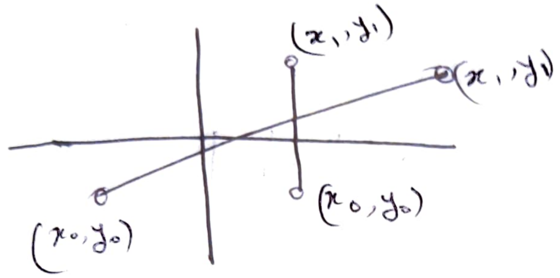
Lect - 1, Monday

9.1.23

Computer graphics course $\to$ Graphical Algorithms
$\to$ Engineering Behind Graphics



$P_0 \longrightarrow P_1$

$|v|$ $\quad \overline{P_1 - P_0}$

$$y = mx + c \quad \cdots \text{(1)}$$

$x_1 - x_0 = dx$

$y_1 - y_0 = dy$

```
for (x0 to x1) {
    doPixel(x, y);
    y += m;
}

for (y0 to y1) {
    doPixel(x, y);
    x ± 1/m;
}
```

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

P.t.o

$y = mx + c$

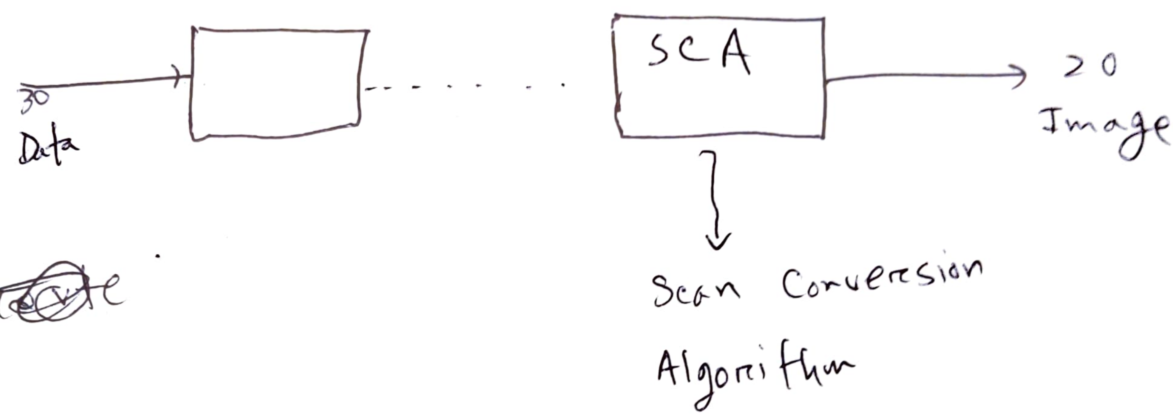
DDA Algorithm (Digital Differential Analyzer)

```
void drawLine( x0, y0, x1, y1) {
        int  dx = x1 - x0,  dy = y1 - y0 ;
        int  maxVal = max(|dx|, |dy|);
                ↓
               step
     dx = dx / step,  dy = dy / step;

     i = 0,  x = x0,  y = y0;

     while (i ≤ step) {
           drawPixel (x, y)
           x ± dx ;
           y ± dy ;
           i++;
     }
```

* 3D Data → CG → 2D Image

① Model —
  ↳ Geometric

* Rendering Pipeline
    - ↳ drawing steps used by machine
    - ↳ strictly maintains sequences



⇒ DDA
- ↳ Symmetric roundoff } minus এর দিকে symmetric roundoff করবো
- ↳ Sign

Lect-2

Wednesday

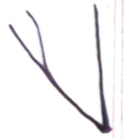
11.1.23

- Line Drawing Algorithm (গতদিন DDA - algo পড়েছি)

  ↳ drawPixel (x, y) ← floating-point values

→ drawPixel (int(x+0.5), int(y+0.5)) করলে কি হয়?

↳ 0.5 এর উপরে থাকলে উপরে যাবে, নিচে থাকলে নিচে



$dx \leq dy$

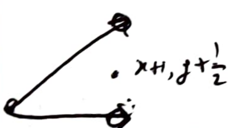
- MidPoint Line-Drawing Algo / Bresenham's Line Drawing Algo.

$y = mx + c$ —(i)

$Ax + By + C = 0$ —(ii)

$\Rightarrow$ if $m = \frac{dy}{dx}$, $A = dy$, $B = -dx$

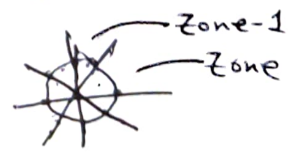
equation (ii) 0 হলে Point (x,y) is on the line.



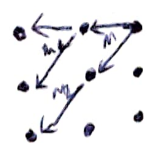
$(x+1, y+\frac{1}{2})$ point বসালে positive/negative হলে বুঝা যায় Line উপরে দিয়ে গেছে নাকি নিচে।

2nd finding of this algo → Finding equation (ii)'s return value using "Rate of change"

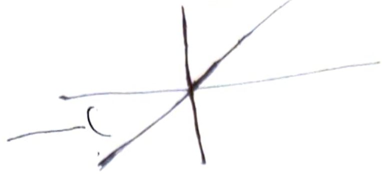
Weakness → Line এর slope দিয়ে Division করলে 8 টি group এ ভাগ করা যায়



আপাতত আলোচনা zone-0 নিয়ে



$dx > dy$,



$\frac{dy}{dx} \le 1$

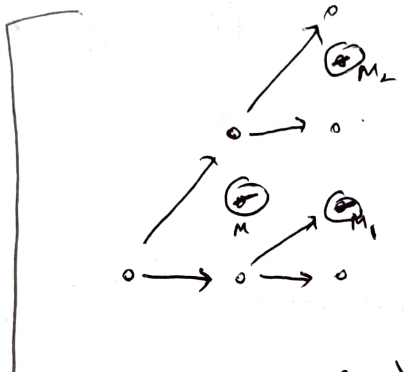
Zone-0 (0°,

→ dx, dy sign equal, $dx \ge dy$.

→ Zone-1 8°,

dx, dy sign equal, $dy \ge dx$

$dy = y_2 - y_1$



$f(x, y) = Ax + By + C = 0$ (from eq-2)

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

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

$d_1 - d = A$ ($x=1$ increment এ "deviation এর change" হলো $= A$)

→ Next Pixel East এ হলে আমাদের deviation এর সবসময় একটা নির্দিষ্ট হবে। so,

$\Delta E = d_1 - d = A$ (East Δ deviation) —— (iii)
$= dy$

likewise, $\Delta NE = d_2 - d = A + B = dy - dx < 0$ —— (iv)

if $\Delta E \ge 0$ then $\Delta NE < 0$.

→ Finding first value of d_{init}

$= A(x_0+1) + B\left(y_0 + \frac{1}{2}\right) + C$ (first point is x_0, y_0)

we know, $Ax_0 + By_0 + C = 0$

so, $\boxed{A + \frac{B}{2} = d_{init}}$

dinit derivatives of zone-0

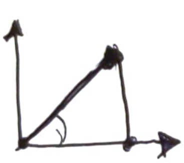
Q1. Derive ~~initial derivation~~

as,

$A = dy \quad B = -dx, \qquad d_{init} = dy - \frac{dx}{2}$ —— ✓

```
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);
    }
}
```

2 গুণ to avoid fraction ↓ we only care about d's 'sign'



$x = -0.8$, $y = 1$

$0.8 - 0.5$

dinit >

$\frac{-0.8}{2}$ -0.8

$d < 0$ $d + delW$

$d > 0$ $d + delSW$

$\frac{-1}{2} + 0.4$

$-0.5 + 0.4$

$= 0.1$

→ "দিয়ে একটি Line (-20,30) থেকে (10,50) তে যাবে, এটার simulation করো"

$dx = 30$, $dy = 20$; $d_{init} = 5$ ($dy - \frac{dx}{2}$)

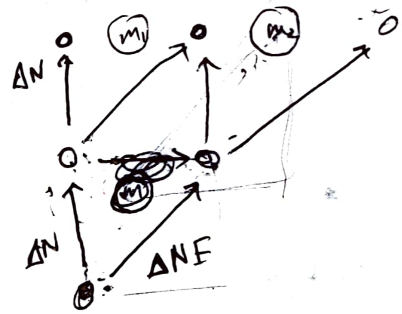
so, $d = d_{init} \times 2 = 10$

x	y	d	$\Delta E / \Delta NE$
			~~40~~, -20 so (ΔNE)
-20	30	10	
-19	31	-10 ($10 + (-20)$)	ΔE
-18	31	30	ΔNE
-17	32	$30 + -20 = 10$	ΔNE
-16	33	-10	ΔE
-15	33	30	ΔNE
-14	34	10	ΔNE

