Lecture - 1

Real life objects (2D/3D) is projected on 2D computer screen and that's what computer graphics does.

Object space : coordinates of object with respect to one. continuous. also called real space.

Multiple polygons are combined together to form complex objects or structures, known as mesh structure. We'll work with small polygons and then combine them together.

Image space : coordinates of the screen. and it is not continuous. pixels are discrete. more resolution, better representation but needs larger space in memory.

dpi = dot per sq. inch

This course is also mostly concerned with matrix multiplications.

Image processing is an extended version of graphics.

Graphics pipelining ⇒ steps related to projection of an object, including transformations, clipping, merging, projection.

Book: Schaum's outline
computer graphics

Library for lab: OpenGL

Lecture - 2

CHAPTER 2
IMAGE REPRESENTATION

Unit of image → pixel

Size of an image $w \times h$

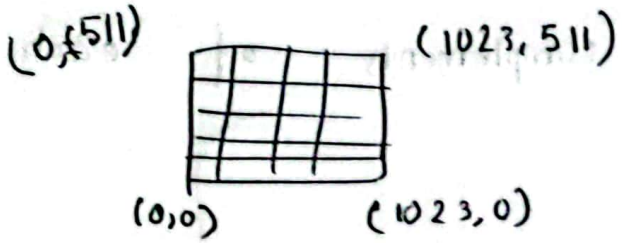
aspect ratio $w : h$

w ↙ h

landscape portrait

more dpi, less resolution of image : the image will appear to look smaller.

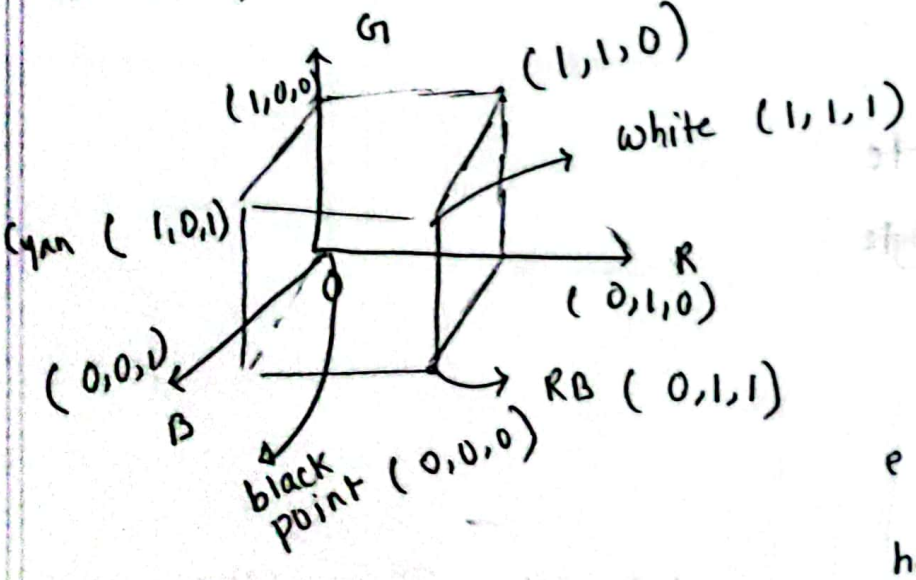
for a screen.



1024 x 512

8 bit — 2^8 values in greyscale, for representing each pixel.

for RGB,



• R ke 1,0,0 dhora ~~ei~~ uchit chilo, oitar respect e baki gulo change hobe.

• CMY model, RGB as diagonal points

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} + \begin{pmatrix} C \\ M \\ Y \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

RGB and CMY are complements of each other.

To color a pixel,

$$\frac{256}{R} \times \frac{256}{G} \times \frac{256}{B}$$

2^{24} ~ can represent ≈ 16.7 million colors using 3 byte.

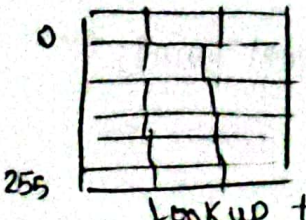
$300 \times 100 \times 3$ byte

$\approx 9 \times 10^4$ byte

$\approx 10^5$ byte

≈ 100 kB

Lookup table is constructed using 256 colors only –



$300 \times 100 \times 1$ byte

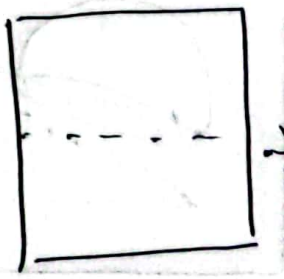
3×10^4 byte → to store lookup table

30KB + 3byte x 256

The effect can be realized as size of the image increases.

Lecture - 3

contents of an image file :

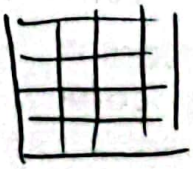


Header → any encoding scheme, total size of file

body ↳ if lookup table is used
↳ pixel content.

To change a format of an image file, you need to change the header.

Screen :

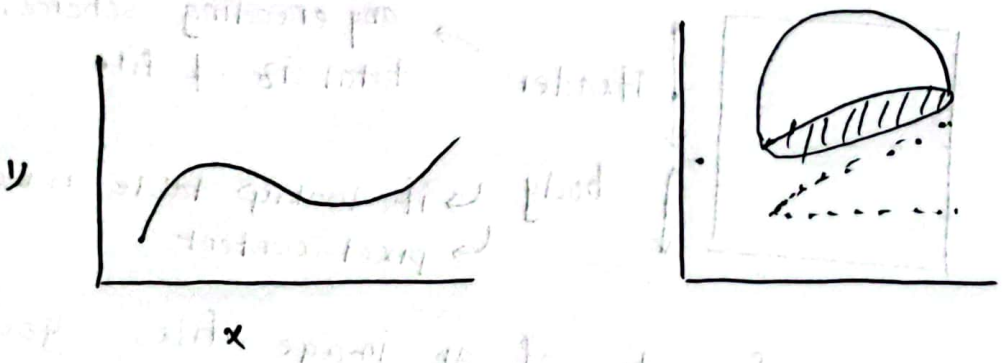


electron gun

CHAPTER 3

SCAN CONVERSION

Projecting a real continuous object to the screen is basically known as scan conversion.

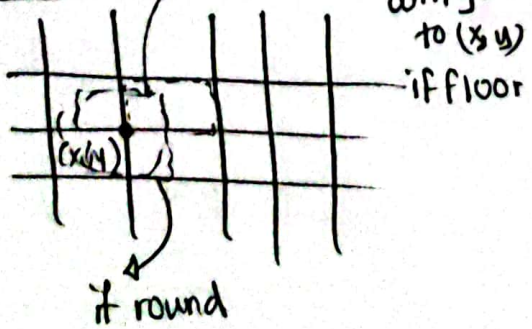


Sampling : which points from which surface

Quantization : Range of colors we'll choose to represent the points. convert a color to its nearest ~~range~~ .color.

