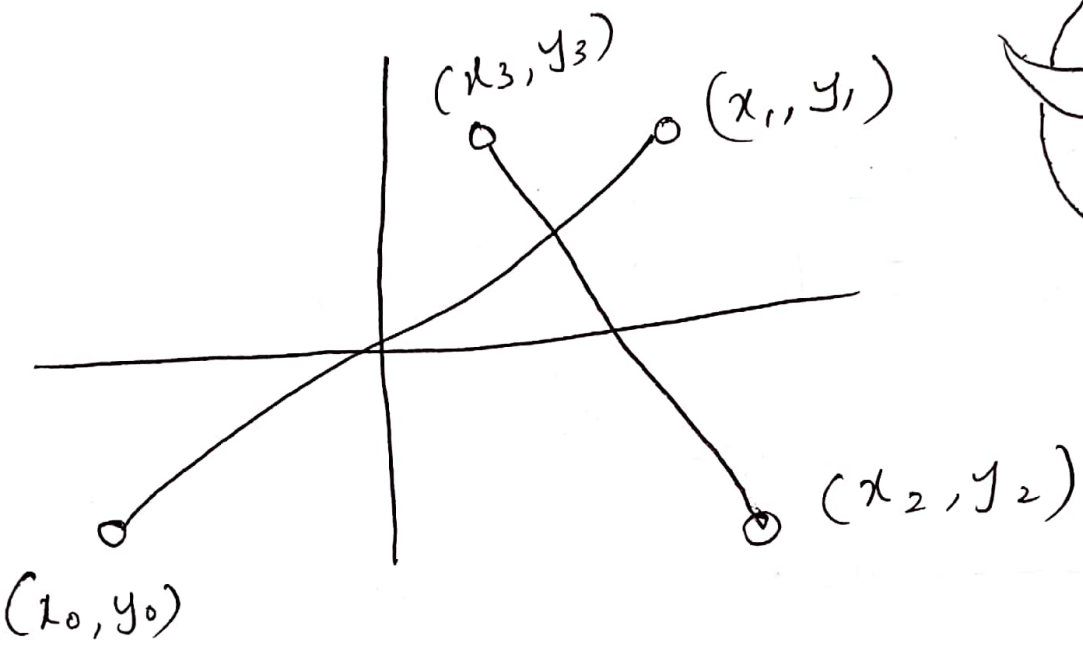
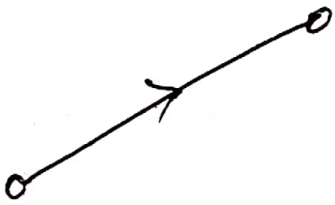
Graphics

① Geraphical Algorithm
② Engineering behind graphics





① Ivi vector has both magnitude & direction
② Ray has starting point & direction

$y = mx + c$ —— ①

$y = y_0$
for (x_0 to x_1)
drawPixel (x, y)

$x \to x+1$ $\quad y \to y+1$
$y \to y+m$ $\quad x \to x+1/m$

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

if $dy = 1$
$dx = \frac{1}{m}$

if $dx = 1$
$dy = m$

(x) Draw a line such that there is no pixel gap

- Input (x_0, y_0) (x_1, y_1)
- calculate dy & dx
- calculate max(dx, dy)
- if max = dx

dy	dx
m	1

$\left[\frac{dy}{max(dx,dy)}, \frac{dx}{max(dx,dy)}\right]$

- if max = dy

dy	dx
1	1/m

- Pixel gap :

* A pixel has 8 neighbouring pixels
* if the next plotted point is outside the 8 neighbouring pixels there is a pixel gap.

no pixel gap

pixel gap

DDA → Digital ~~Data~~ differential analysis

```
step = max(dy, dx);
dx /= step;  dy /= step;
```

DDA algorithm

```
i = 0;
while (i <= step) {
    drawPixel(x, y);
    x += dx;   y += dy;
    i++;
}
```

* DrawPixel takes integer as inputs

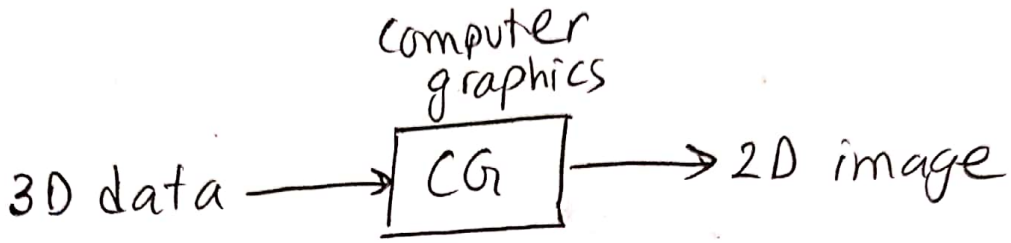
New

```
DrawPixel( int(x + sign(dx) * 0.5),
           int(y + sign(dy) * 0.5) );
```

old

```
drawPixel(round(x), round(y))
    round(x) = float(x) - integer(x)
    if round(x) > 0.5
            integer(x) + 1
    else
            0
```

DDA algorithm: if it has floating point rather than integer, it has 6 times step than with integer.



3D data:

① Model: ~~DA~~ Dummy copy of real 3D data, with all the properties. Prototype of real world.

- Geometric model

$(x-a)^2 + (y-b)^2 + (z-c)^2 - r^2 = 0$

a sphere with centre (a, b, c) and radius r.

- Motion Capturing
- Rendering Pipeline:

↓

Drawing by machine

Pipeline: sequentially in order

(4) scan conversion algorithm:

Colored integer pixel co-ordinate Graphics card for scan conversion algorithm.

Graphics

Line drawing Algorithms

drawPixel (x, y);

drawPixel (int(x+0.5), int (y+0.5))

Mid-point Line drawing algorithm

or, Bresenham's Line drawing algorithm

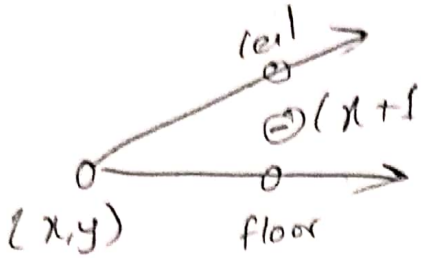
$y = mx + c$ —— (I)

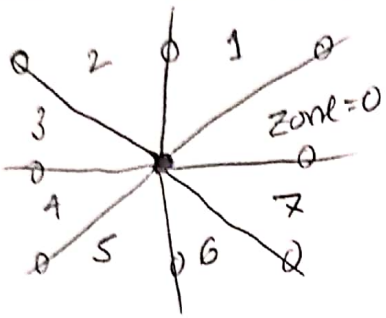
$Ax + By + C = 0$ —— (II)

$A = dy$

$B = -dx$

If $Ax+By+C=0$ then (x,y) midpoint.





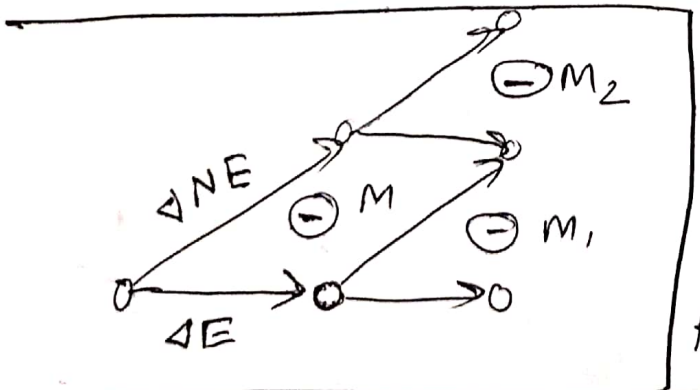
Each pixel is surrounded by 8 pixels.

zone = 0 x, y both increment
zone = 5 x, y both decrement.

zone = 0 $(x+1, y+\frac{1}{2})$ ~~So, $(x+1, y+\frac{1}{2})$~~

Let zone = 0

- dx, dy = +ve
- $dx \geq dy$ < $dx > dy$ / $dx = dy$ [boundary]



