* We can see a line because pixels between two points are colored.

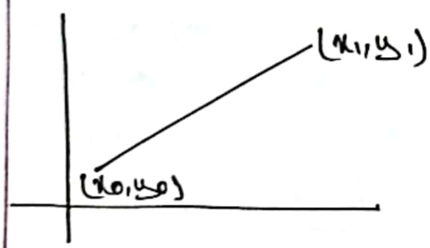
* Pixel হল particular coordinate যাদের which are always integers.

LINE DRAWING ALGORITHM

- → Simple Solution
- → DDA
- → Midpoint line Algorithm

SIMPLE SOLUTION

Equation of a line: $y = mx + c$ (m → slope, c → y-intercept)



$$m = \frac{y_1 - y_0}{x_1 - x_0}$$

$$c = y_0 - mx_0$$

Find out the intermediate pixels of a line from (2,2) to (7,5) using simple solution methods.

$m = \frac{5-2}{7-2} = \frac{3}{5} \Rightarrow m = 0.6$; $c = 2 - (0.6)(2)$, $\Rightarrow c = 0.8$

$y = 0.6x + 0.8$

$y(3) = 0.6(3) + 0.8 = 2.6 \approx 3$

$y(4) = 0.6(4) + 0.8 = 3.2 \approx 3$

$y(5) = 0.6(5) + 0.8 = 3.8 \approx 4$

$y(6) = 0.6(6) + 0.8 = 4.4 \approx 4$

* pixels fraction এ থাকলে round up করি if pixel ≥ 0.5
round down if pixel < 0.5

intermediate pixels :-
(3,3), (4,3), (5,4), (6,4)

Drawbacks:—

(1) multiplication cost ↑

$y = \underbrace{mx}_{} + c$

└→ এই part এর জন্য multiplication cost add হয়

For every value of x, we have to find a value of y using eqⁿ of line. Thus, multiplying the values of x with m and increasing the multiplication cost.

- Hardware efficiency ↓
- Time consuming

(2) Round Off

For pixels coming in fraction, we have to round up/down.

DDA (DIGITAL DIFFERENTIAL ANALYZER)

(2,2), (3, 2.6), (4, 3.2), (5, 3.8) → given points

$x \rightarrow$ increases by 1

$y \rightarrow$ increases by m ($m = 0.6$)

$x_{new} = x_{prev} + 1$

$-1 \leq m < 1$

$y_{new} = y_{prev} + m$

* m বেশি বড় হলে, y এর value তে অনেক বড় বড় jump হবে, line টা ভালো আসবে না, crack দেখা যাবে।

$x_{new} = x_{prev} + \frac{1}{m}$

Otherwise

$y_{new} = y_{prev} + 1$

Adv. DDA is faster than simple soln

Find the intermediate pixels from (2,2) to (5,7) in a line using DDA

$$m = \frac{7-2}{5-3} = 1.67$$

$$\frac{1}{m} = \frac{3}{5} = 0.6$$

$x\left(+\frac{1}{m}\right)$	$y(+1)$	pixels
2	2	(2,2)
2.6→3	3	(3,3)
3.2→3	4	(3,3)
3.8→4	5	(4,5)
4.4→4	6	(4,6)
5	7	(5,7)

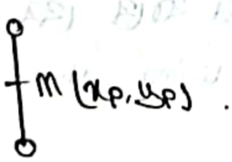
* DDA removed the multiplication cost increment caused by simple solution.

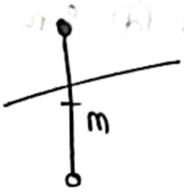
* However, round off problem is still present which is time consuming.
* Accumulation of round off errors can make pixelated line drift away from what was intended.

MIDPOINT LINE ALGORITHM

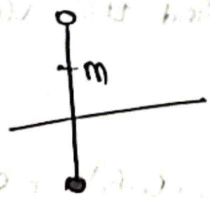
$f(x,y) = ax + by + c$ → new line

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

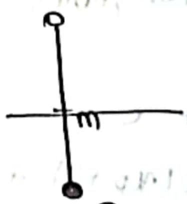




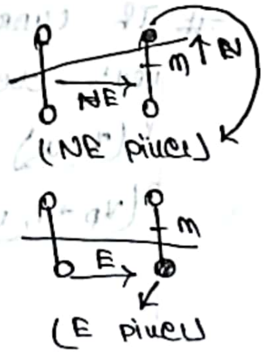
d > 0
NE pixel (upper point) is colored



d < 0
E pixel is selected



d = 0



LECTURE 2

THURSDAY DATE: 05/10/23

MIDPOINT LINE ALGORITHM

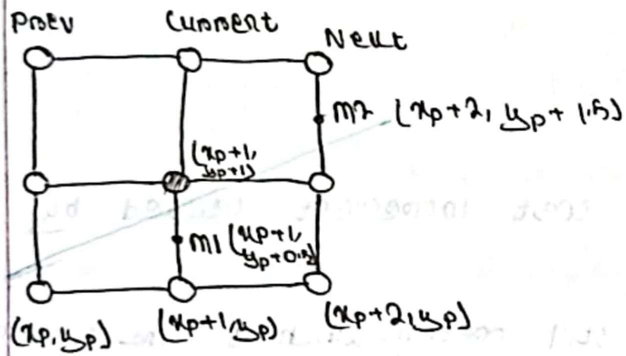
$f(x, y) = ax + by + c$

$m(x_p, y_p)$

$$f(x_p, y_p) = \underbrace{ax_p + by_p + c}_{d}$$

$d > 0$ upper (NE)

$d \leq 0$ lower (E).



* pixels are arranged as ~~rows~~ column-wise pixels.

* আমরা যদি current step এর d এর value জানি, then next step এ d এর value কেমনে করে বের করতে পারি without finding the midpoint, but using the value of d of current step.

If current step = NE, how to find the value of (d) for next step.

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

$f(x_p+1, y_p+0.5) = a(x_p+1) + b(y_p+0.5) + c$

$$d_{current} = ax_p + a + by_p + 0.5b + c \quad \text{—— (1)}$$

$f(x_p+2, y_p+1.5) = a(x_p+2) + b(y_p+1.5) + c$

$$d_{next} = ax_p + 2a + by_p + 1.5b + c \quad —— ②$$

② —— ①

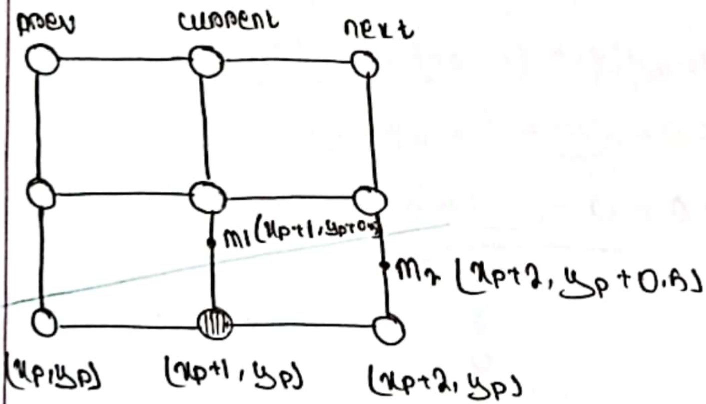
$d_{next} - d_{current} \Longrightarrow$ gives a relation between d_{next} & $d_{current}$

$$\cancel{ax_p} + 2a + \cancel{by_p} + 1.5b + \cancel{c} - \cancel{ax_p} - \cancel{a} - \cancel{by_p} - 0.5b - \cancel{c}$$

$d_{next} - d_{current} = a + b$

$$\boxed{d_{next} = d_{current} + a + b} \rightarrow \text{NE}$$

If the current step is E pixel,



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

$f(x_p+1, y_p+0.5) = a(x_p+1) + b(y_p+0.5) + c$

$$d_{current} = ax_p + a + by_p + 0.5b + c \quad —— ①$$

$f(x_p+2, y_p+0.5) = a(x_p+2) + b(y_p+0.5) + c$

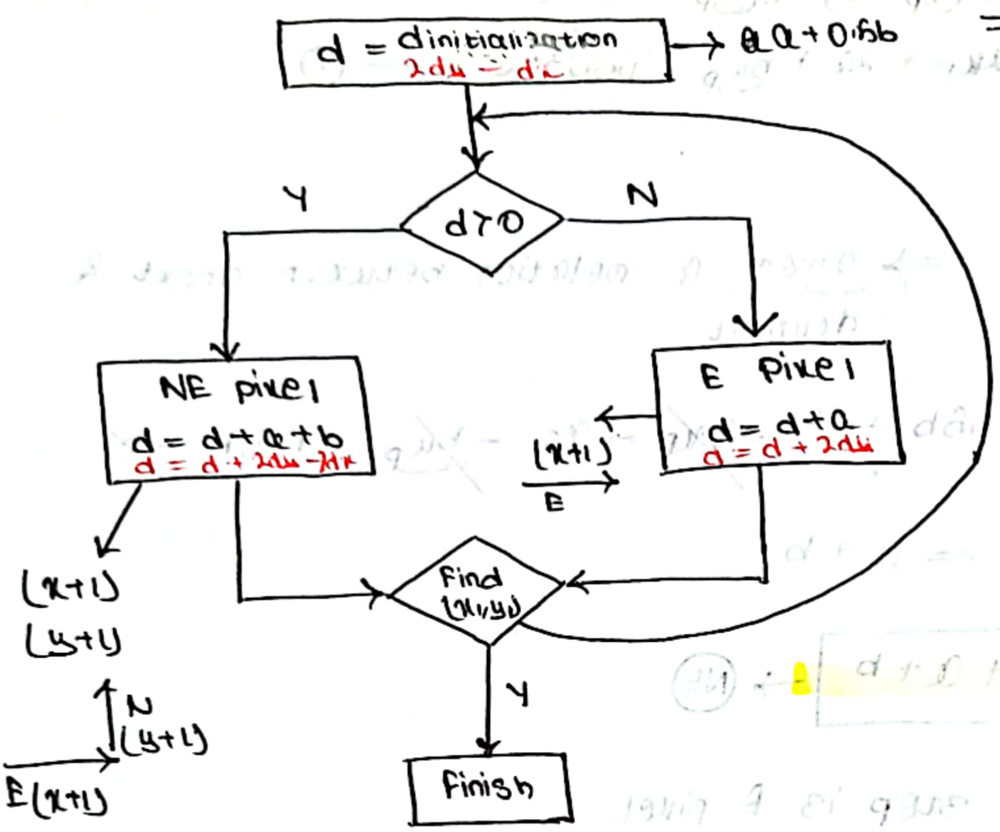
$$d_{next} = ax_p + 2a + by_p + 0.5b + c \quad —— ②$$

$d_{next} - d_{current} = \cancel{ax_p} + 2a + \cancel{by_p} + \cancel{0.5b} + \cancel{c} - \cancel{ax_p} - a - \cancel{by_p} - \cancel{0.5b} - \cancel{c}$

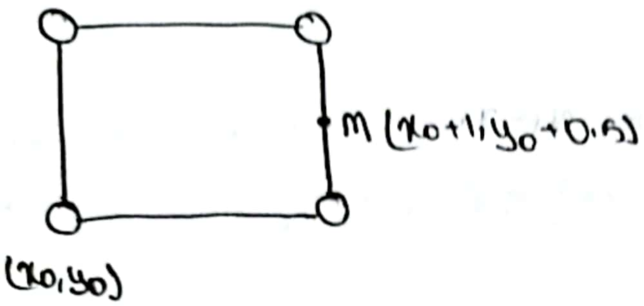
$d_{next} - d_{current} = a$

$$\boxed{d_{next} = d_{current} + a} \rightarrow \text{E}$$

$(x_0, y_0) \rightarrow (x_1, y_1)$



How to find dinitialization



* The picture is taken from the very first pixel. ~~of the~~

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

$$f(x_0,y_0) = ax_0 + by_0 + c = 0$$

→ (x_0, y_0) line এর মধ্যে

$$f(x_0+1, y_0+0.5) = a(x_0+1) + b(y_0+0.5) + c$$

$$= ax_0 + a + by_0 + 0.5b + c$$

$$= \underbrace{ax_0 + by_0 + c}_{f(x_0,y_0) \to 0} + a + 0.5b$$

$$\boxed{d_{initial} = a + 0.5b}$$

Find the value of a and b from (x_0, y_0) and (x_1, y_1)

$$m = \frac{y_1 - y_0}{x_1 - x_0} = \frac{dy}{dx}$$

dy → variable for $y_1 - y_0$

dx → variable for $x_1 - x_0$

$$y = mx + b$$

$$y = \left(\frac{dy}{dx}\right)x + b$$

$$y \cdot dx = x \cdot dy + b \cdot dx$$

$$x \cdot dy - y \cdot dx + b \cdot dx = 0 \xrightarrow[\text{by } 2]{\text{multiplied}} x \cdot 2dy - y \cdot 2dx + 2b \cdot dx = 0$$

$$ax + by + c = 0 \qquad\qquad ax + by + c = 0$$

By comparing coefficient :-

$a = dy$

$b = -dx$

Again, comparing coefficient :-

$a = 2dy$

$b = -2dx$

Replacing the value of a & b in d_{init} :-

$$d_{init} = a + 0.5b$$

$$= dy - 0.5dx$$

↓

a fraction is present, so we have to resolve it.

* কোনো line কে constant দিয়ে multiple করলে, line টা same থাকে

$$d_{init} = a + 0.5b$$

$$= 2dy + 0.5(-2dx)$$

$$\boxed{d_{init} = 2dy - dx}$$

Find out 7 intermediate pixels from $(0,2)$ to $(70,52)$ using MPL.

$(0,2) \rightarrow (x_0, y_0)$, $(70,52) \rightarrow (x_1, y_1)$

$dx = x_1 - x_0 = 70 - 0 = 70$

$dy = y_1 - y_0 = 52 - 2 = 50$

$d_{init} = 2dy - dx = 2(50) - 70 = 30$

$2dy - 2dx = 2(50) - 2(70) = -40 \longrightarrow$ for NE pixel

$2dy = 2(50) = 100 \longrightarrow$ for E pixel

x	y	d	E/NE	dupdate	Pixel
0	2	30	NE	30 − 40 = −10	(0,2)
1	3	−10	E	−10 + 100 = 90	(1,3)
2	3	90	NE	90 − 40 = 50	(2,3)
3	4	50	NE	50 − 40 = 10	(3,4)
4	5	10	NE	10 − 40 = −30	(4,5)
5	6	−30	E	−30 + 100 = 70	(5,6)
6	6	70	NE	70 − 40 = 30	(6,6)
7	7	30	NE	30 − 40 = −10	(7,7)

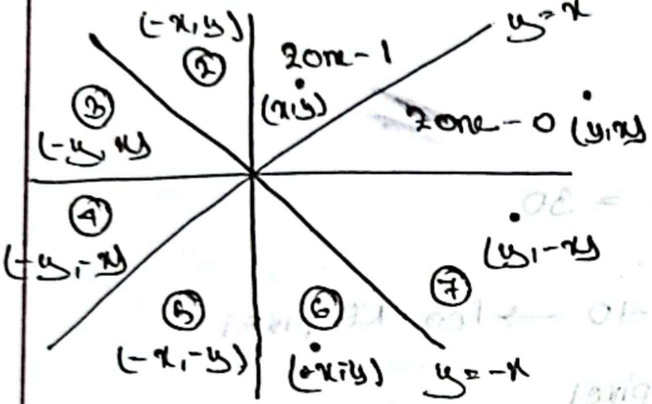
* ৭ টা step দেখাতে হবে including dinit.