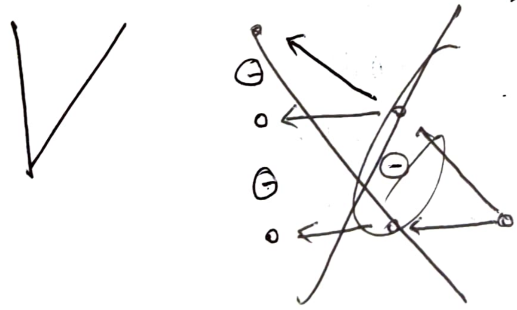
new x

ΔE

$30 = -10 + 40$



Zone - 1



so Zone
M, M_1, M_2

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

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

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

$\Delta N = d_1 - d = B$, $\Delta NE = d_2 - d_1 = dy - dx$

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

$y = 1$, $x = 0.25$ হতে , $d_{init} > 0$ so,

$\boxed{+ve \text{ হলে } \Delta N}$

$d_{init} \le 0$ হলে, ΔNE

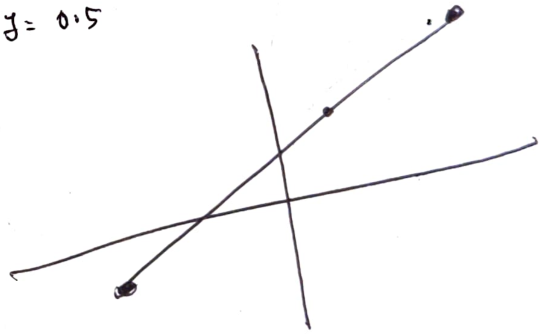
———— x ————

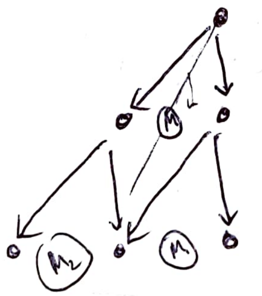
Assignment

→ যেকোনো কাগজে Randomly ২ টা zone derivation করে জমা দিবা, [illegible] অংক করবা, (Monday)

$x = -1$

$y = 0.5$

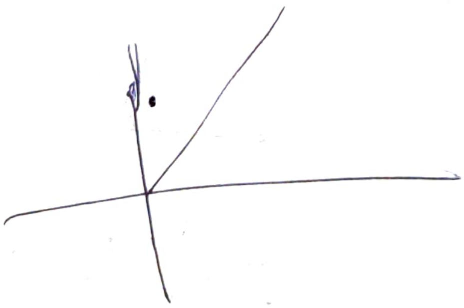




$d < 0$

$d_{init} \le 0$: SW

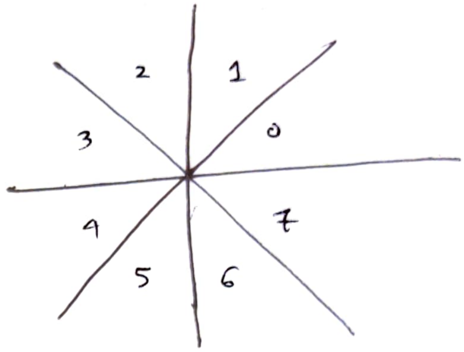
$d_{init} > 0$: W



Lect-3 , Monday

16.1.23

Slope-independent Line:



Algorithm – (1)

(i) Determine slope

(ii) Run Accordingly

or algorithm(2) –

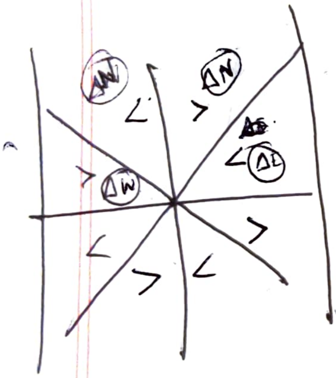
(i) Determine slope

(ii) Convert slope into a specific one

a) say এটা হলো zone-0

(iii) Draw

↳ b) Draw in original but process in zone-0

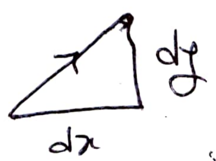


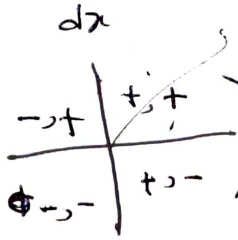
→ Both algo has same complexity, But algo-2 তে Code optimized (ছোট size code)

From (x_0, y_0) (x_1, y_1) → $\boxed{\text{Zone Determination}}$

ধরব zone-0 তে

$dx, dy > 0$, $dx \geq dy$





for sign

→ code এ, বাইরের if এ dx, dy এর sign দেখবো,

Inner if এ দুটোর absolute value compare করবো

অথবা, বাইরের if এ absolute |dx| > |dy|, এর জন্য

0, 3, 4, 7 consider

Inner if এ sign consider.

ভিতরের (Inner) code same থাকবে

$\boxed{\text{Code}}$

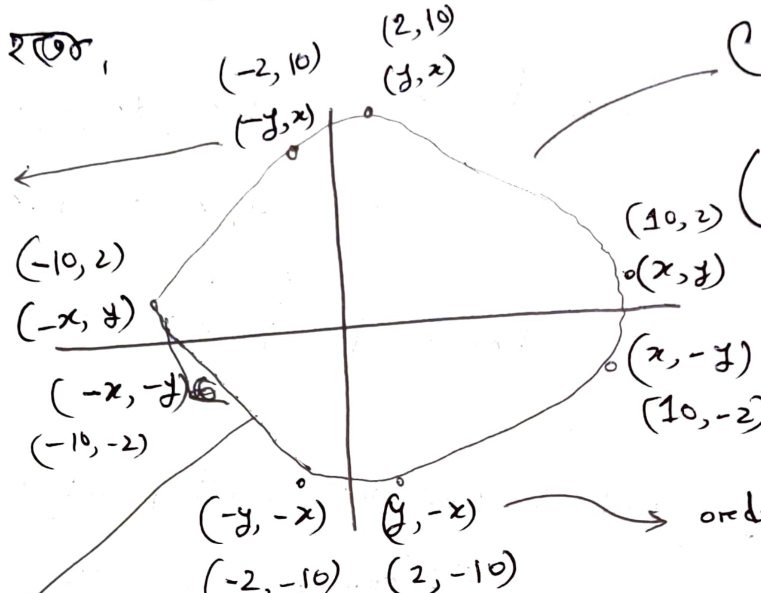
```
void drawLine (int x0, int y0, int x, int y) {
        // Determine zone , then call drawLine0,
                                      drawLine1
        int dx = x1 - x0, dy = y1 - y0;        or something
                                                 else
        [ zone = determineZone (dx, dy)
          ↳ zone এর উপর depend করে call
                  drawLine
```

Lab → return না করে colour set করে দিতে হবে zone অনুযায়ী

```
int determineZone(int dx, int dy) {
        if (|dx| ≥ |dy|) {     //0, 7, 3, 4 zones
            if (dx ≥ 0 & dy ≥ 0) return 0 ;
            elif (dx ≥ 0 & dy ≤ 0) return 7 ;
            elif (dx ≤ 0 & dy ≤ 0) return 4;
            else return 3;
        }
        else → same with //1, 6, 2, 5 ...
```

→ কোনটি Initial আর কোনটি endpoint, সেটি important, তা না থাকলে Zone-0, Zone-4 একই হতো,

" এখন position always x"



Circle
↓
(center থেকে distance সমান!)

→ ordered pairs

→ এই circle এর 8 টা point এর মধ্যকার Relation is called 8-way Symmetry

↳ we can use this to take all to a specific Zone

<u>example</u> (-10,-2)

↳ (10, 2) [দুটিতেই minus দিয়ে গুণ করে]

→ 0, 7, 3, 4 এর জন্য sign change করা লাগে হয়.

Zone ≠ Zone

$(10,2) \rightarrow (-2,10)$

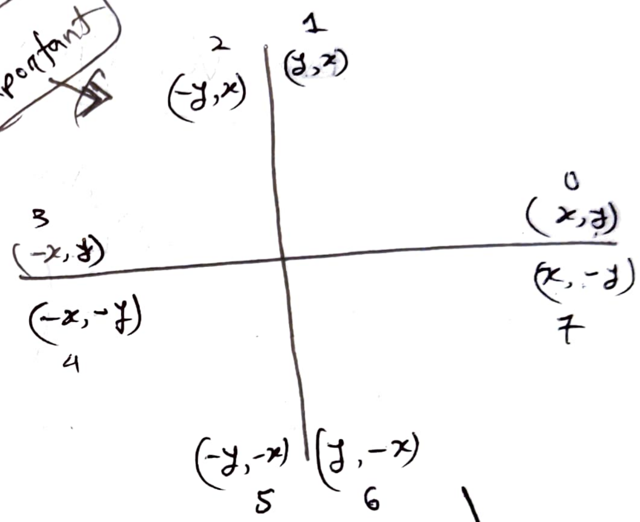
→ 1, 2, 5, 6 এর জন্য আলাদা rule

★ We need a table for Zone-0 to Zone-all

and another " " Zone-all to Zone-0

Zone	Any to 0
1	$(x,y) \rightarrow (y,x)$
2	$(x,y) \rightarrow (y,-x)$
3	$(x,y) \rightarrow (-x,y)$
4	$(x,y) \rightarrow (-x,-y)$
5	$(x,y) \rightarrow (-y,-x)$
6	$(x,y) \rightarrow (-y,x)$
7	$(x,y) \rightarrow (x,-y)$

Important



for 0, 3, 4, 7
↳ natural
for 1, 2, 5, 6
↳ x,y swap but not minus

First এ coordinate আগে 0, 3, 4, 7 চেক

↳ Then 1, 2, 5, 6 চেক

★ ZoneDetermination এর algo এ

Zone-0 এ তে → without any change

Zone-i তে → i এর জন্য mapping (i to 0)