$f(M) = A(x+1) + B(y+\frac{1}{2}) + C = d$

$f(M_1) = A(x+2) + B(y+\frac{1}{2}) + C = d_1$

$f(M_2) = A(x+2) + B(y+\frac{3}{2}) + C = d_2$

$(d_1 - d) = A$

$\Delta E = d_1 - d = A$

$d_2 - d = A + B$

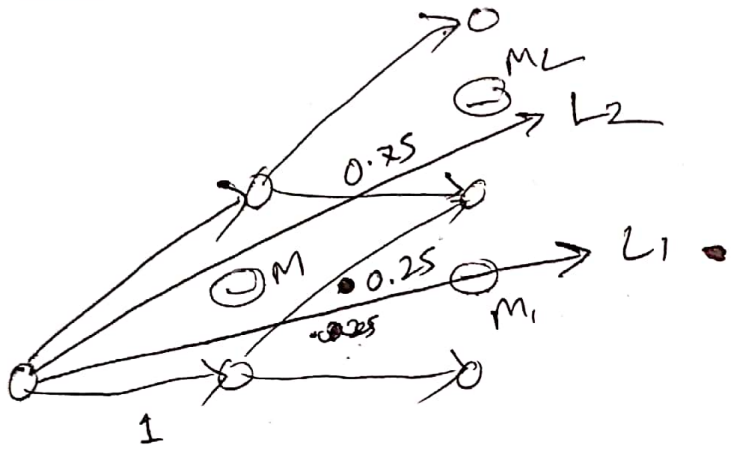
$\Delta NE = A + B$

rate of change of d due to east movement.

$$d_{init} = A(x_0+1) + B\left(y_0+\frac{1}{2}\right) + C$$

$$f(x,y) = A(x_0) + By_0 + C = 0$$

$$\therefore \boxed{d_{init} = A + \frac{B}{2} = dy - \frac{dx}{2}}$$



d_{init} for L_1?

$$\frac{dy}{dx} = 0.25$$

$$d_{init} = -0.25\,dx$$

$\Rightarrow < 0$

Derive initial derivation and the derivatives of a line in zone 0

```
void drawLine0(int x0, int y0, int x1, int y1){
    int dx = x1 - x0, dy = y1 - y0;
    int delE = 2*dy, delNE = 2(dy-dx);
    int d = 2*dy - dx;
    int x = x0, y = y0;
    drawPixel(x0, y0);
```

```
while(x<x1) {
    if(d<0) {//ΔE
        d+=delE;
        x++;
    }
    else {//ΔNE
        d+=delNE;
        x++; y++;
    }
    drawPixel(x,y);
}
}
```

(-20,30) to (10,50)

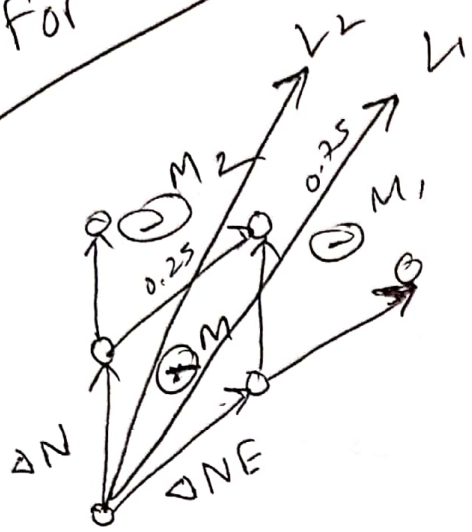
$dx = 30$ $dy = 20$

$d = \left(dy - \frac{dx}{2}\right) * 2$

ΔE = 40 ΔNE = -20

x	y	d	ΔE / ΔNE
-20	30	10	ΔNE
-19	31	-10	ΔE
-18	31	30	ΔNE
-17	32	10	ΔNE
-16	33	-10	ΔE
-15	33	30	ΔNE

For zone 1



$$F(M) = A(x+\tfrac{1}{2}) + B(y+1) + C = d$$
$$f(M_1) = A(x+\tfrac{3}{2}) + B(y+2) + C = d_1$$
$$F(M_2) = A(x+\tfrac{1}{2}) + B(y+2) + C = d_2$$

$$\Delta N = d_2 - d = B$$

$$\Delta NE = d_1 - d = A + B$$

$$d_{init} = A(x_0+\tfrac{1}{2}) + B(y_0+1) + C$$

$$= \frac{A}{2} + B \qquad [Ax_0 + By_0 + C = 0]$$

$$d_{init} = \frac{dy}{2} + -dx$$

$L_2 \rightarrow \frac{dy}{dx} = 0.25$ ....

-ve → ΔNE

+ve → ΔN

Do any two derivation of any 2 zones with code, derivation and simulation

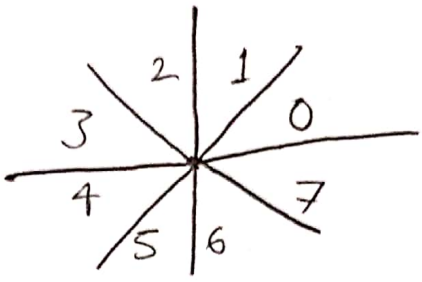
Assignment

Monday

Graphics
~~NUM~~

Slope dependent line

Algorithm : 1



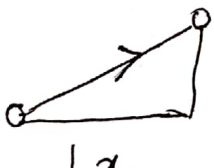
0, 7, 3, 4
1, 2, 5, 6

① Determine slope
② Run Accordingly

Algorithm: 2

① Determine slope
② Convert slope into a specific one (say zone)
③ Draw in original zone but process in zone 0.

Algorithm 1 → bigger, code for each zone
2 → smaller, complexity same, code for one zone



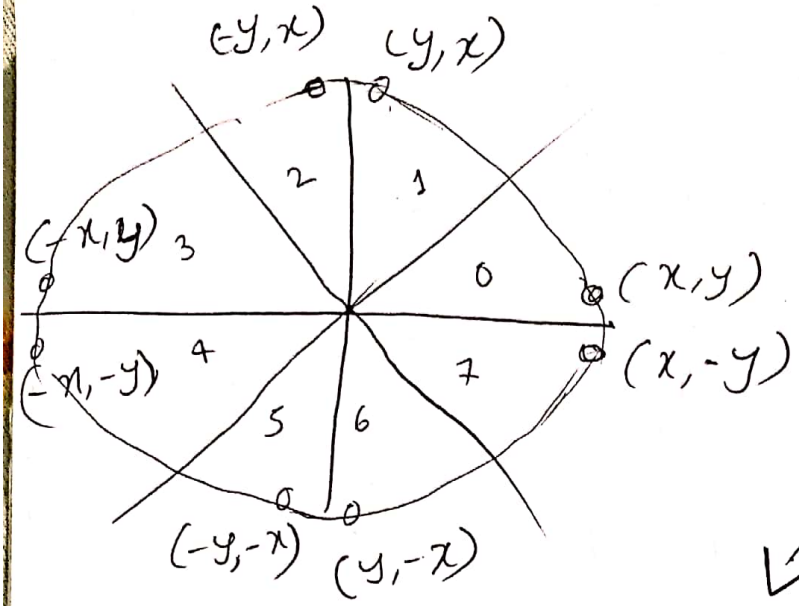
8 combinations of dx & dy possible.

Zone selection	dx +ve, dy +ve → Zone 0, 1
dx=dy 0/1	dx>dy zone 0 dy>dx zone 1

```
void drawline (int x0, int y0, int x1, int y1) {
    int dx = x1 - x0 , dy = y1 - y0 ;

    if (abs(dx) >= abs(dy)) {
        if (dx > 0 && dy > 0)  drawline0(x0, y0, x1, y1);
        if (dx > 0 && dy <= 0) drawline7(x0, y0, x1, y1);
        if (dx <= 0 && dy > 0) drawline3(x0, y0, x1, y1);
        if (dx <= 0 && dy <= 0) drawline4(x0, y0, x1, y1);
    }
    else {
        if (dx > 0 && dy > 0)  drawline1(x0, y0, x1, y1);
        if (dx <= 0 && dy > 0) drawline2(x0, y0, x1, y1);
        if (dx <= 0 && dy <= 0) drawline5(x0, y0, x1, y1);
        if (dx > 0 && dy <= 0) drawline6(x0, y0, x1, y1);
    }
}
```

Algorithm 1



8 way symmetry

Any zone → zone 0

Any zone		zone 0
zone 1	(x,y)	(y,x)
zone 2	(x,y)	(y,-x)
zone 3	(x,y)	(-x,y)
zone 4	(x,y)	(-x,-y)
zone 5	(x,y)	(-y,-x)
zone 6	(x,y)	(-y,x)
zone 7	(x,y)	(x,-y)

1 2 4 6

drawline0(y_0, x_0, y_1, x_1, 1)

Algorithm 2

1	drawline0(y_0, x_0, y_1, x_1)
2	drawline0($y_0, -x_0, y_1, -x_1$)
3	drawline0($-x_0, y_0, -x_1, y_0$)
4	drawline0($-x_0, -y_0, -x_1, -y_1$)
5	drawline0($-y_0, -x_0, -y_1, -x_1$)
6	drawline0($-y_0, x_0, -y_1, x_1$)
7	drawline0($x_0, -y_0, x_1, -y_1$)