Level of quantization ↑ higher quality
higher memory

Point conversion: all points inside will go to (x,y) if floor

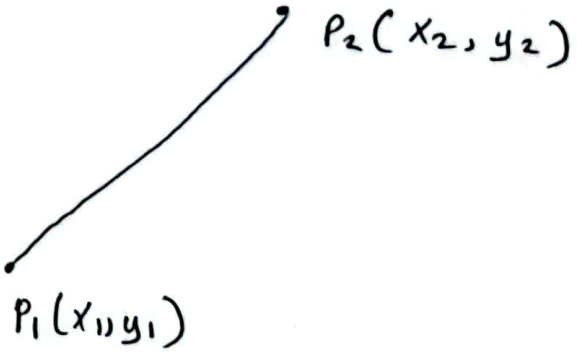


$x, y \in \mathbb{R}^+$

Floor (x) $F(x < x+1) = x$

Line conversion:
For a line, there are 3 algorithms.

Direct equation:



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

$$y = mx + c$$

If slope = 1 or less than one, increase towards x.

x ke increment kore y er value ber korbo.

$$y_i^\circ = m x_i + c$$

$$x_i^\circ = x_{i-1} + 1 \quad \text{till } x_i \neq x_2$$

computational complexity depends on calculation of the slope and then multiplying afterwards.

Lecture - 4

DDA :

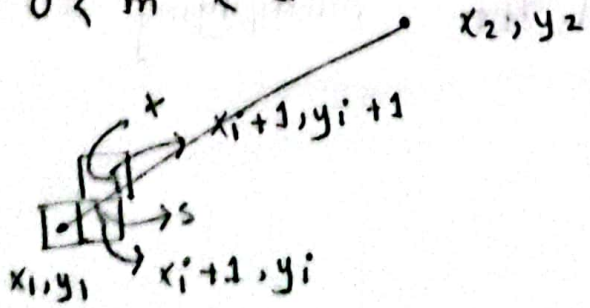
$$m = \frac{\Delta y}{\Delta x}$$

$x_1 + 1 \quad y_1 + m$

less complexity due to addition, but it's floating point addition, needs rounding up as well (since pixel fraction e hoy na). so error is a bit higher.

Bresenhem algorithm :

$0 < m < 1$



$s = y - y_i$

$t = y_{i+1} - y$

$s - t = y - y_i - y_i - 1 + y = 2y - 2y_i - 1$

$$= 2(mx_i + b) - 2y_i - 1$$

$$= 2\left(\frac{\Delta y}{\Delta x} x_i\right) + 2b - 2y_i - 1$$

$$\Delta x (d_i) = 2\Delta y x_i + 2b\Delta x - 2y_i\Delta x - \Delta x$$

$$\Rightarrow d_i = 2\Delta y x_i + \overbrace{2b\Delta x - \Delta x}^{c} - 2y_i\Delta x$$

$$d_{i+1} = d_i + 2\Delta y - 2\Delta x (y_{i+1} - y_i)$$

2 cases :

$$d_{i+1} = \begin{cases} d_i + 2(\Delta y - \Delta x) & \text{if } d_i \geq 0 \\ d_i + 2\Delta y & \text{if } d_i < 0 \end{cases}$$

$$d_1 = \Delta x [2m - 1]$$

For cases other than $0 < m < 1$,

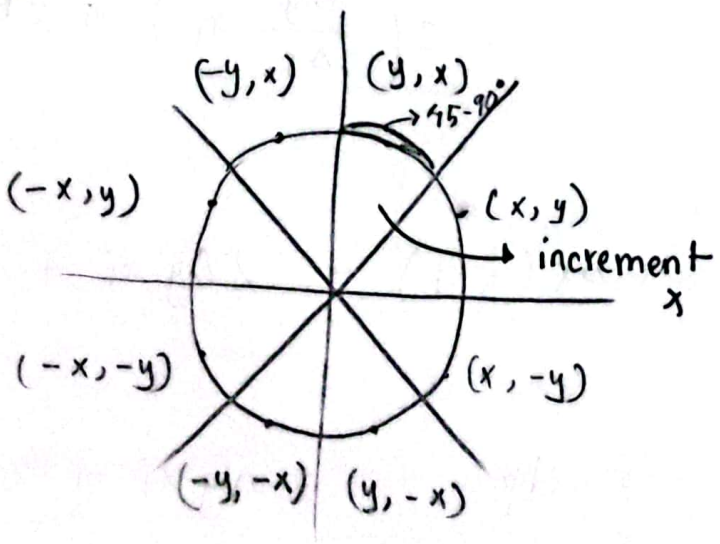
- find formula for the algorithm
- take reflection

Circle drawing :

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

Cartesian:

more overhead



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

plot one segment and then we can plot the rest by taking reflection

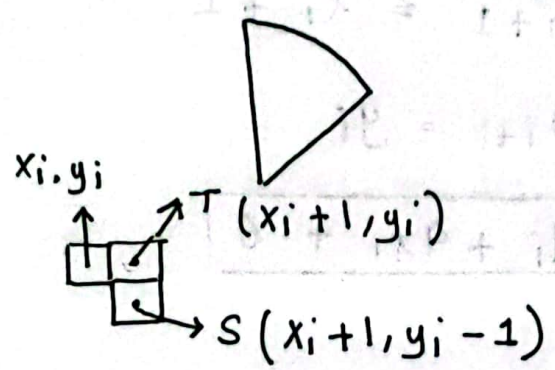
Polar :

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

} much more overhead !

Lecture - 5

Bresenham → tries to minimize error



$$D(T) = (x_i+1)^2 + y_i^2 - r^2$$

$$D(S) = (x_i+1)^2 + (y_i-1)^2 - r^2$$

T, S will always be non-negative and non-positive respectively.

$$d_i = D(T) + D(S)$$

$$= 2(x_i+1)^2 + y_i^2 + (y_i-1)^2 - 2r^2 \quad -(1)$$

If $d_i < 0$,

$$|D(T)| < |D(S)| \longrightarrow T \checkmark$$

$d_i \geq 0$

$$|D(T)| > |D(S)| \longrightarrow S \checkmark$$

$$d_{i+1} = 2(x_{i+1}+1)^2 + y_{i+1}^2 + (y_{i+1}-1)^2 - 2r^2 \quad -(2)$$

(2) − (1) ⇒

$$d_{i+1} = d_i + 4x_i + 2(y_{i+1}^2 - y_i^2) - 2(y_{i+1} - y_i) + 6$$

$d_i < 0$; $x_{i+1} = x_i + 1$

$y_{i+1} = y_i$

$$\boxed{d_{i+1} = d_i + 4x_i + 6}$$

$d_i >= 0$; $x_{i+1} = x_i + 1$

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

$$\boxed{d_{i+1} = d_i + 4(x_i - y_i) + 10}$$

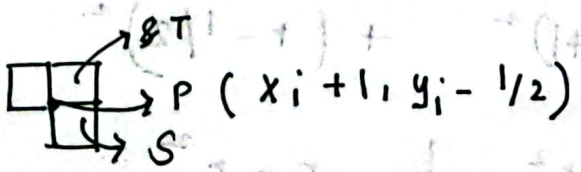
Base case :

$$d_1 = 2(0+1)^2 + r^2 + (r-1)^2 - 2r^2$$

$$= 3 - 2r$$

till $x_1 = r/\sqrt{2}$ [* $r\cos 45$]

