITE);

}
```

#

$x_1 = 10$ $\quad$ $dx = 10$

$y_1 = 10$ $\quad$ $dy = 8$

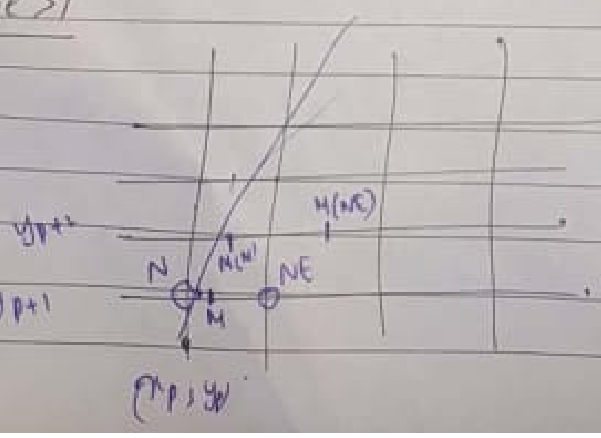
$x_2 = 20$ $\quad$ $d = 2 \times 8 - 10 = 6$

$y_2 = 18$

dry run

x	y	d	
10	10	6	
11	11	2	else
12	12	-2	else
13	12	14	if
14	13	10	else
15	14	6	else
16	15	2	else
17	16	-2	else
18	~~19~~ 16	14	if
19	17	10	else
20	18	6	else

Slope > 1



major movement in y.

Delta

At (x_p, y_p)

$f(x_p, y_p) = 2(ax_p + by_p + c)$

$d = f(M) = f(x_p + 1/2, y_p + 1)$.

$d = 2(ax_p + a/2 + by_p + b + c)$

$d = 2(ax_p + by_p + c + a/2 + b)$

$d = \cancel{2a+2b} \cdot a + 2b = dy - 2 * dx$

if $(d < 0)$.
selection is N.

$f(M_N) = f(\cancel{x_p} + 1/2, y_p + 2)$.

$d_{new} = 2(a(x_p + 1/2) + b(y_p + 2) + c)$

$\Delta N = d_{new} - d$
$= 2b = -2dx$

if $(d > 0)$
selection is NE

$f(M_{NE}) = f(x_p + 3/2, y_p + 2)$

$d_{new} = 2(a(x_p + 3/2) + b(y_p + 2) + c)$

$\Delta NE = d_{new} - d$
$= 2(a+b) = 2(dy - dx)$.

Scanned by CamScanner

Delta

input = x_1, y_1, x_2, y_2

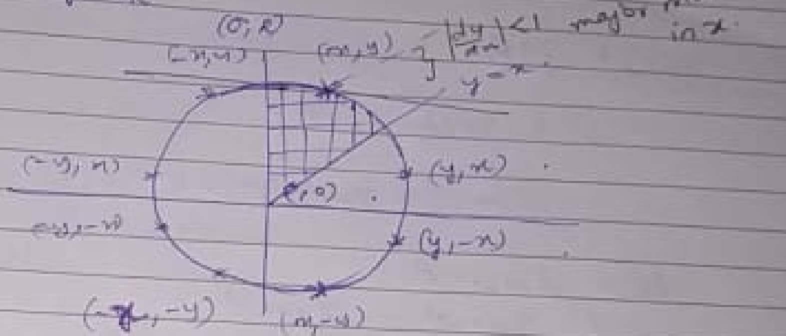
```
dx = x2 - x1
dy = y2 - y1
d = 2*dy - 2*dx

x = x1, y = y1;

put pixel ( x, y, WHITE);

while (y <= y2)
{
    if (d<0)
        d+ = - 2*dx
    else
    {
        d+ = 2*(dy-dx)
        x++;
    }
    y++;
}

putpixel (x, y, WHITE)

}
}.
```

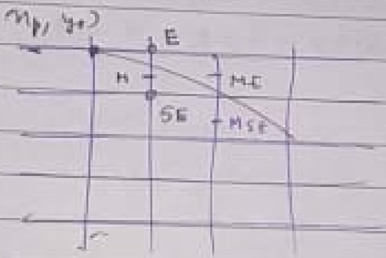
Scanned by CamScanner

Scan Conversion

Circle

$$x^2 + y^2 = R^2$$





At (x_p, y_p)

$$f(x_p, y_p) = x_p^2 + y_p^2 - R^2$$

$$d = f(M) = f(x_p + 1, y_p - \tfrac{1}{2})$$

$$= (x_p+1)^2 + (y_p - \tfrac{1}{2})^2 - R^2$$

$$= \underbrace{x_p^2 + y_p^2 - R^2} + 2x_p + 1 - y_p + \tfrac{1}{4}$$

$$= 2x_p - y_p + \frac{5}{4}.$$

$$(x_p, y_p) = (0, R)$$

$$d = \frac{5}{4} - R \approx 1 - R.$$

if (d<0)

select E

$d_{new} = f(M_E)$

$= f(x_p+2, y_p-\frac{1}{2})$

$= (x_p+2)^2 + (y_p-\frac{1}{2})^2 - R^2$

$\Delta E = d_{new} - d$

$= (x_p+2)^2 - (x_p+1)^2$

$= 4x_p - 2x_p + 4 - 3$

$\Delta E = 2x_p + 3$

else

select SE

$d_{new} = f(M_{SE})$

$= f(x_p+2, y_p-3/2)$

$\Delta E = d_{new} - d$

$= (x_p+2)^2 - (x_p+1)^2 + (y_p-\frac{3}{2})^2 - (y_p-\frac{1}{2})^2$

$= 2x_p+3 + (2y_p-2)(-1)$

$= 2x_p - 2y_p + 5$

$= 2(x_p-y_p) + 5$

```
x=0;
y=R;
d=1-R;

CircleSymmetry (x, y) // put pixel 8 times
                         for all cases

while (x<=y)          // y=x
  {
    if (d<0)
       d+=2*x+3;
    else
       {
          d += 2*(x-y)+5;
          y--;
       }
    x++;
    CircleSymmetry (x, y);
  }
```

R=8

x	y	d.
0	8	-7
1	8	-4
2	8	1
3	7	-6
4	7	~~5~~ 3
5	6	~~8~~ ~~6~~ 2
6	5	~~9~~ 5

Delta

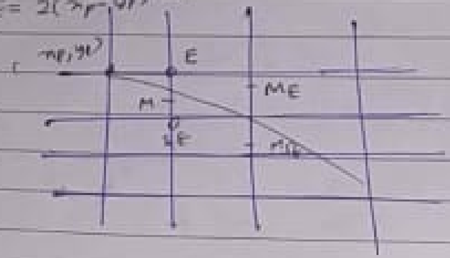
$\boxed{2^{nd} \text{ diff method}}$

from multiplication

why → 2* x → computation reduces to arithmetic ~~multiplication~~ addition

$\Delta E = 2x_p + ?$

$\Delta SE = 2(x_p - y_p) + 5$



A (x_p, y_p)

first differential are

$\Delta E = 2 * x_p + 3$

$\Delta SE = 2 * (x_p - y_p) + 5$

if $d<0$

selection is E.

At E

$(x_p + 1, y_p)$

$\Delta E_{new} = 2 * x_p + 5$

$\Delta SE_{new} = 2(x_p - y_p) + 7$

$\Delta^2 E^{E} = 2 \rightarrow \Delta E_{new} - \Delta E$

$\Delta^2 SE^{E} = 2 \rightarrow \Delta SE_{new} - \Delta SE$

Delta

else

selection is SE

At SE

(x_p+1, y_p-1)

$\Delta E_{new} = 2*x_p + 5$

$\Delta SE_{new} = 2*(x_p - y_p) + 9$

$\Delta^2 E^{SE} = 2$

$\Delta^2 SE^{SE} = 4$

Initialisation

$d = 1 - R$

$\Delta E = 3$ $(2x_p+3) \rightarrow (0,R)$

$\Delta SE = 5 - 2R$ $2(x_p - y_p) + 5 \rightarrow (0,R)$.

```
x = 0;
y = R;
d = 1-R;
deltaE = 3;
deltaSE = 5-2R;

CircleSymmetry (x, y);

while (x <= y)
{
```

Delta

```
if (d<0)
   { d+= deltaE;
     deltaE +=2;
     deltaSE += 2;
   }
else
   {
     d += deltaSE;
     deltaE +=2;
     deltaSE += 4;
   } y--;
   }
x++;
circleSymmetry (x, y);
}
```

R = 8

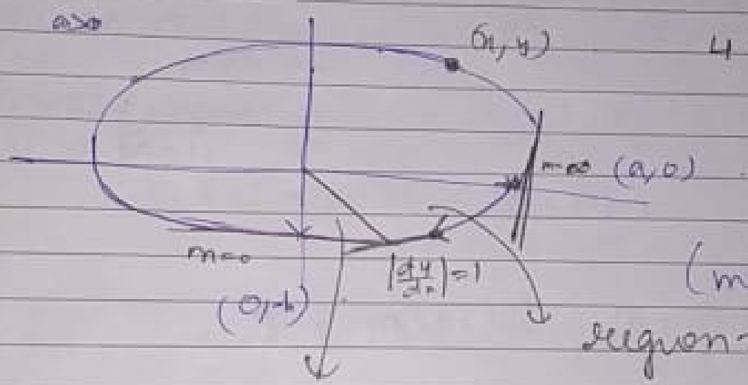
x	y	d	deltaE	deltaSE
0	8	-7	3	-11
1	8	-4	5	-9
2	8	1	7	-7
3	7	-6	9	-3
4	7	3	11	-1
5	6	2	13	3
6	5	5	15	7

Ellipse

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$$

$$f(x,y) = \frac{x^2}{a^2} + \frac{y^2}{b^2} - 1$$

$$f(x,y) = b^2x^2 + a^2y^2 - a^2b^2$$



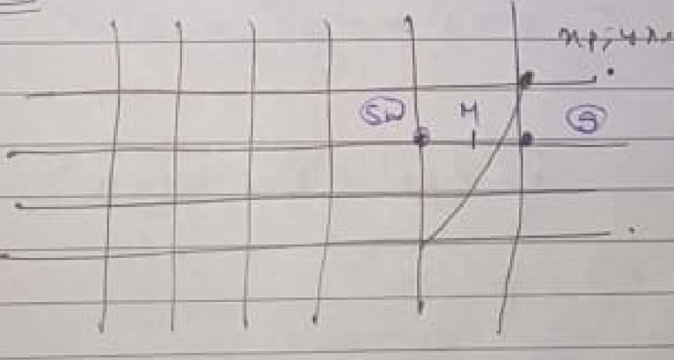
4 way symmetry

(major movement in y)

region I

region II
(major movement in x)

Region I



$$\cancel{d_i = f(x_p, y_p) = b^2x_p^2 + a^2y_p^2 - a^2b^2}$$

$$\cancel{d_i = b^2a^2 - a^2b^2 = 0}$$

Delta

if ~~(d<0)~~

$d = f(M) = f(x_p - 1/2, y_p - 1)$.

$d = b^2(x_p - 1/2)^2 + a^2(y_p - 1)^2 - a^2b^2$.

Gradient(E) $= \frac{\partial f}{\partial x} i + \frac{\partial f}{\partial y} j$.

if i component < j component → region 1

i comp = j comp slope is 1.

d_{new} = ~~if (d<0) selection is S~~

~~$\Delta S = d_{new} - d$.~~

~~$= b^2(x_p^2 + 1/4 - x_p) + a^2(y_p^2 + 1 - 2y_p) - a^2b^2$~~

~~$- b^2x_p^2 - a^2y_p^2 + a^2b^2$~~

~~$= \frac{b^2}{4} - x_p b^2 + a^2 - 2y_p a^2$.~~

if ($d<0$) selection is S

$d_{new} = b^2(x_p - \frac{1}{2})^2 + a^2(y_p - 2)^2 - a^2b^2$

$\Delta S = d_{new} - d_m$

$= a^2[(y_p - 2)^2 - (y_p - 1)^2]$

$= -a^2[(2y_p - 3)]$

else

selection is SW.

$d_{new} = b^2(x_p - 3/2)^2 + a^2(y_p - 2)^2 - a^2b^2$

$\Delta SW = d_{new} - d_m$

$= -a^2[2y_p - 3] + b^2[(x_p - \frac{3}{2}) - (x_p - \frac{1}{2})]$

$= -a^2[2y_p - 3] - b^2[2x_p - 2]$

$d_i = f(M) = b^2(x_p - \frac{1}{2})^2 + a^2(y_p - 1)^2 + a^2b^2$

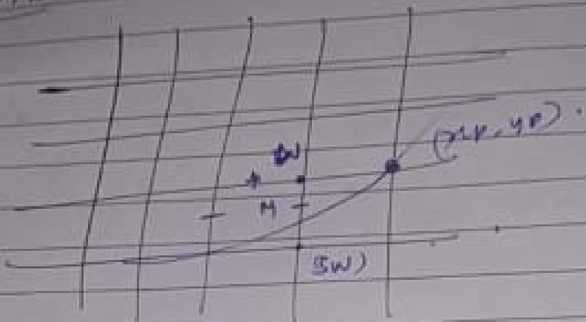
$x_p, y_p \rightarrow a, 0$

$b^2(a - \frac{1}{2})^2 + a^2 - a^2b^2$

$= \cancel{a^2b^2} - ab^2 + a^2 - \cancel{a^2b^2} + \frac{b^2}{4}$

$= a^2 - ab^2 + \frac{b^2}{4}$

Region 2



$f(m) = f(x_p - 1,\ y_p - \tfrac{1}{2})$

$d = f(m) = b^2(x_p-1)^2 + a^2(y_p - \tfrac{1}{2})^2 - a^2b^2$

If $(d<0)$

Selection is W

$d_{new} = f(x_p - 2,\ y_p - \tfrac{1}{2})$

$\Delta W = d_{new} - d$

$= b^2[(x_p-2)^2 - (x_p-1)^2]$

$= -b^2[2x_p - 3]$

Else

Selection is SW.

$d_{new} = f(x_p - 2,\ y_p - \tfrac{3}{2})$

Delta

$\Delta SW = d_{new} - d$

$= b^2[(x_p-2)^2 - (x_p-1)^2] + a^2[(y_p-3/2)^2 - (y_p-1/2)^2]$

$= -b^2[2x_p-3] - a^2[2y_p-2]$.

$\frac{\partial f}{\partial x} = 2b^2(x-1/2) \quad \frac{\partial f}{\partial y} = 2a^2(y-1)$

$\frac{\partial f}{\partial x} < \frac{\partial f}{\partial y}$.

~~if ($b^2x < a^2y$).~~
~~{~~

$d_i = a^2 - ab^2 + \frac{b^2}{4}$.

@ $x = a$; $y = 0$;
$d_i = a^2 - ab^2 + \frac{b^2}{4}$
ellipse symmetry (x, y);
while $(|b^2 x_{-1/2}| < |a^2 y_{-1}|)$
{
if $(d < 0)$
{
$d -= a^2[2y-3]$;
}
else
{
$d -= [a^2 2(y_p-3) + b^2(2x_p-2)]$.
& $x--$;
}
$y--$; ellipse symmetry (x, y); }.

Dt:
Pg:
Delta

~~Algo~~

$d_i =$

$d = b^2(x-1)^2 + a^2(y-1/2)^2 - a^2b^2$

while(~~...~~ x, y are of same sio