LECTURE
3

THURSDAY DATE: 12/10/23

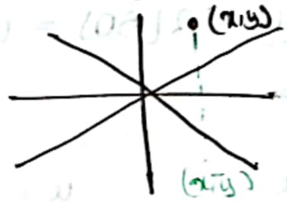
MIDPOINT CIRCLE ALGO

EIGHT-WAY SYMMETRY

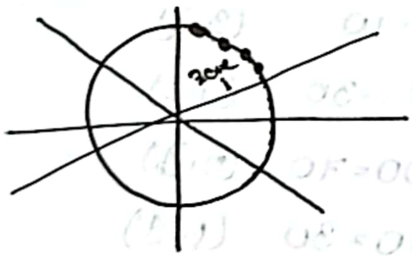


* 2 lines ~~inter~~ at 45° divides axis in 8 parts.

* এক zone এর particular point জানি, বাকি zone গুলোর value এর respect এ values এর বসাতে পারি

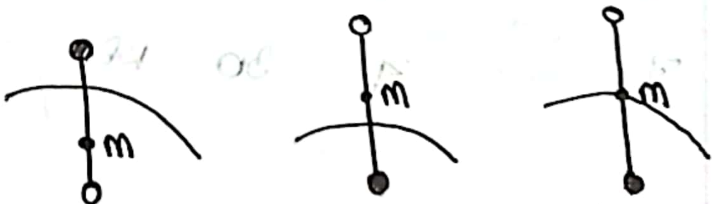


circle ~~এর~~ এর জন্য eight-way symmetry apply করি।
zone-1 এর ~~ff~~ জন্য mid-point algo চালাব

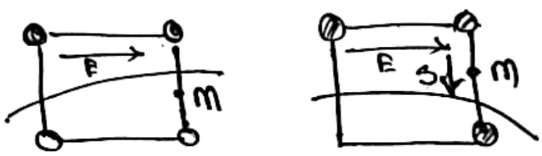


$d < 0 \rightarrow$ up (E)
$d \geq 0 \rightarrow$ below (SE)

Centre (0,0)
radius ~~(r)~~ = r
$x^2 + y^2 - r^2 = 0$
$f(x,y) = x^2 + y^2 - r^2$

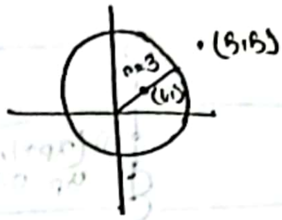


$d < 0$ (E) $d > 0$ (SE) $d = 0$ (SE)



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

$C(0,0)$ $r = 3$



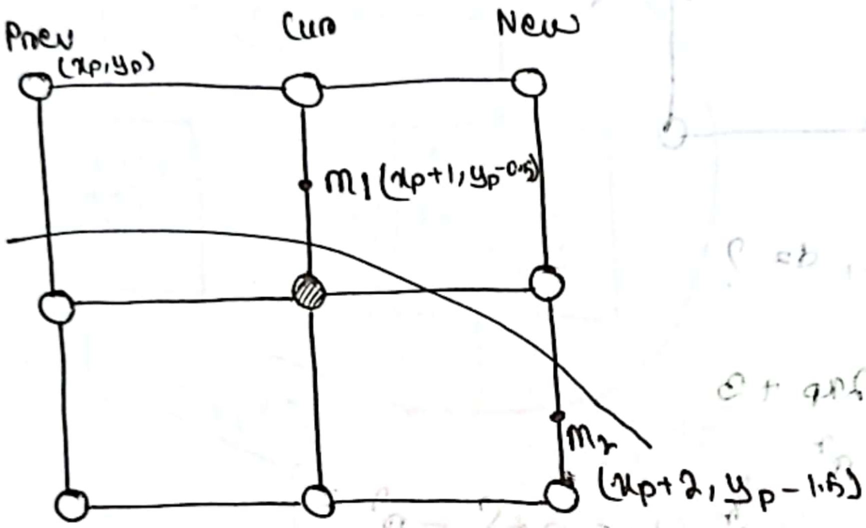
$f(x,y) = x^2 + y^2 - 9$

$f(1,1) = -7 < 0$

~~centre~~ circle এর ভিতরে value থাকলে, $d < 0$

$f(5,5) = 41 > 0$

~~centre~~ circle এর বাইরে value থাকলে, $d > 0$



if cur step = SE, next d = ?

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

$f(x_p+1, y_p - 0.5) = (x_p+1)^2 + (y_p - 0.5)^2 - r^2$

$$d_{current} = x_p^2 + 2x_p + 1 + y_p^2 - y_p + 0.25 - r^2 \quad \text{--- (1)}$$

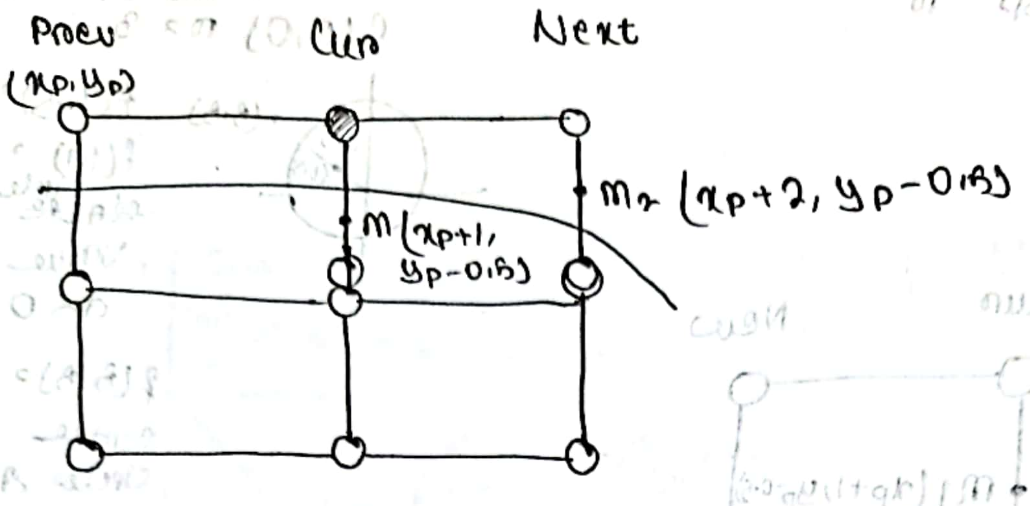
$f(x_p+2, y_p - 1.5) = (x_p+2)^2 + (y_p - 1.5)^2 - r^2$

$$d_{new} = x_p^2 + 4x_p + 4 + y_p^2 - 3y_p + 2.25 - r^2 \quad \text{--- (2)}$$

$$d_{new} - d_{current} = \cancel{x_p^2} + 4x_p + 4 + \cancel{y_p^2} - 3y_p + 2.25 - \cancel{r^2}$$
$$\underline{\phantom{d_{new} - d_{current} =} - \cancel{x_p^2} - 2x_p - 1 - \cancel{y_p^2} + y_p - 0.25 + \cancel{r^2}}$$

$$d_{new} - d_{current} = 2x_p + 3 - 2y_p + 2$$

$$\boxed{d_{new} = 2x_p + 5 - 2y_p + d_{current}} \rightarrow \text{SE}$$



If curr step = E, d = ?

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

$f(x_p+1, y_p-0.5) = (x_p+1)^2 + (y_p-0.5)^2 - r^2$

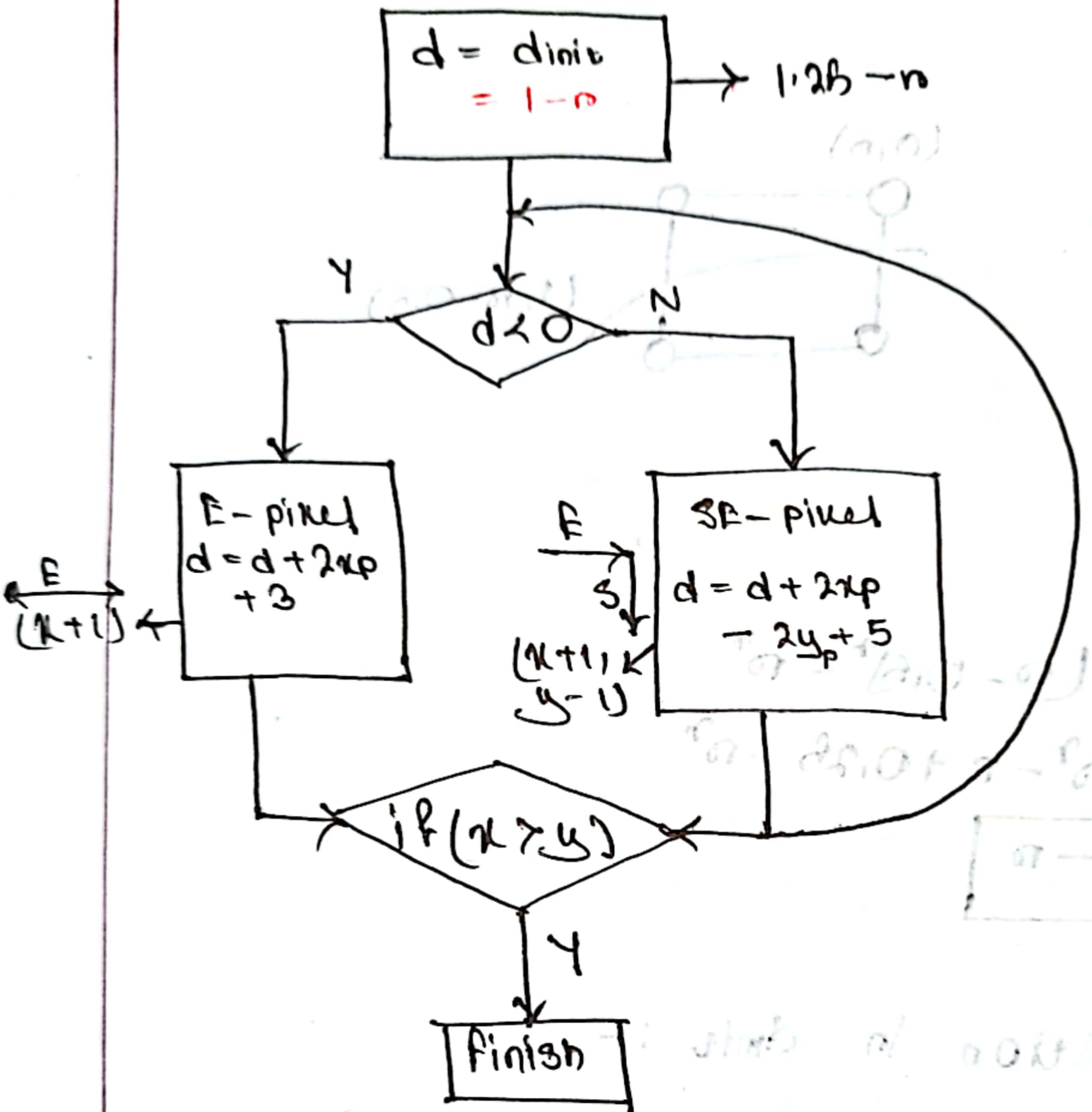
$= x_p^2 + 2x_p + 1 + y_p^2 + 2y_p - 0.25 - r^2$

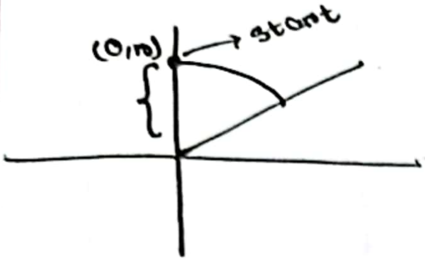
$f(x_p+2, y_p-0.5) = (x_p+2)^2 + (y_p-0.5)^2 - r^2$

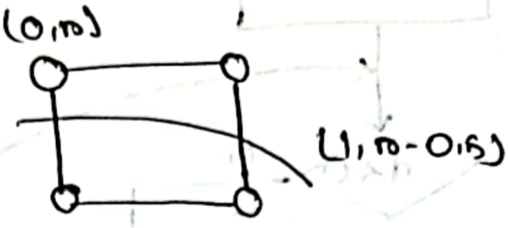
$= x_p^2 + 4x_p + 4 + y_p^2 + 2y_p - 0.25 - r^2$

$d_{next} - d_{current} = \cancel{x_p^2} + 4x_p + 4 + \cancel{y_p^2} + \cancel{2y_p} - \cancel{0.25} - \cancel{r^2}$
$- \cancel{x_p^2} - 2x_p - 1 - \cancel{y_p^2} - \cancel{2y_p} + \cancel{0.25} + \cancel{r^2}$

$\boxed{d_{next} = d_{current} + 2x_p + 3}$


d = dinit
= 1 - r
1.25 - r
Y
d<0
N
E - pixel
d = d + 2xp + 3
E
(x+1)
E
S
(x+1, y-1)
SE - pixel
d = d + 2xp - 2yp + 5
if (x>y)
Y
finish





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

$$f(1, \cancel{0.5}\ r - 0.5) = 1^2 + (r - 0.5)^2 - r^2$$

$$= 1 + r^2 - r + 0.25 - r^2$$

$$\boxed{d_{init} = 1.25 - r}$$

To remove the fraction in dinit :-

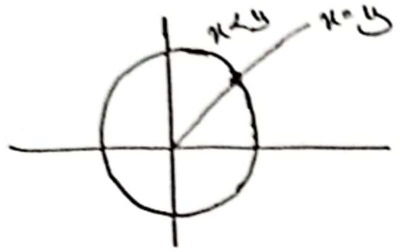
	$d = 1.25 - r$	$d = 1 - r$
$r=1$	0.25 (SE)	0 (SE)
$r=2$	−0.75 (E)	−1 (E)
$r=3$	−1.75 (E)	−2 (E)
$r=4$	−2.75 (E)	−3 (E)

* d এর 2 টা eq^n এর ~~jonno~~ জন্য d এর value different আসবে, but আমরা check করব pixel টা same আসবে কি না

* $d = 1.25 - r$ and $d = 1 - r$ use করলে for specific r, pixel same আসে, so $d = 1.25 - r$ না use করে $d = 1 - r$ use করতে পারি, fraction Round off ও করা লাগবে না।

$$\boxed{d_{init} = 1 - r}$$

$(x_0, y_0) \rightarrow (x_1, y_1)$



* zone 1 এ x এর value always y এর চেয়ে ছোট হবে
* mid point circle algo এর জন্য zone = 1 ধরে নিতে হবে

$x < y \rightarrow$ এর মধ্যে থাকলে, loop চালাতে থাকব

$x > y \rightarrow$ loop stops

Find out the pixels of a circle where centre (0,0) and radius = 15 using MCL.

x	y	d	E/SE	dupdate	Pixel (zone - 1)	zone 0 তে convert ↑ zone (0)
0	15	−14	E	−11	(0,15)	(15,0)
1	15	−11	E	−6	(1,15)	(15,1)
2	15	−6	E	1	(2,15)	(15,2)
3	15	1	SE	−18	(3,15)	(15,3)
4	14	−18	E	−7	(4,14)	(14,4)
⋮	⋮					