Midpoint algorithm :



$$D(P) = (x_i + 1)^2 + (y_i - 1/2)^2 - r^2$$

$D(p) < 0 \rightarrow T$ ✓

$D(p) > 0 \rightarrow S$ ✓

$$P_i = (x_i + 1)^2 + (y_i - 1/2)^2 - r^2 \quad \text{— (1)}$$

$$P_{i+1} = (x_{i+1} + 1)^2 + (y_{i+1} - 1/2)^2 - r^2 \quad \text{— (2)}$$

(2) - (1) $\Rightarrow$

$$P_{i+1} = P_i + 2(x_i + 1) + (y_{i+1}^2 - y_i^2) - (y_{i+1} - y_i)$$

$P_i < 0$; $P_{i+1} = P_i + 2x_i + 2$

$P_i > 0$; $P_i + 2(x_{i+1} - y_{i+1}) + 1$

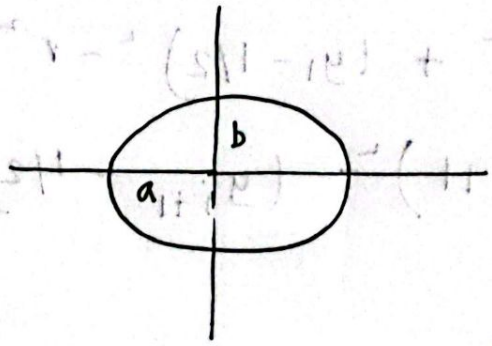
Base case :

$$P_1 = (0+1)^2 + (r - 1/2)^2 - r^2$$

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

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

Lecture - 6



Scan converting an ellipse :

- cartesian

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1 \qquad [\text{center at } 0]$$

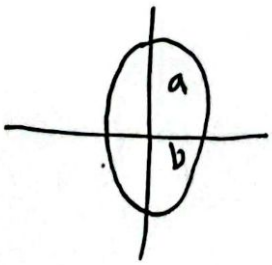
$$y = \frac{\sqrt{a^2 b^2 - b^2 x^2}}{a}$$

- Polar trig. equation :

$$x = a\cos\theta$$
$$y = b\sin\theta$$

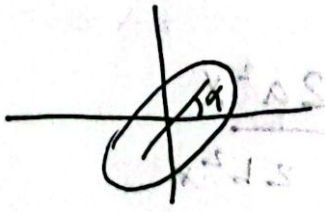
θ 0 to 90°

In case of a rotation to 90° :



$$x = b\cos\theta$$
$$y = a\sin\theta$$

rotation to α :

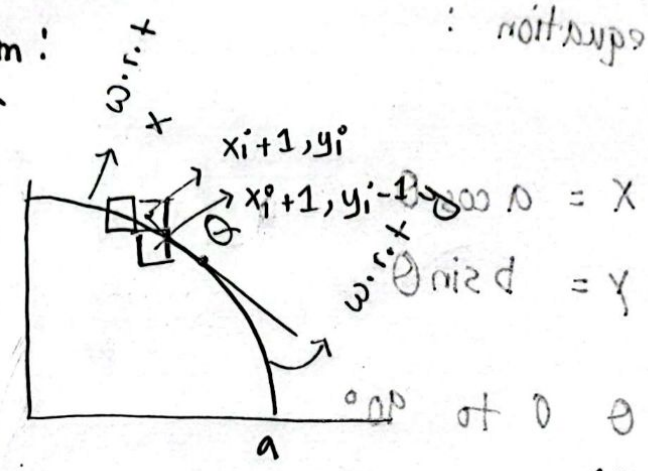


$$x = a\cos\theta - b\sin(\theta + \alpha) \rightarrow +h$$
$$y = b\sin\theta + a\cos(\theta + \alpha) \rightarrow +k$$

if not at center

at some point, the slope will be negative

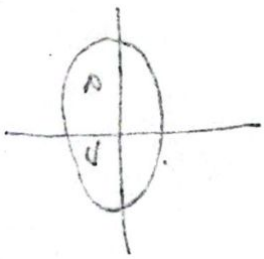
Midpoint algorithm:



$f(x,y) = b^2x^2 + a^2y^2 - a^2b^2$ $\begin{cases} <0 & \text{inside} \\ =0 & \text{on the line} \\ >0 & \text{outside} \end{cases}$

$f'(x) = 2b^2x$

$f'(y) = 2a^2y$



$$\frac{dy}{dx} = -\frac{f_y}{f_x}$$

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

Upper segment:

$P_i = f(x_i, y_i - 1/2) = b^2x_i^2 + a^2(y_i - 1/2)^2 - a^2b^2$

$P_{i+1} = b^2x_{i+1}^2 + a^2(y_{i+1} - 1/2)^2 - a^2b^2$

$$P_{i+1} - P_i = b^2 \left[(x_{i+1} + 1)^2 - x_{i+1}^2 \right] + a^2 \left[(y_{i+1} - 1/2)^2 - (y_i - 1/2)^2 \right]$$

$P_i < 0$:

$$P_{i+1} = P_i + 2b^2 x_{i+1} + b^2$$

$P_i > / 0$:

$$P_{i+1} = P_i + 2b^2 x_{i+1} + b^2 - 2a^2 y_{i+1}$$

Base case:

for $(0, b)$ ~~ba~~ point

lower segment:

$f(x_i + 1/2, y_i - 1)$

$$p_i = f(x_{i+1/2}, y_i - 1) = b^2 (x_{i+1/2})^2$$

$$+ a^2 (y_i - 1)^2 - a^2 b^2$$

$$p_{i+1} = b^2 (x_{i+1} + 1/2)^2 + a^2 (y_{i+1} - 1)^2$$

$$- a^2 b^2$$

$$p_{i+1} = p_i + b^2 \left[(x_{i+1} + 1/2)^2 - (x_i + 1/2)^2 \right]$$

$$- 2a^2 y_{i+1} + a^2$$

base case: point Q

$p_i < 0$:

$$p_{i+1} = p_i + 2b^2 x_{i+1} - 2a^2 y_{i+1} + a^2$$

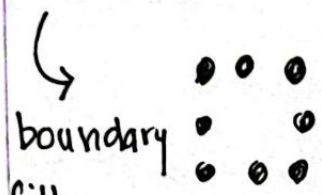
$p_i \geq 0$:

$$p_{i+1} = p_i + 2a^2 y_{i+1} + a^2$$

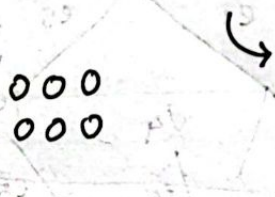
Lecture - 7

Region filling

Boundary defined region vs interior defined region

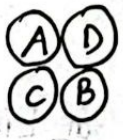


boundary fill algo



flood fill algo

4-connected vs 8-connected



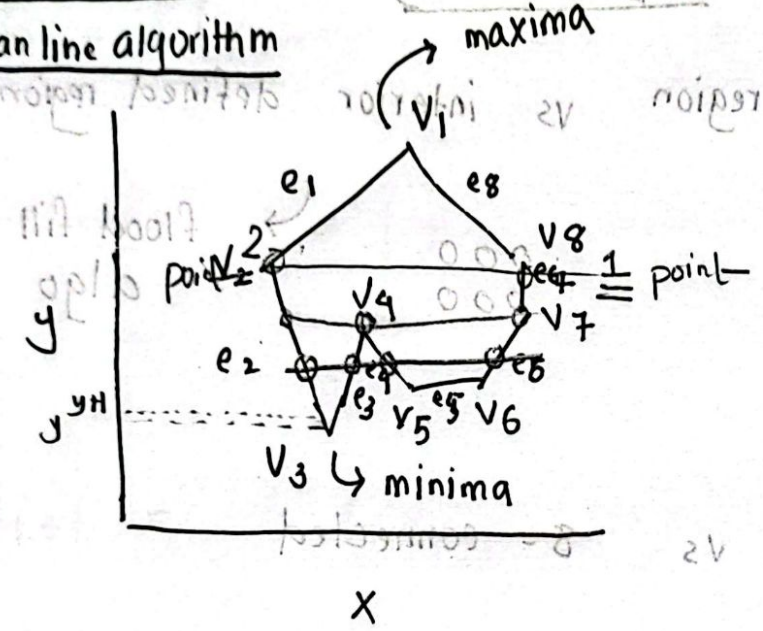
seed → an initial point

calls neighbours like bfs & keeps checking for boundary ; for boundary fill algorithm

keep coloring as long as points exist ; flood fill algorithm

Lecture - 8

Scan line algorithm



y's minima to maxima → scan line

sort according to value of x, gL Line on pairs

- you can ignore horizontal line segments if you want
- local minima / maxima : two points for two edges treat as normal
- monotonously increasing / decreasing

[y bartese / komtese]

for such points, take 1 point instead of 2.

$y_{min} \leq y \leq y_{max}$

edge list

edge	y_{min}	y_{max}	$x\,[y - y_{min}]$	$1/m$
e_1	y_{min_1}	y_{max_1}	$x_1\,[y_{min}]$	$1/m_1$
e_2	⋮	⋮		
e_3	⋮	⋮		
e_4				
e_5				
e_6				
e_7				
e_8				

for monotonously incr/decr points, we can take $y_{max} - 1$ or $y_{min} + 1$

Aliasing effects problem : error increases while transitioning

anti-aliasing methods

- filtering
  - low pass filter
  - super sampling more
- pre post

- staircase effect
- uneven brightness 1 → $\sqrt{2}$ [distance, feels like it has less light]
- picket fencing problem

Lecture-9

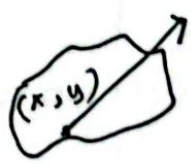
CHAPTER 4
2D TRANSFORMATION

transformation : all kinds of operation on objects

transformation
- geometric transformation
- coordinate transformation

Transformation
① translation:

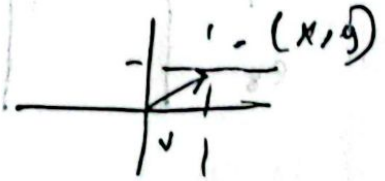
geo



$\hat{v} = h\hat{i} + k\hat{j}$

new coordinates
$x+h, y+k$

coordinate



$\overline{Tv}$ $\quad v = h\hat{i} + k\hat{j}$

$$\overline{Tv} = \begin{pmatrix} 1 & 0 & -h \\ 0 & 1 & -k \\ 0 & 0 & 1 \end{pmatrix}$$

representation
- points as colm vector/ matrix
  - ↳ standard approach
- points as row matrix/ vector

$$\begin{cases} x' = x+h \\ y' = y+k \end{cases}$$

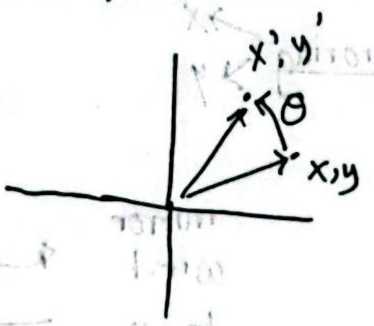
$$\begin{pmatrix} 1 & 0 & h \\ 0 & 1 & k \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix}$$

$$\underset{2\times2}{(\ )}\ \underset{2\times1}{(\ )} = \underset{2\times1}{(\ )}$$

another (transformed) colm matrix

geometric

② ↳ rotation:



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

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

↳ extend row and colm to make them a 3x3 matrix

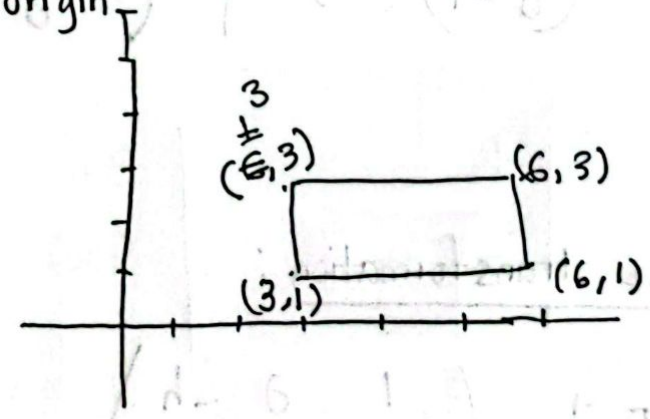
$$\begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

(Teo)

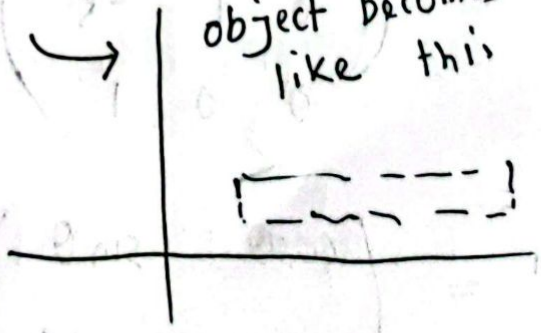
③ ↳ scaling:

along x and y axis, with respect to the origin

$$x' = S_x \cdot x$$
$$y' = S_y \cdot y$$

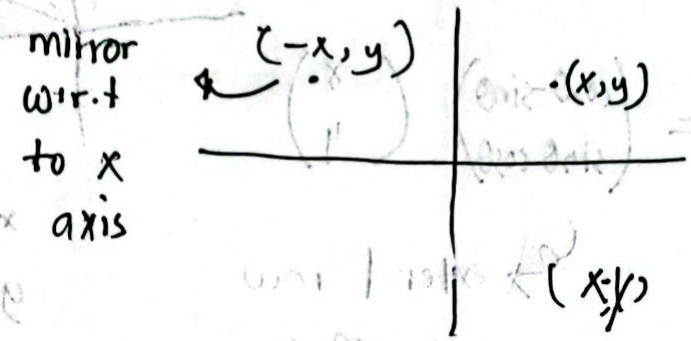


$S_{2,\frac{1}{2}}$ ↳ object becomes like this



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

④ Mirroring → x, ↓ y



y axis	x-axis
$x' = x$	$x' = -x$
$y' = -y$	$y' = y$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} \quad \Big| \quad \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}$$