```
while (x>0)
{
    if (d<0)
    {
        d -= b^2[2x-3];
    }
    else
    {
        d -= [ b^2[2x-3] + a^2[2y-2
        y--;
    }
    x--;
}
ellipse symmetry (x, y);
```

Delta

2^{nd} derivative.

Region - I

At (x_p, y_p).

$\Delta S = a^2(3-2y_p)$
$\Delta SW = a^2[3-2y_p] + b^2[2-2x_p]$

if $d<0$

Selection is S.

$(x_p, y_p - 1)$

$\Delta S_{new} = a^2(5-2y_p)$.

$\Delta SW_{new} = a^2[5-2y_p] + b^2[2-2x_p]$.

$\Delta^2 S^{S} = +2a^2$.

$\Delta^2 SW^{S} = 2a^2$.

else

Selection is SW.

$x_p - 1, y_p - 1$

$\Delta S_{new} = a^2[5-2y_p]$
$\Delta SW_{new} = a^2[5-2y_p] + b^2[4-2x_p]$

$\Delta^2 S^{SW} = +2a^2$
$\Delta^2 SW^{SW} = 2a^2 + 2b^2$

Region 2

At (x_p, y_p)

$\Delta W = b^2(3-2x_p)$
$\Delta SW = b^2(3-2x_p) + a^2(2-2y_p)$

if $d<0$

selection is W

(x_p-1, y_p)

$\Delta W new = b^2(5-2x_p)$
$\Delta SW new = b^2(5-2x_p) + a^2(2-2y_p)$

$\Delta^2 W^{W} = 2b^2$
$\Delta^2 SW^{W} = 2b^2$

else

selection is SW

(x_p-1, y_p-1)

$\Delta W new = b^2(5-2x_p)$
$\Delta SW new = b^2(5-2x_p) + a^2(4-2y_p)$

$\Delta^2 W^{SW} = 2b^2$

$\Delta^2 SW^{SW} = 2b^2 + 2a^2$

x = a;
y = 0;

$d = a^2 - a b^2 + \frac{b^2}{4}$

dels = $3a^2$;
delsw = $3a^2 + 2b^2 - 2b^2 a$

while ellipse symmetry (x, y).

while (| $b^2(x - \frac{1}{2})$ | < | $a^2(y+1)$ | .
{.
if (d < 0)
{
d += dels;
dels += $2a^2$;
delsw += $2a^2$
}
else
{
d += delsw.
delsw += $2a^2 + 2b^2$; dels += $2a^2$;
x--;
}
y--
circle Ellipse symmetry (x, y).
}.

delw = $b^2(3 - 2x)$;
delsw = $b^2(3 - 2x) + a^2(2 - 2y)$;

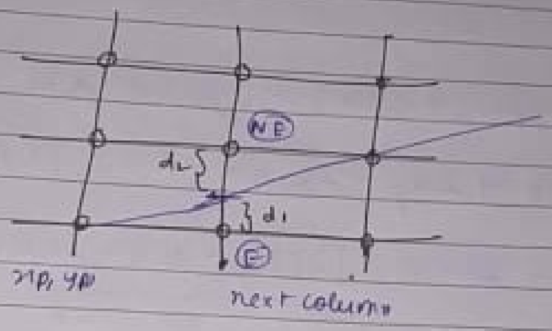
```
while (x>0)
{
    if (d>0)
      {
        d += delw ;
        delw+=2b²;
        delsw+=2b²;
      }
    else
      {
        d += delsw ;
        delw += 2b2,
        delsw += 2b² +2a² ;
        y--;
      }
   x--;
   Ellipse Symmetry (x,y),
}
```

Scanned by CamScanner

Bresenham's Approach

Line

AIM: To avoid floating pt. computation



$m < 1$

We need to find a measurable Quantity (not necessarily distance) → to figure out whether line is close to bottom or top pixel

at (x_p, y_p)

$y_p = mx_p + c$

for every primitive, we have a unique ~~solution~~ equation, similarly every primitive will have different measurable Quantity.

at $(x_p + 1)$

$y = m(x_p + 1) + c$

$d_1 = y - y_p$ — ①

$d_2 = (y_p + 1) - y$ — ②

decision → $d = d_1 - d_2$

$= 2y - y_p - y_p - 1$

if $d < 0$ → line near to bottom

Delta

$d = 2[m(x_p+1)+c] - 2y_p - 1$

floating point → substitute $\frac{\Delta y}{\Delta x}$

$d\Delta x = 2(\Delta y(x_p+1) + c\Delta x) - 2y_p\Delta x - \Delta x$

$d = 2\Delta y x_p + 2\Delta y - 2y_p\Delta x - \Delta x + 2c\Delta x$

$d = 2\Delta y x_p - 2\Delta x y_p + [\underbrace{2(\Delta y - \Delta x - 2c\Delta x}]$

constant as $\Delta x, \Delta y$ constant for line

$d = 2\Delta y x_p - 2\Delta x y_p + c$ — (3)

decision for next point

if $d < 0$
select E

$d_{new} = 2\Delta y(x_p+1) - 2\Delta x y_p + c$ — (4)

eq (4) - (3)

$\Delta E = 2\Delta y$ → Same as midpoint method.

else

select NE

$d_{new} = 2\Delta y(x_p+1) - 2\Delta x(y_p+1) + c$ — (5)

eq (5) - (3)

Scanned by CamScanner

$\Delta NE = 2(\Delta y - \Delta x)$

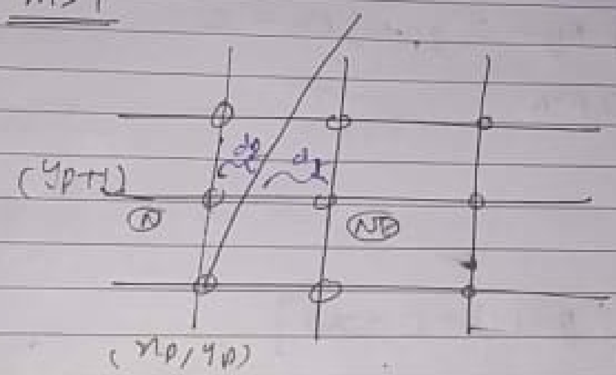
Initialisation

$d = 2[m(x_p+1)+C] - 2y_p - 1$

$d\Delta x = 2\Delta x[\cancel{m x_p - y_p + C}] + 2(\Delta y - \Delta x) \rightarrow 0$

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

m > 1



at (y_p+1)

$x = \frac{1}{m}[(y_p+1) - C]$

$d_1 = (x_p+1) - x$ — (1)

$d_2 = x - x_p$ — (2)

$d = d_1 - d_2$

$= 2x_p + 1 - 2x$

$= 2x_p + 1 - 2\frac{\Delta x}{\Delta y}[y_p + 1 - C]$

$d\Delta y = 2x_p\Delta y + \Delta y - 2\Delta x\, y_p - 2\Delta x + 2\Delta x C$

$d = 2x_p\Delta y - 2y_p\Delta x + C$

Delta

if $d<0$
select NE

$d_{new} = 2(x_p+1)\Delta y - 2(y_p+1)\Delta x + c$

$\Delta NE = 2(\Delta y - \Delta x)$

else
select N

$d_{new} = 2x_p\Delta y - 2(y_p+1)\Delta x + c$

$\Delta N = -2\Delta x$

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

$\Downarrow$

$$d = 2\left[-\frac{1}{m}(y_p+1) + c + x_p\right]$$

$$= 2\left[\frac{-1}{m}y_p + c + x_p\right] - \frac{2\Delta x}{\Delta y} + 1$$

$$d_i = 2x_p + 1 - 2x$$

$$= 2x_p + 1 - 2\left[\frac{1}{m}[y_p+1] - c\right]$$

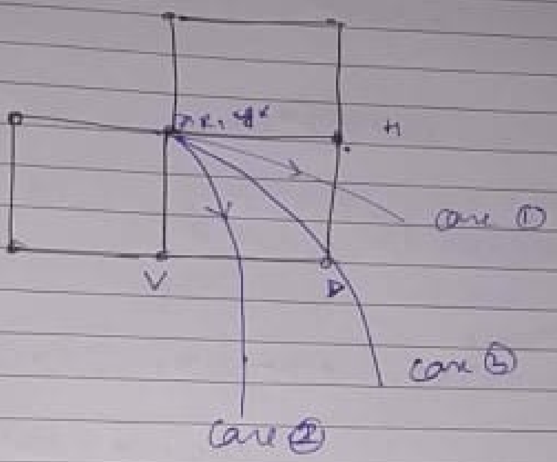
$$= 1 + 2\left[\cancel{x_p - \frac{1}{m}y_p + c} - \frac{1}{m}\right]$$

$$= 1 - \frac{2}{m} = 1 - \frac{2\Delta x}{\Delta y}$$

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

Circle

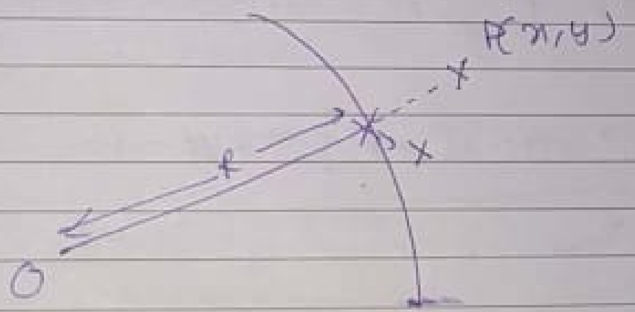
$x^2+y^2=R^2$



Measurable Quantity

quantity which will be >0/<0 depending on which point to choose.

i) proposing a measurable quantity of a circle



$$d_P = |OP^2 - OX^2|$$
$$= |(x+y)^2 - R^2|$$

$$d_H = |(x_i+1)^2 + y_i^2 - R^2|$$

$$d_V = |x_i^2 + (y_i-1)^2 - R^2|$$

$$d_D = |(x_i+1)^2 + (y_i-1)^2 - R^2|$$

Case I (passing above the diagonal)

$OD^2 - R^2 < 0$; $OD^2 < R^2$

Case II

$OD^2 - R^2 > 0$, $OD^2 > R^2$

PART I — selection b/w H and D

$$\delta_{HD} = d_H - d_D$$
$$= |(x_i+1)^2 + y_i^2 - R^2| - |(x_i+1)^2 + (y_i-1)^2 - R^2|$$

case 1 → $OD^2 < R^2$, $OH^2 > R^2$

$\delta_{HD} < 0$ → near to H

$$\delta_{HD} = (x_i+1)^2 + y_i^2 - R^2 - [(R^2 - (x_i+1)^2 - (y_i-1)^2]$$

$$\delta_{HD} = 2\underbrace{[(x_i+1)^2 + (y_i-1)^2 - R^2]}_{\Delta D_i} + 2y_i - 1$$

Part II — Selection b/w V and D

$$\delta_{VD} = d_V - d_D$$
$$= |OV^2 - R^2| - |OD^2 - R^2|$$

case 2 → ∵ $OD^2 > R^2$, $OV^2 < R^2$

$$= 2R^2 - OV^2 - OD^2$$
$$= 2R^2 - [x_i^2 + (y_i-1)^2 + [(x_i+1)^2 + (y_i-1)^2]$$
$$= 2[R^2 - (x_i+1)^2 - (y_i-1)^2] + 2x_i + 1$$

Delta

1) Pixel H is chosen

$x_{i+1} \rightarrow x_i + 1$
$y_{i+1} \rightarrow y_i$

$\Delta D_i = (x_i+1)^2 + (y_i-1)^2 - R^2$ — (1)

$\Delta D_{i+1} = (x_i+2)^2 + (y_i-1)^2 - R^2$ — (2)

eq2 - eq(1)

$$\boxed{\Delta D_{i+1} = \Delta D_i + 2x_i + 3}$$

2) Pixel V is chosen

$x_{i+1} \rightarrow x_i$
$y_{i+1} \rightarrow y_i - 1$

$\Delta D_{i+1} = (x_i+1)^2 + (y_i-2)^2 - R^2$ — (3)

eq (3) - (1).

$$\boxed{\Delta D_{i+1} = \Delta D_i - 2y_i + 3}$$

3) Pixel D is chosen

$x_{i+1} \rightarrow x_i + 1$
$y_{i+1} \rightarrow y_i - 1$

$\Delta D_{i+1} = (x_i+2)^2 + (y_i-2)^2 - R^2$ — (4)

Eq 4 - Eq 1 $\rightarrow$ $\boxed{\Delta D_{i+1} = \Delta D_i + 2(x_i - y_i) + 6}$

Delta

(Initialization)

$\rightarrow (0, R)$

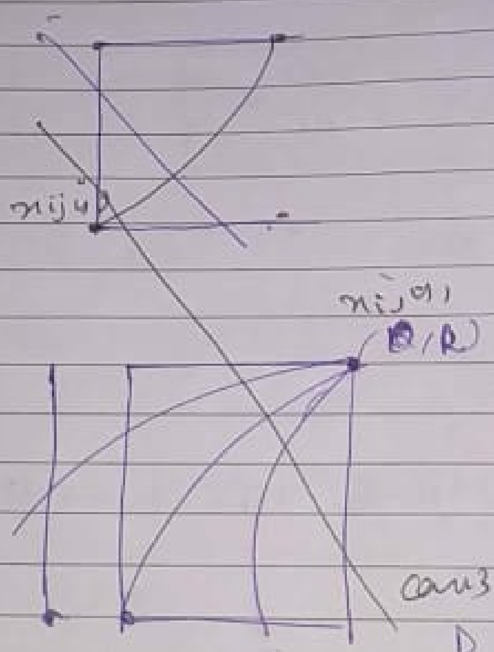
$$\Delta D_i = (x_i + 1)^2 + (y_i - 1)^2 - R^2$$

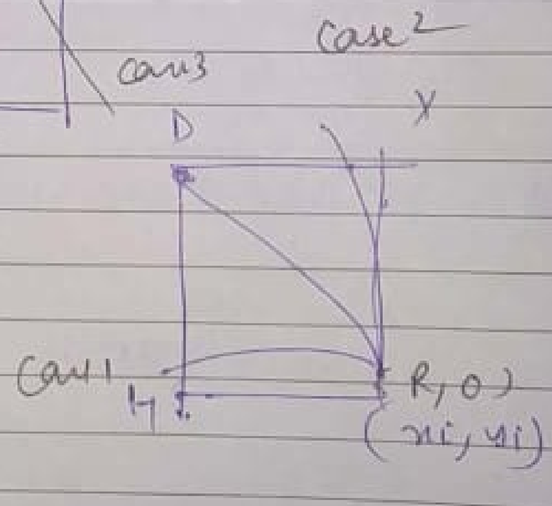