↓ From Table

example for Zone-3 $\xrightarrow{\text{call}}$ drawLine0 $(-x_0, y_0, -x_1, y_1)$

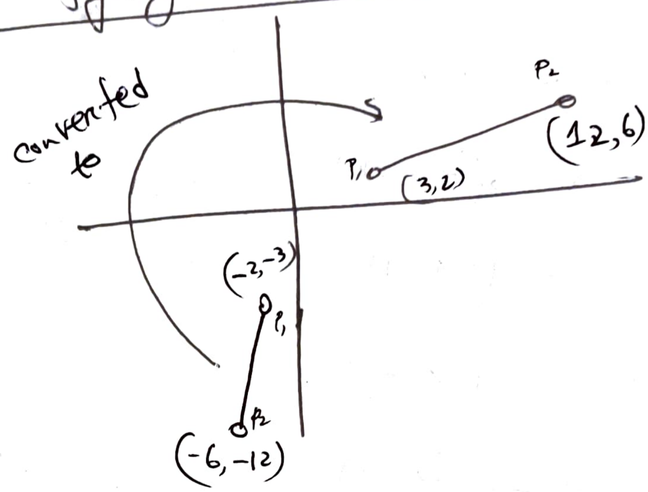
For,

$Z\text{-}0 \Rightarrow d\_L(x_0, y_0, x_1, y_1)$

$Z\text{-}1 \Rightarrow d\_L(y_0, x_0, y_1, x_1)$ ...

$Z\_5 \Rightarrow d\_L(-y_0, -x_0, -y_1, -x_1)$

Verifying with Example



→ এক্ষেত্রে Zone-0 থেকে back করতে হবে

জন্য আরেকটা Argument লাগবে drawLine

→ (So five arguments $\boxed{(x_0, y_0, x_1, y_1, zone)}$)

↓

এখানে 0 থেকে

any তে back এর

জন্য আরেকটা

← table

Zone	Mapping
0	$(x,y) \rightarrow (x,y)$
1	$(x,y) \rightarrow (y,x)$
2	$(x,y) \rightarrow (-y,x)$

→ একটা figure থেকে

Code

```
void drawLine0 (int x0 .... int zone){
    // same algo for Line draw
    ........
    // এখানে drawPixel (x,y, zone) হবে
    drawPixel(x, y, zone);
}
```

drawPixel এ 0 থেকে any জোনে নেবে

↳ unlike 2 inputs before, 3 here

```
void drawPixel (int x, int y, int zone){
    switch (zone){
        case 0: glVertex2i (x,y);
        case 1: glVertex2i (y,x);
        :
        case 7: ...
    }
}
```

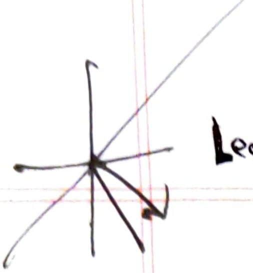
⇒ Comparison between Algo-1 and Algo-2

[LAB → দুইটাই পরে দেখানো লাগবে]

"সাতার কাটার সময় ঘুমাবানা, তাহলে মরে যাব"
— Sir

Lect-4, Wednesday

18.1.23



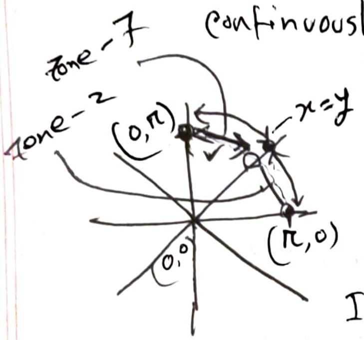
Circle Drawing Algorithm

$$(x-a)^2 + (y-b)^2 = r^2 \quad \cdots\cdots (i)$$

$$x^2 + y^2 - r^2 = 0 \quad \cdots\cdots (ii)$$ → So this equation is used for processing

(Processing এ a, b বাদ দিলেই হবে।

★ Circles first derivative isn't constant → so rate of change (tangent) will continuously change → এখন del constant না, unlike line's

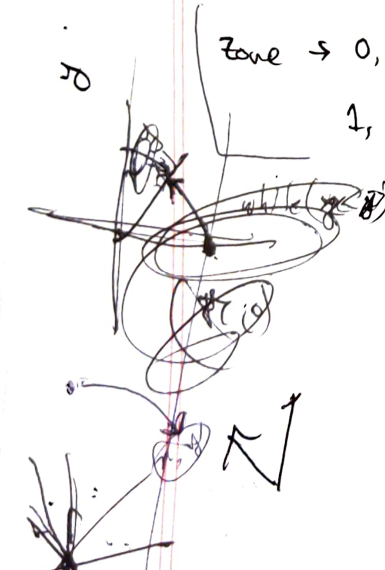


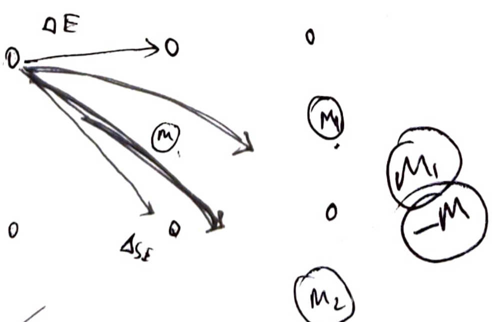
we'll be using 8-way symmetry

Initial point (0, r) or (r, 0)

Zone → 0, 7, 3, 4 → loop controller x

1, 2, 5, 6 → loop controller y



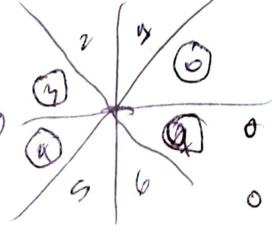


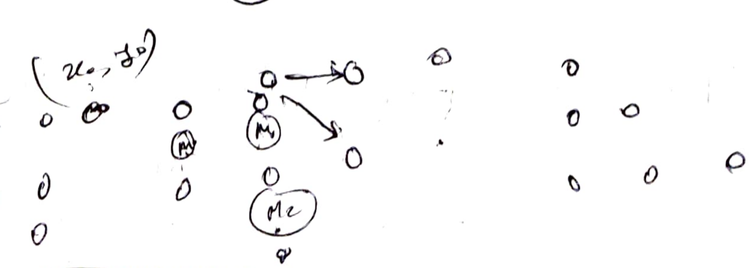
if initial is (x_0, y_0)

$$M = \left(x_0 + 1, y_0 - \frac{1}{2}\right)$$

$$M_1 = \left(x_0 + 2, y_0 - \frac{1}{2}\right)$$

$$M_2 = \left(x_0 + 2, y_0 - \frac{3}{2}\right)$$





-2.3

$(x, \frac{1}{2})$ $-y$ $4+\frac{1}{4}$ $\frac{17}{4}$

$$f(x_0, y_0) = x_0^2 + y_0^2 - r^2$$

Now, M $\rightarrow$ $f\left(x_0 + 2, y_0 - \frac{1}{2}\right) = x_0^2 + 4x_0 + 4 + y_0^2 - y_0 + \frac{1}{4} - r^2 = d_0$

d ~~$\Delta E = d_1 - d$~~ $=$ ~~$4x_0 - y_0 + \frac{17}{4}$~~

$\frac{360}{1+1}$

$M_1 \rightarrow f\left(x_0 + 2, y - \frac{1}{2}\right) = d_1$

$$\boxed{\Delta E = d_1 - d = 2x + 3} \text{ --- (iii)}$$

$M_2 \rightarrow f\left(x_0 + 2, y - \frac{3}{2}\right) = d_2$

$$\boxed{\Delta SE = d_2 - d = 2x + 3 - 2y + 2 = 2x - 2y + 5} \text{ --- (iv)}$$

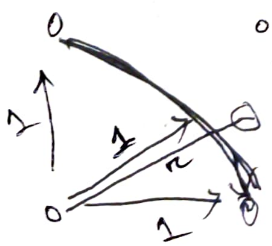
Here, For zone-7, initial point is $(0, r)$

So, first midpoint $= \left(0 + 1, r - \frac{1}{2}\right) = \left(1, r - \frac{1}{2}\right)$

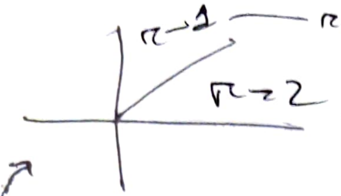
$$f\left(1, r - \frac{1}{2}\right) = 1 + \frac{1}{4} - r = \frac{5}{4} - r = d_{init}$$

$x_0, y_0 = 0, 0$

So, we got dinit by putting in the first midpoint value



$x^2 + y^2 - r^2 = 0 \rightarrow$ কখন Positive হবে: কখন negative?



region 1 (ΔE & ΔSE)

→ (ΔN & ΔNW)