Reverse transformation:

① $T_v^{-1} = \begin{pmatrix} 1 & 0 & -h \\ 0 & 1 & -k \\ 0 & 0 & 1 \end{pmatrix}$ for translation

② $R_\theta^{-1} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$

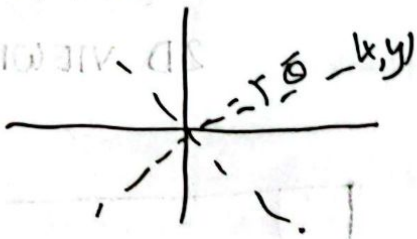
↳ make it clockwise ↘

③ $S_{x,y}' = \begin{pmatrix} \frac{1}{Sx} & 0 \\ 0 & \frac{1}{Sy} \end{pmatrix}$

④ $Mx' = Mx$

$My' = My$

Coordinate:



② $\bar{R\theta} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$ $\quad \bar{T_V}^{-1} = T_V$

$R\theta^{-1} = R\theta$

③ Scaling: change axis [ie units]

$\bar{S}_{x,y}^{-1} = S_{x,y}$ -

$\bar{S}_{x,y} = \begin{pmatrix} \frac{1}{Sx} & 0 \\ 0 & \frac{1}{Sy} \end{pmatrix}$ Rθ

④ Mirroring: change the markings

$\bar{M}_x = Mx$

$\bar{M}_y = My$

$\bar{M}_x^{-1} = Mx$

$My^{-1} = My$

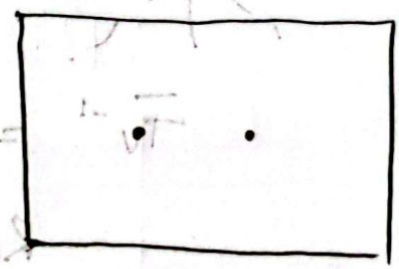
Scaling an obj for proper magnification

T_v $S_{x,y}$ T_v^{-1} ( )

Lecture - 10

CHAPTER 5

2D VIEWING AND CLIPPING

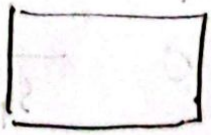


2D projection

World coordinate system

project or view

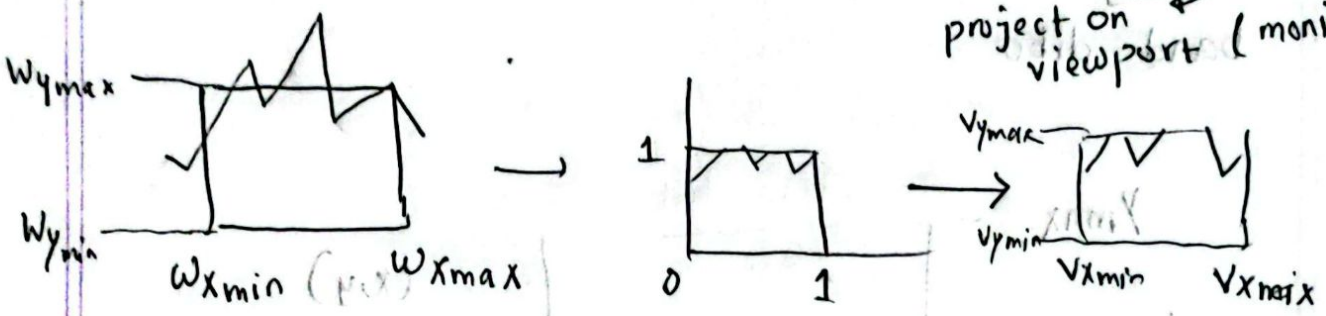
window :
real obj theke jototuku dekhate chacchi



viewport

Select window → bring under a normalized window

project on viewport (monitor)



Task:

Relate W_x, W_y with V_x, V_y

$$\left.\frac{W_x - W_{x\min}}{W_{x\max} - W_{x\min}}\right\} = \frac{V_x - V_{x\min}}{V_{x\max} - V_{x\min}}$$ [scaling factor will be the same]

$$V_x = (W_x - W_{x\min}) \times \frac{V_{x\max} - V_{x\min}}{W_{x\max} - W_{x\min}} + V_{x\min}$$

Similarly,

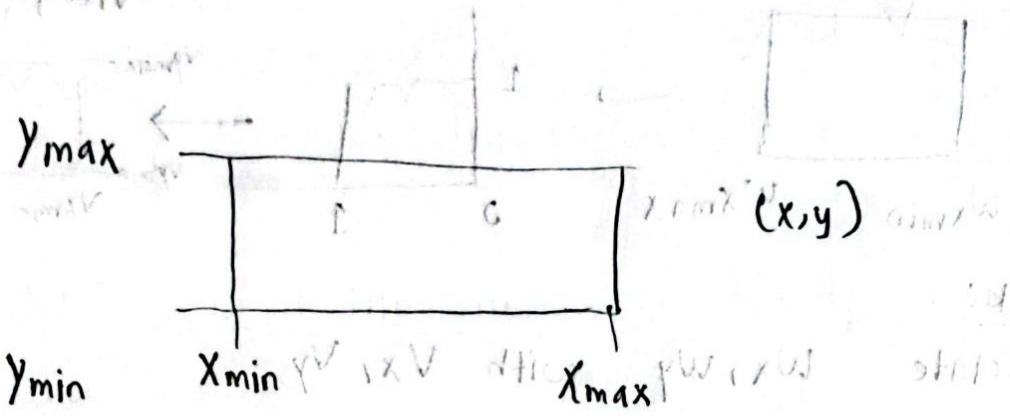
$$\frac{W_y - W_{y\min}}{W_{y\max} - W_{y\min}} = \frac{V_y - V_{y\min}}{V_{y\max} - V_{y\min}}$$

Transformation:

$$\begin{pmatrix} V_x \\ V_y \\ 1 \end{pmatrix} = N \begin{pmatrix} W_x \\ W_y \\ 1 \end{pmatrix}$$

$$N = \begin{pmatrix} 1 & 0 & V_{x\min} \\ 0 & 1 & V_{y\min} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{V_{x\max} - V_{x\min}}{W_{x\max} - W_{x\min}} & 0 & 0 \\ 0 & \frac{V_{y\max} - V_{y\min}}{W_{y\max} - W_{y\min}} & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & -W_{x\min} \\ 0 & 1 & -W_{y\min} \\ 0 & 0 & 1 \end{pmatrix}$$