$$= (1)^2 + (R-1)^2 - R^2$$

$$= 1 + R^2 - 2R + 1 - R^2$$

$$= 2(1-R)$$

Circle - Bresenham's Anticlockwise





Delta

$d_H = |(x_i - 1)^2 + y_i^2 - R^2|$

$d_V = |(x_i)^2 + (y_i + 1)^2 - R^2|$

$d_D = |(x_i - 1)^2 + (y_i + 1)^2 - R^2|$

(Case I) passing below diagonal

$OD^2 > R^2$.

Case II

$OD^2 < R^2$.

(Part I) selection b/w H & D

$\delta_{HD} = d_H - d_D$

$= |(x_i - 1)^2 + y_i^2 - R^2| - |(x_i - 1)^2 + (y_i + 1)^2 - R^2|$

$OD^2 > R^2$
$OH^2 < R^2$

$\delta_{HD} = -(x_i - 1)^2 - y_i^2 + R^2 - (x_i - 1)^2 - (y_i + 1)^2 + R^2$

$\delta_{HD} = 2[R^2 - ((x_i - 1)^2 + (y_i + 1)^2)] + 2y_i + 1$

Part II selection b/w V and D.

$2y_i^2 \; 2y+1$

$\delta_{VD} = d_V - d_D$

$= |x_i^2 + (y_i + 1)^2 - R^2| - |(x_i - 1)^2 + (y_i + 1)^2 - R^2|$

$OD^2 < R^2$
$OV^2 > R^2$

Delta

$x_i^2 - 2x_i + 1$

$\delta_{VD} = x_i^2 + (y_i+1)^2 - R^2 + (x_i-1)^2 + (y_i+1)^2 - R^2$

$\delta_{VD} = 2\underbrace{((x_i-1)^2 + (y_i+1)^2 - R^2)}_{\Delta D_i} + 2x_i - 1.$

1) Pixel H is chosen

$x_{i+1} \rightarrow x_i - 1$

$y_{i+1} \rightarrow y_i$

$\Delta D_i = (x_i-1)^2 + (y_i+1)^2 - R^2$

$\Delta D_{i+1} = (x_i-2)^2 + (y_i+1)^2 - R^2$

$\Delta D_{i+1} = \Delta D_i - 2x_i + 3.$

2) Pixel V is chosen

$x_{i+1} \rightarrow x_i$

$y_{i+1} \rightarrow y_i + 1$

$\Delta D_{i+1} = (x_i-1)^2 + (y_i+2)^2 - R^2.$

$\Delta D_{i+1} = \Delta D_i + 2y_i + 3.$

3) Pixel D is chosen

$x_{i+1} \rightarrow x_i - 1$, $y_{i+1} \rightarrow y_i + 1.$

$\Delta D_{i+1} = (x_i-2)^2 + (y_i+2)^2 - R^2.$

$\cancel{\Delta D_{i+1} = \Delta D_i}$

$\Delta D_{i+1} = \Delta D_i + 2(y_i - x_i) + 6$

Initialization

$\Delta D_i = (x_i - 1)^2 + (y_i + 1)^2 - R^2$

$(R, 0)$

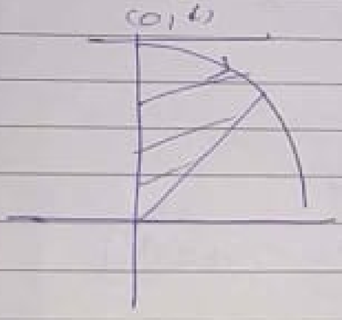
$= (R-1)^2 + 1 - R^2$

$= R^2 - 2R + 1 + 1 - R^2$

$= 2[1 - R]$.

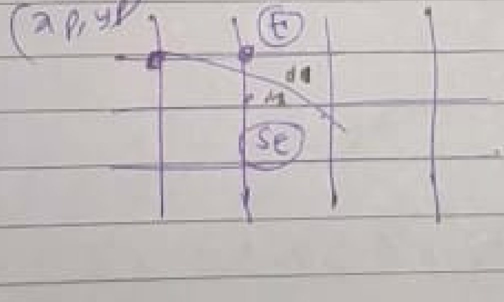
Bresenham's approach

circle (with 1st octant).



Looking into primitive eq, it is decided measurable Quantity.

1st octant major movement in x



for column position (x_i+1)

$d_1 = (x_p + 1)^2 - x^2$

$d_2 = x^2$.

Delta

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

$d_1 = y_p^2 - y^2$ — wrt reference to E

$d_2 = y^2 - (y_p - 1)^2$ — wrt reference to SE

$y^2 = r^2 - (x_p + 1)^2$

$D = d_1 - d_2$ (difference in measurable Quantity)

(decision)

$D < 0$ select E

$D > 0$ select SE

$D = d_1 - d_2$

$= y_p^2 - y^2 - y^2 + (y_p - 1)^2$

$= y_p^2 + (y_p - 1)^2 - 2y^2$

$= y_p^2 - 2[R^2 - (x_p + 1)^2] + (y_p - 1)^2$

$D < 0$ select E

$x_{p+1} \to x_p + 1$, $y_{p+1} = y_p$

$D_{new} = y_p^2 + (y_p - 1)^2 - 2[R^2 - (x_p + 2)^2]$

$\Delta E = D_{new} - D$

$= y_p^2 - 2R^2 + 2(x_p + 2)^2 + 2R^2 - 2(x_p + 1)^2$

$= 2[2x_p + 3]$

$D > 0$

select SE

$x_{p+1} \to x_{p+1}$, $y_{p+1} \to y_{p-1}$

Scanned by CamScanner

41-2-25

Date

$D_{new} = (x_p - 1)^2 + (y_p - 2)^2 - 2[r^2 - (x_p + 2)^2]$

$\Delta SE = D_{new} - D$

$= 2[2x_p + 3] + (y_p + 2)^2 - y_p^2$

$= 2[2x_p + 3] + -2[2y_p - 2]$

$= 2[2x_p + 3 - 2y_p + 2]$

$= 2[2x_p - 2y_p + 5]$

initially

$D(0, r) = -2[R^2 - (R+1)^2] + 1$

$= -2R^2 + 2(R+1)^2 + 1$

$= -2R^2 + 2R^2 + 4R + 2 + 1$

$= 4R + 3$

initially

$D(0, r) = R^2 - 2[R^2 - 1] + (R-1)^2$

$= R^2 - 2R^2 + 2 + R^2 - 2R + 1$

$= 3 - 2R$.

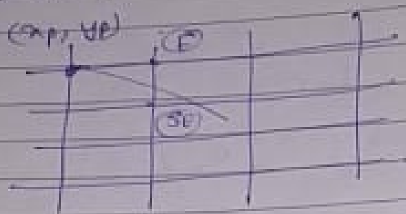
Ellipse. (Bresenham's approach)

$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$

$b^2x^2 + a^2y^2 = a^2b^2$

Circle 2nd Derivative

$d = 3 - 2R$



at (x_p, y_p)

$\Delta E = 2(2x_p + 3)$.
$\Delta SE = 2[2x_p + 3] + 4[1 - y_p]$

$(0, R)$

$\Delta E' = 6$
$\Delta SE' = 6 + 4 - 4R$
$\Delta SE' = 10 - 4R$

if $d < 0$,
selection is E.

i.e $x_{p+1} \rightarrow x_p + 1$
$y_{p+1} \rightarrow y_p$.

$\Delta E_{new} = 2[2x_p + 5]$
$\Delta SE_{new} = 2[2x_p + 5) + 4[1 - y_p]$

$\Delta^2 E_{(E)} = 4$
$\Delta^2 SE_{(E)} = 4$

else $d > 0$

selection is SE.

i.e. $x_{p+1} \rightarrow x_p + 1$

$y_{p+1} \rightarrow y_p - 1$.

$\Delta E_{new} = 2[2x_p + 5]$

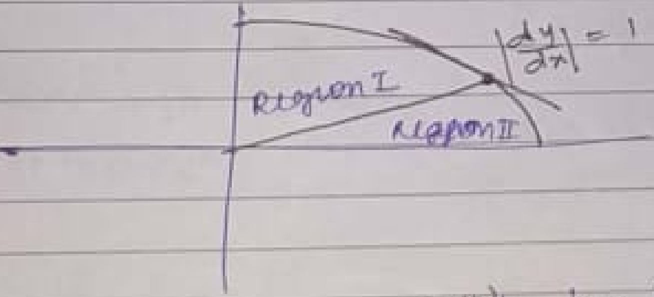
$\Delta SE_{new} = 2[2x_p + 5] + 4[2 - y_p]$

$\Delta^2 E$ (SE) $= 4$

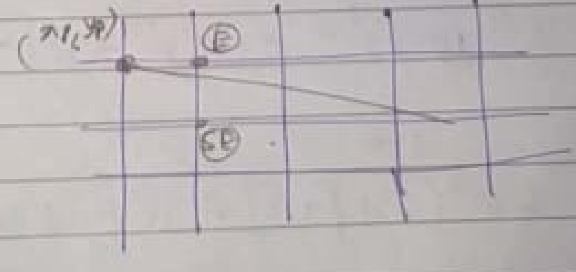
$\Delta^2 SE$ (SE) $= 4 + 4 = 8$.

Ellipse – Bresenham's approach

$b^2x^2 + a^2y^2 = a^2b^2$



Region I →



major movement in x

wrt E $\quad d_1 = a^2y_p^2 - a^2y^2$

$a^2y^2 = a^2b^2 - b^2(x_i+1)^2$

Delta

w.r.t (SE) $d_1 = a^2y^2 - a^2(yp-1)^2$

$D^{[1]} = d_1 - d_2$

$xp+1 \to xp+1$
$yp+1 \to yp$

$D<0$ Selection is (E).

$D = a^2yp^2 - a^2y^2 - a^2y^2 + a^2(yp-1)^2$

$D = a^2yp^2 - 2[a^2b^2 - b^2(xp+1)^2 + a^2((yp-1)^2]$

$D_{new} = a^2yp^2 - 2[a^2b^2 - b^2[xp+2]^2] + a^2[(yp-1)^2]$

$\Delta E = D_{new} - D$

$= 2b^2(xp+2)^2 - b^2(xp+1)^2$

$= 2b^2[2xp+3]$

$= 2b^2[2xp+3]$

else if $D>0$. selection is (SE)

$xp+1 \to xp+1$
$yp+1 \to yp-1$

~~Dnew~~

$D_{new} = a^2(yp-1)^2 - 2[a^2b^2 - b^2(xp+2)^2] + a^2(yp-2)^2$

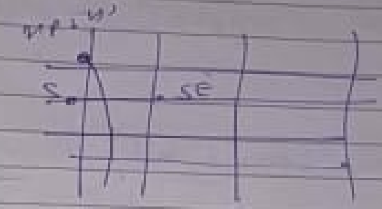
$\Delta SE = 2b^2(2xp+3] + a^2[\cancel{(yp-1)^2} + (yp-2)^2 - yp^2 - \cancel{(yp-1)^2}]$

$= 2b^2(2xp+3] + a^2(-2)[2yp-2]$

$\Delta SE = 2b^2(2xp+3) + 4a^2[1-yp]$.

Region II

$b^2x^2 = a^2b^2 - a^2(yp-1)^2$



$d_1 = -b^2xp^2 + b^2x^2$

$d_2 = -b^2x^2 + b^2(xp+1)^2$.

$D^{II} = d_1 - d_2$

$D^{II} = -b^2xp^2 + b^2(xp+1)^2 + 2b^2x^2$

$D^{II} = 2b^2x^2 + b^2[(xp+1)^2 - xp^2]$

$D^{II} = 2[a^2b^2 - a^2(yp-1)^2] + b^2[(xp+1)^2 + (xp)^2]$

if $D < 0$,
selection is S.

$xp+1 \to xp$
$yp+1 \to yp-1$

$D_{new} = 2[a^2b^2 - a^2(yp-2)^2] + b^2((xp+1)^2 + (xp)^2]$

$\Delta S = -2a^2[(yp-2)^2 - (yp-1)^2]$

$\Delta S = +2a^2[2yp-3]$

Delta

if $D > 0$ selection is SE

$x_{p+1} \rightarrow x_p + 1$
$y_{p+1} \rightarrow y_p - 1$

$D_{new} = 2[a^2b^2 - a^2(y_p - 2)^2] - b^2((x_p+2)^2 + (x_p+1)^2]$

$\Delta SE = D_{new} - D$

$= \cancel{2[\, 2a^2(2y_p - 3] - b^2[(x_p+2)^2 - (x_p+1)^2 + (x_p+1)^2 - (x_p)^2]}$

$= 2a^2(2y_p - 3] - b^2[(x_p+2)^2 + (x_p+1)^2 - (x_p+1)^2 - (x_p)^2]$

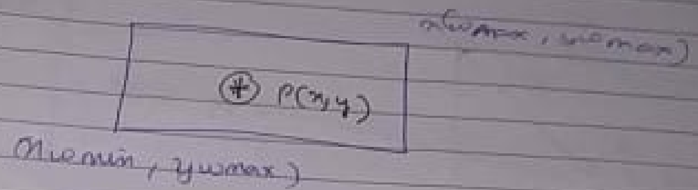
$= 2a^2(2y_p - 3] - b^2[2(2x_p + 2]$

$= 2a^2(4y_p - 3] - 4b^2[x_p + 1]$

Clipping

Delta

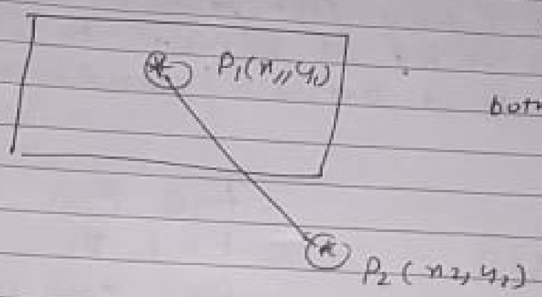
rectangular window



$xwmin \le x \le xwmax$
$ywmin \le y \le ywmax$

Any pt will be inside of the window

Pts inside will satisfy these 4 inequalities



both pts outside

0	0
0	1
1	0
1	1

clipping required (0 1, 1 0)

both points inside

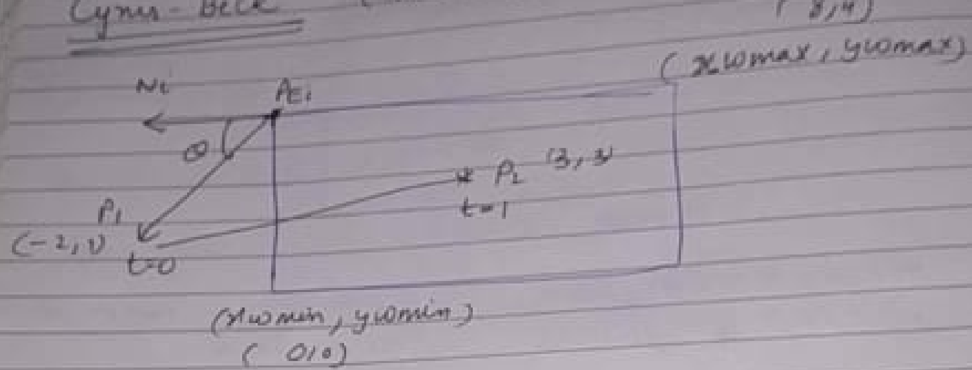
clipping

Line Clipping

Clip the line if one point is inside & one outside