$r = 1, d \geq 0$

⇒ r=1 থাকে r>1 এর জন্য circle m এর

প্রথম ~~...~~ দিয়ে যায়।

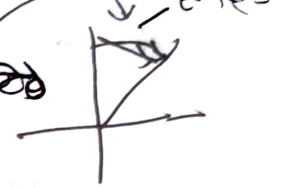
$d \geq 0$

r = 1 হলে ই নিতে পারে যায় → ΔSE

$d \leq 0$

r > 1 হলে → ΔE

→ for ~~...~~

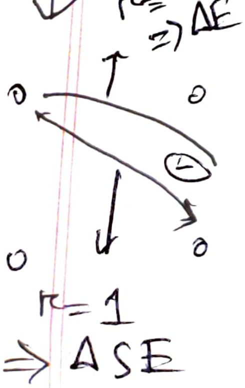


<u>code</u>

প্রথম pixel টা loop এর বাইরে এঁকে নেওয়া → loop controller

Position matters (del আগে কেন e(d))

```
void drawCircle1(int r) {

    int x=0, y=r;
    int d = 5 - 4 x r;
    draw8way (x,y);
    while(x<y) {
        if (d<0) { // ΔE
            d+ = (2x+3) x 4;
            x++;
        }
        else { // ΔSE
            d+ = (2x-2y+5) x 4;
            x++; y--;
        }
        draw8way(x,y);
    }
}
```



Circle এর সব Region এ d same

Simulation : Draw a circle for $r = 8$ and center is $(0,0)$

x	y	d	$\Delta E / \Delta SE$
0	8	-27	$\Delta E\ (+12)$
1	8	-15	$\Delta E\ (+20)$
2	8	5	$\Delta SE\ (-7)$
3	7	-23	$\Delta E\ (+36)$
~~4~~	~~7~~	~~34~~	~~$\Delta SE\ (-1)$~~
4	7	13	$\Delta SE\ (-4)$
5	6	9	$\Delta SE\ (\ )$
6	5		

$d = 5 - 4 \times r$
$= 5 - 32 = -27$

$d \leq 0 \rightarrow \Delta E$
$d > 0 \rightarrow \Delta SE$

$x = y$ হলে, so দুই পর্যন্ত হবে loop — ✱

এখানে termination point হবেনা floating point.

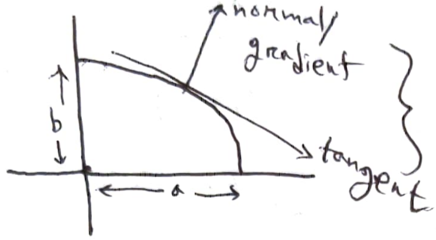
⇒ Region - 2 এর জন্য dinit, ΔN, ΔNW

দেখে নেবে ✱✱✱

8+5
6+5
-4

Lect-5, Monday 23.01.23

Ellipse Drawing → Double radius (a, b)



4-way drawing unlike 8-way
1 portion থেকে শুধু sign swapping করেই 4-টা পাওয়া যাবে

1/4th of an ellipse

Eq

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

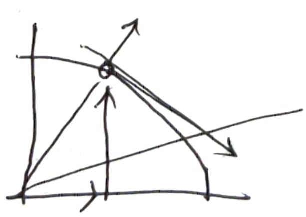
$$\underbrace{b^2x^2 + a^2y^2 - a^2b^2 = 0}\quad \text{------------ (ii)}$$

Implicit form of (i)

deviation বের করতে Implicit form (যাতে 0) type equation লাগে

→ notice normal/gradient and tangent vectors

- Consider (for understanding) a circles tangent -



for circle at point (x, y) grad is (x, y) as well

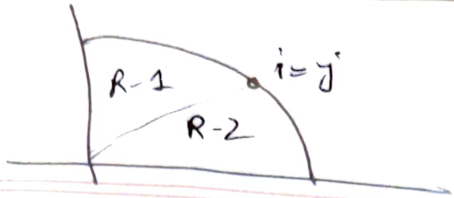
Reasoning বুঝিনাই

এত কুইক তাই

★ Ellipse এ termination এর জন্য এখন একটা ব আমাদের নিতে হবে যেখানে gradient এর

$x = y$. $(i = j)$

AS



→ grad vector এর গুণ গালে $j > i$ অবস্থায় $i > j$,
এবং মাঝে কোন point এ $j = i$.

Region-1 এ → $j > i$

Region-2 এ → $i > j$

- Consider for circle, $x^2 + y^2 = r^2$

x, y, i, j values
main

grad → $2x\,i + 2y\,j = 0$ / $2x\,dx + 2y\,dy = 0$

- For ellipse, $b^2x^2 + a^2y^2 - a^2b^2 = 0$ so,

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

magnitude of x component $2b^2x$, y comp → $2a^2y$

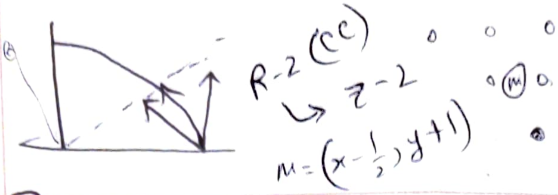
start point
→ $(0, b)$ যখন $2a^2y$ (y comp) > $2b^2x$ (x comp)
তখন R-1 এ.

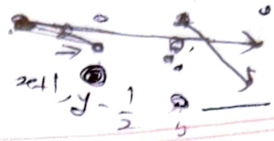
For new point,

Region-1 এ	→	Zone-7
Region-2 এ	→	Zone-6

Region-1 এ x বাড়বে বেশি → loop controller x

Region-2 এ y কমবে বেশি → loop controller y





$a^2(4y-2) > 4b^2(x+1)$

$2a^2y - a^2 > 2b^2x + 2b^2$

$a^2(2y-1) > 2b^2(x+1)$

Derivation ~~for~~ R-1

As Region-1 = Zone-7,

$$M = (x+1,\ y-\tfrac{1}{2})$$
$$M_1 = (x+2,\ y-\tfrac{1}{2})$$
$$M_2 = (x+2,\ y-\tfrac{3}{2})$$

$$d_1 - d = \Delta E = b^2(x+2)^2 + a^2(y-\tfrac{1}{2})^2 - b^2(x+1)^2 - a^2(y-\tfrac{1}{2})^2$$
$$= b^2(2x+3)$$

$$d_2 - d = \Delta SE = b^2(x+2)^2 + a^2(y-\tfrac{3}{2})^2 - b^2(x+1)^2 - a^2(y-\tfrac{1}{2})^2$$
$$= b^2(2x+3) + a^2(-2y+2)$$

$\frac{9}{4} - \frac{1}{4}$

$\frac{8}{4}$

$-3y + y$

$(\frac{3}{2})^2 - (\frac{1}{2})^2$

$$d_{init} = b^2 + a^2(b-\tfrac{1}{2})^2 - a^2b^2$$

↓ at $(0+1,\ b-\tfrac{1}{2})$

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

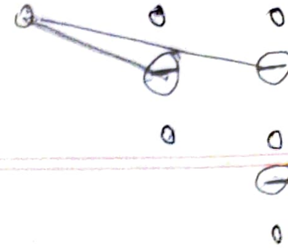
$$\boxed{d_{init} = b^2 - a^2b + \frac{a^2}{4}}$$

Think,

if $a = b = 1$ হলে $d > 0$

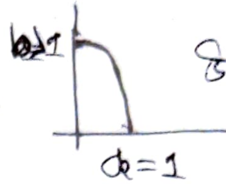
$a = 2,\ b = 1$ হলে $d < 0$

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



- এখানে দুটো কারণে $\boxed{d \text{ positive}}$,

  ① $a=1,\ b>1$ হয় →



উল্টো ellipse

  ⑪ $a=1,\ b=1$ হয় → Circle

d negative হলে ΔE

d positive হলে ΔSE

Code

(for region 1)

```
void drawEllipse1(int a, int b){
    int x=0, y=b;  // initial point of R-1
    int d= 4(b²-a²b)+a²;  // multiplied by 4 to remove fraction
    draw4way(x, y);  // all 4 part এর draw করে
    while( b²(x+1) < a²(y-1/2) ){
        if(d<0){
            d += 4b²(2x+3)
            x++
        }
        else{
            d += 4b²(2x+3)
            x++, y--
        }
        draw4way(x, y);
    }
} ...... // Rest code.
```

$$\underbrace{b^2(x+1)}_{\text{i-comp}} < \underbrace{a^2(y-\tfrac{1}{2})}_{\text{j-comp}}$$

[or we can multiply by 2, $(2b^2x+2b^2) < (2a^2y-a^2)$]

remember to multiply by 4

(x,y update after d update!)

→ Region-2 এর dinit হলো, Region-1 এর last point থেকে.

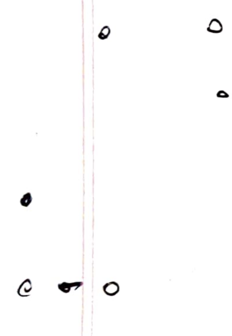
Region-2 এর জন্য Zone-6.

তবে x_0, y_0 দিয়ে,

$\{(x_0+\frac{1}{2}),(y_0-1)\}$ f এ বসালে

dinit পাওয়া যাবে,

Termination criteria in R-2 is simply $y>0$.