$x > y$

* zone convert করলে সব pixel এর জন্য circle create করতে পারব

~~* centre (0,0) না হলে, last এর value এর আগে just~~

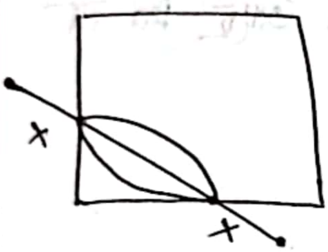
* centre (0,0) না হলে centre (30,20) হলে, then math এ কোনো changes হবে না, just last এ coordinate এর সাথে add করে দিব (x+30, y+20)
* centre (30,20), zone 4 থাকলে → first zone 4 নিয়ে যেতে centre add করব

LECTURE 4

THURSDAY DATE: 19/10/23

LINE CLIPPING ALGORITHM

- Cohen Sutherland
- Cyrus Beck



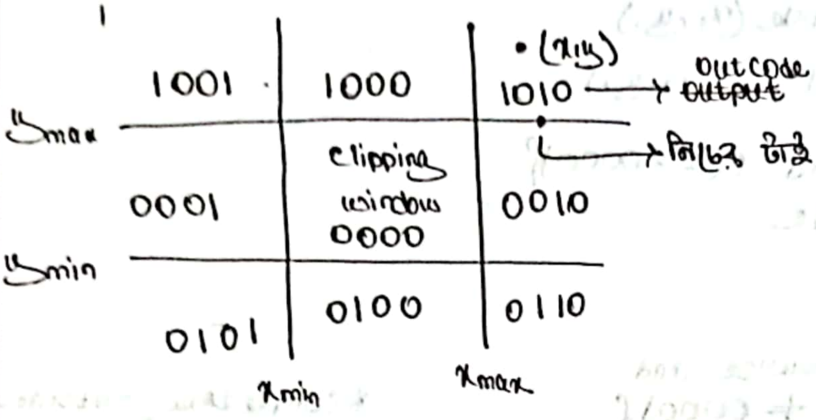
* screen এর ভিতরের line টা থাকবে, বাকি parts গুলোকে বাদ দিয়ে দিব

* If two points of a line are in the ~~win~~ clipping window (screen), then it is called "completely inside". → রেখে দিব

* If both the points of a line are not in the clipping window, then it is called "completely outside". → exclude করবো

* If one point is inside and the other point is outside the clipping window, then it is called "partially inside".
  → intersections বের করবো
  intersections এর ভিতরের যে region clipping window এর ভেতরে থাকবে, সেটা হিসেবে include করবো।
  বাকি part exclude করবো

COHEN SUTTERLAND



* 9 region is denoted by 4 bits.

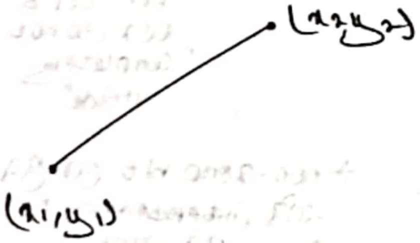
* bit দিয়ে outcode বলে for each region

(x,y) এর region এ থাকে, following condition follow করলে, bit no. 1 হবে

(above) $y > y_{max}$	(bottom) $y < y_{min}$	(right) $x > x_{max}$	(left) $x < x_{min}$	
1	0	1	0	
3	2	1	0	bit

* যেকোন two ten region এর মাঝে থাকলে, যে region এ minimum no. of 1 থাকবে, সে region এ point টা থাকবে।

* point যদি কোনো corner এ থাকে, then also same rule follow করবে।



First task : outcode বের করব

```
cohen (x1, y1, x2, y2){
  oc1 = create_outcode (x1, y1)
  oc2 = create_outcode (x2, y2)
  if (oc1 = 0000 and oc2 = 0000){
     completely inside
     break
  }
  else if (oc1 & oc2 ≠ 0000){      // & → bitwise and
     completely outside
     break
  }
  else {
     if (oc1 ≠ 0000){
        find non-zero bit of oc1
        find intersection point
        update oc1
     }
     else {
        find non-zero bit of oc2
        find intersection point
        update oc2
     }
     continue;
  }
```

* completely outside এর জন্য 2 টা point এর bit similar থাকবে। OC1 and OC2 কে multiple করবো। যদি, there's a similarity & is completely outside.

$$\begin{array}{r} OC1 = 1010 \\ OC2 = 0010 \\ \hline 0010 \end{array}$$

* $OC1 = 0101$ (Y_{min} (non-zero), X_{min}) — not 0000, so not "completely inside"

$$\begin{array}{r} OC1 = 0101 \\ OC2 = 1010 \\ \hline 0000 \end{array}$$

no similarity betn OC1 & OC2, so not "completely outside"

LEFT BOUNDARY INTERS

→ non-zero bit কে বের করি, ~~intersection এর point বের করি~~ line এর জন্য non-zero bit টা আছে সেই line এর সাথে intersection point বের করব।

① Left Boundary Intersection

$x = x_{min}$

$y = y_1 + m(x_{min} - x_1)$

② RIGHT Boundary Intersection

$x = x_{max}$

$y = y_1 + m(x_{max} - x_1)$

③ BOTTOM boundary Intersection

$y = y_{min}$

$x = x_1 + \frac{1}{m}(y_{min} - y_1)$

④ TOP boundary Intersection

$y = y_{max}$

$x = x_1 + \frac{1}{m}(y_{max} - y_1)$

Given Clip Region from $(-250, -200)$ to $(250, 200)$

P1 $(100, 50)$

P2 $(300, 100)$

$x_{min} = -250$, $x_{max} = 250$

$y_{min} = -200$, $y_{max} = 200$

$y > 200$	$y < -200$	$x > 250$	$x < -250$

~~OC1 from P1 →~~

OC1 from P1 →

$50 > 200$	$50 < -200$	$100 > 250$	$100 < -250$
F	F	F	F

; OC1 = 0000

OC2 from P2 →

$100 > 200$	$100 < -200$	$300 > 250$	$300 < -250$
F	F	T	F

; OC2 = 0010

OC1 = 0000 ⎫
OC2 = 0010 ⎭ not completely inside

$\underbrace{\text{0010}}_{\to x_{max}}$

0000 — not completely outside

x_{max} এ 1 তাই, x_{max} এর সাথে point এর intersection বের করব

$x = x_{max} \Rightarrow x = 250$

$y = \underbrace{y_1}_{} + m(x_{max} - \underbrace{x_1}_{})$ → যার জন্য math করব তার coordinate. Here, it's the coordinates ~~[illegible]~~ of P2.

$\cancel{= 100 + \quad (250 - 300)}$

$= 100 + \left(\dfrac{100 - 50}{300 - 100}\right)(250 - 300)$

$= 100 + \left(\dfrac{1}{4}\right)(-50)$

$= 100 - 12.5$

$y = 87.5$

$P_2'(250, 87.5)$

update OC2 from P_2' →

87.5 > 200	87.5 < -200	250 > 250	250 < -250
F	F	F	F

OC2 = 0000

Now,

OC1 = 0000 ⎫
OC2 = 0000 ⎭ loop আবার চালালে দেখা যাবে OC1 and OC2 both are 0000, so completely inside, loop break করে দিবে

★

$\boxed{\begin{array}{l} P_1(100, 50) \\ P_2'(250, 87.5) \end{array}}$

(Continued LECTURE 5

SUNDAY DATE: 29/10/23

EIGHT WAY SYMMETRY FOR MIDPOINT LINE ALGO

*Lecture 2 তে line টান দেখে বাকা দিয়ে যায়। সেটার জন্য eight way symmetry use করে mid-point line algo use করেছি।

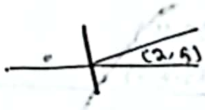
*অন্য direction এর জন্য mid point algo কেমন use করব সেটা eight way symmetry এর মাধ্যমে corner case handle করে করা লাগবে।

*একটা line কোন একটা zone এর property always follow করে সেটা বের করা লাগবে।

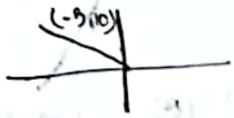
initial (x_1, y_1) and find (x_2, y_2)

$dx = x_2 - x_1$, $dy = y_2 - y_1$

$dx > 0, dy > 0$ and $|dx| > |dy| \rightarrow$ এই property follow করলে, points are in zone 0.



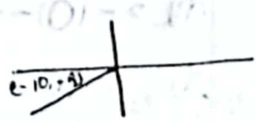
$dx > 0, dy > 0$ and $|dx| < |dy| \rightarrow$ zone 1



$dx < 0, dy > 0$ and $|dx| < |dy| \rightarrow$ zone 2

$dx < 0, dy > 0$ and $|dx| > |dy| \rightarrow$ zone 3

$dx < 0, dy < 0$ and $|dx| > |dy| \rightarrow$ zone 4



$dx < 0, dy < 0$ and $|dx| < |dy| \rightarrow$ zone 5

$dx > 0, dy < 0$ and $|dx| < |dy| \rightarrow$ zone 6

$dx > 0, dy < 0$ and $|dx| > |dy| \rightarrow$ zone 7

(1) Find out the zones from given points. zone = m

* zone = 0, হলেই direct mid point algo use করতে পারব।

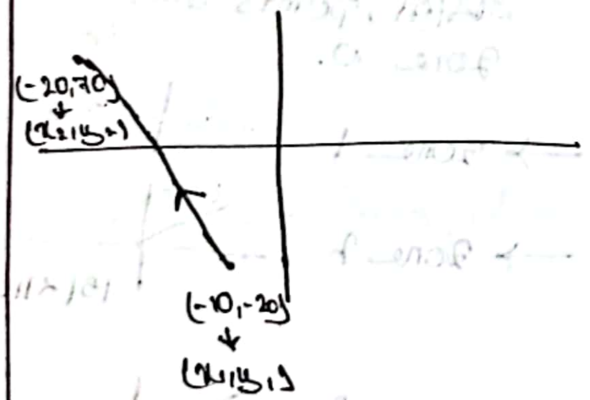
(2) Convert points from zone = m to zone = 0

* mid-point line apply করার জন্য zone = 0 হওয়া লাগবে, that's why conversion করা লাগবে

(3) Apply mid-point line algo on converted points.

(4) Convert back from zone = 0 to zone = m

Find out the intermediate points of a line from $(-10,-20)$ to $(-20,70)$



* initial point (x_1,y_1) and ↓ from final point (x_2,y_2) ↓ to

Step 1: Find out the zones from given points. zone = m

$dx = -20 - -10 = -10$

$dy = 70 - -20 = 90$

$dx < 0$, $dy > 0$ and $|dx| < |dy| \rightarrow$ zone $2 \rightarrow m$

Step 2: Convert points from zone = 2 to zone = 0

$(-x, y) \rightarrow (y, x)$ $(-10,-20) \dashrightarrow (-20,10)$

zone 2 $\rightarrow$ zone 0 $(-20,70) \dashrightarrow (70,20)$

(3) Apply midpoint line algo to converted points from (-20, 10) to (70, 20)

$dx = 70 - (-20) = 90$

$dy = 20 - 10 = 10$

$d_{init} = 2dy - dx = 2(10) - 90 = -70$

if NE : $2(dy - dx) = 2(10 - 90) = -160$

if E : $2dy = 2(10) = 20$

x	y	d	E/NE	dupdate
-20	10	-70	E	-50
-19	10	-50	E	-30
-18	10	-30	E	-10
-17	10	-10	E	10
-16	10	10	NE	-150
-15	~~10~~ 11	-150	E	-130
-14	11	-130	E	-110
-13	11	-110	E	-90
-12	11	-90	E	-70
-11	11	-70	E	-50
-10	11	-50	STOP.	

$\lvert -10 \rvert \cancel{/} \lvert 11 \rvert$

(4) convert back from zone 0 to zone 2

$(y, x) \rightarrow (-x, y)$

zone 0 $\rightarrow$ zone 2

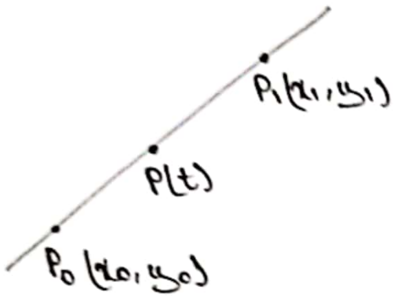
x	y	convert (x)	convert (y)
-20	10	-10	-20
-19	10	-10	-19
-18	10	-10	-18
-17	10	-10	-17
-16	11	-11	-16
-15	11	-11	-15
-14	11	-11	-14
-13	11	-11	-13

(online) LECTURE 6

THURSDAY — DATE: 02/11/23

CYRUS BECK

Parametric equation of line.



Parametric equation

$$P(t) = P_0 + t(P_1 - P_0)$$
$$= [(x_0, y_0) + t(x_1 - x_0, y_1 - y_0)]$$
$$= (x_0 + (x_1 - x_0)t, y_0 + (y_1 - y_0)t)$$

$P(t) = P_0 + t(P_1 - P_0)$

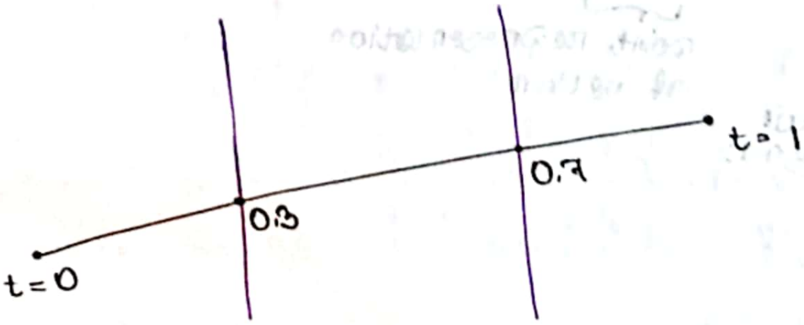
$t = 0$, $P(0) = P_0 + (0)(P_1 - P_0) \rightarrow P(0) = P_0$ → initial point

$t = 1$, $P(1) = P_0 + (1)(P_1 - P_0)$

$= \cancel{P_0} + P_1 - \cancel{P_0} \rightarrow P(1) = P_1$ → final point

* t এর value 0 বসালে, initial point এ যাই
* t এর value 1 বসালে, final point এ যাই

$0 \leq t \leq 1$ এর মধ্যে line টা এ যাবে, এ range এর বাইরে line এর extended portion যাবে



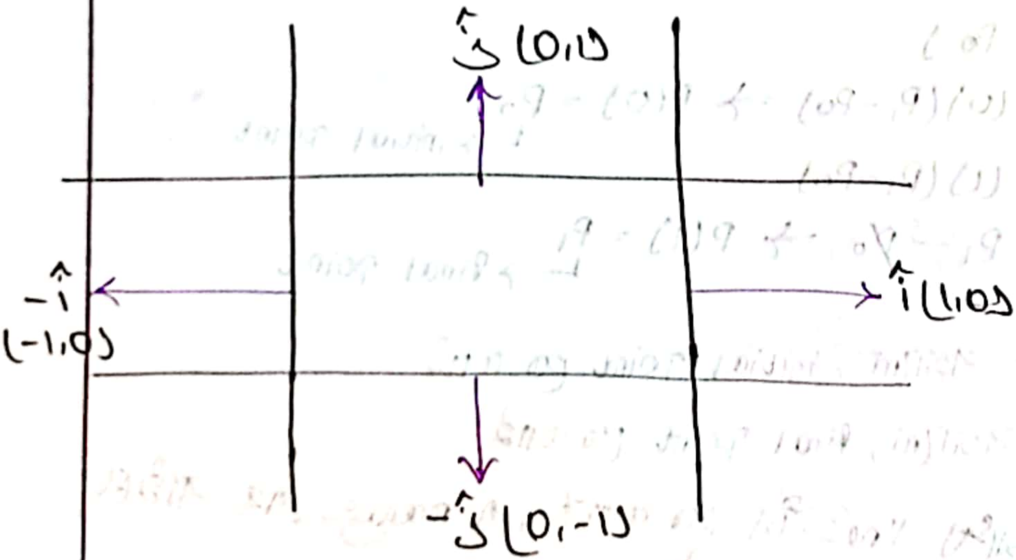
$0 \leq t \leq 1$ এ থাকলে আমরা guaranteed বলতে পারি যে line টা ("purple" line)/boundary কে intersect করবে, Otherwise, boundary cross করবে না,