```
int inside ( )
{
  // outside
  return 0
  // inside
  return 1
}
```

Cyrus-Beck (vector)



Take a first edge (say left edge) & drop a vector from a vertex of window to line → it will tell if it is outside / inside wrt that left edge.

$$P(t) = P_1 + (P_2 - P_1)\,t$$

$t=0$ P_1
$t=1$ P_2
$P(t)$ any point on a line variable with t

$$\vec{V} = [P(t) - P_{E_i}]$$

$\vec{V} \cdot \vec{N_i}$ → <0, $=0$, >0

positive means point outside ($\theta < 90$)
-ve means inside
0 means on Edge

If points on different sides, we need to find intersection.

Delta

At intersection

$\overline{v}\cdot\overline{N_i} = 0$

$(P(t) - P_{E_i}]\cdot\overrightarrow{N_i} = 0$

$[P_1 + (P_2 - P_1)t - P_{E_i}]\cdot\overrightarrow{N_i} = 0$

$N_i \cdot (P_1 - P_{E_i}) + \overrightarrow{N_i}\cdot(P_2 - P_1)t = 0$

$$t = \frac{N_i\cdot(P_1 - P_{E_i})}{-N_i(P_2 - P_1)}$$

<u>left edge</u>

$N_i = -\hat{i}$

$P_1 = -2i + j$

$P_2 = 3i + 3j$

$P_{E_i} = 4j$

$P(t) = P_1 + (P_2 - P_1)t$

$t = 2/5$

$N_i\cdot(P_2 - P_1) \neq 0$

$> 0 \rightarrow$ exiting (the left edge)

$< 0 \rightarrow$ Entering left edge

$0 \leq t \leq 1$

if $t < 0$ or $t > 1 \rightarrow$ no intersection.

Delta

Eg 1

LEFT

$PE_i = i + 3j$ $\qquad$ $N_i = -i$

$P_1 = -2i - 2j$

$P_1 - PE_i = -3i - 5j$

$P_2 = 11i + 7j$ $\qquad$ $P_2 - P_1 = 13i + 9j$

$t_L = 3/13$

BOTTOM.

$PE_i = 8i + 3j$ $\qquad$ $P_1 - P_E = -10i - 5j$

$P_1 = -2i - 2j$ $\qquad$ $P_2 - P_1 = 13i + 9j$

$P_2 = 11i + 7j$

$N_i = -j$ $\qquad$ $t_B = 5/9$

lly $\rightarrow$ $t_R = 10/13$ $\qquad$ $t_T = 10/9$

$N_i(P_2 - P_1)$

$\cancel{\leq 0}$ $\qquad$ $\cancel{\leq 0}$ $\qquad$ ≥ 0 $\qquad$ ≥ 0 $\qquad$ > 1 X.

$t_L(3/13)$ $\qquad$ $t_B(5/9)$ $\qquad$ $t_R(10/13)$ $\qquad$ $t_T(\cancel{10/9})$.

$(0 \quad t_L \quad t_B)$

max $\qquad\qquad$ $\min(t_R, 1)$

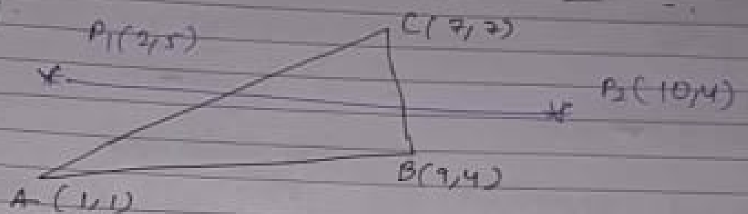
t_B ✓ $\qquad$ t_R ✓

clipped line (t_B, t_R) ✓ $\quad P_t = P_1 + (P_2 - P_1)\,t$ ✓

Scanned by CamScanner

* Cyrus Beck → non-rectangular window

Eg



$t = \frac{N_i(P_1 - P_{Ei})}{-N_i(P_2 - P_1)}$ if $t<0$ or $t>1$ line & edge not intersecting

$0 < t \le 1$

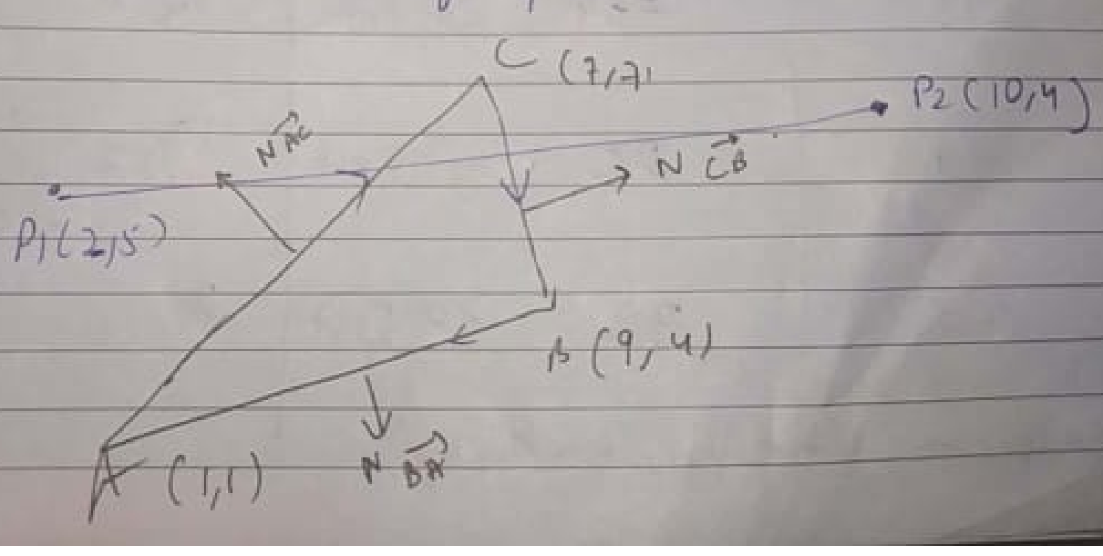
Potentially Entering − max, $-D < 0$

Potentially Exit → min. $-D > 0$

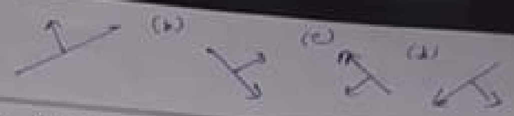
non-rectangular window → convex
→ concave

Convex

if we move clockwise, right part is inside & left part is outside



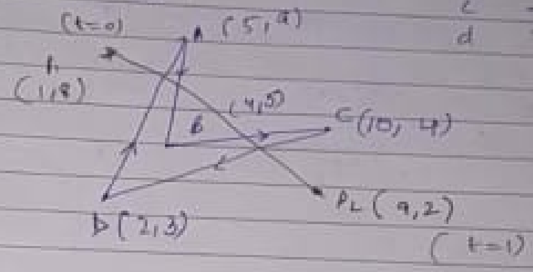
(a) (b) (c) (d)



Line Clipping

non-Rectangular windo.

Concave - window.



for clockwise

	edge	normal
a	+ +	− +
b	+ −	+ +
c	− +	− −
d	− −	+ −

Cyrus-beck-approach

{clockwise}

A → B → C → D → A

edge		normal (N)
$\vec{AB}$	$-i - 4j$	$4i - j$
$\vec{BC}$	$6i - j$	$+ i + 6j$
$\vec{CD}$	$-8i - j$	$i - 8j$
$\vec{DA}$	$+3i + 6j$	$-6i + 3j$

$P_2 - P_1 = 8i - 6j$

$P_2 - P_1 = 8i - 6j$

$P_1 = i + 8j \quad P_2 = 9i + 2j$

	P_{Ei}	$P_1 - P_{Ei}$	$P_1 - P_{Ei} \cdot N$	$N \cdot (P_2 - P_1)$
AB	$5i+9j$	$-4i-j$	-15	38 P exit
BC	$4i+5j$	$-3i+3j$	15	-28 P enter
CD	$10i+4i$	$-9i+4j$	-41	56 P Exit
DA	$2i+3i$	$-i+5j$	21	-66 P enter

t

AB	$\frac{15}{38}$
BC	$\frac{15}{28}$
CD	$\frac{41}{56}$
DA	$\frac{21}{66}$

To show clipped line →

~~draw line b~~

draw line b/w Enter/Exit pairs.

concavity

t

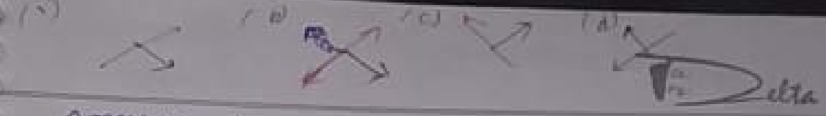
$S: \left(0, \frac{21}{66}, \frac{15}{38}, \frac{15}{28}, \frac{41}{56}, 1\right)$

concave

Convex

Enter Exit Enter Exit } switches

$(0, t_{Enter}, t_{ent}, t_{exit}, t_{exit}, 1)$



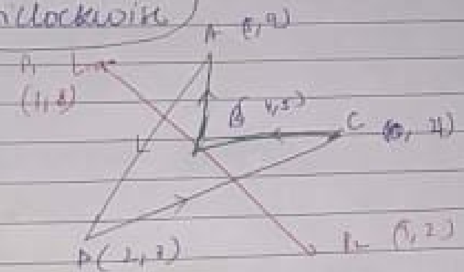
convex →

$(0, t_{ent}, t_{ent}, t_{ex}, t_{ex}, 1)$

line b/w enter & exit

if one enter / Exit → one clipping
more than one Enter / Exit
→ more than 1 clipping

Anticlockwise



Normal

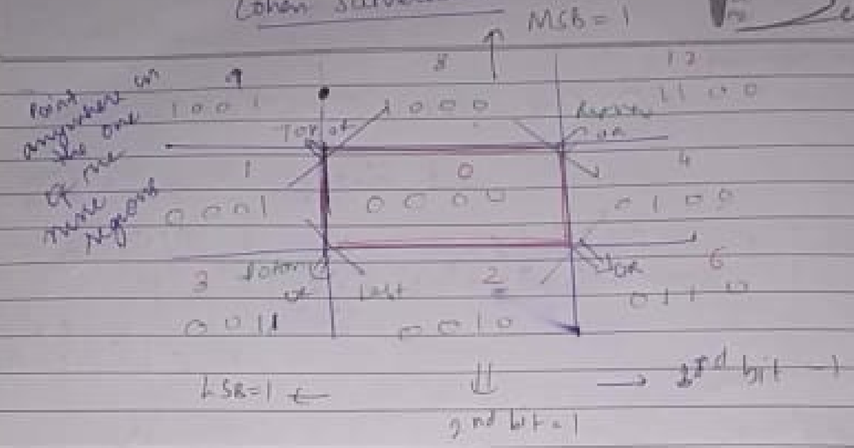
	Edge		Normal	
a	+	+	+	−
b	+	−	−	−
c	−	+	+	+
d	−	−	−	+

Edge $P(E_i)$	~~Normal~~ Edge	Normal	$P_1 - PE_i$	$(P_1 - PE_i)N$	$N(P_2 - P_1)$
$\vec{AD}$ (5i+9j)	$-3i - 6j$	$-6i + 3j$	$-4i - j$	21	−66 Enter
$\vec{DC}$ (2i+3j)	$8i + j$	$i - 8j$	$-1i + 5j$	−41	56 exit
$\vec{CB}$ (10i+4j)	$-6i + j$	$i + 6j$	$-9i + 4j$	15	−28 Enter
$\vec{BA}$ (4i+5j)	$i + 4j$	$4i - j$	$-3i + 3j$	−15	38 Exit

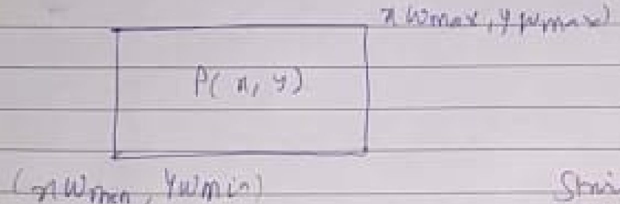
Rest procedure same

$P_1 = i + 8j$
$P_2 = 9i + 2j$ $\quad P_2 - P_1 \rightarrow 8i - 6j$

Cohen Sutherland



point anywhere in the one of the nine regions



$xwmin \le x \le xWmax$

$ymin \le y \le ymax$

input: $P_1(x_1, y_1)$, $P_2(x_2, y_2)$

Method:

Code1 = gencode(point P_1)

Code2 = genCode (Point P_2);

address assigned to end point