Region-2 তে ΔS ও ΔSE হবে,

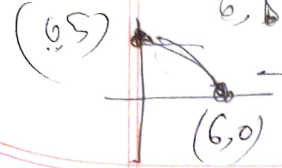
যেটা derive করা যাবে,

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

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

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

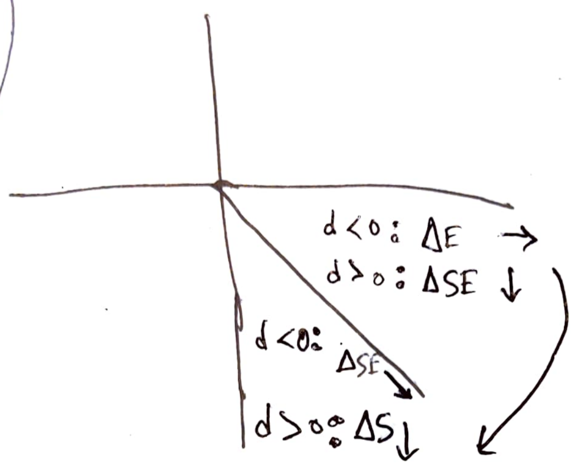
So, find ΔS and ΔSE



- region-2 তে d negative-positive:

ΔSE $d < 0$ হলে ΔSE
$d > 0$ হলে ΔS } region-1 এ যেমন clockwise, এখানেও

এই week এ শুধু Line আর Circle



R-1

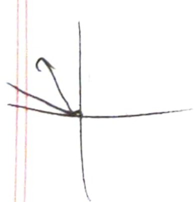
- Program → শুরুতে d এর লিখবে

প্রতি loop এ x, y, d, ΔE/ΔSE

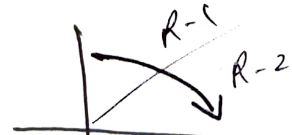
R-2 শুরুতে আবার d-value

Repeat for x, y, d, ΔSE/ΔS

↳ Do for (6, 5) → 7 টা value পাবো (a, b)

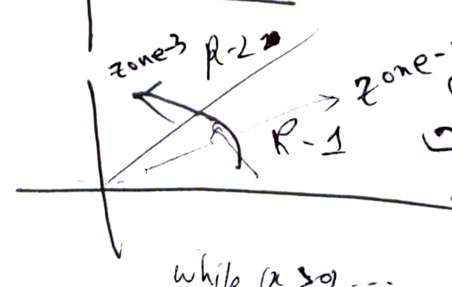


ক্লকওয়াইজ

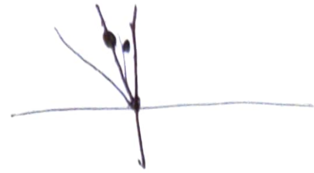


এ direction

বিপরীত



এই direction (oi dicke)



(0, 0) → (-2, 5)

Questions   loop controllers — code (

Circle 8-way

Lect-6, Wednesday 25.01.23

Equation of a Line : $\frac{x}{7} + \frac{y}{5} = 1$; Draw the following line using line drawing algo.

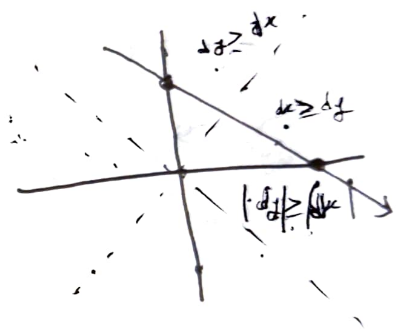
↳ Make range yourself if not given

So, range will be

$(0,5)$ to $(7,0)$

$(x=0)$ $(y=0)$

আবার, $y = 3$ to 10 চাইলে x বের করে Point বের করো।



① Find the zone

$y = mx + c$; $m = -\frac{1}{7}$

But if you multiply equation by 5 to keep y এর সহগ 1,

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

$\frac{5x}{7} \rightarrow m$

$m = \frac{dy}{dx} = \frac{-5}{7}$; $dx \le dy$

and ~~dx and~~

either dy or dx neg, so zone-3 or 7

↳ So, 7

DDA is floating point algo.

(ii) Compare the pixel co-ordinates from Midpoint and DDA algorithm.

(iii). For zone-7, find ΔE, ΔSE and dinit.

$$\boxed{\begin{aligned} &Ax + By + C = d \\ &A = dy, \quad B = -dx \end{aligned}}$$

~~$d - C - By$~~

$By = -Ax + (d-C)$

$y = -\frac{A}{B} + (d-c)$

$\frac{dy}{dx} = -\frac{A}{B}$

~~$f(M) = d =$~~

$$d_1 - d = f(x+2, y-\tfrac{1}{2}) - f(x+1, y-\tfrac{1}{2})$$

$$= A = \Delta E = dy$$

$$d_2 - d = f(x+2, y-\tfrac{3}{2}) - f(x+1, y-\tfrac{1}{2})$$

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

$$\text{dinit} = A - \frac{B}{2} = dy + \frac{dx}{2}$$

(0,5) to (7,0)

$dy = -5$, $dx = 7$ $\quad$ dinit $= -\frac{3}{2}$

$$\text{dinit} = 2 \times \left(dy + \frac{dx}{2}\right) = -10 + 7$$

$$= -3$$

$$\Delta E = \Delta E_{act} \times 2 = -10$$

$$\Delta SE = \Delta SE_{act} \times 2 = 4$$

$L_1 > 0$

$L_2 < 0$

$$\text{dinit at } (1, -0.25) = 2 \times \left(1 + \frac{0.25}{2}\right) > 0$$

positive হলে ΔE

neg হলে ΔSE

Simulation

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~~

$\Rightarrow$ Now we do DDA and Compare.

DDA Simulation

$dx = 7$, $dy = -5$, step/max_val $= \max(|7|, |-5|) = 7$

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

- loop_controller $\rightarrow x$ (max)

~~i=0, x=0, y=5~~

i	x	y
	0	5
1	1	4
2	2	4
3 4	3	3
5	4	2
6	5	1
7	6	1
7	7	0

⇒ Symmetric roundoff

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

x		y actual	y
0	0	5	5
1	1	$5-\frac{5}{7}$	
2	2	$5-\frac{2\times5}{7}$	
3	3	$5-\frac{3\times5}{7}$	
4	4	$5-\frac{4\times5}{7}$	

→ 4 আসছে কেন!
from Midpoint

So actually হবে,

drawPixel (int (x + sign(x_{curr} ~~curr − x~~) × 0.5),
int (y + sign(y_{curr}) × 0.5))

dynamically x or y এর sign হবে

উপর ভিত্তি করে $+0.5$ বা -0.5 হবে.

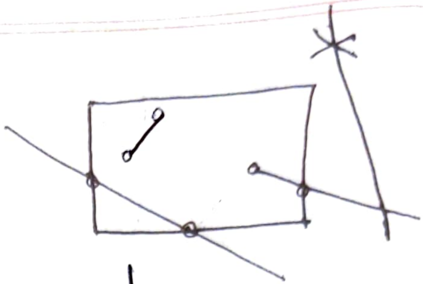
$\Rightarrow$ বইতে দেওয়া sign(dx) sign(dy) লিখেছে,

আসলে sign(x) sign(y) হবে।

Lect-7, Monday

30.01.22

Line ~~c~~lipping Algorithm



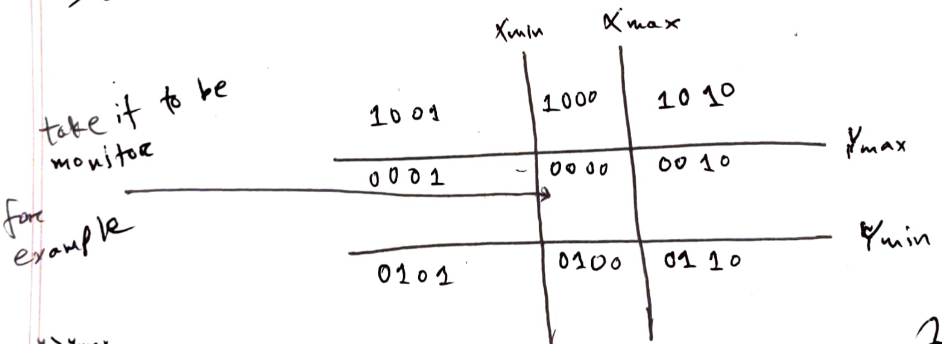
↓ Draw lines within the Box

Cohen-Sutherland LCA:

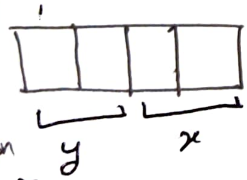
→ Two separate ~~invention~~ discovery

→ প্রতি Line এর দুইটা endpoint এর Binary code বের করে → using the code, deciding whether it'll be drawn

→ for both 2D and 3D



$y > y_{max}$



প্রত্যেক point এর (x,y) এর কোনো নির্দিষ্ট পরে 4-bit code তৈরি করা যায় — Region Output

(x, y position interchangeable)

if $y > y_{max} \rightarrow 10$

else $y < y_{min} \rightarrow 01$

in-between হলে → 00

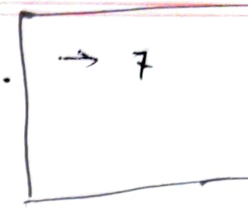
same for xs.

→ 9 combinations (instead of 2^4)

Because → none has more than 2 ones.