LINE VECTOR

$$P_1 - P_0 = (x_1, y_1) - (x_0, y_0)$$

$$D = (x_1 - x_0,\ y_1 - y_0)$$

NORMAL VECTORS TO BOUNDARY



* normal vector draw করব boundary এ

* boundary গুলোকে vector ধরব

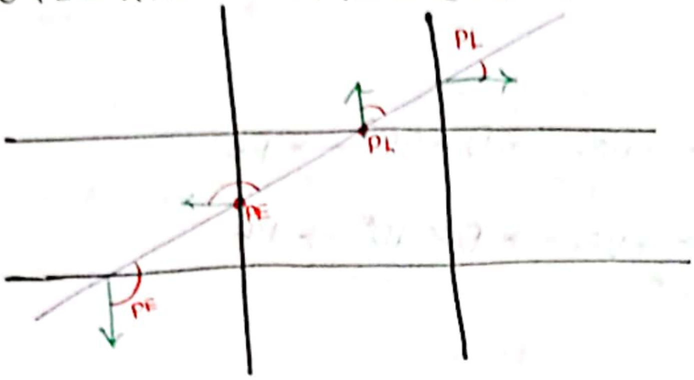
$P = 3\hat{i} + 2\hat{j} \longrightarrow \underbrace{(3,2)}_{\text{point representation of vector}}$

$\hat{i}$: unit vector of x-axis

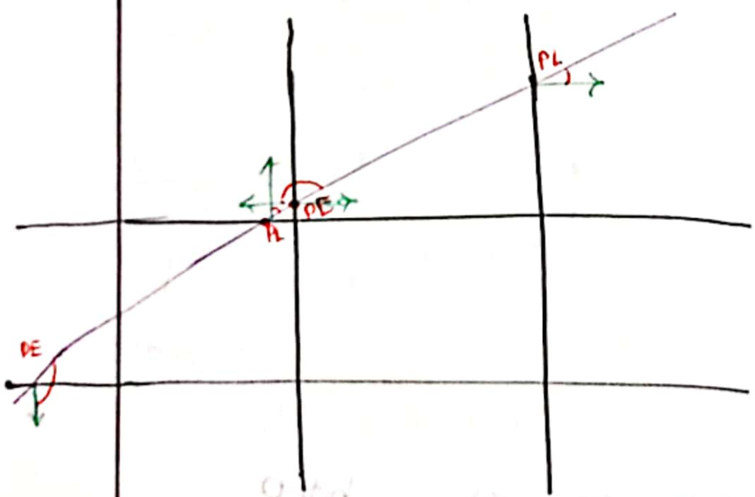
$\hat{j}$: unit vector of y-axis

POTENTIAL ENTERING (PE) & POTENTIAL LEAVING (PL)

max (PE)
min (PL)



* Normal vector + line এর angle যদি :-
  - 90° এর বড় হয় (obtuse) → PE
  - 90° এর ছোট হয় (acute) → PL
* PE এর মধ্যে আমরা সবসময় max টা কে নিব as actual PE.
* PL এর মধ্যে আমরা সবসময় min টা কে নিব as actual PL.



max (PE) > min (PL) → DRAW করতে হবে না
min (PL)

clipping window এ line নাই

এখানে leave আগে and enter পরে করেছে, which doesn't make sense. So, there's no clipping window.
Also, max(PE) > min(PL) তাই no clipping window is here.

normal vector → N

line vector → D

$\vec{a}\cdot\vec{b} = ab\cos\theta$

$N \cdot D = ND\cos\theta$ → $-ve \rightarrow \theta > 90° \rightarrow PE$

→ $+ve \rightarrow \theta < 90° \rightarrow PL$

$\theta > 90°, \cos\theta < 0$

$\theta < 90°, \cos\theta > 0$

Finding out the value of t :

$$t_{left} = \frac{-(x_0 - x_{min})}{(x_1 - x_0)}$$

$$t_{right} = \frac{-(x_0 - x_{max})}{(x_1 - x_0)}$$

$$t_{top} = \frac{-(y_0 - y_{max})}{(y_1 - y_0)}$$

$$t_{bottom} = \frac{-(y_0 - y_{min})}{(y_1 - y_0)}$$

Outcode :

bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
$z > z_{max}$	$z < z_{min}$	$y > y_{max}$	$y < y_{min}$	$x > x_{max}$	$x < x_{min}$
Far	Near	Above	Below	Right	Left

STEPS :-

(1) Parametric equation, t, $0 \le t \le 1$

(2) Line vector, $D = P1 - P0 = (x_1 - x_0, y_1 - y_0)$

(3) Normal vector (N), Right $(1,0)$, Left $(-1,0)$, Top $(0,1)$, Bottom $(0,-1)$

(4) PE, PL.

if max(PE) < min(PL) : Draw

else : Skip

$N \cdot D < 0$ (PE)

$N \cdot D > 0$ (PL)

$$t_{left} = \frac{-(x_0 - x_{min})}{(x_1 - x_0)}$$

$$t_{right} = \frac{-(x_0 - x_{max})}{(x_1 - x_0)}$$

$$t_{top} = \frac{-(y_0 - y_{max})}{(y_1 - y_0)}$$

$$t_{bottom} = \frac{-(y_0 - y_{min})}{(y_1 - y_0)}$$

Clip Regions are from (-100, -120) to (150, 200). Find the clip region for the given line from P0 (-125, 260) to P1 (195, -140)

$x_{min} = -100$ $\quad x_0 = -125$ } P0

$x_{max} = 150$ $\quad y_0 = 260$

$y_{min} = -120$ $\quad x_1 = 195$ } P1

$y_{max} = 150$ $\quad y_1 = -140$

Line vector, $D = P1 - P0$

$= (x_1 - x_0, y_1 - y_0)$

$= (195 - (-125), -140 - (260))$

$= (320, -400)$

$$t_{left} = -\frac{(x_0 - x_{min})}{(x_1 - x_0)} = -\frac{(-125 - (-100))}{195 - (-125)} \Rightarrow t_{left} = 0.078$$

$$t_{right} = -\frac{(x_0 - x_{max})}{(x_1 - x_0)} = -\frac{(-125 - 150)}{(195 - (-125))} \Rightarrow t_{right} = 0.859$$

$$t_{top} = -\frac{(y_0 - y_{max})}{(y_1 - y_0)} = -\frac{(260 - 150)}{(-140 - 260)} \Rightarrow t_{top} = 0.15$$

$$t_{bottom} = -\frac{(y_0 - y_{min})}{(y_1 - y_0)} = -\frac{(260 - (-120))}{(-140 - 260)} \Rightarrow t_{bottom} = 0.95$$

CamScanner

Boundary	N	N.D	PE/PL	t	t(E)=0	t(L)=1
left	(-1,0)	-320	PE	0.078	max (0, 0.078) = 0.078	1
right	(1,0)	320	PL	0.859	0.078	min (1, 0.859) = 0.859
top	(0,1)	-400	PE	0.15	max (0.078, 0.15) = 0.15	0.859
bottom	(0,-1)	400	PL	0.95	0.15	min (0.859, 0.95) = 0.859

$t(E) = 0.15,\ t(L) = 0.859$

$t(E) < t(L) \rightarrow$ DRAW

Parametric equation $\rightarrow P(t) = P(0) + t(P1 - P0)$

$= (x_0, y_0) + t(x_1 - x_0,\ y_1 - y_0)$

$t = 0.15,\ P(0.15) = (-125, 260) + 0.15(320, -400)$

$= (-125, 260) + \cancel{0.15}(48, -60)$

$= (-77, 200)$

$t = 0.859,\ P(0.859) = (-125, 260) + 0.859(320, -400)$

$= (150, -83.6)$

From $(-77, 200)$ to $(150, -83.6)$

DDA

(a) (20,5),(19,50)

$m = \frac{50-5}{19-20} \Rightarrow m = -45$

$\frac{1}{m} = \frac{1}{-45} = -0.022$

$x + \frac{1}{m}$	$y+1$	Pixel
20	5	(20,5)
19.97→20	6	(20,6)
19.95→20	7	(20,7)
19.93→20	8	(20,8)
19.90→20	9	(20,9)

(b) (-5,50),(-15,0)

$m = \frac{0-50}{-15--5} \Rightarrow m = 5$

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

$x + \frac{1}{m}$	$y+1$	Pixel
-5	50	(-5,50)
-4.8→-5	51	(-5,51)
-4.6→-5	52	(-5,52)
-4.4→-4	53	(-4,53)
-4.2→-4	54	(-4,54)

(c) (-10,-10),(48,24)

$m = \frac{24-(-10)}{48-(-10)} \Rightarrow m = 0.586$

$x+1$	$y+m$	Pixel
-10	-10	(-10,-10)
-9	-9.4→-9	(-9,-9)
-8	-8.83→-9	(-8,-9)
-7	-8.24→-8	(-7,-8)
-6	-7.66→-8	(-6,-8)

CLIPPING

(a) def Calculate_outcode (x, y, z):

```
bit0 = bit1 = bit2 = bit3 = bit4 = bit5 = 0
    if x < xmin :
        bit0 = 1
    if x > xmax :
        bit1 = 1
    if y < ymin :
        bit2 = 1
    if y > ymax :
        bit3 = 1
    if z < zmin :
        bit4 = 1
    if z > zmax :
        bit5 = 1
```

(b) if both points are completely inside / completely outside, no clipping is needed

if one point is inside and the other one is outside the clipping region, one clipping is needed.

Near	Far	Above	Below	Right	left

Name of intersection pts

(C) $x_{min} = -120$ | $y_{min} = -100$ | $x_{max} = 120$ | $y_{max} = 100$

P0 (200, −140)

P1 (−190, 130)

$D = P1 - P0$
$= (-190, 130) - (200, -140)$
$D = (-390, 270)$

N
above → (0, 1)
below → (0, −1)
right → (1, 0)
left → (−1, 0)

$t_{above} = \dfrac{-(y_0 - y_{max})}{(y_1 - y_0)} = 0.889$

$t_{below} = \dfrac{-(y_0 - y_{min})}{(y_1 - y_0)} = 0.148$

$t_{right} = \dfrac{-(x_0 - x_{max})}{(x_1 - x_0)} = 0.205$

$t_{left} = \dfrac{-(x_0 - x_{min})}{(x_1 - x_0)} = 0.821$

t	N.D	PE/PL	t(E) = 0	t(L) = 1
above 0.889	270	PL	0	min (0.889, 1) = 0.889
below 0.148	−270	PE	max (0, 0.148) = 0.148	0.889
right 0.205	−390	PE	max (0.148, 0.205) = 0.205	0.889
left 0.821	390	PL	0.205	min (0.889, 0.821) = 0.821

(E) < (L) ---→ DRAW

$P(t) = P0 + t(P1 - P0) \Rightarrow P0 + t(D)$

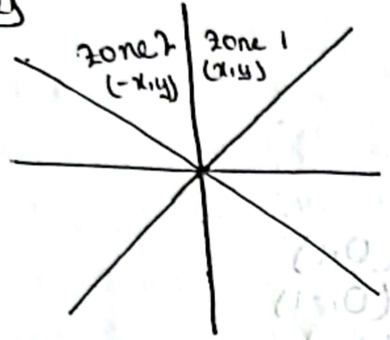
$t = 0.205,\ P(0.205) = (200, -140) + 0.205(-390, 270)$
$= (120.05, -84.65)$

$t = 0.821,\ P(0.821) = (200, -140) + 0.821(-390, 270)$
$= (-120.19, 81.67)$

from (120.05, −84.65) to (−120.19, 81.67)

CIRCLE DRAWING ALGORITHM

(a)



For zone 2 derive:
ΔW, ΔSW, dinit

For zone 1 derive:
ΔE, ΔSE, dinit

$$\boxed{d_{init} = 1 - r}$$

For zone 1 :-

if current = SE :- $[d > 0]$

$$f(x_p+1, y_p-0.5) = (x_p+1)^2 + (y_p-0.5)^2 - r^2$$
$$= x_p^2 + 2x_p + 1 + y_p^2 + y_p - 0.25 - r^2$$

$$f(x_p+2, y_p-1.5) = (x_p+2)^2 + (y_p-1.5)^2 - r^2$$
$$= x_p^2 + 4x_p + 4 + y_p^2 - 3y_p + 2.25 - r^2$$

$$\boxed{d_{next} = d_{current} + 2x_p - 2y_p + 5}$$

if current = E : $[d < 0]$

$$f(x_p+1, y_p-0.5) = x_p^2 + 2x_p + 1 + y_p^2 + y_p - 0.25 - r^2$$

$$f(x_p+2, y_p-0.5) = x_p^2 + 4x_p + 4 + y_p^2 - y_p - 0.25 - r^2$$

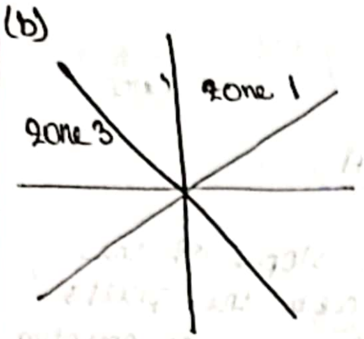
$$\boxed{d_{next} = d_{current} + 2x_p + 3}$$

zone 0 → zone 1
(X,Y) → (-X,Y)

x	y	d	SE/E	dup	Pixel
0	r				(-X,Y)
⋮	⋮	⋮	⋮	⋮	⋮

$x > y \to$ STOP

(b)



Using zone = 1, (0, n)

ΔN ↑N (y+1)

12 times → (0, n+12)

ΔNE N↑(y+1), E → (x+1)

7 times → (0+7, n+12+7)
(7, n+19)

zone = 1 ⇒ (x, y) ⇒ (7, n+19)

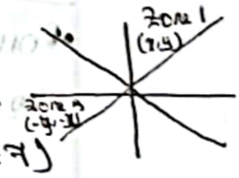
zone = 3 ⇒ (−y, x) ⇒ (−n−19, 7)

(c) dinit = 1.25 − n

Using this, we have to round off up/down the pixels which increases the time.

We can use dinit = 1 − n instead of dinit = 1.25 − n because both gives same pixel at given values of n.

However, dinit = 1 − n resolves the rounding off.