```
String gencode(point p)
{
  String String Code = "0000";
  if(P.x < xwmin)
     code[0] = '1';
  else
  if (P.x > xWmax)
     code[2] = '1';
  else
  if (P.y < ywmin)
     code[1] = '1';
  if ( P.y > yWmax)
     code[3] = '1';
} return code;
```

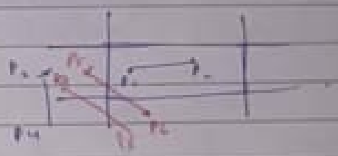
Delta

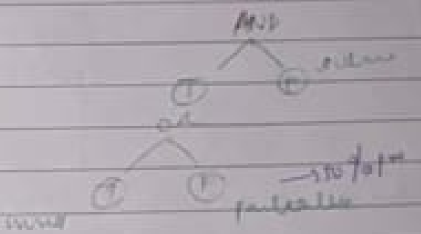
GRAPHICS

define LEFT 0x1 BOTTOM 0x2 TOP 0x8 RIGHT 0x4

code = 0x0; hexadecimal value

```
genCode ( point P )
{
    if P.x <= xwmin
        code |= LEFT

    if P.x >= xwmax
        code |= RIGHT

    if P.y <= ywmin
        code |= BOTTOM

    if P.y >= ywmax
        code |= TOP
}
```





Case I

code(P1) = 0000   P1 AND P2 = 0   completely inside

code(P2) = 0000   P1 OR P2 = 0

Case II

code(P3) = 0001   P3 AND P4 ≠ 0 } → Totally outside

code(P4) = 0011

Case III

code(P5) = 0001   P5 AND P6 = 0 } Line passing through rectangular window

code(P6) = 0010   P5 OR P6 ≠ 0

50% probability

as P7, P8 → condition true but line outside

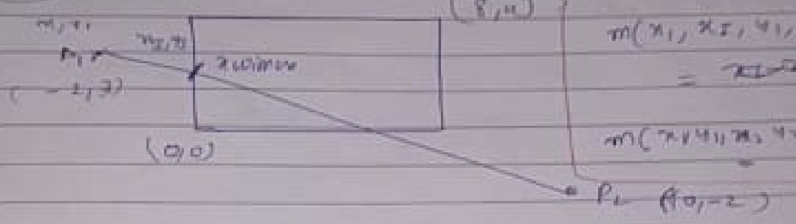
* When AND=0, OR≠0, one goes for clipping & keep on iteratively checking that after clipping line is falling in case 2 or case 1.

method