কোনটা value → int $y_{max} = 100$, $y_{min} = -100$, $x_{max} = 150$, $x_{min} = -150$

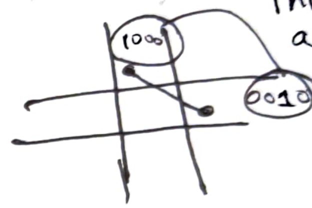
1000 ↑   0100 ↑   0010 ↑   0001 ↑



<u>code</u>

```
int TOP = 8, BOTTOM = 4, RIGHT = 2, LEFT = 1;
int makeCode (int x, int y) {
    int code = 0;

    if ( y > ymax)   code += TOP
    else if (y < ymin)   code += BOTTOM
    if (x > xmax)   code += RIGHT
    else if (x < xmin)   code += LEFT

    Return code;
}
```

(Boundary points are also taken as is used)



Think about this line

⇒ Second discovery, দুইটা point এর

★

→ Trivially accepted: OR is zero

→ Trivially rejected: AND is !zero

else → বড় test and clipping

↳ এক একটা one কে zero করে until trivially accepted or rejected

→ "এটা 3D হলে কি হবে ?" → 6 bit Region Output

→ near user = +Z

32 ← far user = −Z

16 ←

6 bit এর ক্ষেত্রে accepted rejected same.

But → 4-টা loop maximum for 2D, 3D এর জন্য আরেক

Work 88 101
100
100

→ At a glance বললে পারা trivially accepted/rejected.

→ Clipping Algo আগে called হয়,
Clipping এ accepted হলে draw করে।

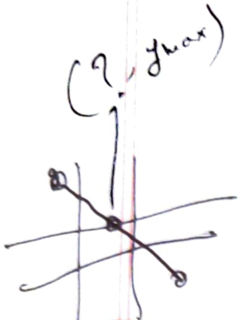
// Code

```
void CohenSutherland (int x0, int y0, int x1, int y1){
    int code0, code1, code, x, y;
    code0 = makeCode (x0, y0) ; Code 1 = makeCode(x1, y1);
    while (1) {
        if (code0 | code 1) = = 0) { // Accepted
            drawLine (x0, y0, x1, y1);
            break ;
        }
        else if (code0 & code1) { // Rejected
            break;
        }
        ~~else~~
```

Pto

$$\cancel{\frac{y+200}{-100}} \quad \frac{y-200}{400} = \frac{x+300}{-700}$$

partial এ ঢোকানোর 1 আগেই



next page

```
else { // partial

    if (code0)  code = code0
    else        code = code1          } safety measure

    if (code & TOP){        // above ymax
        y = ymax;
        x = x0 + (ymax - y0)*(x1 - x0) / (y1 - y0);
    }
    else if (code & BOTTOM){
        y = ymin;
        x = x0 + (ymin - y0)*(x1 - x0) / (y1 - y0);
    }
    else if (code & RIGHT){
        x = xmax;
        y = y0 + (xmax - x0)*(y1 - y0) / (x1 - x0);
    }
    else {
        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)}
} //else শেষ.
} //while শেষ
```

As (x_0, y_0) এর দিক clip করা হয়ে গেছে

Parametric Equation of a line

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

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

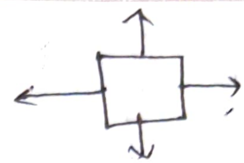
y থেকে t বের করে x এ বসাই.

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

so, $$x_t = x_0 + \frac{y_{max} - y_0}{y_1 \cancel{y_{max}} - y_0}(x_1 - x_0)$$

Lect-8, Wednesday 01.02.23

→ Quiz!?



⇒ Parametric Line-Clipping Algorithm

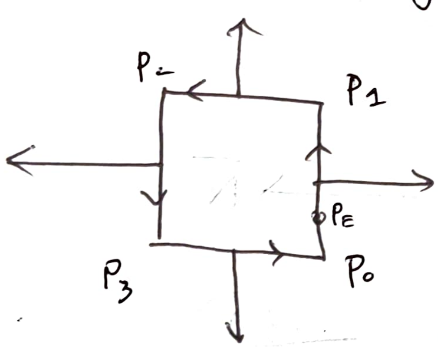
Cyrus Beck LCA

→ Polygon has more than two points

→ Boundary direction of a polygon হলো each side এর normal line বা gradient vectors

→ On-face and off-face of a surface

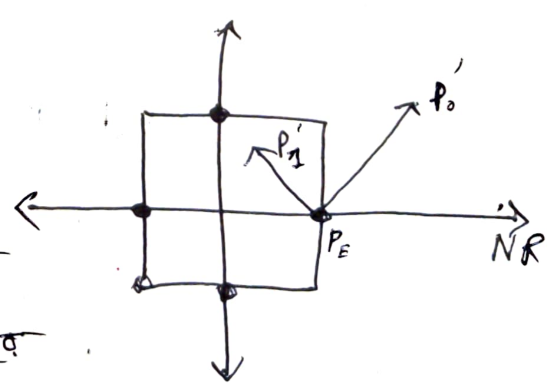
→ On face থেকে দেখলে, tangents are always anti-clockwise.



it's a $P_0P_1P_2P_3$ Polygon (anti-clockwise)

Let, P_E be any point

How do we know mathematically if a point is inside the polygon?



$(P_0' - P_E) \cdot NR > 0$ → বাইরে

< 0 → ভিতরে

$= 0$ → on line

$(P_1' - P_0) \cdot NR > 0$ হলে leaving, < 0 হলে entering,

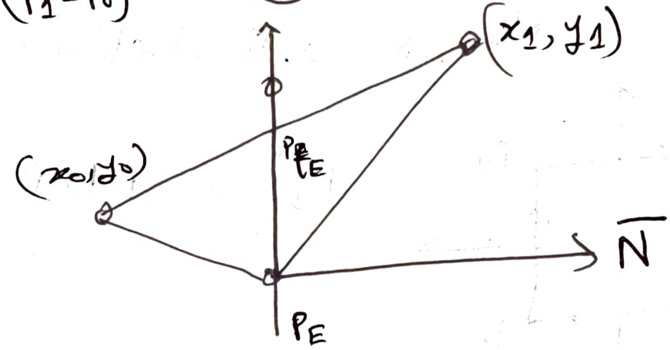
⇒ Entering t_E and Leaving t_L

দুটি t পাবো, ~~একটির~~ একটা entering, একটা leaving

⇒ (x_0, y_0) হতে দূরত্ব t এর মধ্যে সবচেয়ে বড় t হলো t_E,

(x_1, y_1) থেকে দূরত্ব t এর মধ্যে সবচেয়ে ছোট t হলো t_L.

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



$$(P_{t_E} - P_E) . N = 0 \quad \text{(ii)}$$

From (i) and (ii) $\Rightarrow \left(P_0 + t_e(P_1 - P_0) - P_E\right) . N = 0$

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

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

- Learn Derivation of t_e

Left $\dfrac{x_{min} - x_0}{x_1 - x_0}$

→ এখানে t_e ও একটি known, একটি unknown.

Right → $N = (1,0)$, Left $N = (-1,0)$

Top → $N = (0,1)$ Bottom $N = (0,-1)$

For right, $N = (1,0)$

Right → $x_e = x_{max}$

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

$$= - \frac{(x - x_{max}) \cdot 1}{(x_1 - x_0) \cdot 1} = \frac{x_{max} - x_0}{x_1 - x_0}$$

Like

→ this we can proof for TOP,

↳ $\dfrac{y_{max} - y_0}{y_1 - y_0}$

Left → $\dfrac{x_{min} - x_0}{x_1 - x_0}$

Bottom → $\dfrac{y_{min} - y_0}{y_1 - y_0}$

Lecture-9, Monday

06.02.23

Cyrus-Beck LCA (last class এ যা ~~করছি~~ করেছিলো)

we saw, $t_E = \dfrac{-(P_0 - P_E)\cdot N}{(P_1 - P_0)\cdot N}$

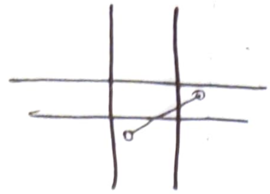
A List of t, for all edges:

Boundary	Normal	$\frac{(P_0 - P_E)\cdot N}{}$ (top এর উপর N)
→ TOP	$(0,1)$ ⟵	$y_0 - y_{max}$
		$(x_0 - x,\ y_0 - y_{max})\cdot(0,1)$

⟶

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

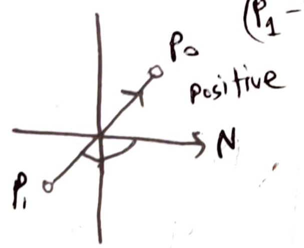
2 ★ মোটমাট → make a list of t for all edges

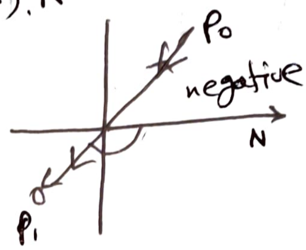


t_{top}, t_{left} এই দুটিও হবে
abnormal → not less than 0

t_E, t_L এ group করতে হলে Intersection গুলো —

positive → leaving, negative → entering





So, for top,
$y_1 - y_0 > 0 \equiv y_1 > y_0$
means leaving.
↓
bottom এ ~~entering~~
entering

⇒ According to cyrus-beck, a line is from (t_E max to t_L min). t_E max যেটা, সেটা new (x_0, y_0) t_E min হলে new (x_1, y_1)

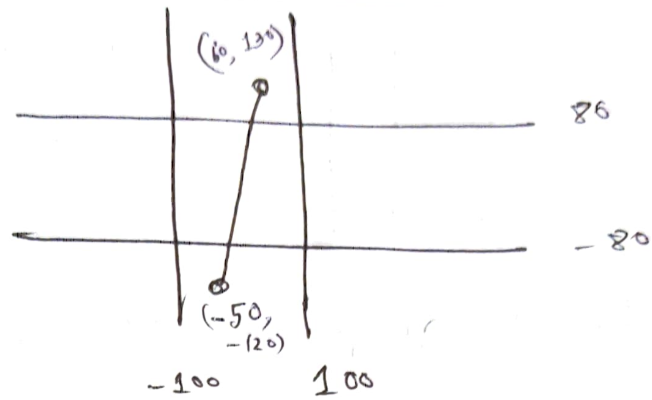
t_L min ?