Clipping : object er kototuku dekhabo ar kototuku baad dibo

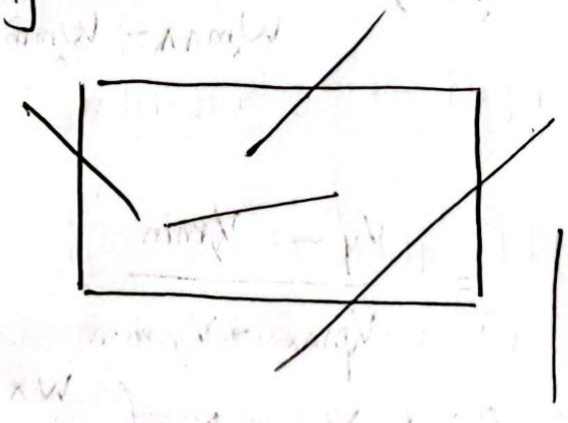


$x_{min} \leq x \leq x_{max}$

$y_{min} \leq y \leq y_{max}$

} point clipping

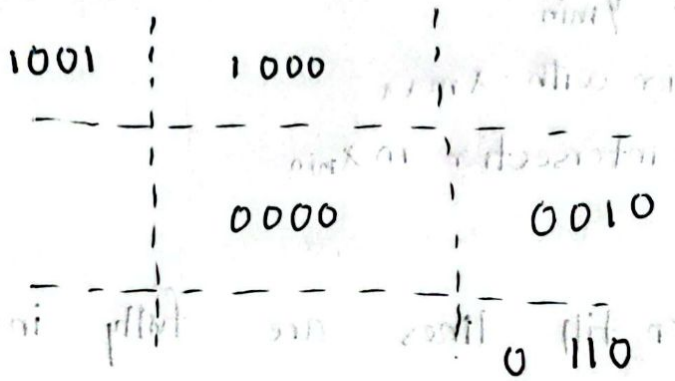
Line clipping :



Categories :

1. fully visible : puratai dekhay dibo
2. fully invisible : cont see in window
3. partially visible ← clipping candidate

Cohen - Sutherland algorithm :



Region is represented using 4 bit code

1st bit
window er upore : 1
~~niche~~ bhetor : 0

2nd bit :
window er niche : 1

3rd bit : → right

4th bit : ← left

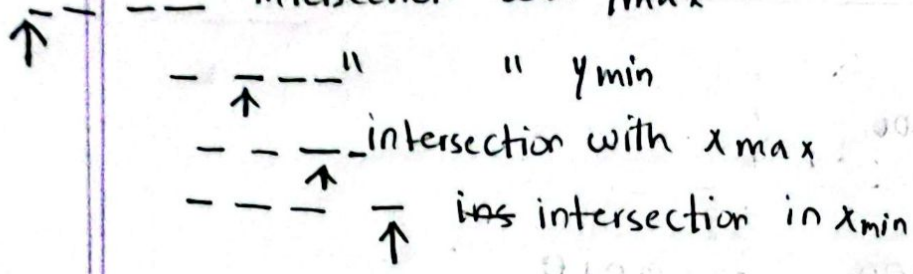
Check endpoints of line segment → which region

fully visible : both endpoints are 0000

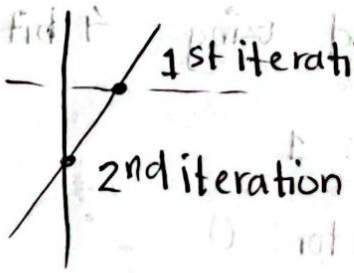
fully invisible : AND of end points region code is not 0

partially visible : both endpoints are not 0000, but their AND is 0.

↙ know that theyre clipping candidates



iteration goes on till lines are fully invisible or visible



Lecture - 11

Midpoint subdivision ~~substitution~~ → larger complexity

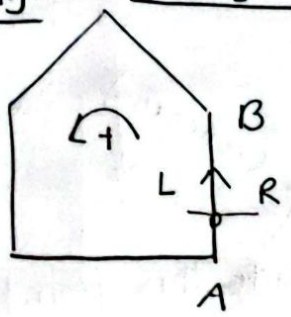
- 2 lines – take midpoint,
  check if within window,
  if fully invisible, discard

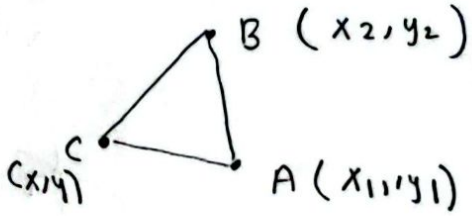
and then continue like b-search

Lecture - 12

• Convex polygon • Concave polygon

Sutherland - Hodgeman algorithm :





$P_1, P_2 \ldots P_N$ → concave / convex polygon

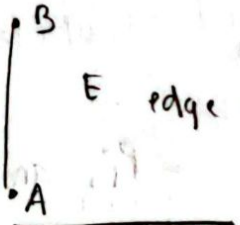
clipping window → convex polygon

~~Show~~ shoelace formula

$$(x_2 - x_1)(y - y_1) - (y_2 - y_1)(x - x_1)$$

+ve : L of AB line

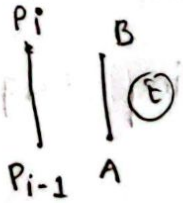
-ve : R of AB line



edge of clipping window

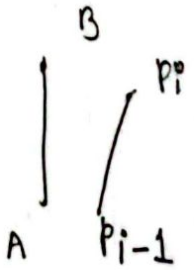
an edge P_{i-1} P_i

Case 1 :



include P_i to output point list

Case 2 :



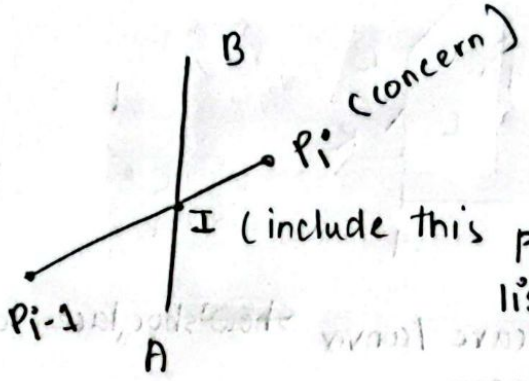
do nothing

Case 3:

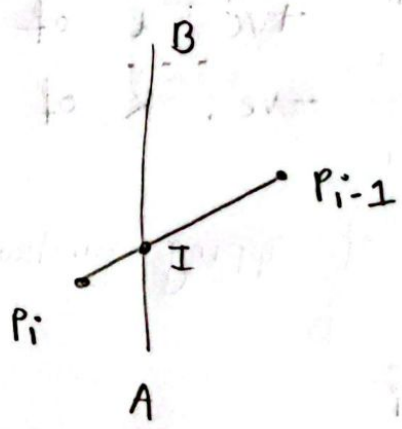
inside to outside

Pi excluded

I included



Case 4:



outside to inside

Pi, and I included to output point list

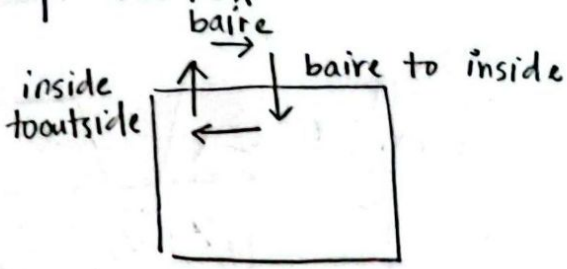
Do the same for all existing edges

Weiler - Atherton algorithm :

polygon to be clipped → subject polygon

clipping window → clip polygon

start with an arbitrary vertex



Cases

① outside to inside :