← Another parameter needed ie zone no. to map back to the respective zones

Zone 0	any zone	
(x, y)	(y, x)	1
(x, y)	(-y, x)	2
(x, y)	(-x, y)	3
(x, y)	(-x, -y)	4
(x, y)	(-y, -x)	5
(x, y)	(y, -x)	6
(x, y)	(x, -y)	7

for converting back

```
void drawLine_0(int x0, int y0, int x1, int y1,
                int zone) {

    //same line drawing algorithm

    drawPixel(x, y, zone);

    .
    .
    .

}
```

```
void drawPixel(int x, int zone) {
               int y,
    switch {
        case 0 : glVertex2i(x, y); break;
        case 1 : glVertex2i(y, x); break;
        case 2 : glVertex2i(-y, x); break;
```

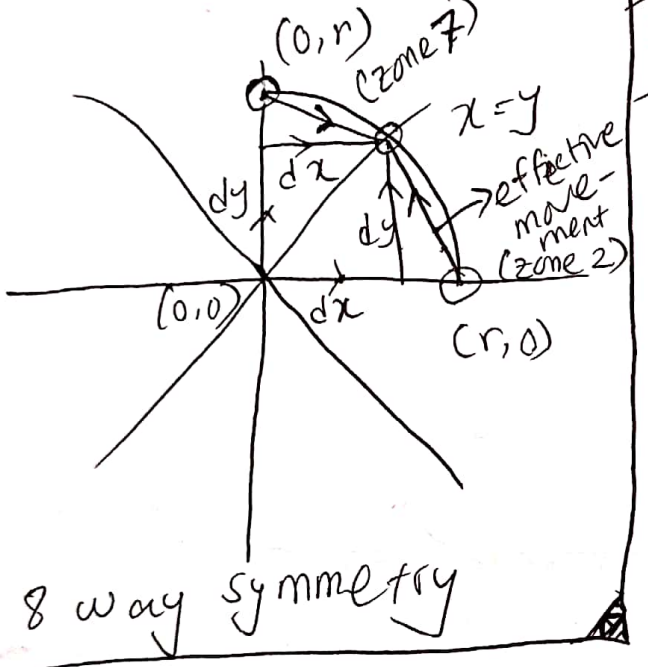
```
case 3: glVertex2i(-x, y); break;
case 4: glVertex2i(-x, -y); break;
case 5: glVertex2i(-y, -x); break;
case 6: glVertex2i(y, -x); break;
case 7: glVertex2i(x, -y); break;
```

~~NUM~~ | Graphics

Circle drawing Algorithm

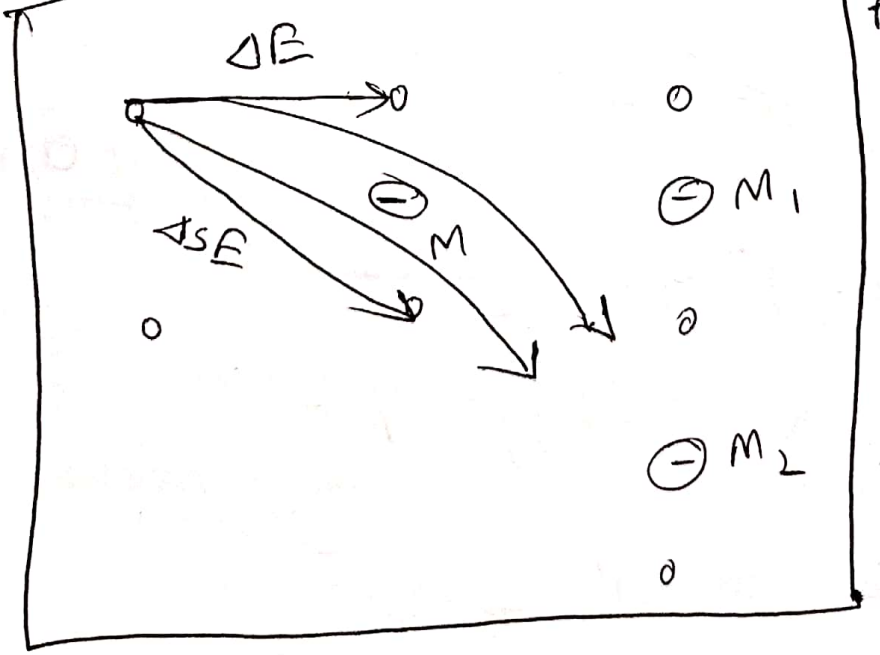
$$(x-a)^2 + (y-b)^2 = r^2 \quad \text{——— (I)}$$

$$x^2 + y^2 - r^2 = 0 \quad \text{——— (II)}$$



8 way symmetry

Read
- Why a & b don't matter?
- Zone controller
- Direction & zone relation
- We pick (0,r) or (r,0)?



(0,r) Picked zone 7

$f(M) = (x+1)^2 + (y-\frac{1}{2})^2 - r^2 = d$

$f(M_1) = (x+2)^2 + (y-\frac{1}{2})^2 - r^2 = d_1$

$f(M_2) = (x+2)^2 + (y-\frac{3}{2})^2 - r^2 = d_2$

Now,

$\Delta E = d_1 - d$

$= 4x - 2x + 4 - 1$

$= 2x + 3$

$\Delta SE = d_2 - d$

$= 2x + 3 - \frac{3}{2} \times 2y + 2y + \frac{9}{4} - \frac{1}{4}$

$= 2x + 3 - 2y + 2$

$= 2x - 2y + 5$

~~$d_{init} = (x+1)^2 + (y - \frac{1}{2})^2 = r^2$ (1, r)~~

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

~~$= 1^2 + r^2 = r^2$~~

(0, r)

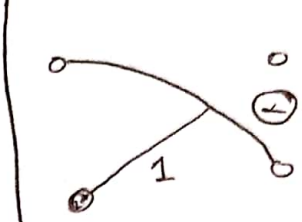
$d_{init} = (x+1)^2 + (y - \frac{1}{2})^2 - r^2$

$= (0+1) + (r - \frac{1}{2})^2 - r^2$

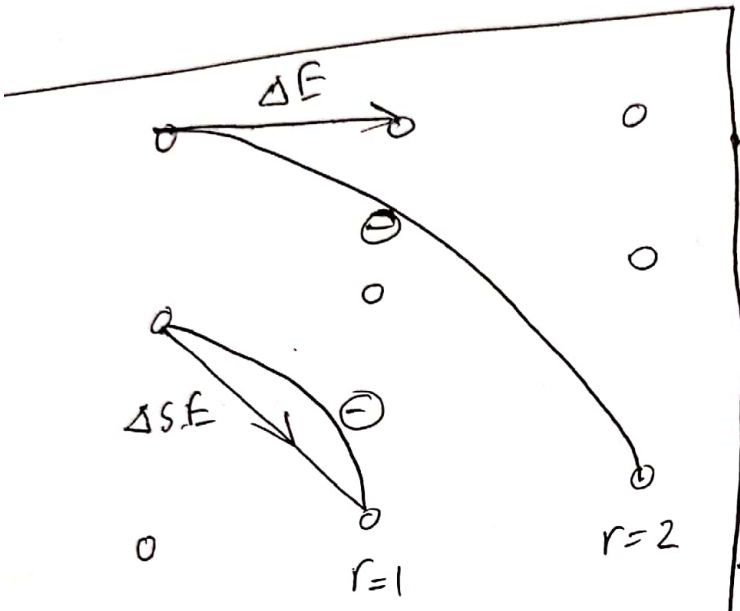
$= 1 + r^2 - r + \frac{1}{4} - r^2$

$= \frac{5}{4} - r.$

$x^2+y^2 < r^2$ -ve



radius 1 pixel



For r = 1, the circle goes from below mid

for r > 1, the circle goes above mid.