↳ শুরুতে $t_E \to 0$ (যেটাকে max করবে)

$t_L \to 1$ (যেটাকে min করবে)

কারণ $(P_1 - P_0).N = y_1 - y_0$, $(P_1 - P_0).N > 0$
means $(y_1 - y_0) > 0 \Rightarrow y_1 > y_0$

example



Q A line from (-50, -120) to (60, 130)

→ value of t for all edges

Boundary	t
TOP	0.8
Bottom	0.16
Right	~~0.722~~ ~~1.484~~ 1.36
LEFT	~~0.466~~ -0.45

t_E

Now, যেহেতু

$y_1 > y_0$

So, top এর জন্য line টা leaving!

Automatically, bottom এর জন্য entering.

একইভাবে,

RIGHT এ leaving

as $x_1 > x_0$,

so, left এ

entering.

SO → $t_{E\,max} = max(0.16, -0.45) = 0.16$

$t_{L\min} = \min(0.8, 1.36) = 0.8$

(তৃতীয় Quiz)

⇒ Code করে দাও from given line

⇒ (x_0, y_0) (x_1, y_1) থেকে চারটি t বের করে
ও কোনটি t এর মধ্যে t_E, t_L সেগুলো

// for top

if $(y_1 > y_0)$:

~~$t_E = t$~~

if $(t < t_{L\min})$: $t_L = t$

else ~~if $y_0 > y_1$~~ :

if $(t > t_{E\max})$: $t_E = t$

তাহলে accepted,
↓
(x_0, y_0) কেমনে from function?

⇒ if $(t_{E\max} > t_{L\min})$ then rejected

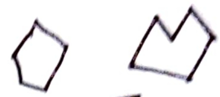
- $d>0$, $d<0$ alternate for line but not circle

08.02.23

Lect-10, Wednesday

Polygon-filling Algorithms

→ Polygons → Convex & concave



 → twisted polygon

⇒ OpenGL শুধু convex polygon filling পারে

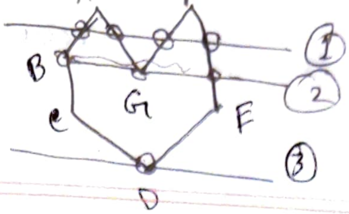
⇒ আমরা এটা শিখবো যেটা both convex and concave পারে কিন্তু twisted পারে না।

⇒ ScanLine algorithm of Polygon filling আমরা পড়বো।

Concept → Polygon's pixels are thought of as set of lines. First এ boundary pixels set হয়, then scanline করা হয়। Parity-based অনুসারে Parity change হলে filling শুরু হয়, হয় বা হলে filling শেষ হয়।

Parity - কি?

Polygonal – mesh



Problems

Polygon এর কিছু অসুবিধা নিয়ে –

- Solving color in overlapping polygons.

↳ এক side এর pixel দেবে (মনে)

এগুলো solve করতে vertically উপরের Pixel দেবে (মনে)।

↳ AB line এ A কে ছেড়ে B কে ধরবে.

A = 0 as AB A কে ছাড়বে

B = 1 as AB B কে ধরবে, BC B কে ছাড়বে

C = 1 D = 2, E = 1, F = 0, G = 2

→ number of intersections

এখানে – Problem এর 3 types এর line এর যোগ হবে even.

⇒ ScanLine এর Bucket array

Bucket array: edge Table → Corner line এর info সব থাকবে

P.T.O

ABCDEFG তে প্রতিটা entry

Edge-Table → horizontal intersections ছাড়া info

Active-Edge Table → Edge Table এর Gap

↓

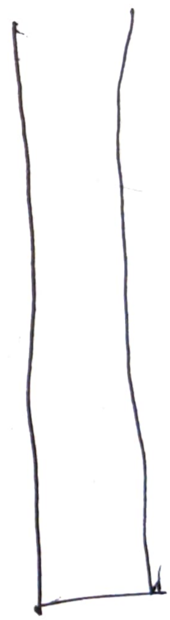
(all the boundary pixels (প্রত্যেক scan-line এর হবে!)

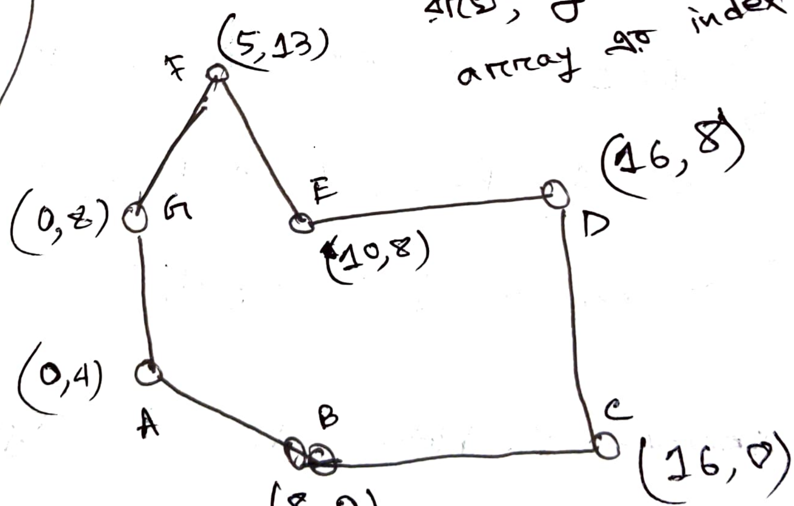
↳ with info

y বাড়লে y = 1 করলে $x = \frac{1}{m}$ বাড়ে,

② Edge-Table এ $y_{min} = y_{max}$ হলে ছেড়ে দেয়

প্রতিটা intersection এ প্রতিটা info ধরবো, y value to array এর index ← 2D array

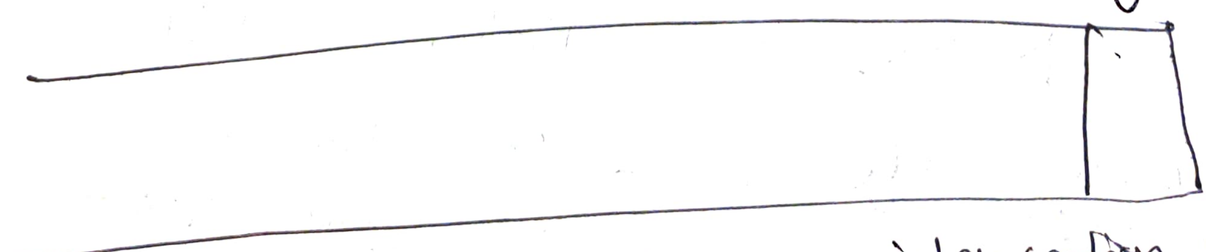




y_{min} to y_{max} পর্যন্ত index

ভুল

y = 0



☆ start values এ null রাখলে কোন intersection নেই।

↳ নাহলে 1 আছে

⇒ $y=0$ তে intersection আছে এবং link ~~[illegible]~~

কোন আছে, link এ হবে 4 টা value

আছে

AB — line is from 0 to 3

$y=0$ ⇒

y_max	x_value	$\frac{1}{m}$	link
4	8	-2	→ CD

CD

y_max	x_value	$\frac{1}{m}$	link
8	16	0	λ

link নেই

BC ধরতে আসবেনা কারণ $y_{min} = 0$ and $y_{max} = 0$

$y_{min} = y_{max}$

$y = 1$ to 3 ⇒ λ

AG

$y=4$ ⇒

8	0	0	λ

$y = 5$ to 7 ⇒ λ

FG, EF

$y = 8$ ⇒

13	0	1	→	13	18	-1	λ

So, five intersections.