(d) dinit = 1 − n ⇒ 1 − 9 = −8

x	y	d	SE/E	dupd	(zone = 1) Pixel [x, y]	(zone = 5) Pixel [−y, −x]	(x − 12, y + 7) Pixel
0	9	−8	E	−5	(0, 9)	(−9, 0)	(−21, 7)
1	9	−5	E	0	(1, 9)	(−9, −1)	(−21, 6)
2	9	0	SE	−9	(2, 9)	(−9, −2)	(−21, 5)
3	8	−9	E	0	(3, 8)	(−8, −3)	(−20, 4)
4	8	0	SE	−3	(4, 8)	(−8, −4)	(−20, 3)
5	7	−3	E	10	(5, 7)	(−7, −5)	(−19, 2)
6	7	10	SE	13	(6, 7)	(−7, −6)	(−19, 1)
7	6	→ STOP [x > y]					

LINE DRAWING ALGORITHM

(a)(i)

MPL	DDA
• updates the value of d to get pixels of the line to be generated by increasing the value of x for E pixel and the value of x & y for ~~Pixel~~ NE pixel	• uses the slope of the line to get the pixels of the line by incrementing the value of x & y.
• faster ~~th~~ and more accurate than DDA.	• faster and more accurate than simple sol^n, but slower than midpoint line algo. This is mainly because of the rounding off problem in DDA
• Rounding off of DDA is resolved here.	• multiplication cost of simple sol^n is resolved here.
• Difficult to implement	• Easier to implement

(ii) If speed and accuracy is a critical factor, then midpoint line algo can be used. Otherwise, DDA can do perform the task.

(b) $1.5x - 1.2y - 120.0 = 0$

At Q, $y = 0$

$1.5x - 1.2(0) - 120.0 = 0$

$x = 80$

$Q(80, 0)$

At P, $x = 0$

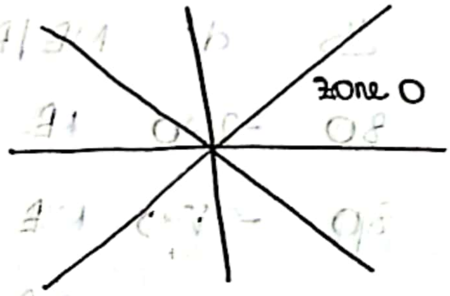
$1.5(0) - 1.2y - 120.0 = 0$

$y = -100$

$P(0, -100)$

$x_1 = 80$ | $x_0 = 0$

$y_1 = 0$ | $y_0 = -100$



$dx = x_1 - x_0 = 80$

$dy = y_1 - y_0 = 100$

$dx > 0$, $dy > 0$ and $|dx| < |dy| \rightarrow$ zone 1

$d_{init} = 2dy - dx$

$d = d + 2dy \rightarrow E\ [d \leq 0]$

$d = d + 2dy - 2dx \rightarrow NE\ [d > 0]$

zone 1 $\longrightarrow$ zone 0

$(X, Y) \longrightarrow (Y, X)$

$(80, 0) \longrightarrow (0, 80)$

$(0, -100) \longrightarrow (-100, 0)$

$P(-100, 0)$; $Q(0, 80)$

(c) $d_{init} = 2dy - dx$

$d_{init} = 60$

$dy = y_1 - y_0 = 80$

$dx = x_1 - x_0 = 100$

$NE \rightarrow 2dy - 2dx = -40$

$E \rightarrow 2dy = 160$

x	y	d	NE/E	dupd	Zone 0 Pixel (Y,X)	Zone 1 Pixel (X,Y)
-100	0	60	NE	20	(-100,0)	(0,-100)
-99	1	20	NE	-20	(-99,1)	(1,-99)
-98	2	-20	E	140	(-98,2)	(2,-98)
-97	2	140	NE	100	(-97,2)	(2,-97)
-96	3	100	NE	60	(-96,3)	(3,-96)
-95	4	60	NE	20	(-95,4)	(4,-95)
-94	5	20	NE	-20	(-94,5)	(5,-94)

(d) Line following in zone = 0 can be handled by midpoint line algo. Lines which follow in other regions cannot be handled by midpoint line algo solely. Using 8-way symmetry, conversion of x,y coordinates from zone = m to zone = 0 is possible. Corner cases of the midpoint line algo can be resolved.

FINAL

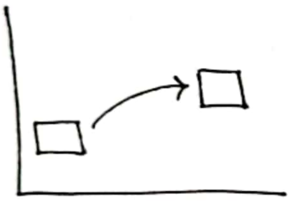
LECTURE 7

SATURDAY DATE: 18/11/23

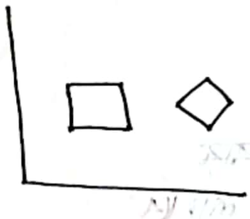
TRANSFORMATION

Transformation দ্বারা object এর হয়, individual pixels এর transformation করা possible.

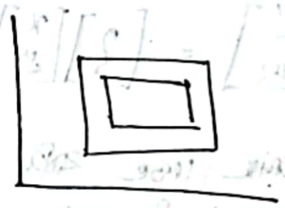
Object — set of points



Translation
[Object এর position shift করি]

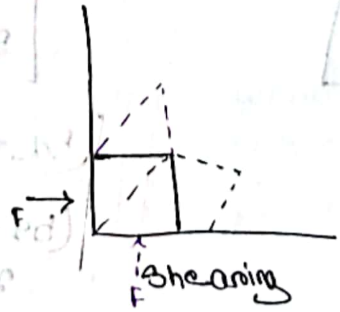


Rotation
[particular direction এ rotate করাই]

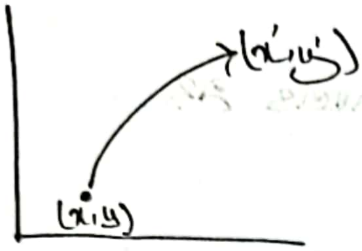


Scaling
[object কে ছোট/বড় করা]

Reflection
[particular line কে mirror ধরে, object কে reflect করাই]



Shearing
[x-axis বরাবর force দিই, then y-axis এ change হয় না, but x-axis এ change হয় shape change হয়]



* input point থাকবে

* input এর সাথে transformation matrix কে multiply করে output বের করবো

[output] = [matrix][input]

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} ? \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix}$$

* Matrix use করবো cuz যাতে যাতে different formulae use করা যায় যা for different ~~mat~~ transformation

$$w = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \times \begin{bmatrix} 4 & 5 \\ 6 & 7 \end{bmatrix}$$

$r \times c = 2 \times 3$ $\quad$ $\cancel{3}\, 2 \times 2$

shape of w = 2×2

$$\underset{2\times 2}{\begin{bmatrix} x & y \\ z & w \end{bmatrix}} \underset{2\times 1}{\begin{bmatrix} 5 \\ 6 \end{bmatrix}}$$

$$\underset{2\times 1}{\begin{bmatrix} (5x + 6y) \\ (5z + 6w) \end{bmatrix}}$$

$x' = x + t_x$

$y' = y + t_y$

$$\underset{2\times 1}{\begin{bmatrix} x' \\ y' \end{bmatrix}} = \begin{bmatrix} & \\ & \end{bmatrix} \underset{2\times 1}{\begin{bmatrix} x \\ y \end{bmatrix}}$$

* output and input 2x1
* input এর জন্য 2x2 এমন matrix multiple করলে ~~ø~~ 2x1 output matrix ~~এর~~ shape পায়

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$ → Cartesian ~~fo~~ Representation

Problem → $\begin{bmatrix} t_x \\ t_y \end{bmatrix}$ is an extra portion to the general form.

#

Homogenous representation

- 2×2 matrix টাকে এই ইচ্ছা করে এক dimension । বাড়াই দেয়
- 2×2 matrix কে 3×3 matrix বানাই

2D → 3D

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}_{3\times1} = \underbrace{\boxed{\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}}}_{3\times3} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}_{3\times1}$$

→ translation matrix

1 → dummy value [mathematical calculation এ কোন effect ফেলবে না

* x' এর জন্য x and tx লাগবে, 3×3 matrix এর first column

$$= \begin{bmatrix} x + t_x \\ y + t_y \\ 1 \end{bmatrix}$$

3D → 4D

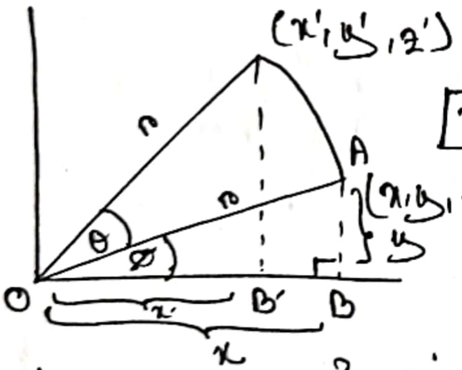
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix}_{4\times1} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}_{4\times4} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}_{4\times1}$$

$$= \begin{bmatrix} x + t_x \\ y + t_y \\ z + t_z \\ 1 \end{bmatrix}$$

Translate point $(3,2,7) \rightarrow$ 3 units in x-axis, 4 units in y-axis and ~~z~~ -3 in z-axis

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 3 \\ 0 & 1 & 0 & 4 \\ 0 & 0 & 1 & -3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 2 \\ 7 \\ 1 \end{bmatrix}$$

Rotation (z-based)
w.r.t origin



$\boxed{z' = z}$

* input point টা x-axis এর সাথে φ angle ~~move~~ make করেছে
* output point টা x-axis এর সাথে (θ+φ) angle make করেছে

For input (x, y, z):-

$\cos\phi = \frac{OB}{OA} = \frac{x}{r}$

$x = r\cos\phi$

$\sin\phi = \frac{AB}{OA} = \frac{y}{r}$

$y = r\sin\phi$

For output (x', y', z')

$$\cos(\theta+\phi) = \frac{OB'}{OA'} = \frac{x'}{r}$$

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

$$\sin(\theta+\phi) = \frac{A'B'}{OA'} = \frac{y'}{r}$$

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

$$x' = r\cos\theta\cos\phi - r\sin\theta\sin\phi$$
$$= \underbrace{r\cos\phi}_{x}\cos\theta - \underbrace{r\sin\phi}_{y}\sin\theta$$

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

$$y' = r\sin\phi\cos\theta + r\cos\phi\sin\theta$$
$$= \underbrace{r\sin\phi}_{y}\cos\theta + \underbrace{r\cos\phi}_{x}\sin\theta$$

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

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

rotation matrix (z – based)

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

CamScanner

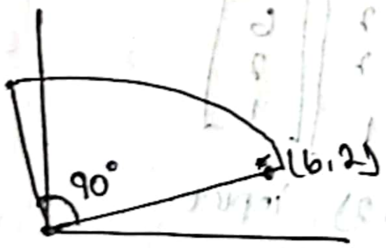
Rotate (z-based) point (9,2,5) at an angle 45° counterclockwise with respect to origin

* counterclockwise = +90°
* clockwise = −90°

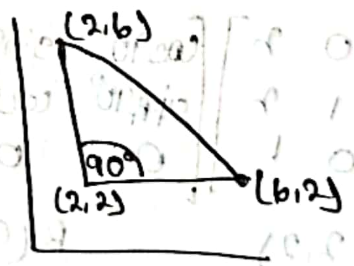
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos 45^\circ & -\sin 45^\circ & 0 & 0 \\ \sin 45^\circ & \cos 45^\circ & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 9 \\ 2 \\ 5 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 4.95 \\ 7.78 \\ 5 \\ 1 \end{bmatrix} \approx (4.95, 7.78, 5)$$

Rotate (z-based) point $(\cancel{9},2\cancel{5})$ (6,2) at an angle 90° counterclockwise with respect to (2,2)



rotate wrt (0,0)



rotate wrt (2,2)

* Question এ wrt (0,0) বলেনি, so point টা কে somehow (0,0) এর তে আনা লাগবে
* rotation apply করতে পারি
* এই output point কে (0,0) থেকে question এর wrt point এ পাঠায় দিব

Self-Study: Rotation (x-based)
Rotation (y-based)

① Take (2,2) → (0,0) → T(-2,-2)

* Translation করতে লাগবে

② Apply Rotation → R(90°)

③ Take (0,0) → (2,2) → T(2,2)

Composite Transformation

$$[\text{output}] = \underset{③}{T(2,2)} * \underset{②}{R(90)} * \underset{①}{T(-2,-2)} * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

* sequence of steps break করা যাবে না

* Composite transformation — more than one transformation takes place.

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \underset{T(2,2)}{\begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix}} \underset{R(90°)}{\begin{bmatrix} \cos 90° & -\sin 90° & 0 \\ \sin 90° & \cos 90° & 0 \\ 0 & 0 & 1 \end{bmatrix}} \underset{T(-2,2)}{\begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix}} \underset{\text{input}}{\begin{bmatrix} 6 \\ 2 \\ 1 \end{bmatrix}}$$

$$= \begin{bmatrix} 2 \\ 6 \\ 1 \end{bmatrix} \approx (2,6)$$

* Right to left calculate করি

* Here, ~~Mult Multiply~~ T(-2,2) * input, the output is then used with R(90°). This output is used with T(2,2) to get the final output.

Rotation (x-based)

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

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

$$= \begin{bmatrix} x \\ y\cos\theta - z\sin\theta \\ y\sin\theta + z\cos\theta \\ 1 \end{bmatrix}$$

Rotation (y-based)

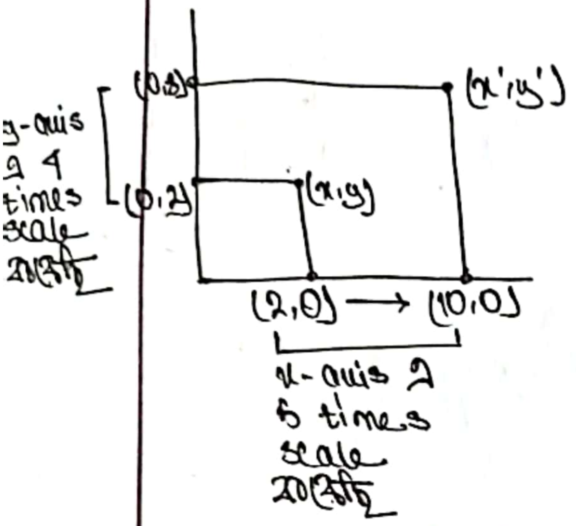
$y' = y,\ x' = x\cos\theta + z\sin\theta,\ z' = -x\sin\theta + z\cos\theta$

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

$$= \begin{bmatrix} x\cos\theta + z\sin\theta \\ y \\ -x\sin\theta + z\cos\theta \\ 1 \end{bmatrix}$$

LECTURE 7 FRIDAY DATE: 24/11/23

Scaling



→ multiply

x axis, S_x unit scal

y axis, S_y unit scal

$x' = x \cdot S_x$

$y' = y \cdot S_y$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{scaling matrix}} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

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

Let's assume a point with coordinates (6,3) from a rectangle has to be scaled 3 units and 5 units in x and y-axis respectively.