$d = \frac{5}{4} - r$ [for r = 1 +ve, r > 1 -ve]

-ve ΔE
+ve ΔSE

let zone7 = region1 for circle

Code

```
void drawCircle1(int r){
    int x=0, y=r;
    int d = 5-4*r;
    draw8way(x,y); // last class draw pixel
    while(x<y){
        if(d<0) { // ΔE
            d += 4*(2*x+3);
            x++; }
```

```
else { //ΔSE
    d+= 4*(2*x - 2*y + 5)
    x++;
    y--;
}
draw8way(x,y);
}
}
```

8 - 14+5

```
draw8way(x,y){
    for(i=0, i<8; i++) drawPixel(x,y,i);
}
```

* Draw a circle for r = 8 ; center (0,0)

x	y	d	ΔE/ΔSE
0	8	-27	ΔE
1	8	-15	ΔE
2	8	5	ΔSE
3	7	-23	ΔE
4	7	13	ΔSE
5	6	9	ΔSE

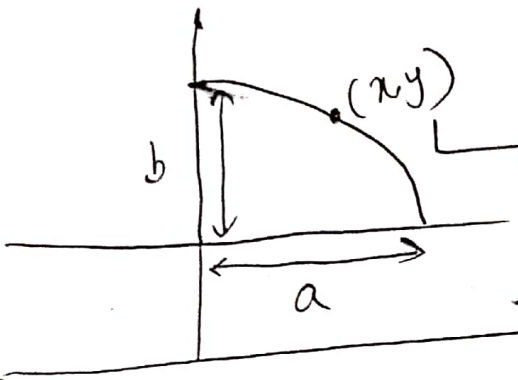
d = 5 - 4r
= 5 - 32
= -27

36
-23
13

Graphics

Ellipse Drawing

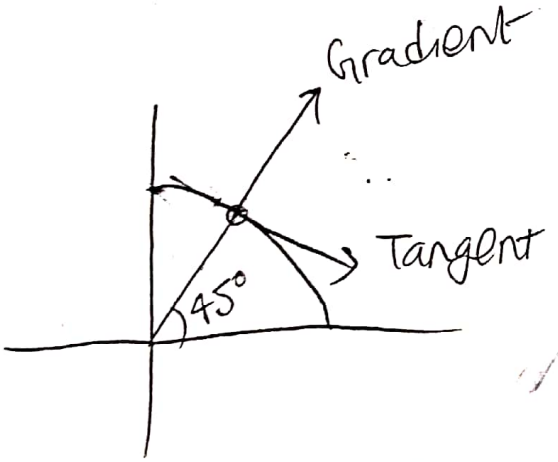
4 way symmetry

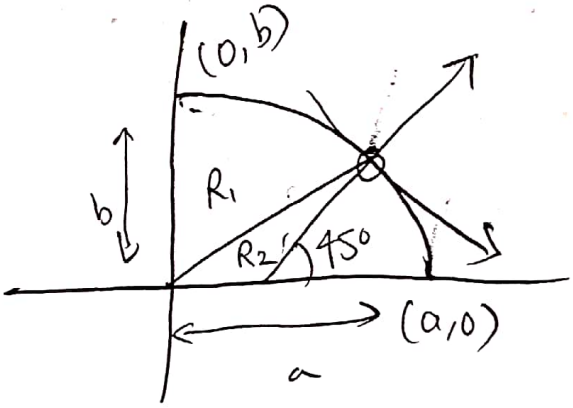


→ other points $(-x, y)$, $(x, -y)$, $(-x, -y)$
[No swapping necessary]

$$\left(\frac{x}{a}\right)^2 + \left(\frac{y}{b}\right)^2 = 1 \quad \text{——} \quad (1)$$

$$\Rightarrow b^2x^2 + a^2y^2 - a^2b^2 = 0 \quad \text{——} \quad (11)$$ [Implicit form]





As long as $j > i$ → region 1
$i = j$ → in between region 1 & 2
$i > j$ → region 2

Circle gradient vector:

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

Gradient $\Rightarrow 2xi+2yj=0$

$\Rightarrow 2x\,dx+2y\,dy=0$

Ellipse:

Gradient $\Rightarrow 2b^2xi+2a^2yj=0$

↳ (grad(x,y))

Magnitude of x component $= 2b^2x$

Magnitude of y component $= 2a^2y$

Region 1 → like circle [zone 7]

Region 2 → zone 6

Derivation for R-1

$M=(x_p+1,\ y_p-\frac{1}{2})$ $\quad M_1=(x+2,\ y-\frac{1}{2})$

$M_2=(x+2,\ y-\frac{3}{2})$

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

$f(M_1)=b^2(x+2)^2+a^2(y-\frac{1}{2})^2-a^2b^2=d_1$

$f(M_2)=b^2(x+2)^2+a^2(y-\frac{1}{2})^2-a^2b^2=d_2$

$$\therefore \Delta E = d_1 - d$$

$$= b^2(2x+3)$$

$$\Delta SE = d_2 - d$$

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

$\boxed{(0, b)}$

$$d_{init} = b^2(x+1)^2 + a^2\left(y-\frac{1}{2}\right)^2 - a^2b^2$$

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

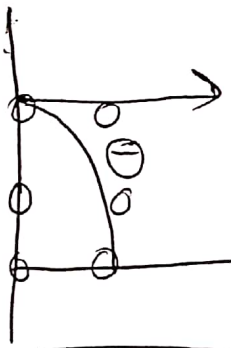
$$= b^2 + a^2\left(b^2 - b + \frac{1}{4}\right) - a^2b^2$$

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

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

d -ve ΔE

+ve ΔSE

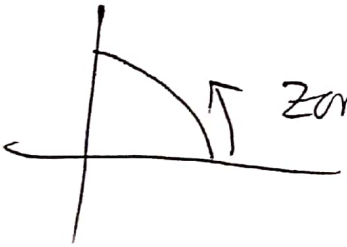


-ve for

$a = b = 1$

$a = 1, b = 2$

-ve $\rightarrow \Delta E$



Code:

```
void drawEllipse(int a, int b) {
    int x=0, y=b;
    int d = 4(b^2 - a^2 b) + a^2;
//int delE = 4b^2(2x+3); x
//int delSE = 4b^2(2x+3) + 4a^2(-2y+2); x
    drawfourway(x,y);
    while(2b^2(x+1) < 2a^2(y-1/2)){
        if(d<0){ //ΔE
            d+ = 4b^2(2x+3);
            x++;
        }
        else { //ΔSE
            d+ = 4b^2(2x+3) + 4a^2(-2y+2);
            x++;
            y--;
        }
        drawfourway(x,y);
    }
}
```

grad(x,y) = $2b^2x\,i + 2a^2y\,j = 0$

---

region 2 → termination (y > 0)

ΔS, ΔSE
(+ve) (-ve)

$M = (x+\frac{1}{2}, y-1)$ $M_2 = (x+\frac{3}{2}, y-2)$

$M_1 = (x+\frac{1}{2}, y-2)$

Graphics

$$\frac{x}{7} + \frac{y}{5} = 1 \quad \text{— (1)}$$

(0,5) to (7,0)

$$y = -\frac{5}{7}x + 1$$

Steps
1. find the zone
2. Compare the pixel co-ordinates from midpoint & DDA algorithms

(1) Zone 7 [x inc., y dec. $dx > dy$, dx +ve dy -ve]

since, $|dy| < (dx)$

so, 1/3/4/7

$dy < 0$ & $dx \geq 0$ zone 7

$dy \geq 0$ & $dx < 0$ zone 3

---

(2)

$$f(M) = A(x+1) + B\left(y - \frac{1}{2}\right) + C = d$$

$$f(M_1) = A(x+2) + B\left(y - \frac{1}{2}\right) + C = d_1$$

$$f(M_2) = A(x+2) + B\left(y - \frac{3}{2}\right) + C = d_2$$

Mid point (1)

$$\Delta E = A = dy$$

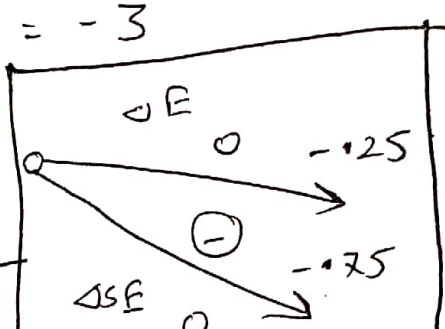
$$\Delta SE = A - B = dy + dx$$

$$d_{init} = A - \frac{B}{2} = dy + \frac{dx}{2}$$

$dx = 7$
$dy = -5$

$\Delta E = 2dy = -10$
$\Delta SE = 2(dy + dx) = 4$
$dinit = 2dy + dx$
$= -3$

+ve ΔE
-ve ΔSE



if $dy > 0.5$ -ve else +ve
$dx = 1$
$dinit = dy + \frac{dx}{2}$
$= dy + 0.5$

x	y	d	ΔE/ΔSE
0	5	-3	ΔSE
1	4	1	ΔE
2	4	-9	ΔSE
3	3	-5	ΔSE
4	2	-1	ΔSE
5	1	3	ΔE
6	1	-7	ΔSE
7	0	-3	ΔSE

$\Delta E = x++ ; d+ = -16$
$\Delta SE = x++ ; y-- ; d+ = 4$

(11) DDA

$dx = 7$, $dy = -5$

$step = 7$

$dx = 1$, $dy = \frac{-5}{7}$

```
i=0;
x=0, y=5;
while (i<step) {
    drawPixel(x, y);
    x+=dx;  y+=dy;
    i++;
}
```

```
drawPixel(int(x+sign(dx)*0.5),
          int(y+sign(dy)*0.5));
```

x		y	
0	0	5	5
1	1	30/7	3
Alternate:		// no rounding	
0	0	5	5
1	1	4.28	4
2	2	3.57	4
3	3	2.85	3
4	4	2.14	2
5	5	1.42	1
6	6	0.71	1
7	7	0	0