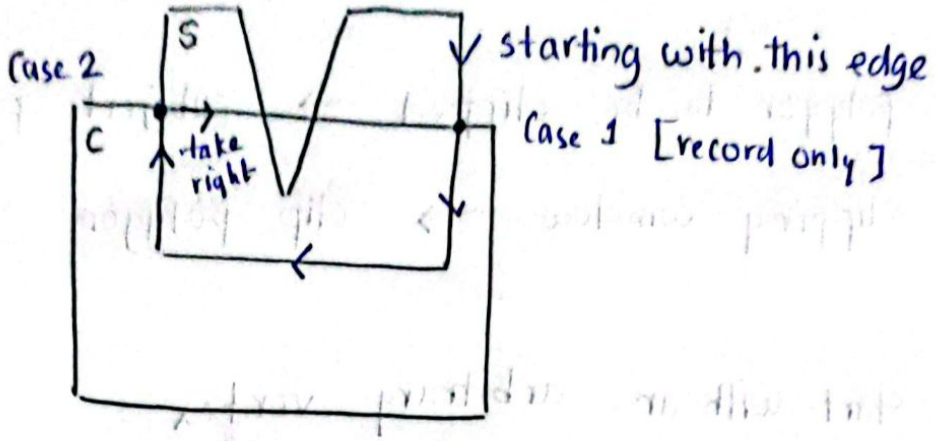
record the intersecting point

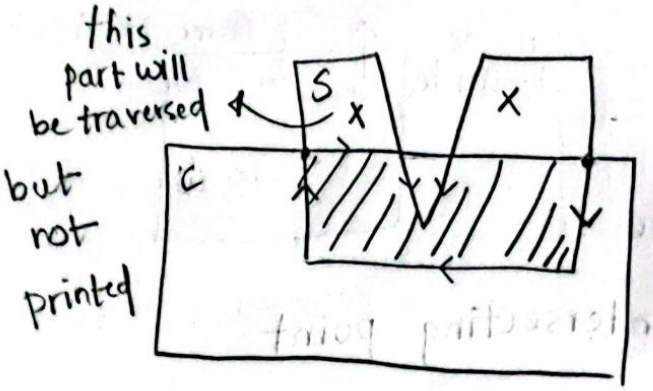
② inside to outside :

- take a right turn
- swap clipping polygon and subject polygon

at one point, we might reach an intersecting point that was previously recorded [like a circuit being completed] - then we print the polygon.

Example :



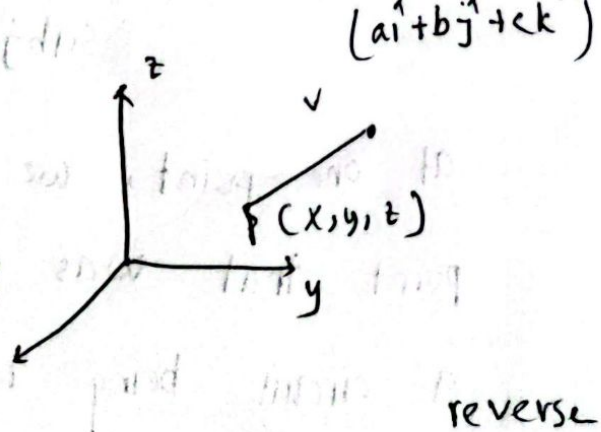


Lecture - 13 CHAPTER 6

3D Trans formation

- Translation

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ a & b & c & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x+a \\ y+b \\ z+c \\ 1 \end{pmatrix}$$



$$x' = x+a \qquad y' = y+b \qquad z' = z+c$$

reverse

$$x' = x-a \qquad y' = y-b \qquad z = z-c$$

- **Scaling**

$$x' = S_x \cdot x$$
$$y' = S_y \cdot y$$
$$z' = S_z \cdot z$$

$$\begin{pmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = S_{x,y,z}$$

- **Rotation**

Rotation may be with respect to 3 axes

x-axis : x ke fix rekhe rotation hobe, yz plane e

z axis

$R_{\theta,k} =$

$$\begin{cases} x' = x\cos\theta - y\sin\theta \\ y' = x\sin\theta + y\cos\theta \\ z' = z \end{cases}$$

y axis

$R_{\theta,j} =$

$$\begin{cases} x' = x\cos\theta + z\sin\theta \\ y' = y \\ z' = -x\sin\theta + z\cos\theta \end{cases}$$