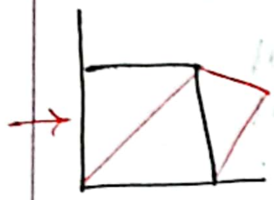
Calculate the new-coordinate

$S_x = 3, S_y = 5$

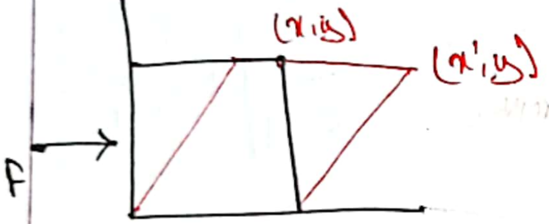
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 6 \\ 3 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 18 \\ 15 \\ 1 \end{bmatrix} \Rightarrow (18, 15)$$

Shearing



x-axis [shearing x-axis বরাবর হয়]



$y' = y$

→ যত amount এ shear করব তা বলা থাকবে (a amount)

* ছোট rectangle এ shearing এর distance কম হবে। y এর value কম।

* বড় rectangle এ shearing distance বড় হবে। y এর value বড়।

* shearing y এর উপর depend করে।

y ছোট — shearing distance ছোট

y বড় — shearing distance বড়

So, we multiply a with y. → $x' = x + ay$ [for x-axis shearing]

$x' = x + ay$

$y' = y$

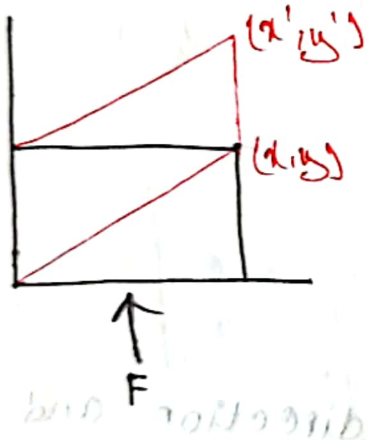
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \boxed{\begin{bmatrix} 1 & a & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

shearing matrix for x-axis

$$= \begin{bmatrix} x + ay \\ y \\ 1 \end{bmatrix}$$

y- axis : (b amount)

⟶ shear করা হয়েছে



$x' = x$

$y' = y + bx$

[x axis এর reflect করলে y-axis পাচ্ছি]

shearing matrix for y-axis

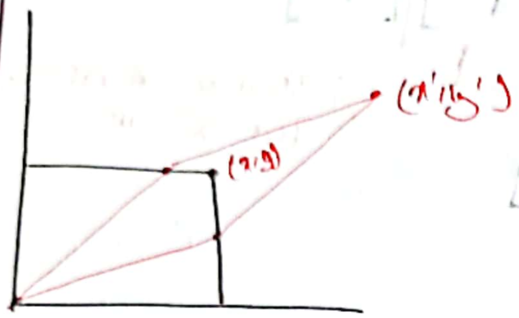
$x' = x$

$y' = y + bx$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \boxed{\begin{bmatrix} 1 & 0 & 0 \\ b & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} x \\ y + bx \\ 1 \end{bmatrix}$$

xy



$$x' = x + ay$$
$$y' = y + bx$$

shearing matrix for xy axis

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \boxed{\begin{bmatrix} 1 & a & 0 \\ b & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} x + ay \\ y + bx \\ 1 \end{bmatrix}$$

Shear the point (4,7), 2 units in x direction and 5 units in y direction wrt (2,-1).

origin এ নাই
so origin এ
নিয়ে এসে, solve
করব using shearing.
তারপর translate
করে ফিরে back to
initial position

(1) $(2,-1) \longrightarrow (0,0)$ $\quad T(-2,1)$

(2) Shearing $\quad SH(2,5)$

(3) $(0,0) \longrightarrow (2,-1)$ $\quad T(2,-1)$

Output = $T(2,-1) * SH(2,5) * T(-2,1) *$ Input

$$= \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 0 \\ 5 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 4 \\ 7 \\ 1 \end{bmatrix}$$

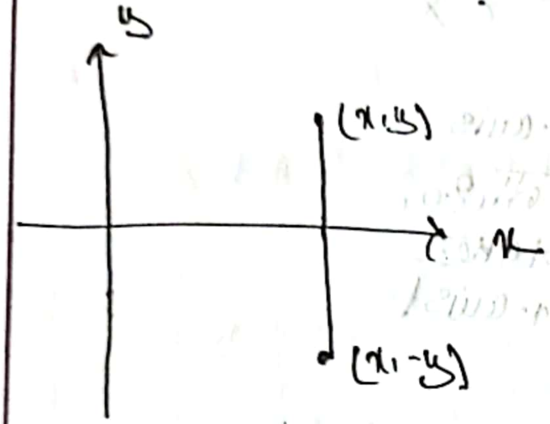
$$= \begin{bmatrix} 20 \\ 17 \\ 1 \end{bmatrix} = (20, 17)$$

Reflection

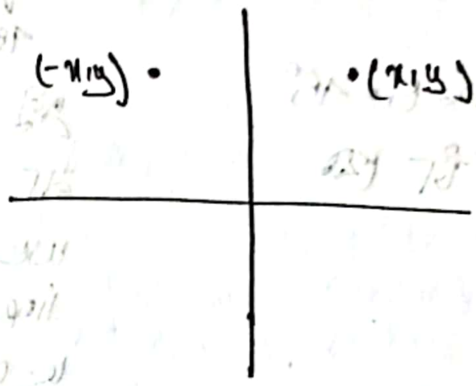
→ Reflection এর জন্য mirror

→ Reflection can occur wrt x-axis or wrt y-axis.

x-axis



y-axis



$x' = x$

$y' = -y$

reflection matrix at x-axis

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \boxed{\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ -y \\ 1 \end{bmatrix}$$

$x' = -x$

$y' = y$

reflection matrix at y-axis

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \boxed{\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} -x \\ y \\ 1 \end{bmatrix}$$

* এইগুলো হল x-axis and y-axis এর respect এ reflection হলে and তাদের reflection matrix বের করি,

* যখন given points এ reflection হবে, তখন method use করব।

*** Reflect point (10,8)

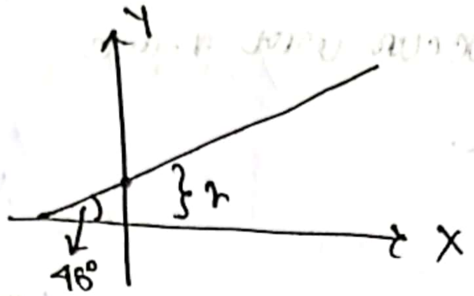
$y = mx + c$

$y = x + 2$

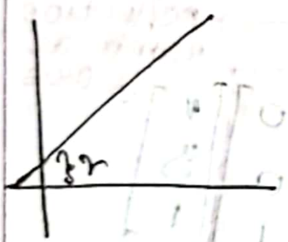
$m = 1, c = 2$

$m = \tan\theta$

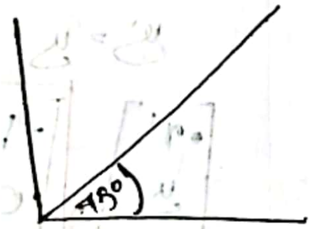
$\tan\theta = 1 \Rightarrow \theta = 45^\circ$



এই line টা x-axis এ না, y-axis এ না, so, we have to take line either at x-axis/ y-axis.

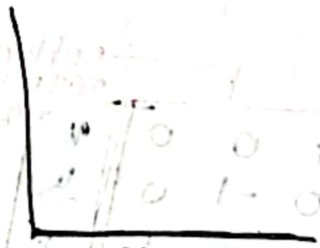


$\xrightarrow{T(0,-2)}$



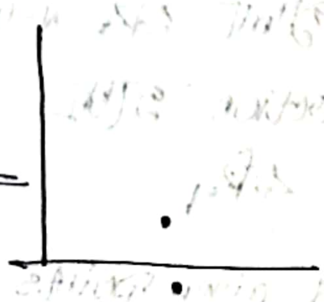
(0,0)
line টা origin এ চলে আসবে

$\xrightarrow{R(-45^\circ)}$

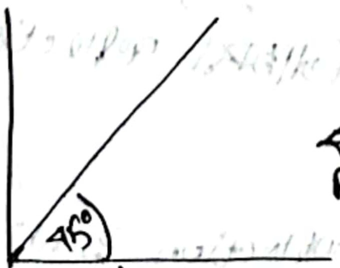


line টাকে 45° clockwise rotate করায় x-axis এ নিয়ে যাবে।

↓ Reflection (x)

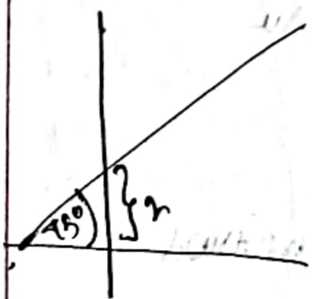


$\xleftarrow{R(45^\circ)}$



line টাকে counter clockwise rotate করে আগের angle এ ফেরত আনলাম

$\xleftarrow{T(0,2)}$



line টা কে আগের position নিয়ে গেলাম

output = T(0,2) * R(45°) * Ref (x) * R(-45°) * T(0,-2) * P

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos 45° & -\sin 45° & 0 \\ \sin 45° & \cos 45° & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(-45°) & -\sin(-45°) & 0 \\ \sin(-45°) & \cos(-45°) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 10 \\ 5 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 3 \\ 12 \\ 1 \end{bmatrix} \Rightarrow (3, 12)$$

LECTURE 10

Color model in Computer Graphics

→ কি কি type এর color generate করা যায়

→ Fundamental model of color model

① Additive Color model

② Subtractive Color model

Additive Color model	Subtractive color model
1. RGB, HSV, HSL	1. CMY, CMYK
2. ACM generates color by adding multiple colors.	2. SCM generates color by subtracting different color.
3. Active display: monitors, phone.	3. devices which deposite color: Printer, Compiler
4. No data → Black	4. No data → White
5. Increase brightness	5. Decrease Brightness

RGB Color model

R = Red, G = Green, Blue = Blue

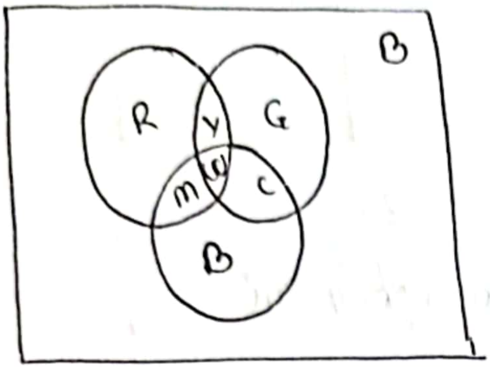
combination

2^8 → bit দিয়ে represent করা

2^8 2^8

2^{24} = 16 million combination of different color

* কোন data না থাকলে by default black থাকবে screen



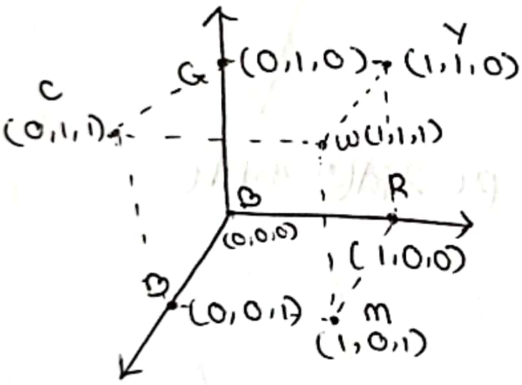
$R + G = Y$

$G + B = C$ (Cyan)

$R + B = m$ (magenta)

Basic Structure of RGB color data

RGB Color Cube



* প্রত্যেক component এর value 0 to 1 এর মধ্যে থাকে

* এই color cube এর প্রত্যেকটা point different different color দেয়। 8 টা different produce করে

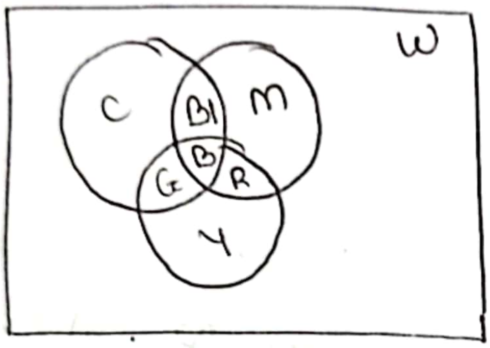
* তার জন্য আমরা 2^{24} combination of color পাই using RGB color model

* Intensity বাড়ানো এবং কমানো দিয়ে different color পাই

CMY

C = Cyan   M = Magenta   Y = Yellow

* By default, data না থাকলে screen white হবে

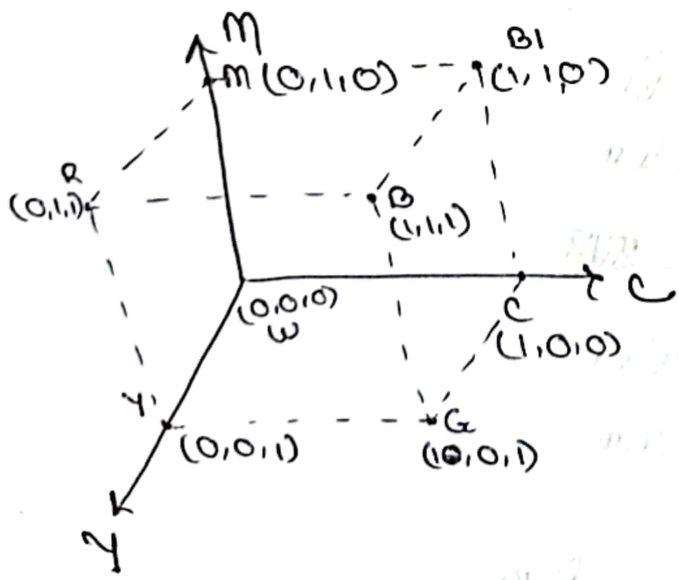


basic structure of CMY color model

C + M = Blue
C + Y = Green
M + Y = Red

* RGB এর উল্টো model হল CMY হল বলতে পারি

CMY Color Cube



CMY
↓
2^{24} color combination

RGB to CMY conversion & vice versa

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

(Q) RGB (0.6, 0.2, 0.9) $\longrightarrow$ CMY (0.4, 0.8, 0.1)

CMY (0.3, 0.4, 0.5) $\longrightarrow$ RGB (0.7, 0.6, 0.5)

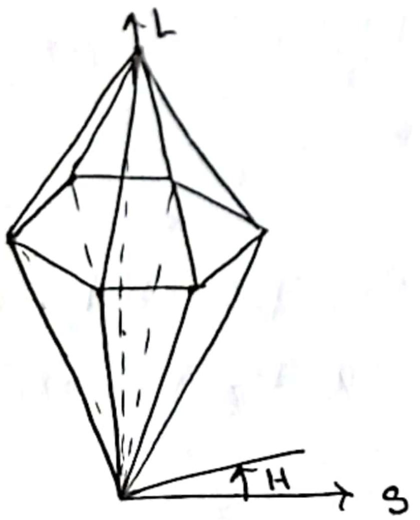
HSL

H = Hue, S = Saturation, L = Lightness

- H = Hue $\downarrow$ Helps to select color $\downarrow$ [0° - 360°] $\downarrow$ angle change করে color change করা যায়
- S = Saturation $\downarrow$ intensity of the color $\downarrow$ [0 - 1]
  - $\swarrow$ low intensity color $\downarrow$ grey দেখায়
  - $\searrow$ high intensity color $\downarrow$ dark color হয়
- L = Lightness $\downarrow$ Background color $\downarrow$ [0 - 1]
  - $\swarrow$ Fully black
  - $\searrow$ fully white

* H এর মাধ্যমে যে color পাবো তার intensity control করে S.