$\frac{30}{7} - \frac{1}{2}$

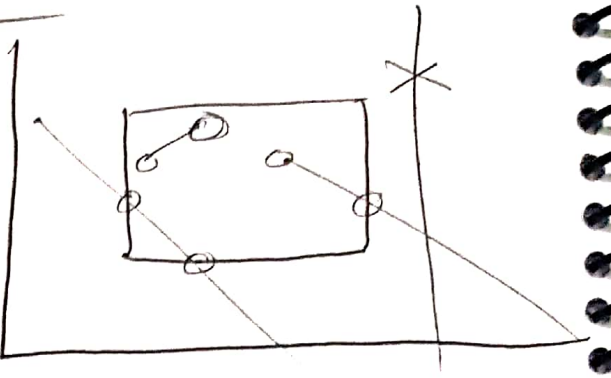
$= \frac{53}{14} \approx 3$

But $\frac{30}{7} \approx 4$

So, rounding for +ve values bad

Graphics

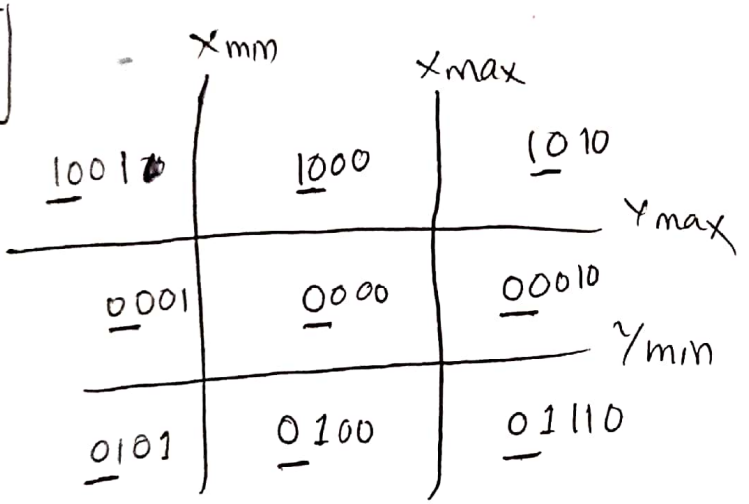
Line-clipping Algorithm



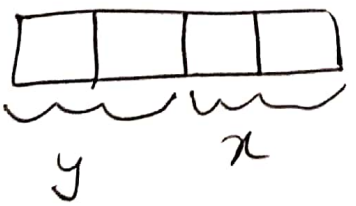
1/ Cohen-Sutherland LCA:

- Equally applicable for 2D & 3D
* Create binary code of the end points of the line

2D



Region outcode:



→ 4 bit code

if y > ymax ↑ 10

else if y < ymin ↓ 01

within range 00

→ same for x

```
int makeCode(int x, int y) {
    int code = 0;

    int TOP = 8, BOTTOM = 4, RIGHT = 2, LEFT = 1;
    int ymax = 100, ymin = -100,
        xmax = 150, xmin = -150;

    if(y > ymax)
        code += TOP;
    else if(y < ymin)
        code += BOTTOM;

    if(x > xmax)
        code += RIGHT;
    else if(x < xmin)
        code += LEFT;

    return code;
}
```

↑↑ Global

Region Outcode generation

* checking if the line can be accepted.
  - 0000 → trivially accepted (bitwise OR = 0)
  - common position of 1 → completely rejected / trivially rejected
  - bitwise AND Nonzero
  - otherwise check

3D → Far 16 (-ve)
Near 32 (+ve)

z	y	x

Quiz → 20 3D endpoints, find outcode | check & comment if accepted, rejected or partial

```
void CohenSatherland (int x0, int y0, int x1, int y1)
{
    int Code0, Code1, Code, x, y;
    Code0 = makeCode(x0,y0);
    Code1 = makeCode(x1,y1);
    While(1){
      if (Code0 | Code1 == 0) { //Accepted
            drawLine (x0, y0, x1, y1);
            break;
      }
```

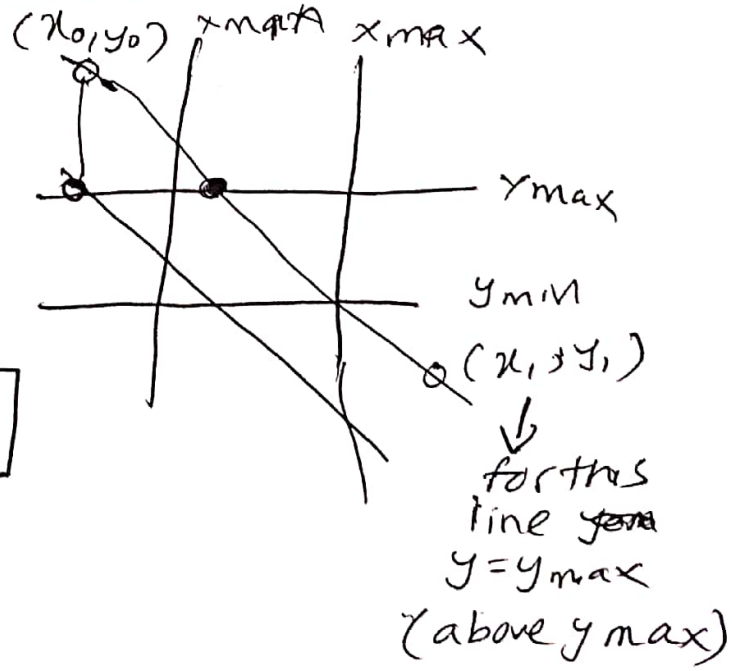
```
if (code0 & code1 != 0) { //Rejected
        break;
}
else { // Partial
    if (code0)  code = code0;
    else  code = code1;
    if (code & TOP) { //above ymax
        y = ymax;
        x = x0 + (ymax - y0) * (x1 - x0)/(y1 - y0);
    }
    else if (code & Bottom) { //below ymin
        y = ymin;
        x = x0 + (ymax - y0) * (x1 - x0)/(y1 - y0);
    }
}
```

$$x_t = x_0 + t(x_1 - x_0)$$

$$y_t = y_0 + t(y_1 - y_0)$$

$$t = \frac{y_t - y_0}{y_1 - y_0}$$

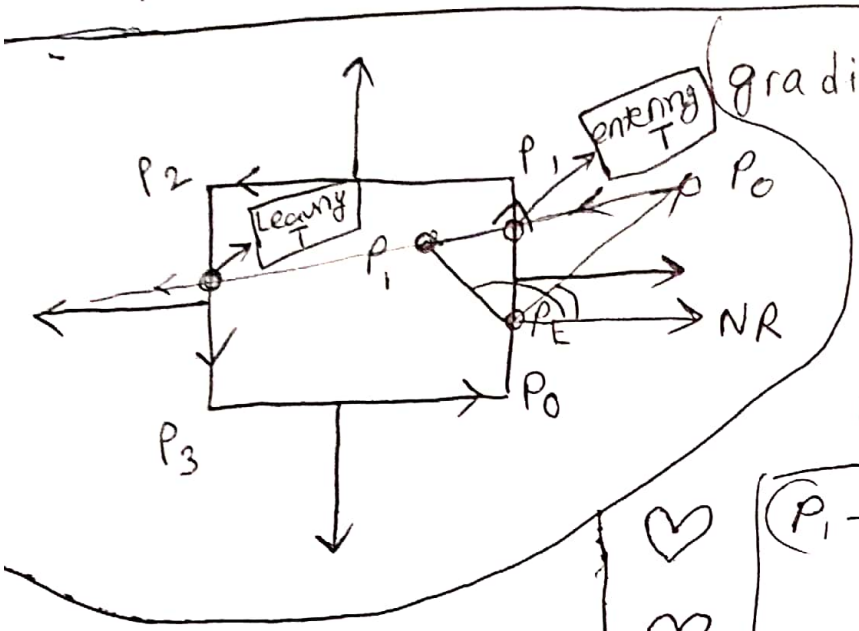
$$\therefore x_t = x_0 + \left(\frac{y_t - y_0}{y_1 - y_0}\right) * (x_1 - x_0)$$



```
else if (code & Right) { // above xmax
        x = xmax;
        y = y0 + (xmax - x0) * (y1 - y0)/(x1 - x0);
}
else {   // below xmin
    x = xmin
    y = y0 + (xmin - x0) * (y1 - y0) / (x1 - x0);
}
if (code == code0) {
    x0 = x;  y0 = y;
    code0 = makeCode(x0, y0);
}
else {
    x1 = x;  y1 = y;
    code1 = makeCode(x1, y1);
}
}
}
```