```
do {      flag = 0; done = 0;
  code1 = genCode (point P1);
  code2 = genCode ( point P2 );

case1  if (code1 & code2 = =0 ) && (code1 | code2 = =0)
       {   /* line is visible */
             flag =1, done=1;
             exit();
       }
case2:  if ( code1 & code2 != 0)
        {
           /* line is not visible */   done= 1,
           exit( ),
        }
case3:  if ((code1 & code2==0) && ( code1 | code2 != 0))
        {
           findintersection ( );
        }
} while (done != 1);
```

Scanned by CamScanner

Eg 1



$m(x_1, x_I, y_1, y_2)$

$= \frac{y_I - y_1}{x_I - x_1}$

$m(x_1, y_1, x_2, y_2) = \frac{y_2 - y_1}{x_2 - x_1}$

y_I

code(P_1) = 0001
code(P_2) = 0010

flag=0, done = 0

flag	done
0	0

Code(P_1) = 0001
Code(P_2) = 0110

case 3
LEFT = 0x1.
if((LEFT & Code(P_1)==1) — intersection with left
{
// clipping from left edge:
x_I = xwmin;
$y_I = y_1 + \frac{(y_2 - y_1)}{(x_2 - x_1)} (-x_1 + xwmin)$
$P_1 \leftarrow (x_I, y_I)$
}

four if condition
left, right, top, bottom

code(P_1) = 0000
Code P_2 = 0110

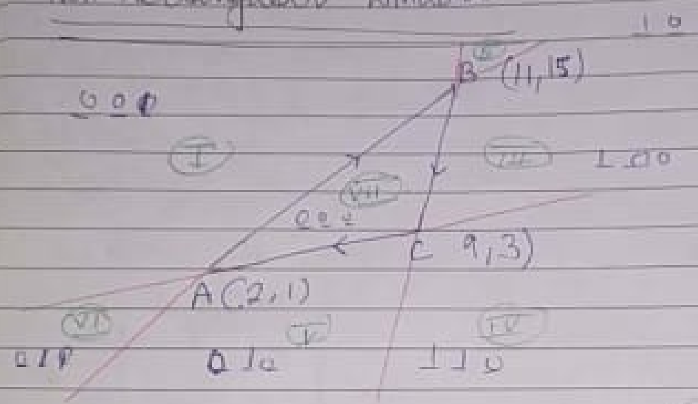
case 3.
→ 4 if condition again

↳ intersection with right ✓

so on → till it falls in case 1, case 2

if (flag)
drawline (P1, P2)

Non rectangular Window → Cohen Sutherland.



Eq $\vec{AB}$ = $y-1 = \frac{14}{9}(x-2)$

$9y - 9 = 14x - 28$

$f_1(x,y)$ $\{ 14x - 9y - 19 = 0 \}$

Eq $\vec{BC}$ → $y-3 = \frac{12}{2} 6 (x-9)$

$f_2(x,y)$ $\{ -6x + y + 51 = 0 \}$

Eq $\vec{CA}$ → $y-1 = \frac{2}{7}(x-2)$

$7y - 7 = 2x - 4$

$f_3(x,y)$ $\{ -2x + 7y - 3 = 0 \}$

Inside test

$f(x,y)$
C(9,3) >0

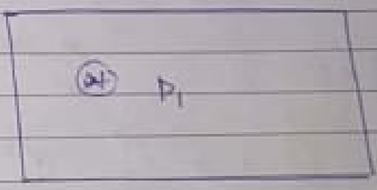
$f(x,y)$
A(2,1) >0

$f(x,y)$
B(11,15)
>0

```
# define   L1   001
# define   L2   002
# define   L4   004
code = 000
int genCode ( point P )
{
    if ( f1( P.x, P.y) < 0)
        code |= L1

    if ( f2(P.x, P.y) < 0)
        code |= L4

    if ( f3(P.x, P.y) < 0)
        code |= L2
return code.
}
```

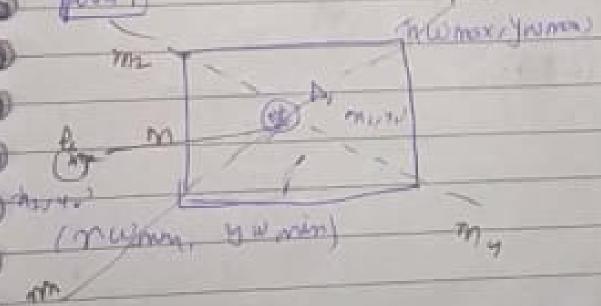
Nicholl – Lee – Nicholl (fastest computation, code length long)

Step-1 Locate one of the end point of a line



Step2

part 1



part 1 → end point is inside the rectangular window

part 2 → end point is on the edge side (left, bottom, right, top edge)

part 3 → end point is on the corner

4 corners → left top, left bottom, right top, right bottom.

We can find b/w which 2 slopes m is lying

10

Y

$m_1 \le m \le m_2$ | $m_2 \le m \le m_3$ | $m_3 \le m < m_4$ | $m_4 \le m < m_1$

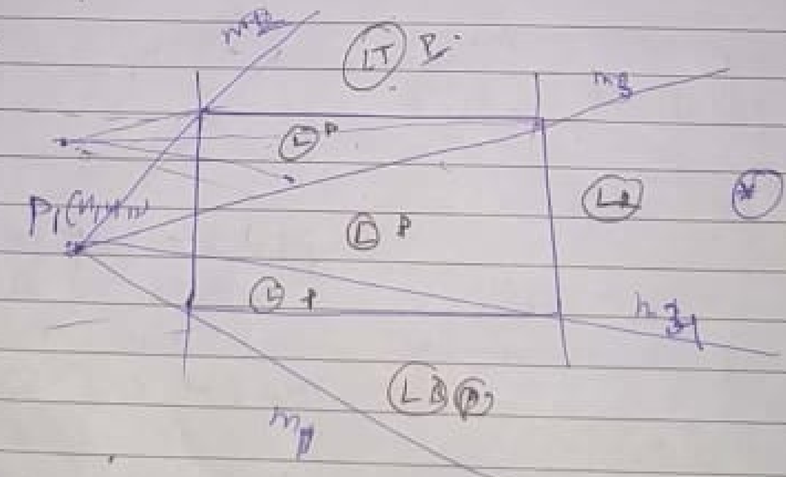
y | $P_2.x < x_{wmin}$

P_1 — clipping

P_2 — no clipping

$m_1 \le m \le m_2 \to$ if ($m < m_1$) && ($m > m_2$) && ($P_1.x_1 > P_2.x_2$)

Part II



der

line b/w m_1 & m_4 → invisible.

WL + Edge

LEFT | BOTTOM | RIGHT | TOP

2nd end POH wrt slope

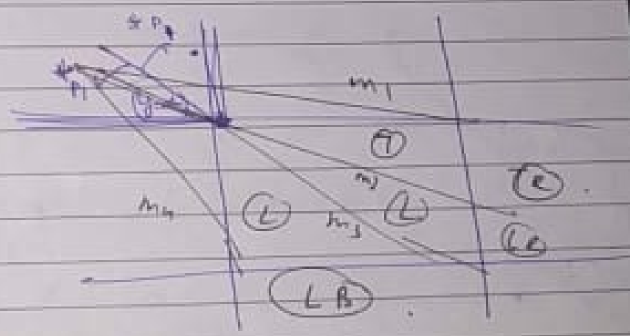
$m_1<m<m_2$ | $m_1<m<m_2$ | $m_3<m<m_4$ | X

reject

L | LT | X

intersection equation

Part LT



LT | RT | BL | BR

acc to above

m=y or lower

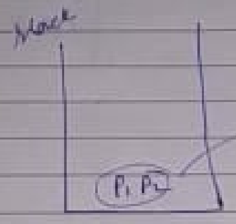
upper | lower

4 cases

$m_1<m_0<m_2$

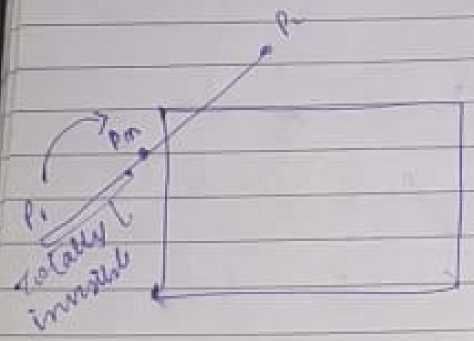
T TR X

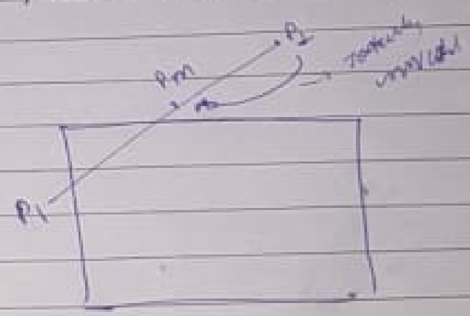
Line Clipping by Recursion.

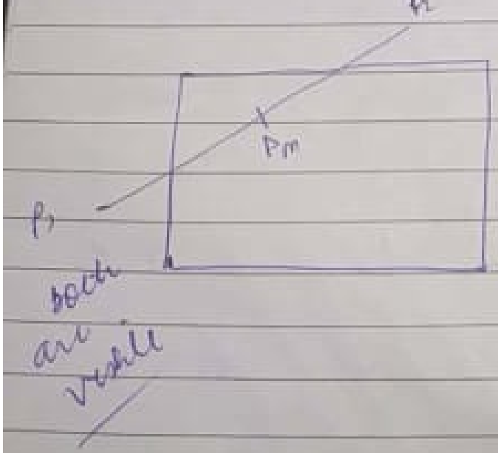


Cohen sutherland condition
→ totally visible
→ totally invisible
→ partially visible

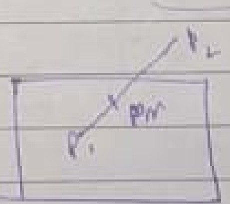
divide the line by mid point
P_1P_m , P_mP_2
push in stack

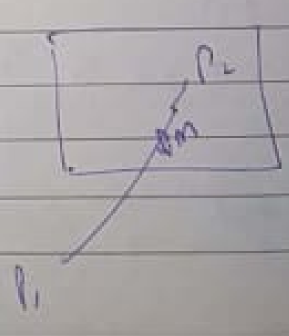






we will do till line comes inside the box.





Delta

Input → $P_1(x_1, y_1)$, $P_2(x_2, y_2)$
xwmin, xwmax, ywmin, ywmax

code1 = genCode(P_1), code2 = gencode(P_2)

Step1

if Code1, code2 both = 0
draw line $P_1 P_2$

Step2 if code1 & code2 != 0.
// line is invisible

Step3 Pm = (P_1 + P_2)/2 codem = genCode(Pm).

Step4 if codem != 0
then
if (code1 & codePm != 0)
then
P_1 = Pm ;
goto step 1.
else
if (code2 & codem != 0
then
P_2 = Pm, goto step 1

Step5 if codem = 0
if code1, codem both = 0
then consider PmB
else
if code2, codem both = 0
then consider P_1 Pm

Step ⑥

Consider $P_1 P_m$

do
{
$P_{m_1} = \frac{P_1 + P_m}{2}$;

code m1 = gencode (P_{m_1}).

if (code m1 != 0)
  $P_1 = P_{m_1}$; ed
else
  $P_m = P_{m_1}$

}
while (Pm.x != xwmax && Pm.x != xwmin && pm.y != ymin && pm.y != ywm

$P_1 \leftarrow P_m$

Step ⑦ Consider $P_2 P_m$

do
{
  (Same)
}
while( );

$P_2 \leftarrow P_m$

Scanned by CamScanner

Delta

Eg1

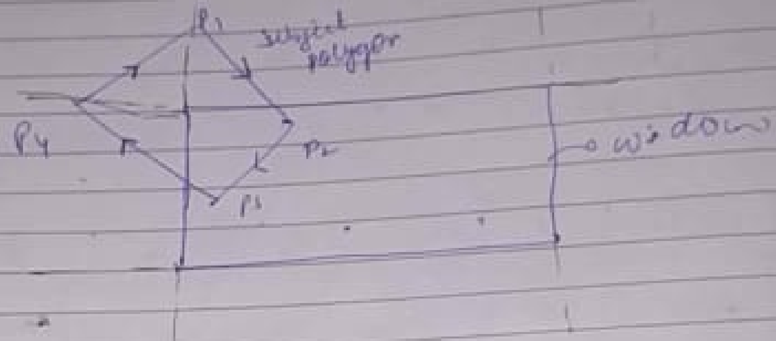
A (120,5)
P_2 (180,30)

Window ABCD
A (100, 10) B (160, 10) C (160, 10), D (100 40)

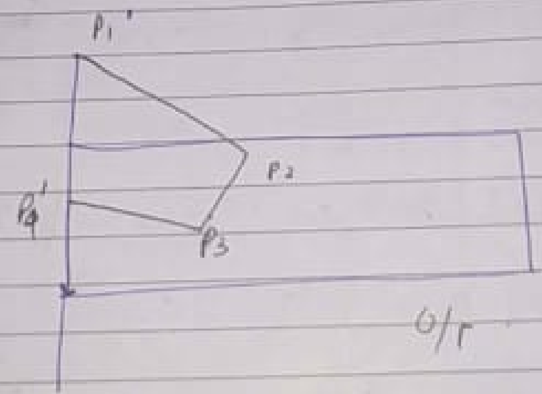
P_1	Code 1	P_2	Code 2	P_m	Code m	Remark
(120,5)	0010	(180,30)	0100	(150,18)	0000	Consider $P_1 P_m$ & Consider $P_m P_2$
Consider $P_1 P_m$						
(120,5)	0010	(150,18)	0000	(135,12)	0000	
(120,5)	0010	(135,12)	0000	(128, 9)	0010	$P_1 P_m$ → invisible
(128,9)	0010	(135,12)	0000	(132, 11)	0000	both visi-
(128,9)	0010	(132,11)	0000	(130,10)	0000	intersection point Broadcast assign P_1 ← (130,10) intersection
128(10)						
Consider $P_m P_2$						
150,18	0000	(180,30)	0100	(165,24)	0100	$P_2 P_m$ invisible
150,18	0000	(165,24)	0100	(158,21)	0000	
158,21	0000	(165 24)	0100	(162,23)	0100	$P_m P_2$ in ~
158,21	0000	(162,23)	0100	(160,22)	0000	P_2 ← (160,22)

(intersection)

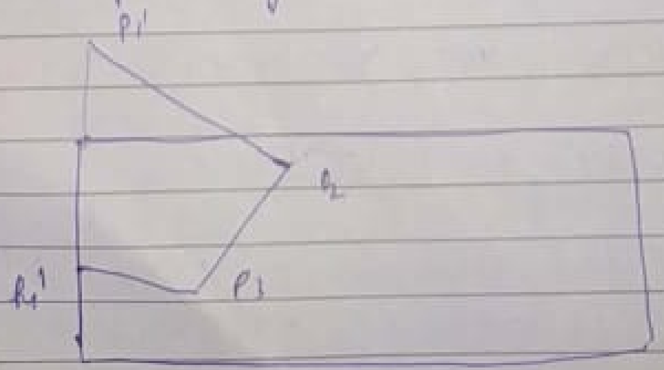
Clipping of a polygon {Sutherland - Hodgman}



Step 1 → clip along the left edge

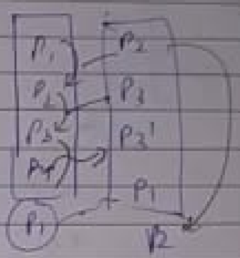


Step 2/3 → clip along the bottom edge / right



Delta

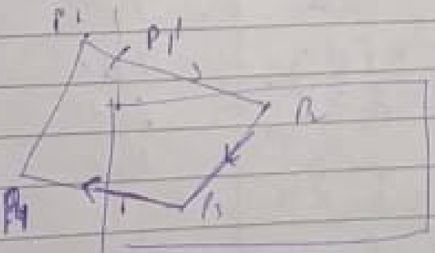
step 4 → clip it along top edge.

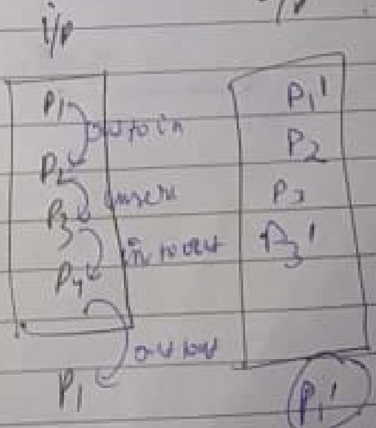


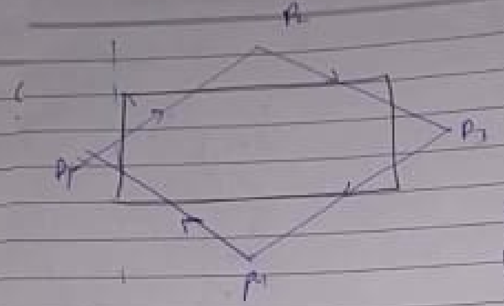
Edge

E_1	E_2	(line end points of a subject polygon)
inside	inside	retaining 2nd end point (E_2)
inside	outside	intersection point is computed and stored.
outside	inside	intersection point & 2nd ~~vertex~~ endpoint both to be stored.
outside	outside	Nothing is to be stored.

(all lost edge)



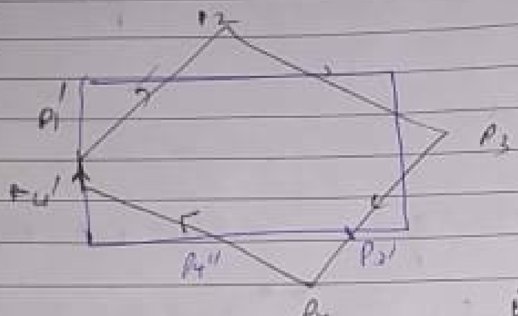




I/P | **left edge**

- out to in: P_1 → P_2
- in to in: P_3
- in to in: P_4
- in to out: P_1

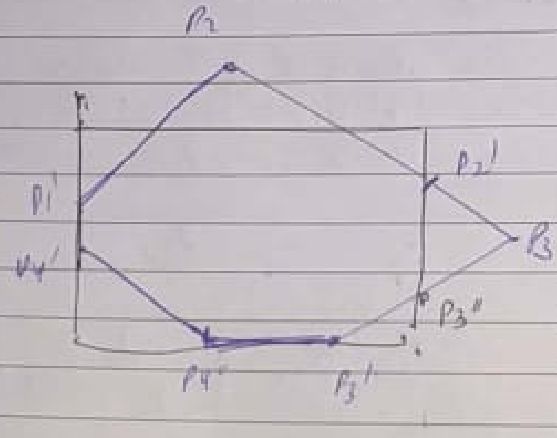
Left edge: P_1', P_2, P_3, P_4, P_4', P_1'



i/p (left edge) | **bottom**

- in to in: P_1' → P_2
- in: P_3
- in to out: P_3 → P_4
- out to in: P_4'
- in: P_1'

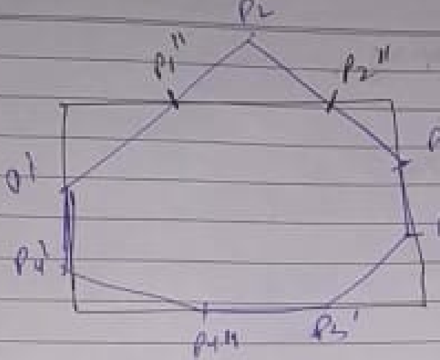
Bottom: P_2, P_3, P_3', P_4'', P_4', P_1', P_2



i/p bottom | **right**

- in to out: P_2 → P_3
- out to in: P_3'
- in: P_4''
- in: P_4'
- in: P_1'
- in: P_2

Right: P_2', P_3'', P_3', P_4'', P_4', P_1', P_2, P_2'



implement

in (P2'
in (P3''
P3'
in (P4''
P4'
P1'
in to out (P2'
P2''

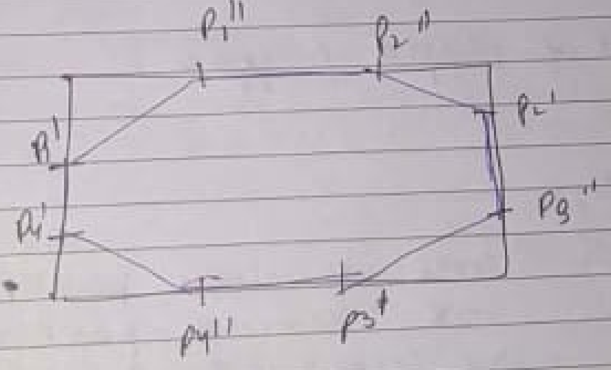
outpoin

or top

P3''
P3'
P4''
P4'
P1'
P1''
P2''
P2'

P3''

O/P



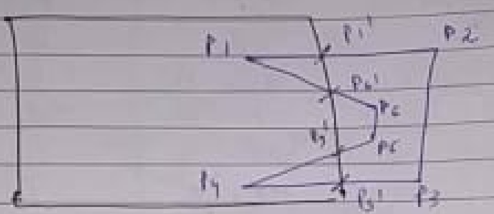
int windowcollec
{