* Brightness(L) 0 বা 1 এ নিলে কোনো saturation (S) & Hue (H) কাজ করবে না



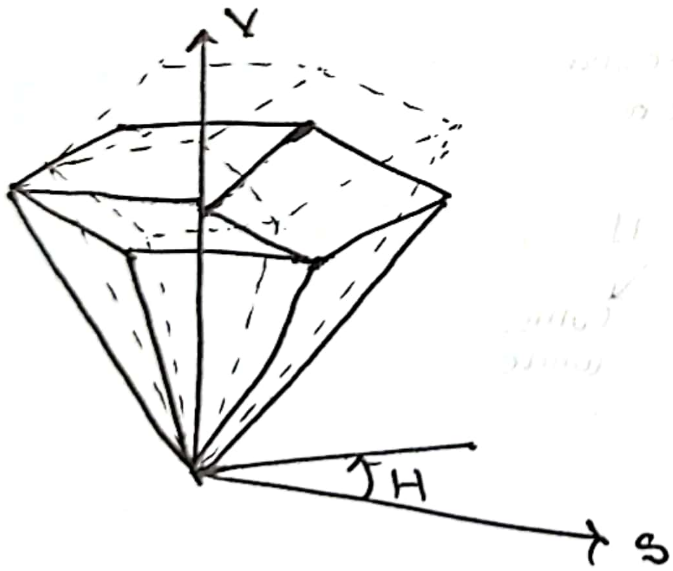
田 RGB color model → HSV/HSL color model

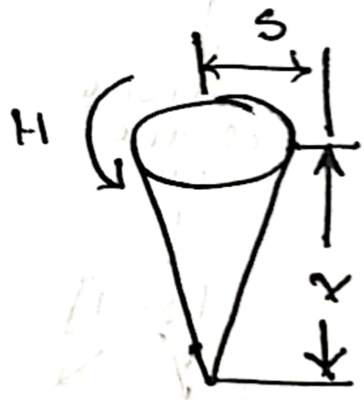
HSV Color Model

H = Hue , S = Saturation, V = Value (Brightness)

↓

It works in conjunction with saturation and disturbs the intensity or the brightness of the color.





RGB color model $\longrightarrow$ HSV/HSL color model

RGB(0.2, 0.4, 0.8) $\longrightarrow$ HSL

Conversion Algorithm:

Step 1: $max = MAX(R,G,B)$ $\longrightarrow$ R,G,B এর মধ্যে যার value max তাকে নিব

$min = MIN(R,G,B)$ $\longrightarrow$ R,G,B এর মধ্যে যার value min তাকে নিব

$l = max - min$

Step 2:

(HSL) $\Rightarrow$ $L = \dfrac{max + min}{2}$

$$S = \frac{l}{1 - |2L - 1|}$$

H:

if $max == R$:

$$H = \left[\frac{(G-B)}{l}\right] * 60^\circ \quad \{\text{if } H < 0, H = H + 360^\circ\}$$

elif $max == G$:

$$H = \left[\frac{(B-R)}{l}\right] * 60^\circ + 120^\circ$$

elif $max == B$

$$H = \left[\frac{R-G}{l}\right] * 60^\circ + 240^\circ$$

HSV :

$V = max$

$S = \frac{\ell}{max} = \frac{max - min}{max}$

$H =$ same as HSL

Math $\longrightarrow$

RGB $(0.7, 0.25, 0.3) \longrightarrow$ HSV / HSL

$R = 0.7$, $G = 0.25$, $B = 0.3$

$max = 0.7$

$min = 0.25$

$\ell = max - min$

$= 0.7 - 0.25$

$\ell = 0.45$

HSV :

$V = max = 0.7$

$S = \frac{\ell}{max} = \frac{0.45}{0.7} = 0.64$

$H = \left(\frac{G-B}{\ell}\right) * 60°$

$= \left(\frac{0.25 - 0.3}{0.45}\right) * 60°$

$= -6.67° + 360°$

$\downarrow$

to convert back θ to [0°-360°]

$H = 353.33°$

HSL :

$L = \frac{max + min}{2}$

$= \frac{0.7 + 0.25}{2}$

$L = 0.475$

$S = \frac{\ell}{1 - |2L - 1|}$

$= \frac{0.45}{1 - |2(0.475) - 1|}$

$S = 0.474$

$H = 353.33°$

CMY $(0.7, 0.3, 0.2) \longrightarrow$ HSV/HSL

CMY $\longrightarrow$ RGB

$= 1 - \text{CMY}$

$\text{RGB} = (0.3, 0.7, 0.8)$

$max = 0.8, \; min = 0.3, \; \ell = 0.8 - 0.3 = 0.5$

HSV:

$V = max = 0.8$

$S = \dfrac{\ell}{max} = \dfrac{0.5}{0.8} = 0.65$

$H = \left(\dfrac{R-G}{\ell}\right) * 60^\circ + 240^\circ$

$= \left(\dfrac{0.3-0.7}{0.5}\right) * 60^\circ + 240^\circ$

$= 192^\circ$

HSL

$L = \dfrac{max + min}{2} = \dfrac{0.8 + 0.3}{2} = 0.55$

$S = \dfrac{\ell}{1 - |2L - 1|} = \dfrac{0.5}{1 - |2(0.55) - 1|} = 0.5$

$H = 192^\circ$

LECTURE 11

Lightning & Illumination

Lightning

→ Light এর কারণে কি কি benefits পায় জানতে পারি

→ Pixel + Lightning use করলে different angle এর view কেমন হতে পারে সেটাই view আনতে পারি

Illumination

Direct illumination

- Light direct main object এ যায়, আমরা view point থেকে সেই object দেখতে পারব।

Indirect Illumination

- Light অন্য object এ যায়, reflect হয়ে main object যায়, এরপর আবার view point থেকে আমরা main object দেখতে পাই।

Reflection

Ambient Reflection
(Environmental light)

- environment light এর কারণে যে reflection হয়
- always ই থাকে

Diffuse reflection
(main light)

- object এর উপর আমরা proper ভাবে main light দিলে, যে reflection আসবে তাকে আগে view করায়।

Specular reflection
(main light) (viewpoint/shininess)

- main light এর জন্য reflection হয়
- view point এর depend করে
- Shininess এর উপর

* Ambient + Diffuse reflection merge করলে ভালো visualization পাই (realistic হয় না)

* Specular reflection ও যদি ambient + ~~reflection~~ diffuse reflection এর সাথে merge করি তাহলে আরো clear view পাই।
  ↳ Phong's Reflection Model

Ambient Light

$A = I_a K_a$ → $K_a = 1$ → object clearly দেখা যায়। as maximum reflection হয়

→ $K_a = 0$ → object কে দেখতে পাবে না as no reflection occurs

K_a ↓ (ambient coefficient) ↳ একটা object কত টুকু reflect করতে পারে

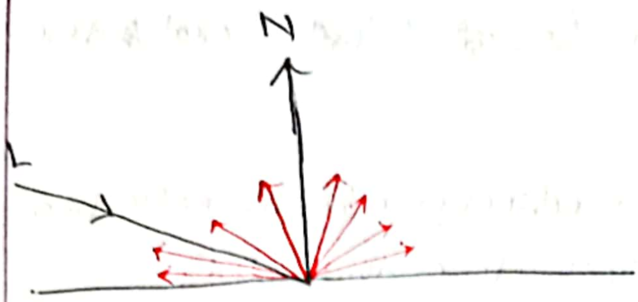
I_a ↓ (intensity of ambient light) ↓ দিনে বেশি light থাকে, তাই intensity of ambient light বেশি থাকে and vice versa

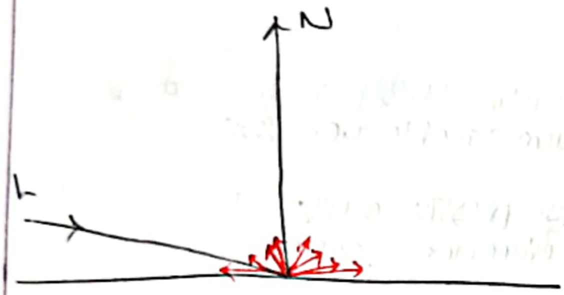
Diffuse Reflection

* Light গুলো যখন surface এ পড়ে, different জায়গায় light distribute হয়

* চারদিকে light spread হয়ে যায়

* এখন reflection ছাড়া কালো object কে আমরা ভালো করে দেখতে পারি





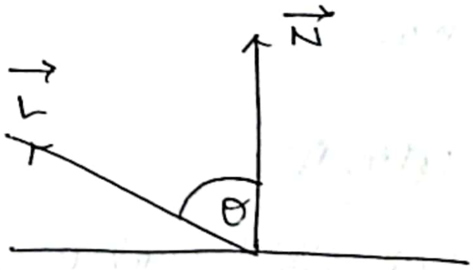
* Light normal থেকে দূরে
তাই diffuse বেশি হয়.
but diffuse এর amount
টা কমে গেছে

$D = I_p K_p \cos\theta$

$\cos\theta$ → L and N এর মাঝের angle

K_p → diffuse co-efficient

I_p → intensity of point line



$\theta\uparrow$, $\cos\theta\downarrow$

$\theta = 30^\circ$, $\frac{\sqrt{3}}{2}$

$\theta = 60^\circ$, $\frac{1}{2}$

$\theta = 90^\circ$, 0

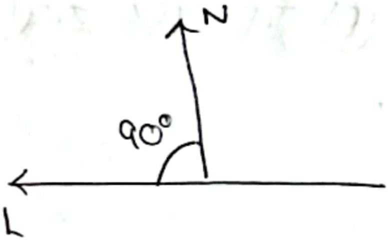
$\theta\uparrow, \cos\theta\downarrow, D\downarrow$

* L and N এর মধ্যে angle (θ)
বাড়লে, cosθ কমে,
cosθ কমলে D এর value
কমে, তাই diffuse reflection
কমে যায়

$\theta = 90^\circ$

$\cos\theta = 0$ →

$D = 0$



$\theta > 90^\circ$

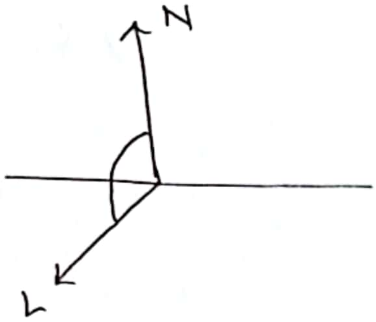
$\cos\theta < 0$ →

$D < 0$

$D = -ve$

↓

not possible



* So আমাদের যে D এর -ve value যে 0 করতে হবে, তার জন্য system করা না পারে জন্য $\theta > 90^\circ$ হলে D এর value by-default 0 হয়ে যাবে।

↓

$D = I_p K_p \cos\theta$

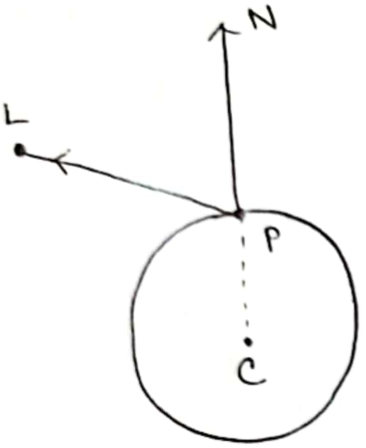
↓ updated

$D = I_p K_p \max(\cos\theta, 0)$ → কোন কারণে $\theta > 90^\circ$ হলে, D by default 0 হয়ে যাচ্ছে

$\theta < 90^\circ$, $\cos\theta = +ve$, $D = I_p K_p \cos\theta$

$\theta > 90^\circ$, $\cos\theta = -ve$, $D = 0$

* computer কিভাবে angle (θ) টা বের করতে পারবে, we have to use an algorithm.



$a \longrightarrow b$

$\underbrace{\vec{b}}_{\text{final}} - \underbrace{\vec{a}}_{\text{initial}}$

$\vec{L} = L - P$ [Line vector]

$\vec{N} = P - C$ [Normal vector]

unit vector of $\vec{L}$ and $\vec{N}$ বের করব

$\hat{L} = \dfrac{\vec{L}}{|\vec{L}|}$, $\hat{N} = \dfrac{\vec{N}}{|\vec{N}|}$

$a \cdot b = |a||b|\cos\theta$

$\hat{L} \cdot \hat{N} = |\hat{L}||\hat{N}|\cos\theta$

$= (1)(1)\cos\theta$

$\hat{L} \cdot \hat{N} = \cos\theta$

$\boxed{\cos\theta = \hat{L} \cdot \hat{N}}$

$D = I_D K_D \max(\hat{L} \cdot \hat{N}, 0)$

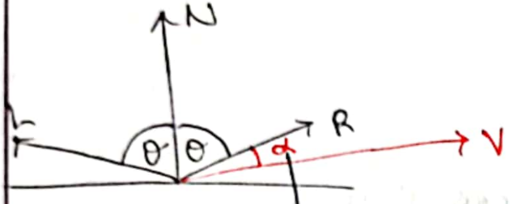
→ final formula for diffuse reflection.

কি shininess কোন উপর depend করে?
- angle between reflected ray and viewpoint (α)

Specular Reflection

→ Object এর view point depend করে

→ shininess measure করে



determines shininess
(একটা Object এর কোন দিক কতটা shine করে determine করে)

$$S = I_p K_s (\cos\alpha)^n$$

n → specular exponent

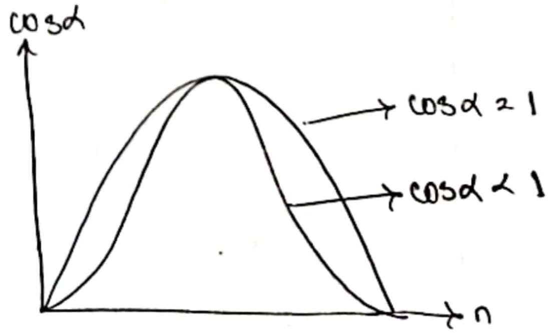
$\cos\alpha$ → angle betⁿ reflected ray (R) & view point (V)

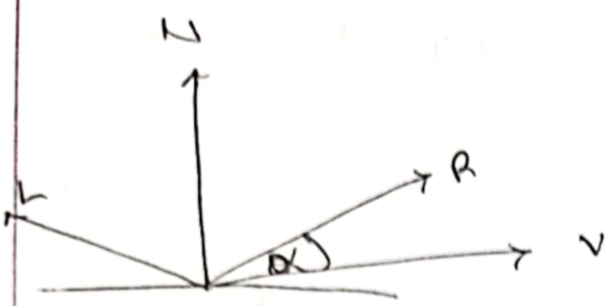
$-1 \le \cos\alpha \le 1$

$n\uparrow$, $(\cos\alpha)^n \downarrow$

$\cos\alpha$ এর value 1 এর ছোট হবে and power বাড়ালে value টা আরো ছোট হয়ে যায়।

shininess এর range ছোট হয়ে যাবে।





$S = I_p K_s (\cos\alpha)^n$

$\downarrow$ updated to compute vectors instead of $\cos\alpha$

$S = I_p K_s (\hat{R}\cdot\hat{V})^n$

$\downarrow$ updated to avoid $\cos\alpha < 0$

$S = I_p K_s \max((\hat{R}\cdot\hat{V})^n, 0)$

* Question এ reflected ray দেওয়া থাকে না,
so, we use a formula:-

$\vec{R} = 2(\vec{L}\cdot\vec{N})\cdot\vec{L} - \vec{L}$

$\vec{L} = L - P$

$\vec{N} = P - C$

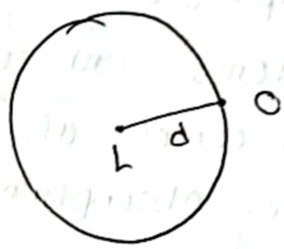
$I = I_a K_a + I_p K_D \max(\hat{L}\cdot\hat{N}, 0) + I_p K_s (\max(\hat{R}\cdot\hat{V})^n, 0)$

Attenuation

→ Loss of light energy over space

→ যত দূরে light source থাকবে from object, তত less visible/shiny হবে object.

$$f_{att} = \frac{1}{d^2}$$

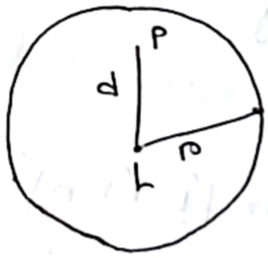


* d অনেক বড় হলে, f_{att} অনেক কমবে মানে, intensity ও তাই কমে যাবে and object less ~~visib~~ clear হয়ে যাবে.