Graphics

Parametric Line Clipping Algorithm:



$(P_0 - P_E) \cdot N_R = +ve$

$(P_1 - P_E) \cdot N_R = -ve$

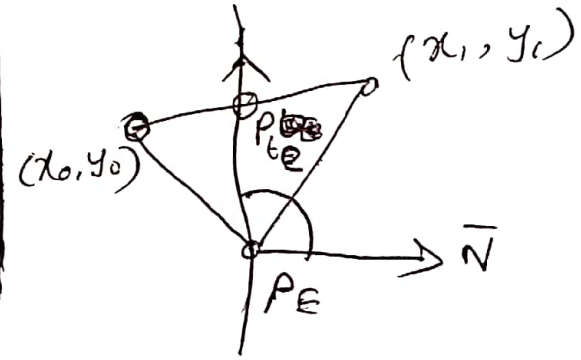
$(P_1 - P_0) N_R = -ve$ entering

$+ve$ leaving

$(T_E(max), T_L(min))$ → Line

$T_E \rightarrow$ entering T

$T_L \rightarrow$ Leaving T



$$P(t) = P_0 + t(P_1 - P_0) \quad \text{——— (I)}$$

$$(P_{te} - P_E) \cdot N = 0 \quad \text{——— (II)}$$

$$(P_0 + t_e(P_1 - P_0) - P_E) \cdot N = 0 \quad [\text{From (I) \& (II)}]$$

$$\Rightarrow (P_0 - P_E) \cdot N + t_e(P_1 - P_0) N = 0$$

$$\Rightarrow t_e = -\frac{(P_0 - P_E).N}{(P_1 - P_0).N} \quad \text{---(iii)}$$

$$t_e = \frac{(x_0 - x_e)N_x + (y_0 - y_e)N_y}{(x_1 - x_0)N_x + (y_1 - y_0)N_y}$$

$$= \frac{(x_0 - x_E)N_x}{(x_1 - x_0)N_x} \quad [N_y = 0]$$

$$= \frac{x_0 - x_{max}}{x_1 - x_0} \quad x_E = x_{max}, \; N_x = 1.$$

→ we have taken N to be $(0,1)$, $(0,-1)$, $(1,0)$ & $(-1,0)$ depending on which ~~do~~ ~~bon~~ boundary.

In this case $N = (1,0)$.

for, P_E, $x_E = x_{max}$ [~~con~~ assuming right boundary].

and find y interms of P_0 & P_1

Graphics

(*) Cyrus-Beak LCA:

$$t = \frac{-(P_0 - P_E).N}{(P_1 - P_0).N}$$

A list of 't' for all edges:

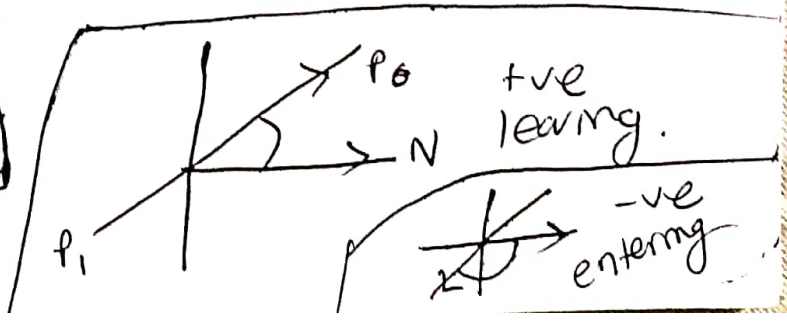
Boundary	Normal	$(P_0 - P_E).N$	$(P_1 - P_0).N$	t
TOP	(0,1)	$y_0 - y_{max}$	$y_1 - y_0$	$\frac{y_0 - y_{max}}{y_1 - y_0}$
BOTTOM	(0,-1)	$y_{min} - y_0$	$y_0 - y_1$	$\frac{y_{min} - y_0}{y_0 - y_1}$
LEFT	(-1,0)	$x_{min} - x_0$	$x_0 - x_1$	$\frac{x_{min} - x_0}{x_0 - x_1}$
RIGHT	(1,0)	$x_0 - x_{max}$	$x_1 - x_0$	$\frac{x_0 - x_{max}}{x_1 - x_0}$

$t > 0 \mid t < 1 \rightarrow$ extend করলে will intersect

Entering $t \rightarrow t_E$
Leaving $t \rightarrow t_L$

$(P_1 - P_0).N \rightarrow$ +ve leaving
-ve entering

if right leaving then left entering



According to Cyrus Beak, line is from

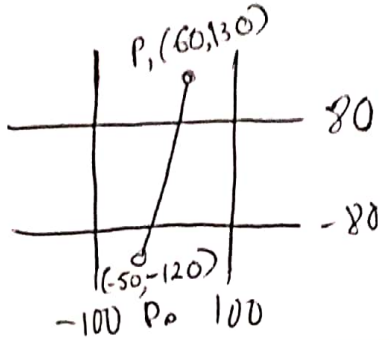
$T_{E\,max}$ to $T_{L\,min}$

↓ new (x_0, y_0)

↓ new (x_1, y_1)

Start value $T_E = 0$

$T_L = 1$



Determine t for all edges.

$$TOP = -\frac{y_0 - y_{max}}{y_1 - y_0} = -\frac{-120 + 80}{130 + 120} = 0.8 \quad t_L$$

$$BOTTOM = -\frac{y_{min} - y_0}{y_0 - y_1} = -\frac{-80 + 120}{-120 - 130} = 0.16 \quad t_E$$

$$LEFT = -\frac{x_{min} - x_0}{x_0 - x_1} = -\frac{-100 + 50}{-50 - 60} = -0.45 \quad t_E$$

$$RIGHT = -\frac{x_0 - x_{max}}{x_1 - x_0} = -\frac{-50 - 100}{60 + 50} = 1.36 \quad t_L$$

As, $y_1 > y_0$ top $t = t_L$

Bottom $t_d = t_E$

As $x_1 > x_0$ Right $t = t_L$

Left $t = t_E$

Here t_L min = 0.8.

$t_{E\,max} = 0.16$

Line is from 0.16 to 0.8

Do with if else

then use $x_t = x_0 + t(x_1 - x_0)$

$y_t = y_0 + t(x_1 - x_0)$

for t_E & t_L to find clipped points

if $(t_E > t_L)$ reject

Quiz

- Region outcode & comment if accepted, rejected as partial
- Find 4 ∈ & set for a set & label t_E & t_L

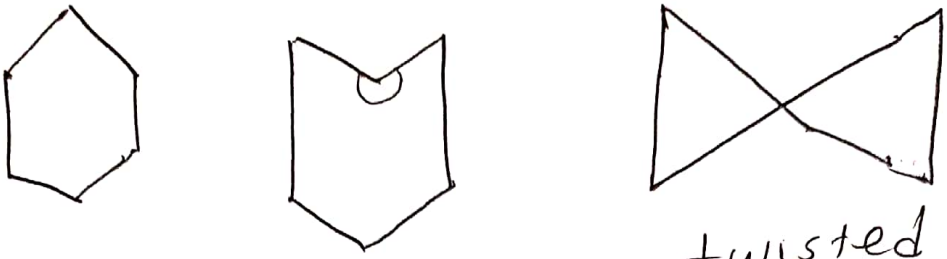
(*) Reject ection cond → $t_{Emax} > t_{Lmin}$

(*) if accepted then find x & y of the t from the parametric equation

---

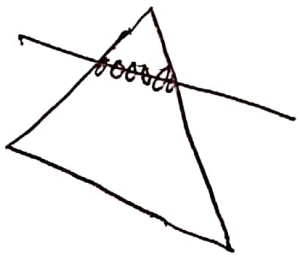
Graphics

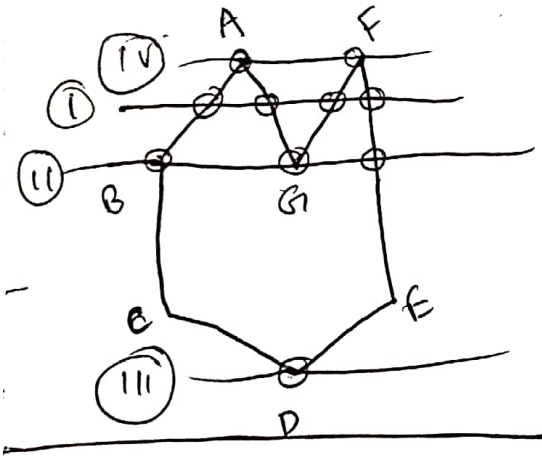
Polygon Filling Algorithms



	Convex	concave	twisted
opengl	✓	×	×
this.algo	✓	✓	×

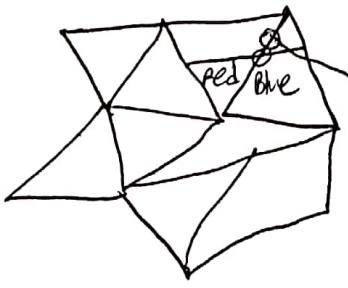
Scan line Algorithm:





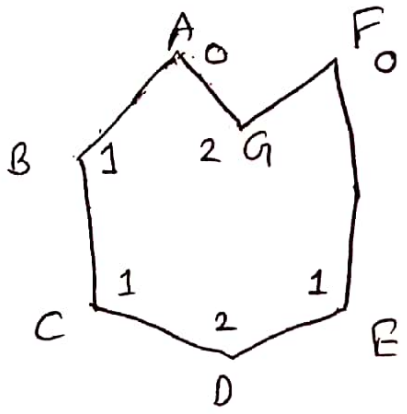
works for (I) but not (II) & (III) & (IV)

parity system:
- 1st intersection with line start filling
- next intersection stop filling.



polygonal mesh
→ This pixels (boundary pixels) red/blue?
- boundary pixels chere dae kono ekta

if always nicher pixel নাহয়



G point has 2 intersection
A point has 0.
Now, the parity system can be used.

AB → B
BC → C
CD → D
DE → D
EF → E
FG → G
AG → G

B → 1
C → 1
D → 2
E → 1
G → 2
A → 0
F → 0

Edge table: → has intersection information of the ~~lines~~ edges i.e the corners

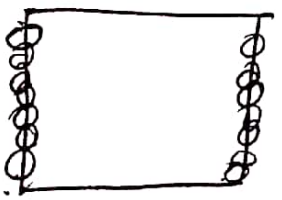
~~if x++ y ... x = 1/m~~

if y++
$x \pm = \frac{1}{m}$

active edge table

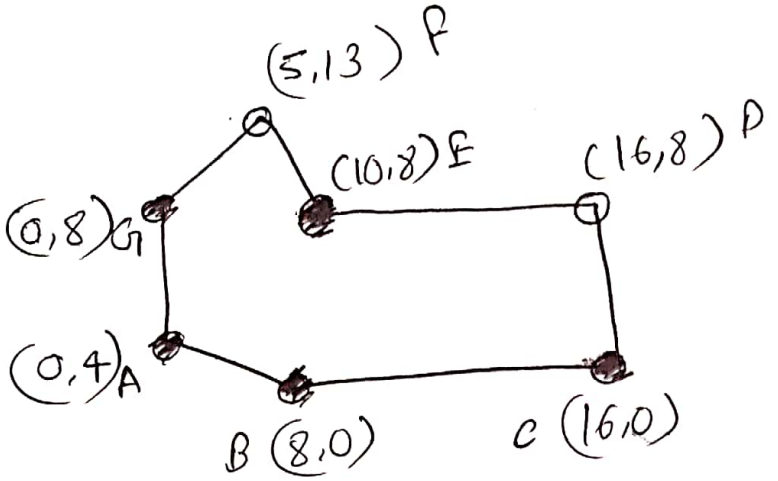
Active Edge table: Active edge table contains all the boundary points, that will be scan lined.

edge table points

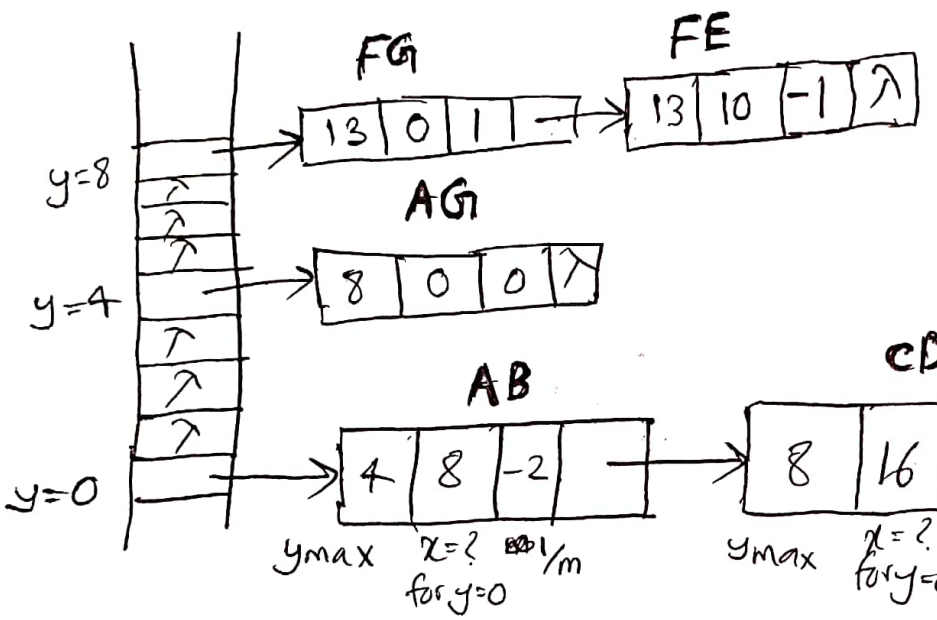


Active edge table

scan line horizontal, so only horizontal bandary points taken here.



$y = 0$ min
$y = 13$ max
λ → null / end of link list



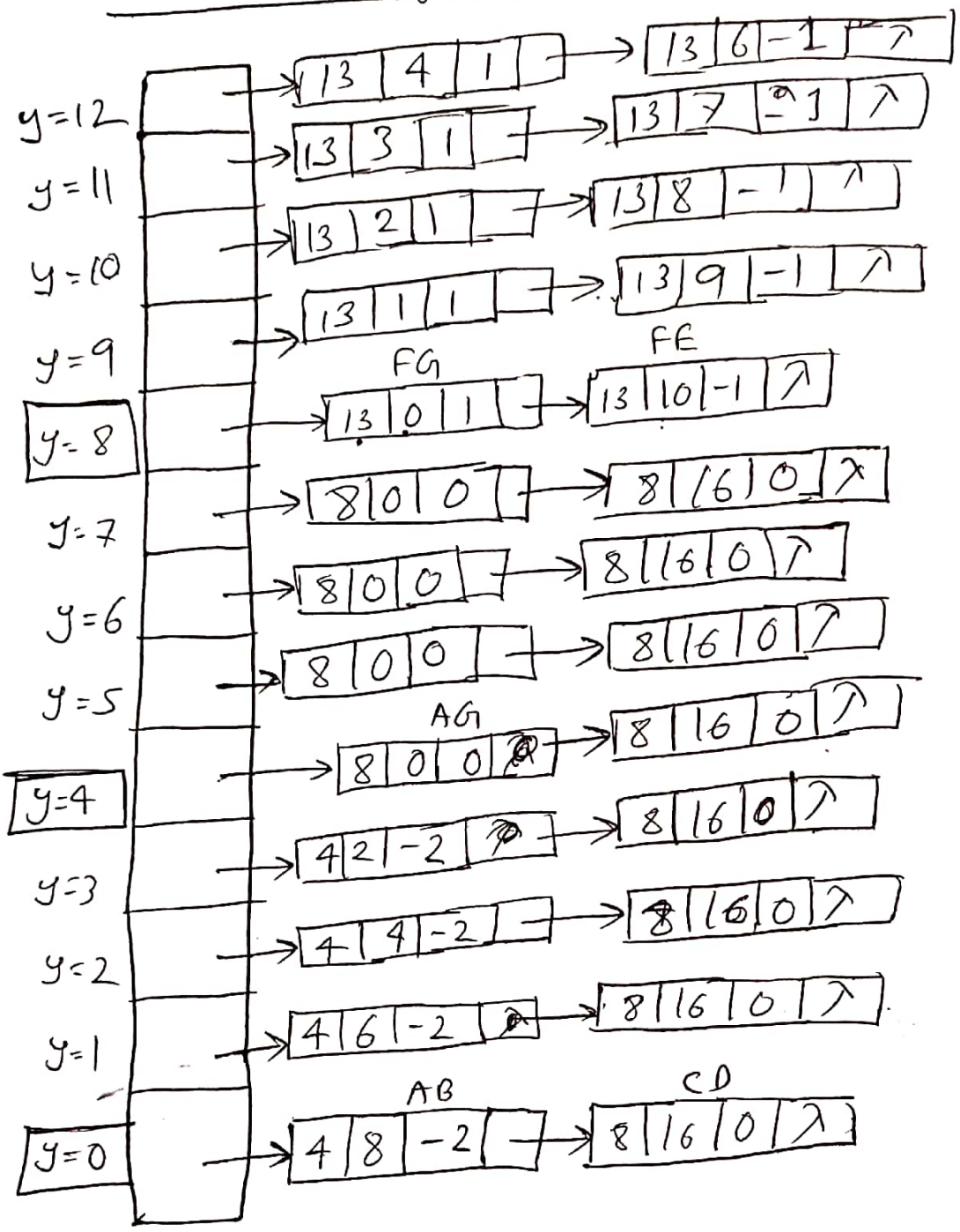
Edge Table

AB → upto $y = 3$
FG → upto $y = 12$
] bc fills upto $y_{max} - 1$

AB → till 3 then GA & CD till 7 then
FG & FE till 12

Active edge table, y++, x += $\frac{1}{m}$

Active Edge Table:

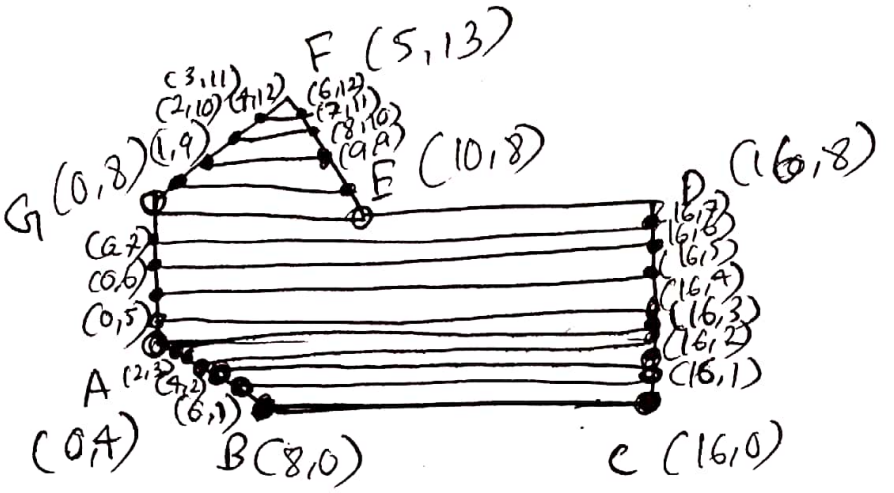


x

list must be sorted by ~~y~~ x

↳ At point 4, 8 0 0 first then

8 16 0



(8,0) → (16,0)
(6,1) → (16,1)
(4,2) → (16,2)
(2,3) → (16,3)
(0,4) → (16,4)
(0,5) → (16,5)
(0,6) → (16,6)
(0,7) → (16,7)
(0,8) → (10,8)
(1,9) → (9,9)
(2,10) → (8,10)
(3,11) → (7,11)
(4,12) → (6,12)

1/ Take edge & sort according to ymin.
if ymin same, put under same y.

2/ Fill the edge table

3/ y++, x += 1/m → fill active edge table

4/ Draw lines.

Graphics

① Draw edges

② sort by ymin

③ find max $(y_{max} - y_{min})$ and take array of that size

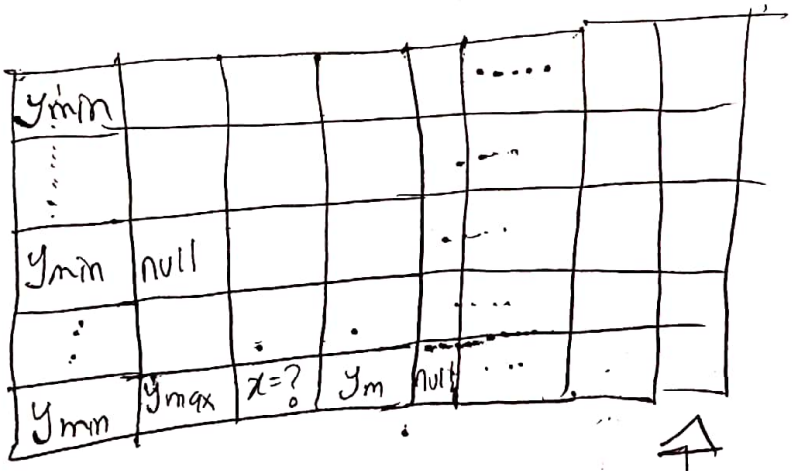
Array size = $max(y_{max} - y_{min}) * 20$

↳ 2D array

↳ if max 4 intersection per intersection 4 values.
4 * 4 = 16
extra emni → 20

poly [20] [$(y_{max} - y_{min})$]

④



x sort the ymin edges

~~Active~~ edge table

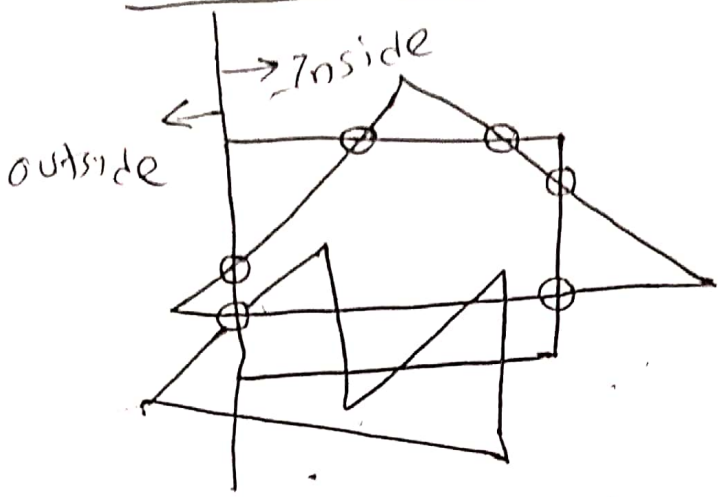
⑤ Active edge table → y++ $x += \frac{1}{m}$

Assignment write code

— Must input from a poly.txt file

Graphics

Polygon Clipping:



Sutherland-Hodgeman:

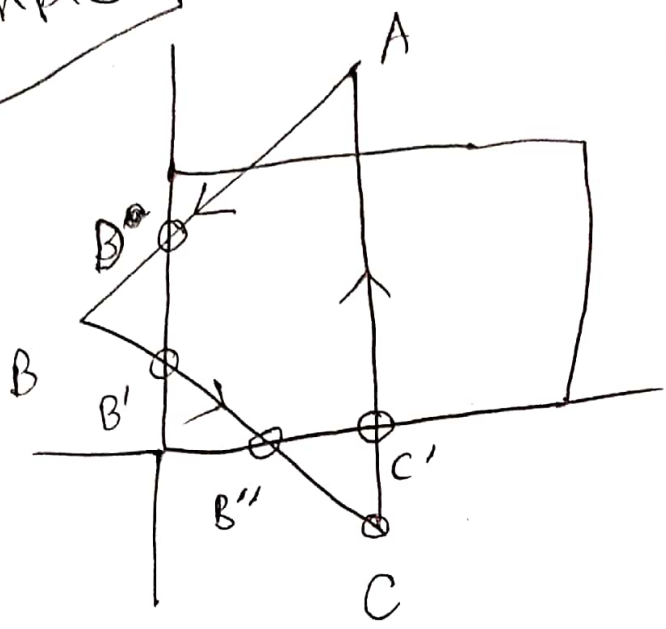
1/ leaving intersection
2/ entering intersection
3/ inside intersection
4/ outside intersection

} Intersection

Boundary group:

1/ leaving group
2/ entering group
3/ completely inside group
4/ completely outside group.

Example



① Entering : 2 vertex
↳ entering এর সময় ২ টা vertex হবে
① Border ② Endpoint

Considering left,
entering edge → BC, new output B'C
↓ ↓
border endpoint

② Inside : 1 vertex
① End point only.

③ Leaving : 1 vertex
① Border only

considering left
new output → D
leaving AB

④ Outside : 0 vertex

Considering bottom, CA → entering, → C'A
B'C → leaving, border only → B''

Similarly Top

+ve leaving -ve entering

↳ cyrus beak but t within $0 \leq t \leq 1$

inside/outside

↳ check if both end-points inside/outside

$[(P - P_E) \cdot N]$

↳ If both points +ve → outside

↳ If both points -ve → inside

$$\pi(a \pm x)(b \pm y) = \pi ab$$

$$\Rightarrow ab - ay + bx - xy = ab$$

$$\Rightarrow x(b - y) = ay$$

$$\Rightarrow x = \frac{ay}{b - y}$$

$$a^2 + b^2 = 2r^2$$