```
int inside (Point P, Edge E)
{
   switch(edge)
   {
//left     case 0:  if (P.x >= xwmin)
                        return 1;
//bottom   case 1:  if (P.y >= ywmin)
                        return 1;
//right    case 2:  if (P.x <= xwmax)
                        return 1;
//top      case 3:  if (P.y <= ywmax) return 1;
   default: return 0; } }
```

Delta

Sutherland – Hodgman (Polygon clipping).

Concave Polygon



Input :- a) Window coordinates
b) Subject Polygon coordinates

O/p → set of clipped polygon

Method:

Rules

a) i → o   Store intersection point (I))
b) o → i   Store intersection point (I), Second Vertex (P2)
c) i → i   Store Second vertex P2)
d) o → o   nothing is stored.

I/P		Right edge	
P1	i → o	P1'	line (P1', P3') — Extra line
P2	o → o	P3'	line (P3', P4)
P3	o → i	P4	line (P4, P4')
P4	i → o	P4'	line (P4', P6') — repeatation
P5	o → o	P6'	line (P6', P1)
P6	o → i	P1	line (P1, P1')
P1		P1'	

line (P6', P1')

line (P3', P4')

Step 1

$i \to 0$ (i→0) visit (traverse)

P_1'	1	$\cancel{0}$ 1
P_3'	0	$\cancel{0}$ 1
P_4	0	$\cancel{0}$ 1
P_4'	1	$\cancel{0}$ 1
P_6'	0	$\cancel{0}$ 1
P_1	0	$\cancel{0}$ 1
P_1'	0	$\cancel{0}$ 1

$i \to 0$

Step 2 (Intersection points)

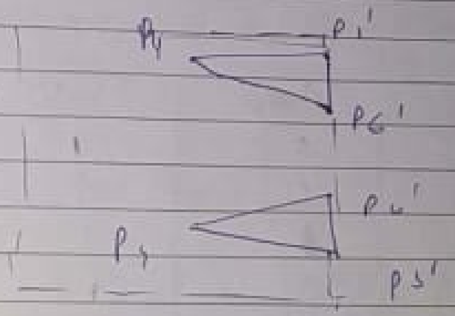
P_1'
P_3'
P_4'
P_6'

$\xrightarrow{\text{Sort } (y)}$ (as on right edge)

P_1'
P_6'
P_4'
P_3'

if $i \to 0$ 1 → jump to 2nd step & draw the line
$i \to 0$ 0 → go to next point & draw line

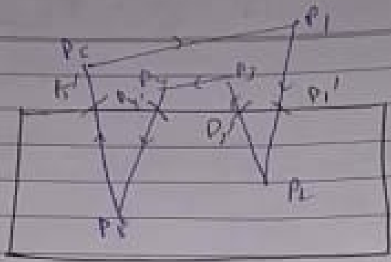
line (P_1' P_0')
line (P_6' P_1)
line (P_1, P_1')
line (P_3', P_4)
line (P_4, P_4')
line (P_4' P_3')

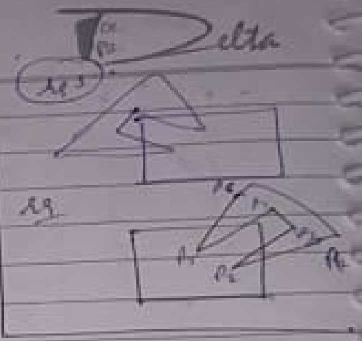


Continue the process till the visit of all the points are 1

Rules are same: but we have changed the traversal for concave polygon

(II)



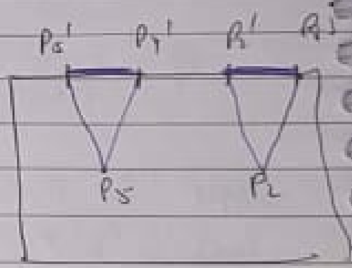


(i/p)

$0 \to 1$ P1
P2 $1 \to 0$
P3
$0 \to 1$ P4
P5 $0 \to 1$
$1 \to 0$ P2
P1 $0 \to 1$

Top

P1'
P2
P2' $\longrightarrow$ ($1 \to 0$)
P4'
P5
P5' $\longrightarrow$ ($1 \to 0$)
P1'



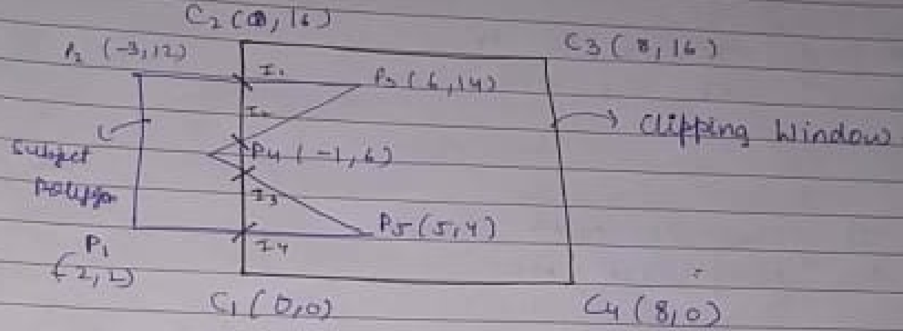
	$i \to 0$	visit
P1'	0	1
P2	0	1
P2'	0	1
P4'	0	1
P5	0	1
P5'	0	1
P1'	0	1

P1'
P2'
P4'
P5'

$\longrightarrow$ sort(n) $\longrightarrow$
(increasing)

P5'
P4'
P2'
P1'

Polygon Clipping (Weiler-Atherton)



whenever exit → turn right

$C_2 - C_1 = 16j$

Edge (Clipping Window)

C_1C_2

$$t = \frac{N \cdot (C_1 - P_{E_C})}{-N \cdot (C_2 - C_1)}$$

	Normals	Entry / Exit (1 / 0)	t	Intersection point
P_1P_2 ~~...~~ $(-i+10j)$	$-10i - j$	(-ve) -16	$\frac{18}{-16} = \frac{-9}{8}$	X $0 \le t \le 1$
P_2P_3 $(9i+2j)$	$-2i + 9j$	144 Entry	$\frac{114}{144}$	I_1
P_3P_4 $(-7i-8j)$	$8i - 7j$	-112 Exit	$\frac{50}{112}$	I_2
P_4P_5 $(6i-2j)$	$2i + 6j$	96 Entry	$\frac{34}{96}$	I_3
P_5P_1 $-7i+2j$	$2i - 7j$	-112 Exit	$\frac{18}{112}$	I_4

← increasing

~~$t\left(\frac{19}{24}, \frac{30}{112}, \frac{1}{12}, \frac{9}{56}\right)$~~

En Exit Ent Exit

$t\left(\frac{114}{144}, \frac{50}{112}, \frac{34}{96}, \frac{18}{112}\right)$

Delta

Step 1

SP P_1 P_2 P_3 P_4 P_5

CW C_1 C_2 C_3 C_4

Step 2

(1) Intersection point Evolute

(b) Entry / Exit note

Step 3

entry

SP P_1 P_2 I_1 $\boxed{1}$ P_3 I_2 $\boxed{0}$ P_4 I_3 $\boxed{1}$ P_5 I_4 $\boxed{0}$

CW C_1 I_4 $\boxed{1}$ I_3 $\boxed{0}$ I_2 $\boxed{1}$ I_1 $\boxed{0}$ C_2 C_3 C_4

to more elements

(exit)

next entry

Step 4

Entry list $\{I_1, I_3\}$

$I_1 \rightarrow P_3 \rightarrow I_2 \rightarrow I_1$

$\boxed{0}$

$I_3 \rightarrow P_5 \rightarrow P_4$

line (I_1, P_3)

line (P_3, I_2)

line (I_2, I_1)

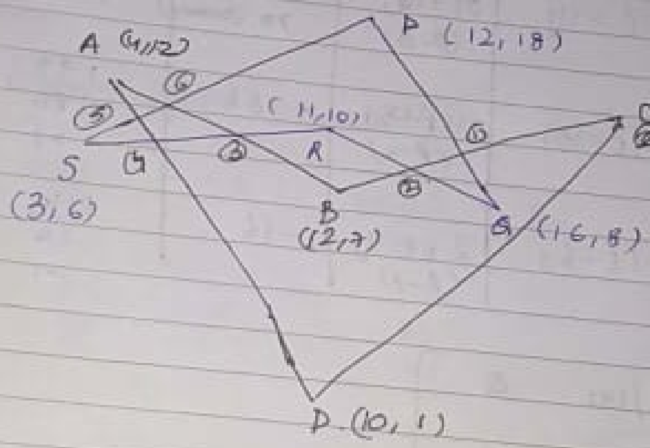
line (I_3, P_5)

line (P_5, I_4)

line (I_4, I_3)

Weiler Atherton.

Non-Rectangular Clipping Window.



PQRS → Subject Polygon

ABCD → Clipping Window

Step 1

SP: P Q R S

CW: A B C D

Step 2

intersection point computation

SP: Edge 1 (PQ) 4i−10j

(point not edge)

$t = \frac{N\cdot(P-E_i)}{-N\cdot(Q-P)}$

Edge (CW)	Normal	$P-E_i$	Dr $(-N\cdot(Q-P))$	t
AB $(8i-5j)$	$5i+8j$	$8i+6j$ $(P-A)$	60	$\frac{88}{60} = \frac{22}{15}$ >1 X
BC $(11i+8j)$	$-8i+11j$	$0i+11j$ $(P-B)$	142 >0 (Entering)	$\frac{121}{142}$ $0\to i$
CD $(-13i-14j)$	$14i-13j$	$-11i+3j$ $(P-C)$	-186	$\frac{193}{186}$ >1 X
DA $(-6i+11j)$	$-11i-6j$	$2i+17j$ $(P-D)$	-16	$\frac{124}{16}$ >1 X

SP $\begin{bmatrix} P & 121/142 & Q \\ -1 & \underbrace{0}_{0\to i} & -1 \end{bmatrix}$.

SP : Edge 3 (RS) $-8i-4j$

$t = \frac{N\cdot(R-E_i)}{-N(S-R)}$

Edge (CW)		Normal	$R-E_i$	Dr $(-N\cdot(Q-P))$		t
AB	$(8i-5j)$	$5i+8j$	$7i-2j$	72	$0\to i$	$\frac{19}{72} = 0.26$ ③
BC	$(11i+8j)$	$-8i+11j$	$-i+3j$	-20	$i\to 0$	$\frac{\cancel{82}}{20}$ $-\frac{41}{20}$ X
CD	$(-13i-14j)$	$14i-13j$	$-12i-5j$	60	$0\to i$	$-\frac{103}{60}$ X
DA	$(-6i+11j)$	$-11i-6j$	$i+9j$	$\cancel{-186}$ -112	$i\to 0$	$\frac{65}{112}$ 0.58 ④

enter 6 → 0 → 0
exit 1 → 0 → 1

entry → 0
exit → 1

SP

P	121/142	Q	R	0.26 ⑤	0.58 ④	S
−1	0	−1	−1	0	1	−1

for all edges → PQ, QR, RS, SP

CW

for clipping window →

Edge 1 — AB	Edge 2 — BC — — — for all edges
PQ	PQ
QR	QR
RS	RS
SP	SP

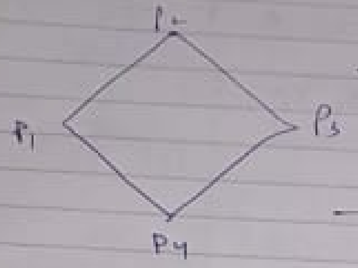
SP	P	①	Q	②	R	③	④	S	⑤	⑥
Tagging	−1	0	−1	1	−1	0	1	−1	0	1
Traversal	~~0~~ 1	~~0~~ 1	~~0~~ 1	~~0~~ 1	~~0~~ 1	~~0~~ 1	~~0~~ 1	~~0~~ 1	~~0~~ 1	~~0~~ 1

CW	A	⑥	③	B	②	①	C	D	④	⑤
Tagging	−1	0	1	−1	0	1	−1	−1	0	1
Traversal	0	~~0~~ 1	~~0~~ 1	0	~~0~~ 1	~~0~~ 1	0	0	~~0~~ 1	~~0~~ 1

① Q ② ① → line (1,Q), line (Q,2) line (2,1)
③ ④ ⑤ ⑥ ③ → line (3,4), line (4,5), line (5,6) line (6,3)

$\underbrace{\hspace{6em}}_{\text{required polygon}}$

Polygon filling

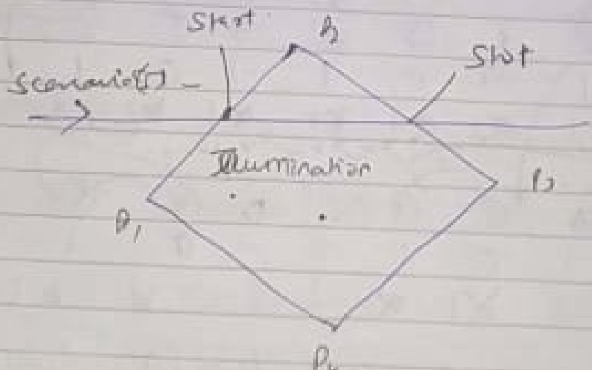


(I)
→ non horizontal
(no horizontal edges)

(II)
→ non vertical
(no vertical edges).

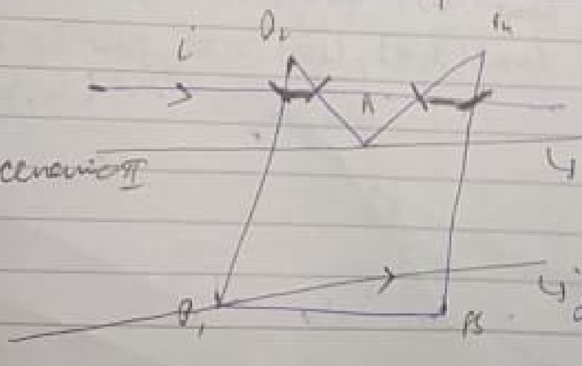
Scan line in filling the polygon

→ horizontal scanning
→ vertical scanning.



left to right.

at vertex
→ 2 edges



Scenario II

↳ 2 edges on same side

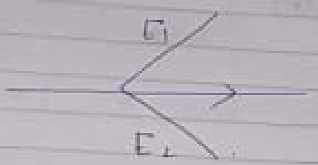
↳ 2 edges on either side.

Scanned by CamScanner

→ vertices are formed from 2 consecutive edges.

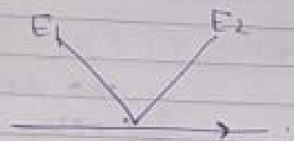
→ when a scan line passes through vertices

a)



$E_1.y_{min} = E_2.y_{max}$

b)



$E_1.y_{min} = E_2.y_{min}$

c)

$E_1.y_{max} = E_2.y_{max}$

Global Edge Table (Polygon)

left to right (moving from bottom to top).

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

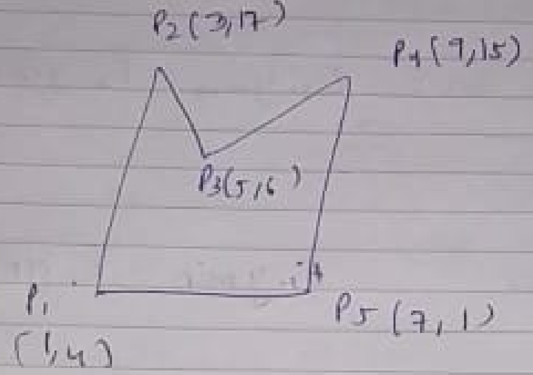
$x_{i+1} = x_i \pm 1/m$

Structure of an edge

ymax	x_{ymin}	$1/m$	$\rightarrow$ next

ymin → min y of all edges

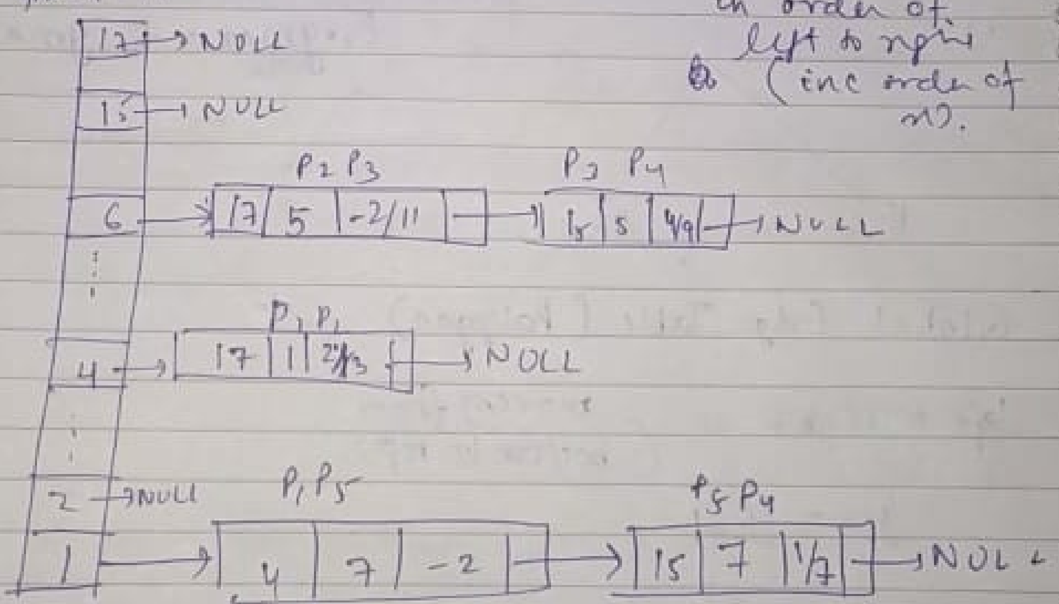
ymax → max y of all edges



no redundancy (take one edge only once)

in order of left to right (inc order of m).

ymax = 17



ymin = 1

Top to bottom.

ymin	x_{ymax}	$1/m$	$\rightarrow$

filling the polygon. (Scan line).

Start → no edges.

finish → no edges

[|] → NULL

~~for (i~~

for (i = ymin to ymax).

$\boxed{i=1}$

① MERGE

[|] → [4 | 7 | -2 |] → [15 | 7 | 1/7 |] → NULL

line (, i, , i).

line (7, 1, 7, 1)

② update if the nodes are previous one)

$x_{i+1} = x_i \pm 1/m$.

update

③ Draw line

→ if i = ymax of any node delete the edge

$\boxed{i=2}$

Step1 → merge

[|] → [4 | 7 | -2 |] → [15 | 7 | 1/7 |] → NULL.

Step 2 → update

[|] → [4 | 5 | -2 |] → [15 | 50/7 | 1/7 |] → NULL

Step 3 → draw line   line (5, 2, 50/7, 2).

$i = 3$

merge

[ ] → [4 | 5 | -2 |] → [15 | 3/7 | 1/7 |] → NULL.

update

[ ] → [4 | 3 | -2 |] → [15 | 51/7 | 1/7 |] → NULL

display

line (3, 3 , 51/7 , 3)
(i) (i)

$i = 4$

Step 1 → Merge

[ ] → [17 | 1 | 2/13 |] → [4 | 3 | -2 |] → [15 | 51/7 | 1/7 |] → NULL

(sorting according x)

Step 2 update

[ ] → [17 | 1 | 2/13 |] → [4 | 3 | -2 |] → [15 | 52/7 | 1/7 |] → NULL

(new)