* That's why we don't ~~use~~ use this formula.

→ works for diffuse & specular

$$f_{att} = 1 - \left(\frac{d}{r}\right)^2 \quad , \cancel{d > r}$$



* Adv: d এর value যত ই হোক, there'll be no problem.

if $d > r$, $\frac{d}{r} > 1$

$f_{att} < 0$

↓ updating the f_{att}

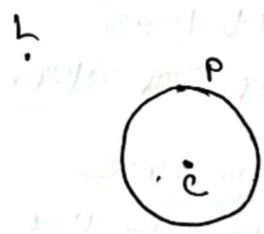
$$f_{att} = \max\left(1 - \left(\frac{d}{r}\right)^2, 0\right)$$

$$I = I_a K_a + f_{att} I_p K_D \max(\hat{L}\cdot\hat{N}, 0) + f_{att}\cdot I_p K_S \max((\hat{R}\cdot\hat{V}, 0)^n)$$

*বেশি light source use করতে চাইলে :—

$$I = I_a K_a + \sum_{i=1}^{m} I_{p_i} f_{att} (K_D \max(L_i \cdot n, 0) + K_S(\max(V\cdot R_i, 0))^n)$$

Let (-70, 500, 420) be the coordinate of the light source of intensity $I_p = 0.80$ unit. The light is illuminating a point on a sphere with coordinates (-25, 100, 75).
Given that the centre of the sphere is at the origin (0, 0, 0) and the absorption coefficient for diffuse reflection is $K_d = 0.80$ unit.
Calculate the intensity of diffuse reflection for the point.



$L = (-70, 500, 420)$

$I_p = 0.80$ unit

$P = (-25, 100, 75)$

$C = (0, 0, 0)$

$K_d = 0.80$

$D = I_p K_D \max(\hat{L}\cdot\hat{N}, 0)$

$= (0.80)(0.80)\max(0.99088, 0)$

$= (0.80)(0.80)(0.99088)$

$D = 0.63416$ unit

$\vec{L} = L - P$

$\vec{L} = (-45, 400, 345)$

$\hat{L} = \dfrac{(-45, 400, 345)}{\sqrt{(-45)^2 + (400)^2 + (345)^2}}$

$\hat{L} = (-0.0848, 0.7545, 0.65)$

$\vec{N} = P - C$

$\vec{N} = (-25, 100, 75)$

$\hat{N} = \dfrac{(-25, 100, 75)}{\sqrt{(-25)^2 + (100)^2 + (75)^2}}$

$\hat{N} = (-0.196, 0.78, 0.588)$

$\hat{L}\cdot\hat{N} = (-0.0848, 0.7545, 0.65)\cdot(-0.196, 0.78, 0.588)$

$\hat{L}\cdot\hat{N} = 0.99088$ unit

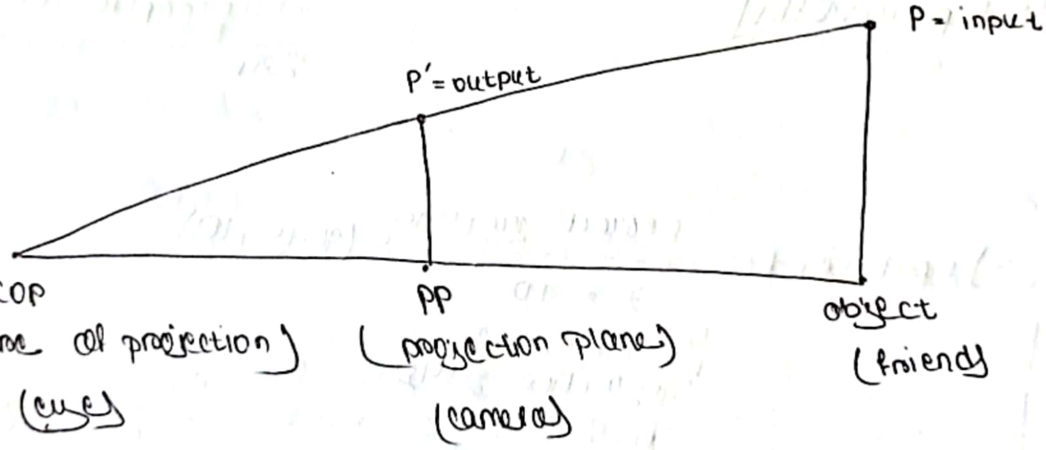
LECTURE 12

Projection

→ process where a 3D model is converted to 2D model for viewing on a plane

3D input → projection → 2D output

For example:
While clicking a picture of a person, the person is the 3D input and the image you are viewing on the mobile screen is the 2D output. The camera is the projection body.



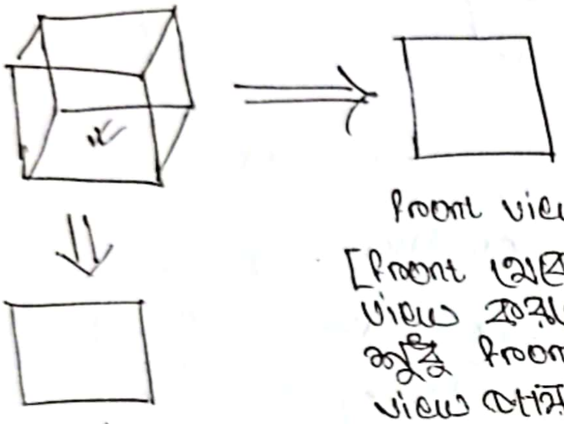
* input থেকে output এর conversion এর জন্য projection matrix এর দ্বারা সাহায্যে

$P' = MP$

M → projection matrix

Projection type based on the Projection plane (PP)

Orthographic Projection ⟶ 3D object এর একটা particular portion view দেখাতে পারি

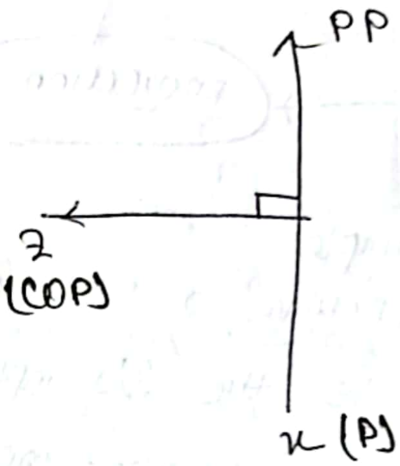


front view

[front থেকে view করলে শুধু front view পাবো]

top view

[bird eye angle] top থেকে দেখলে শুধু top view পাবো]



* projection plane এর উপর view টা যদি 90° হয়, তাহলে Orthographic projection হয়।

(x, y, z) input point $\xrightarrow[z = 10]{\text{project } xy \text{ plane}}$ $(x, y, 10)$

[xy plane এ projection হচ্ছে, so x and y remains unchanged in output]

$\xrightarrow[y = -13]{\text{project } zx \text{ plane}}$ $(x, -13, z)$

[x and z same and $y = -13$ হবে output এ]

$(x, y, z) \rightarrow (x, y, 10)$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 10 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} x \\ y \\ 10 \\ 1 \end{bmatrix}$$

orthographic matrix for the above scenario.

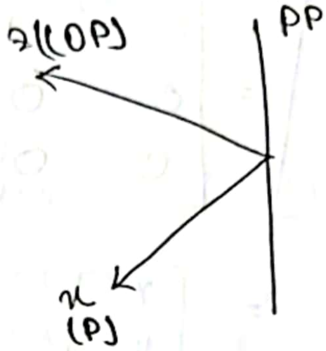
$(x, y, z) \rightarrow (x, -13, z)$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -13 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ -13 \\ z \\ 1 \end{bmatrix}$$

* orthographic matrix fixed থাকে না, scenario to scenario change হয়।

* So, কত distance থেকে, কোন plane এ আছে, এইভাবে দেখে orthographic matrix বানাবো for the given scenario.

Oblique projection → কোনো একটি side এর দেখার সাথে সাথে আরো কিছু part দেখাতে

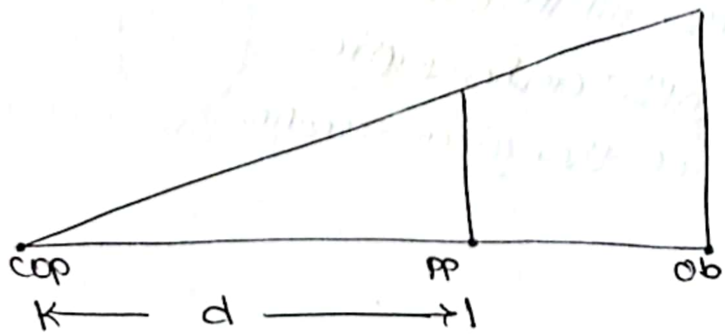


* view and plane projection plane এ 90° তে নাই, that's why একটি specific view এর পাশাপাশি extra জিনিস দেখা যায়।

Based on the distance (d)

Perspective projection → d/ data is limited

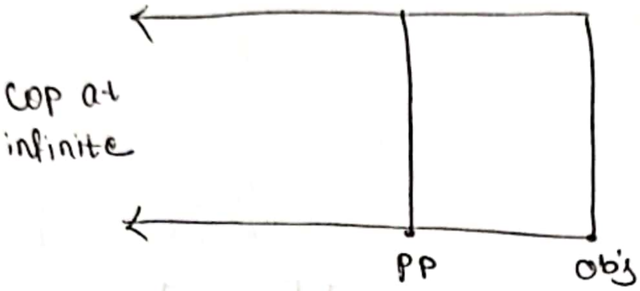
~~[illegible]~~



distance between cop and pp

* Here, d is limited. So, follows under perspective projection.

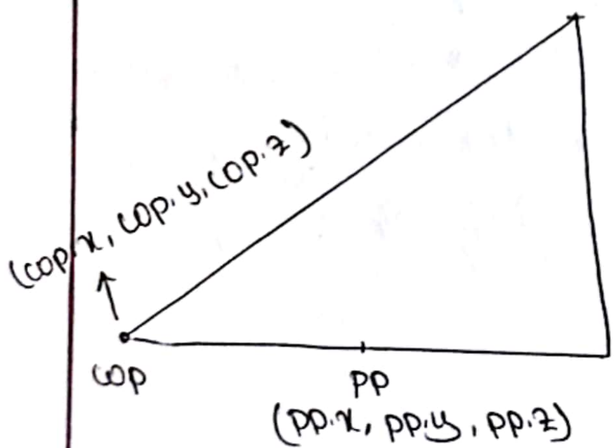
Parallel Projection → d/data is infinite



* projection plane & cop এর distance infinite হলে ধরছি, so এমন scenario parallel projection এর under এ পড়ে

General Projection Matrix → 4 * 4 projection

$$m = \begin{bmatrix} 1 & 0 & -\frac{dx}{dz} & \frac{dx}{dz} * pp.z \\ 0 & 1 & -\frac{dy}{dz} & \frac{dy}{dz} * pp.z \\ 0 & 0 & -\frac{pp.z}{dz} & pp.z\left(1 + \frac{pp.z}{dz}\right) \\ 0 & 0 & -\frac{1}{dz} & 1 + \frac{pp.z}{dz} \end{bmatrix} \quad 4 * 4$$



$dx = cop.x - pp.x$

$dy = cop.y - pp.y$

$dz = cop.z - pp.z$

Given COP (50, 40, 100) and PP (0, 0, -200). find out the projected output for the given point (30, 50, -250)

$dx = 50 - 0 = 50$

$dy = 40 - 0 = 40$

$dz = 100 - (-200) = 300$

$$P' = \begin{bmatrix} 1 & 0 & \frac{-50}{300} & \frac{50}{300}(-200) \\ 0 & 1 & \frac{-40}{300} & \frac{40}{300}(-200) \\ 0 & 0 & \frac{200}{300} & -200\left(1 + \frac{-200}{300}\right) \\ 0 & 0 & \frac{-1}{300} & 1 + \frac{-200}{300} \end{bmatrix} \begin{bmatrix} 30 \\ 50 \\ -250 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 115/3 \\ 170/3 \\ -700/3 \\ 1.167 \end{bmatrix} \xrightarrow{\text{divide by } 1.167} \begin{bmatrix} 32.847 \\ 48.557 \\ -200 \\ 1 \end{bmatrix}$$

to make the dummy value '1', the entire matrix is divided by 1.167.

Scenario 1

Derive a projection matrix for the xy plane, where the COP is at (0,0,0) and PP is at d distance from the COP.

$COP.x = COP.y = COP.z = 0$

$PP.x = 0$

$PP.y = 0$

$PP.z = d$

$dx = 0 - 0 = 0$

$dy = 0 - 0 = 0$

$dz = 0 - d = -d$

$$m = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix}$$

Scenario 7

Derive a projection matrix for the xy plane where the PP is at (0,0,0) and COP is a distance from the PP.

$PP.x = 0$

$PP.y = 0$

$PP.z = 0$

$COP.x = 0$

$COP.y = 0$

$COP.z = d$

$dx = 0$

$dy = 0$

$dz = d - 0 = d$

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -1/d & 1 \end{bmatrix}$$

Suppose a perspective projection where the eye is on origin and the camera is on the xy plane and 5 distance away from the eye in the z axis. for the input point (13, 12, 10). find out the projected output point.

COP = (0, 0, 0) , PP(0, 0, 5)

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/5 & 0 \end{bmatrix} \begin{bmatrix} 13 \\ 12 \\ 10 \\ 1 \end{bmatrix} = \begin{bmatrix} 13 \\ 12 \\ 10 \\ 3 \end{bmatrix} = \underbrace{\begin{bmatrix} 13/3 \\ \cancel{12}\,4 \\ 10/3 \\ 1 \end{bmatrix}}_{\text{output}}$$

① Projection

② Lightning

③ Color model

Complicated question হবে না

④ Transformation → complicated আসবে

$$(3,4) \xrightarrow[(4,2)]{} (7,6)$$

$$\begin{bmatrix} 1 & 0 & 4 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 4 \\ 1 \end{bmatrix} = \begin{bmatrix} 7 \\ 6 \\ 1 \end{bmatrix}$$

* This is normal transformation.

এখন (7,6) থেকে (3,4) যাওয়ার জন্য inverse transformation করা লাগবে বা আগের translation matrix ঘুরায় দিব

$$(7,6) \xrightarrow[(-4,-2)]{} (3,4)$$

$$\begin{bmatrix} 1 & 0 & -4 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 7 \\ 6 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 4 \\ 1 \end{bmatrix}$$

[ ] [ ] [inp]

→ compositive matrix