Polygon এর F, D survive করেনা তাদের list of (x_{min}, y_{max}) নিয়ে থাকবে।

এবার Active-Edge → y_{min} থেকে ($y_{max} - 1$) পর্যন্ত যাবে।

$y + 1$ হলে $x + \frac{1}{m}$ হবে।

table এ সব দেওয়া

~~Active~~

Pto

Active-Edge Table

$y_{max}-1$

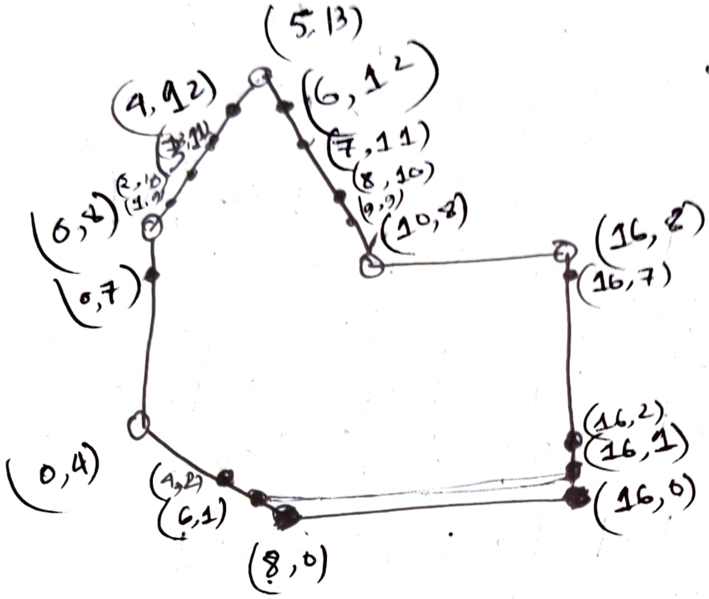
12	13 | 4 | 1	13 | 6 | -1 | λ
11	13 | 3 | 1	13 | 7 | -1 | λ
10	13 | 2 | 1	13 | 8 | -1 | λ
9	13 | 1 | 1	13 | 9 | -1 | λ
8	13 | 0 | 1 (FG)	13 | 10 | -1 | λ (EF)
7	8 | 0 | 0	8 | 16 | 0 | λ
6	8 | 0 | 0	8 | 16 | 0 | λ
5	8 | 0 | 0	8 | 16 | 0 | λ
4	8 | 0 | 0 (GA)	8 | 16 | 0 | λ
3	4 | 2 | -2 (AB)	8 | 16 | 0 | λ
2	4 | 4 | -2 (AB)	8 | 16 | 0 | λ
1	4 | 6 | -2 (AB)	8 | 16 | 0 | λ
0	4 | 8 | -2 (AB)	8 | 16 | 0 | λ

x মান দিয়ে sorted হয়ে থাকতে হবে যেকোন case এ

★ এখানে Edge-table to Active-Edge-Table conversion দেখানো হয়েছে

★ @

আগের আঁকা ছবি :



এক্ষেত্রে Active-edge table এ সব row এ line হবে

Active-edge table features → even number of intersections guaranteed.

☆ ছবি দেখে Edge Table, Active-Edge বানানো

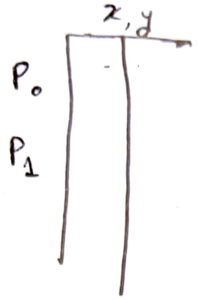
LAB এ Edge table বানাতে কি কি Rules follow করছি,

A, B, C, D, E, F, G থেকে Edge table বানানো

↳ প্রত্যেকটা steps ধরে (internet) দেখতে হবে

Lecture-11, Wednesday 15.02.23

⇒ Per line এ দুটি value যদি দেওয়া থাকে -



→ এটি থেকে Edge table বানাতে হবে,

কিভাবে করা যেতে পারে ?

① প্রথমে vertex থেকে edge বানাতে হবে, → (P_0P_1, P_1P_2 ... P_nP_0)

vertex 5 টি হলে 5 টি edge বানিয়ে নাও

② y_{min} দিয়ে sort করতে হবে প্রত্যেক Edge কে।

③ $|y_{min} - y_{max}|$ কত — বের করতে হবে, অর্থাৎ size এর array নিতে হবে। ধরো প্রতিটা y এ 4 টি intersection। তাহলে $|y_{min} - y_{max}| \times \cancel{(4 \times 4)}$ 20

↓

Poly[X][Y] একটা Array নিয়ে
20 40

④ y_{min} যে value হয় তা → first location এ বসাও

যেখানে বসাবে অর্থাৎ
দিয়ে ~~Edge table~~ বসাবে

বাকি জায়গা 'Blank' বসাও

⑤ যেখানে y_{min} আছে, সেখানে x sort করে ফেলো.

↳ y_{min} এর x এর sort করে lower x আগে, higher x পরে।

First location এ null থাকলে → intersection নেই।

(6) 4 টা value একসাথে নিতে হবে to make edge table.

(7) এর পর Blank জায়গা fill করতে করতে Active edge table বানাতে হবে ($y_{max} - 1$) পর্যন্ত।

⇒ Poly.txt নামের data file আছে, যেখানে point থাকবে।

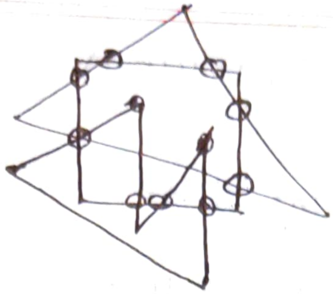
H.W

22 ই ফেব্রুয়ারী Algo টা পেপার এ লিখে জমা।

Lect-12, Wednesday

22.02.23

Polygon-Clipping:



→ Polygon clip করলে এখানে ভিন্ন হতে পারে

→ একাধিক ও হতে পারে

• We'll see Sutherland-Hodgman Algorithm for Polygon Clipping

→ Point inside-outside হিসাব করে (like cyrus-beak)

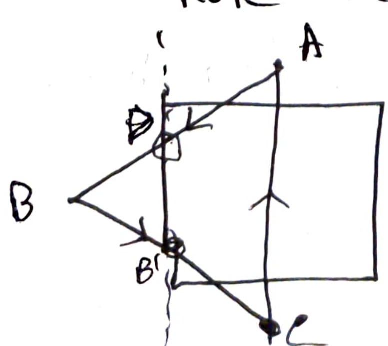
→ Edge গুলোকে vector ধরে করে (with direction)

↳ চারটি Property of Edge while intersecting → entering, leaving, (চার ধরনের intersection) inside, outside

→ যখন left boundary দেখা হবে, Left এর বাইরে সবকিছু "outside" ~~Inside~~, ভেতরে সবকিছু Inside

→ Edge গুলো চার group এর → leaving, entering, totally outside, totally inside.

⇒ $\boxed{\text{Example}}$ → চার group এর চারটি Rule আছে —



Output
(i) Border
(ii) End

Rule #1 (Entering) : ২ vertex

Entering Edge এর জন্য দুইটা output vertex থাকে।

From left (in image) : entering → $\vec{BC}$

output from edge → B', C
B' → border
C → end

Rule #2 (Inside) : output → 1 vertex (End only)

Image এ, completely inside edge $\vec{CA}$ → output: A (end)

Rule #3 (Leaving) : output → 1 vertex (Border only)

Image এ, leaving edge AB → output: D

Rule #4 (Outside) : outside হলে discard → 0 vertex

no vertex will be considered

Now, for lower border,

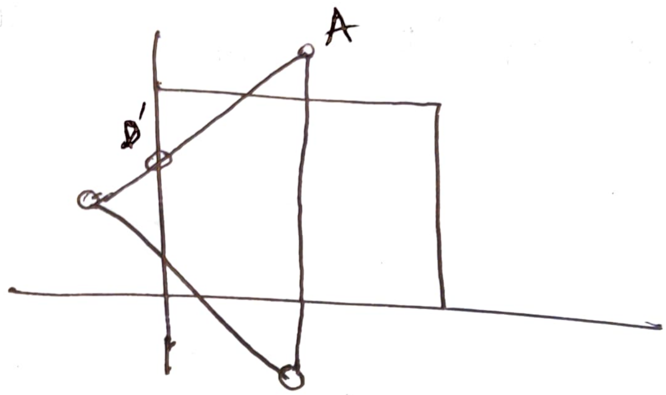
$(P_i - P_0) \cdot N > 0 \rightarrow$ leaving

$< 0 \rightarrow$ entering

B'C is leaving → border only → B''

CA is entering → ~~end only~~ border end → C', A

AD is inside → end only → D



⇒ Cyrus-beak এর মতোই leaving-entering বের করে। However, $0 \le t \le 1$ না হলে intersect করবে না,

যেমন: Cyrus-beak এ AD was entering but এখানে inside.

Inside Outside ~~N-এর~~

• $(P - P_E) \cdot N >$ +ve , inside

−ve , outside

দেও এটা

?

Lect 14, Wednesday

Q1 → Line-Drawing Algorithm

(x_0, y_0) equation or intersection দেওয়া।

- (i) Zone বের করো
- (ii) derive করো d, del
- (iii) Algorithm
- (iv) Simulation

Q2 → Circle এর জন্য (0, r) অথবা (r, 0) থেকে শুরু।

R-1 দেওয়া থাকলে R-3

এরকম stuff দেওয়া থাকবে

- Z-3 থেকে যেকোনো Zone এ আঁকাতে দিতে হবে।
- zone-determination algo লিখবে।
- ☆ 8-way symmetry apply করে slope-independent line

- Ellipse R1 to R2 এর derivation ও R2 to R1 derivations

- Cohen-Sutherland এর makeCode লিখে for 2D or 3D.

  → Basic Principle of cohen-sutherland
  3 টা (accepted

  → Algo for 2D / 3D

  →

- Cyrus-Beak (i) Derivation of t and list of t

  (ii) Determining inside or outside points and entering / leaving edges

  "~~t এর ভিত্রে~~

  (iii) Line এর জন্য t_E, t_L বের করে new point বের করা

- Polygon এর Corners দেওয়া থাকলে Edge-Table ও AET বের

- Polygon Clipping → "4 টা Basic Rule আছে"

"এই 4 টা Rule important"

→ Clip area 3 Triangle দেওয়া থাকলে, একটা Boundary clip করার পরে কি হবে।