delete

Step 3.

line (1, 4, 52/7 , 4)
(i) (i)

i=5

Step 1 → Merge

[|] → [17 | 1 | 2/13 |] → [15 | 53/7 | 1/7 |] → NULL

Step 2 → Update

[|] → [17 | 15/13 | 2/13 |] → [15 | 53/7 | 1/7 |] → NULL

Step 3 → draw

line (15/13, 5, 53/7, 5).

i=6

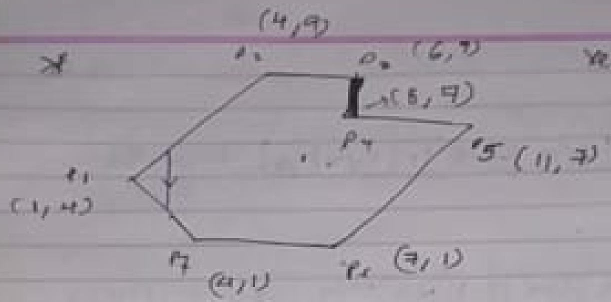
Step 1 → Merge

[|] → [17 | 15/13 | 2/13 |] → [17 | 5 | -2/11 |] → [|] → [15 | 5 | 4/9 |] → [15 | 53/7 | 1/7 |] → NULL

Step 2 Update

[|] → [17 | 17/13 | 2/13 |] → [17 | 5 | -2/11 |] → [15 | 5 | 4/9 |] → [15 | 54/7 | 1/7 |] → NULL

*



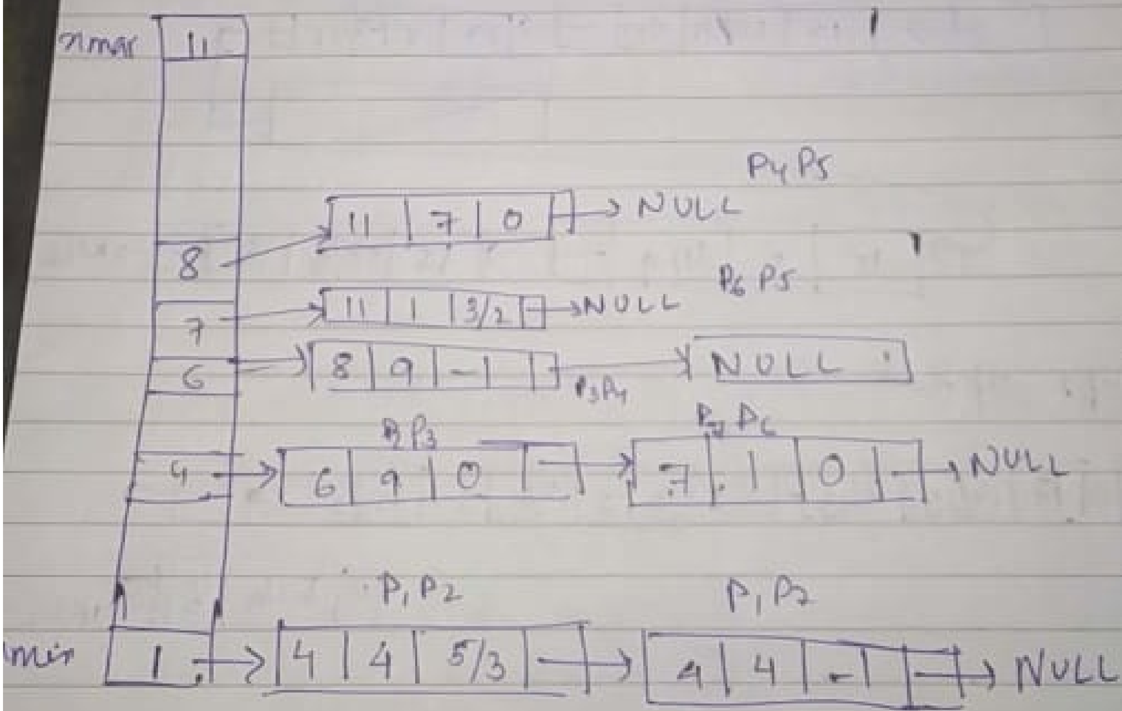
vertical scanning
(no vertical edges)
(left to right)

Step 1 → Edge-Structure to be defined.

xmax	yxmin	m	→ next

edge node

Step 2 → Global Edge Table Construction



Step 3 → Filling of polygon
(Active edge table)

$x=1$ [→] → NULL   for ($x = x_{min}$ to x_{max})

1) Merge

[→] → | 4 | 4 | 5/3 | → | → | 4 | 4 | -1 | → | → NULL

2) Update (delete $x_{max} = x$).
3) Delete x.
4) draw

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

line (1, 4 , 1 4)

$x=2$

1) Merge
2) Update   (+m).

[→] → | 2 | 17/3 | 5/3 | → | → | 4 | 3 | -1 | → | → NULL

3) Delete x.
4) draw
line (2, 17/3 , 2 , 3)

$x=3$

1) Merge
2) update

[→] → | 4 | ~~22~~/3 | 5/3 | → | → | 4 | 2 | -1 | → | → NULL

3) Delete
4) Draw   line (3, 22/3, 3, 2)

sort → descending

~~n=4~~

~~1) Merge~~

~~[ ] → [4 | 22/3 | 5/3 |] → [4 | 2 | -1 |] →~~

~~[ ] → [6 | 9 | 0 |] → [4 | 22/3 | 5/3 |] → [4 | 2 | -1 |] → [ ]~~

n=4

1) Merge

[ ] → [6 | 9 | 0 |] → [4 | 22/3 | 5/3 |] → [4 | 2 | -1 |] → [7 | 1 | 0 |] → NULL

2) update

[ ] → [6 | 9 | 0 |] → [4 | 9 | 5/3 |] → [4 | 1 | -1 |] → [7 | 1 | 0 |] → Null

delete

3) delete

[ ] → [6 | 9 | 0 |] → [7 | 1 | 0 |] → NULL

4) draw

line (4, 9, 4, 1, 0

x=5

1) merge
2) update

[ ] → [6 | 9 | 0 |] → [7 | 1 | 0 |] → ~~upd~~ NULL

3) delete x
4) draw

line (5, 9, 5, 1)

x=6.

1) Merge

[ ] → [6 | 9 | 0 |] → [8 | 9 | 4 |] → [7 | 1 | 0 |] → NULL

2) update

[ ] → [~~6 | 9 | 0~~ |] → [8 | 9 | -1 |] → [7 | 1 | 0 |] → NULL

3) delete

[ ] → [8 | 9 | -1 |] → [7 | 1 | 0 |] → NULL

4) draw

line (6, 9, 6, 1).

x=7

1) merge

[ ] → [8 | 9 | -1 |] → [7 | 1 | 0 |] → [11 | 1 | 3/2 |] → NULL

2) update

[ ] → [8 | 8 | -1 |] → [7 | 1 | ~~0~~ |] → [11 | 1 | 3/2 |] → NULL

delete

3) delete

[ ] → [8 | 8 | -1 |] → [11 | 1 | 3/2 |] → NULL

4) draw

line (7, 8, 7, 1).

$x=8$

1) Merge

[ ] → [8 | 8 | -1 |] → [11 | 7 | 0 |] → [11 | 1 | 3/2 |] → Null

2) update

delete

[ ] → [8 | ~~7~~ | -1 |] → [11 | 7 | 0 |] → [11 | 5/2 | 3/2 |] → NULL

3) delete

[ ] → [11 | 7 | 0 |] → [11 | 5/2 | 3/2 |] → NULL

4) draw

line (8, 7, 8, 5/2)

Scanned by CamScanner

x=9

1) Merge

2) update

[ ] → [11 | 7 | 0 |] → [11 | 4 | 3/2 |] → NULL

3) delete x

4) draw

line (9, 7, 9, 4).

x=10

1) Merge

2) update

[ ] → [11 | 7 | 0 |] → [11 | 11/2 | 3/2 |] → NULL

3) delete x.

4) draw

line (10, 7, 10, 11/2).

x = 11.

1) Merge

2) update

delete

[ ] → [11 | 7 | ~~0~~ |] → [11 | ~~7~~ | 3/2 |] → NULL

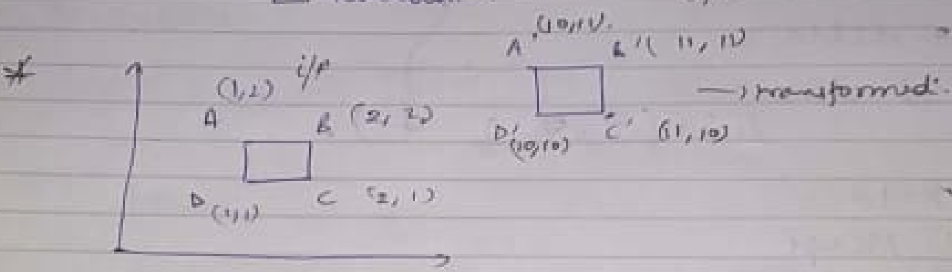
3) delete

[ ] → NULL

4) draw

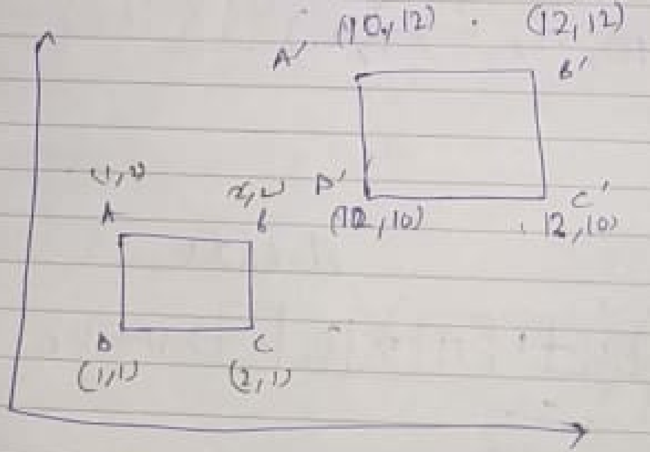
2-D Transformation

- Translation
- Scaling
- Rotation



$X' = x + \Delta x$
$Y' = y + \Delta y$

Transformation operator → + 'here'

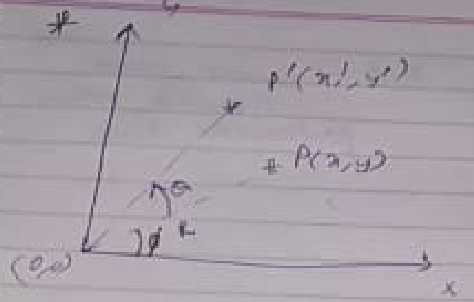


+ zooming

Scale up:
Sx:
Sy:

$X' = (x-1)Sx + \Delta x + 1$
$Y' = (y-1)Sy + \Delta y + 1$

Transformation operator
*, +
'here'



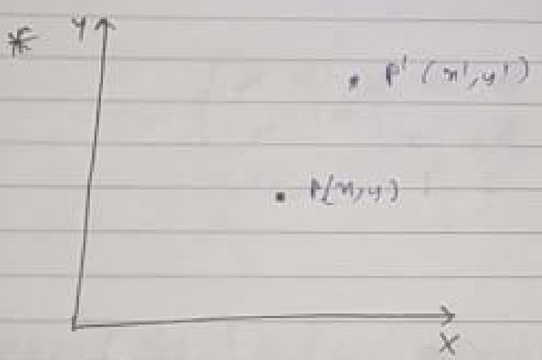
$R\cos\phi = x$
$R\sin\phi = y$
$R\cos(\theta+\phi) = x'$
$R\sin(\theta+\phi) = y'$

$x' = R[\cos\theta\cos\phi - \sin\theta\sin\phi]$
$x' = [x\cos\theta - y\sin\theta]$
$y' = R[\sin\theta\cos\phi + \cos\theta\sin\phi]$
$y' = [x\sin\theta + y\cos\theta]$

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

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} * \begin{pmatrix} x \\ y \end{pmatrix}.$$

→ Transformation operator is *



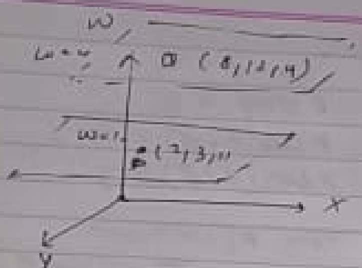
Scaling

→ Transformation operator is *

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} S_x & 0 \\ 0 & S_y \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

* Operator is behaving as non-homogenous + in some case, * in some case.

* Make it homogenous



P, Q → same when viewed from particular view

Homogenous coord system → $\left(\frac{x}{w}, y/w, 1\right)$

Translation

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & \Delta x \\ 0 & 1 & \Delta y \\ 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Scaling

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

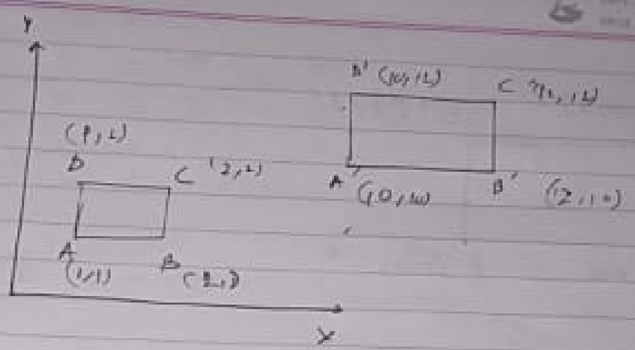
rotation

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

Homogenous operator → '*'

same for everything

Eg 1



Solution

Step 1: Obj is brought to origin

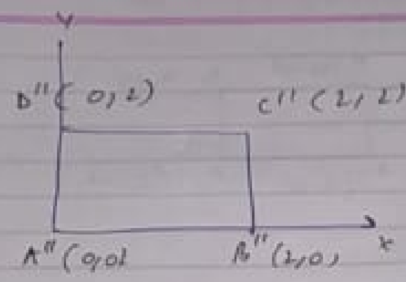
$$T = \underbrace{\begin{pmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{pmatrix}}_{\Delta x, \Delta y} * \begin{array}{cccc} A & B & C & D \\ \end{array} \begin{bmatrix} 1 & 2 & 2 & 1 \\ 1 & 1 & 2 & 2 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$T = \begin{array}{cccc} A' & B' & C' & D' \\ \end{array} \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

Step 2: Object to be scale up

$Sx = 2, \; Sy = 2$

$$S = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix} * \begin{array}{cccc} A' & B' & C' & D' \\ \end{array} \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix} = \underbrace{\begin{array}{cccc} A'' & B'' & C'' & D'' \\ \end{array} \begin{bmatrix} 0 & 2 & 2 & 0 \\ 0 & 0 & 2 & 2 \\ 1 & 1 & 1 & 1 \end{bmatrix}}_{\text{Result object}}$$



Step 3: Object is translated
$\Delta x = 10, \ \Delta y = 10$

$$T = \begin{pmatrix} 1 & 0 & 10 \\ 0 & 1 & 10 \\ 0 & 0 & 1 \end{pmatrix} \begin{array}{c} \begin{array}{cccc} A'' & B'' & C'' & D'' \end{array} \\ \begin{pmatrix} 0 & 2 & 2 & 0 \\ 0 & 0 & 2 & 2 \\ 1 & 1 & 1 & 1 \end{pmatrix} \end{array}$$

$$T = \begin{array}{c} \begin{array}{cccc} A''' & B''' & C''' & D''' \\ A_F & B_F & C_F & D_F \end{array} \\ \begin{pmatrix} 10 & 12 & 12 & 10 \\ 10 & 10 & 12 & 12 \\ 1 & 1 & 1 & 1 \end{pmatrix} \end{array}$$

$\leftarrow$ order

$$C = \underset{T_2}{\begin{pmatrix} 1 & 0 & 10 \\ 0 & 1 & 10 \\ 0 & 0 & 1 \end{pmatrix}} * \underbrace{\underset{S}{\begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix}} \times \underset{T_1}{\begin{pmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{pmatrix}}}$$

$\Downarrow$

$$C = \begin{pmatrix} 1 & 0 & 10 \\ 0 & 1 & 10 \\ 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} 2 & 0 & -2 \\ 0 & 2 & -2 \\ 0 & 0 & 1 \end{pmatrix}$$

Composite

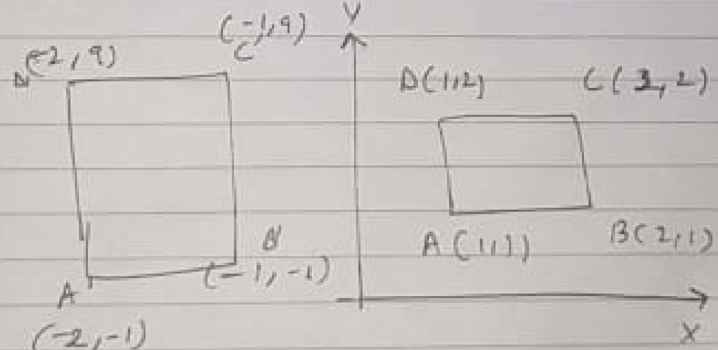
$$C = \begin{pmatrix} 2 & 0 & 8 \\ 0 & 2 & 8 \\ 0 & 0 & 1 \end{pmatrix}$$

O/P = C * i/p

$$O/P = \begin{pmatrix} 2 & 0 & 8 \\ 0 & 2 & 8 \\ 0 & 0 & 1 \end{pmatrix} \times \begin{array}{cccc} A & B & C & D \\ \end{array} \begin{bmatrix} 1 & 2 & 2 & 1 \\ 1 & 1 & 2 & 2 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$O/P = \begin{bmatrix} 10 & 12 & 12 & 10 \\ 10 & 10 & 12 & 12 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Eg 2



$$C = \begin{pmatrix} 1 & 0 & -2 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 10 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{pmatrix}$$

$$C = \begin{pmatrix} 1 & 0 & -2 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & -1 \\ 0 & 10 & -9 \\ 0 & 0 & 1 \end{pmatrix}$$

$$C = \begin{pmatrix} 1 & 0 & \cancel{-2}\ -3 \\ 0 & 10 & \cancel{8}\ -10 \\ 0 & 0 & 1 \end{pmatrix}$$

reflection about y axis →

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$