→ x-axis

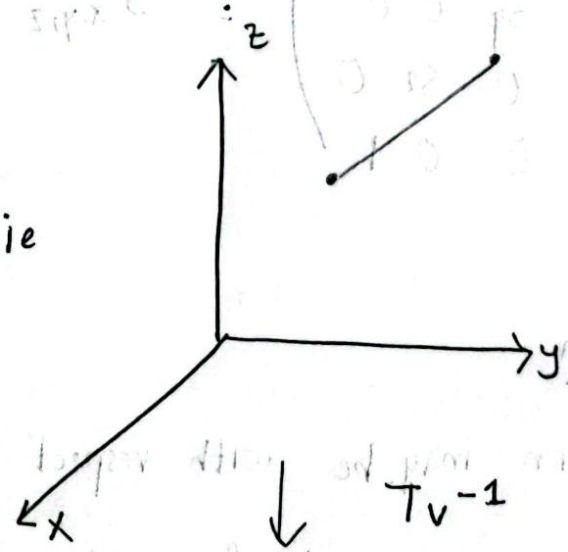
$R_{\theta, I}$

coordinate transformation = reverse transformation

complex transformation

① dhore nie axis e niye ashte hobe ie

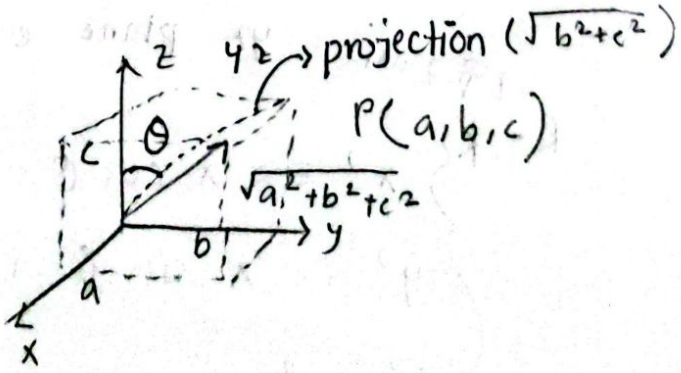
reverse translation



T_v^{-1}

②

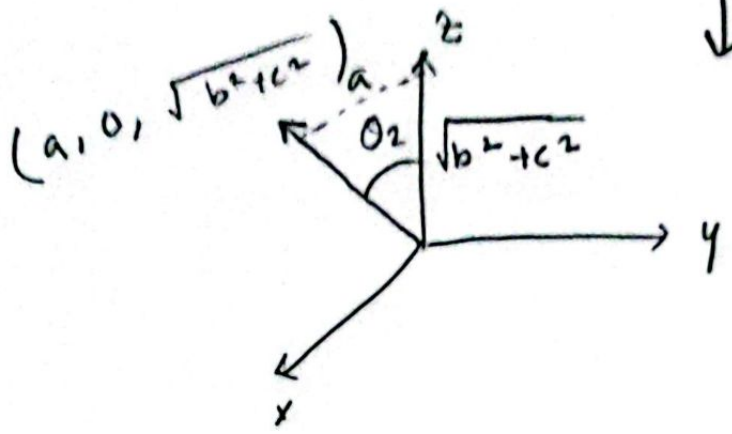
find $\sin\theta$, $\cos\theta$



$$\sin\theta_1 = \frac{b}{\sqrt{b^2+c^2}}$$

$$\cos\theta_1 = \frac{c}{\sqrt{b^2+c^2}}$$

θ_1 → x axis er respect e rotation



$\downarrow R_{\theta,I}$

(3) need to rotate with respect to y.

$\downarrow R_{\theta,J}$

$$\sin\theta_2 = \frac{a}{\sqrt{a^2+b^2+c^2}}$$

$$\cos\theta_2 = \frac{\sqrt{b^2+c^2}}{\sqrt{a^2+b^2+c^2}}$$

(main rotation)

$\downarrow R_{\theta,v}$

$\downarrow$

$R_{\theta,J}^{-1}$

$\downarrow$

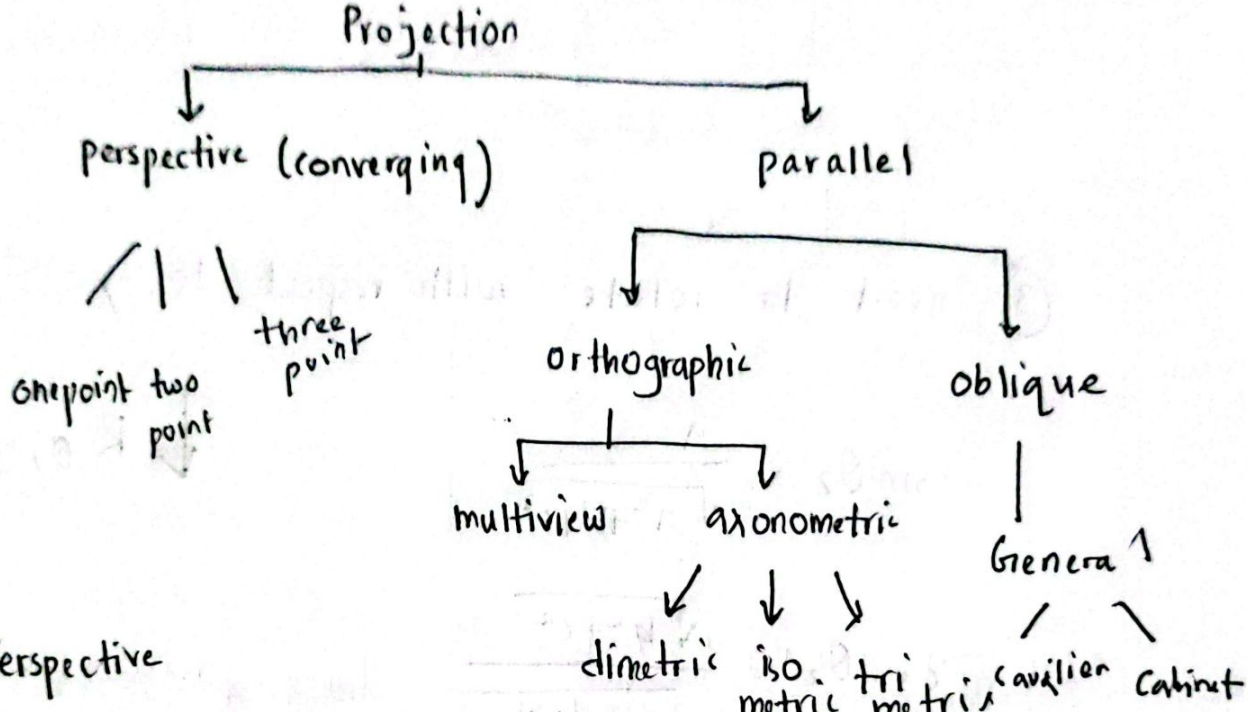
$R_{\theta,I}^{-1}$

$\downarrow$

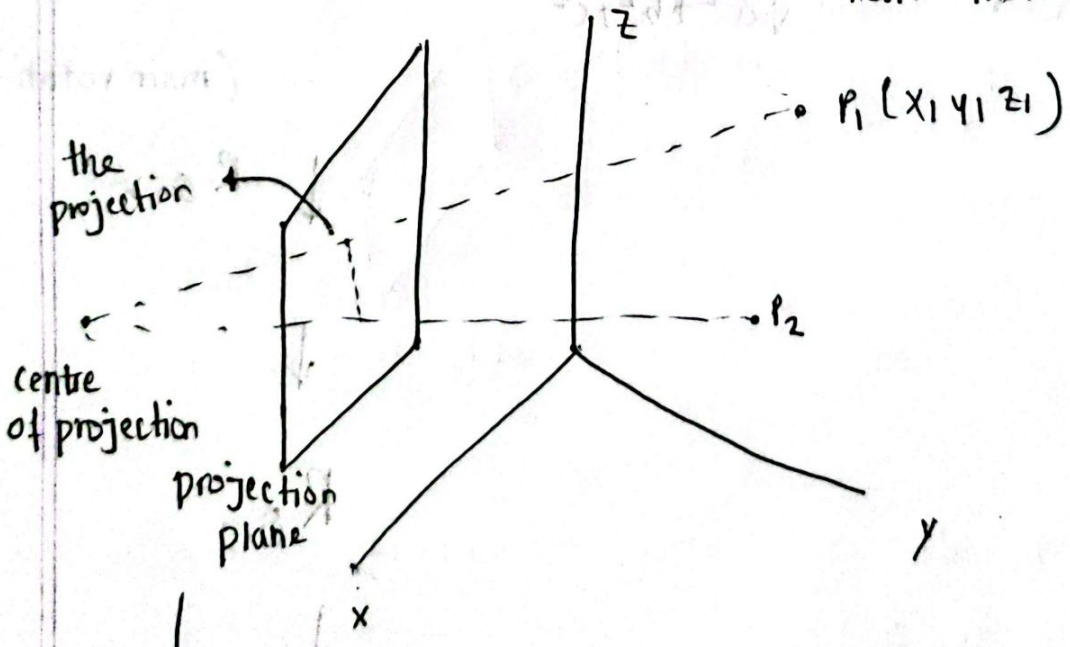
T_v^{-1}

Lecture -14

CHAPTER 7
MATHEMATICS OF PROJECTION



Perspective



→ plane joto dure shorabo projection choto hote jabe → known as perspective foreshortening

at one point, its called vanishing point

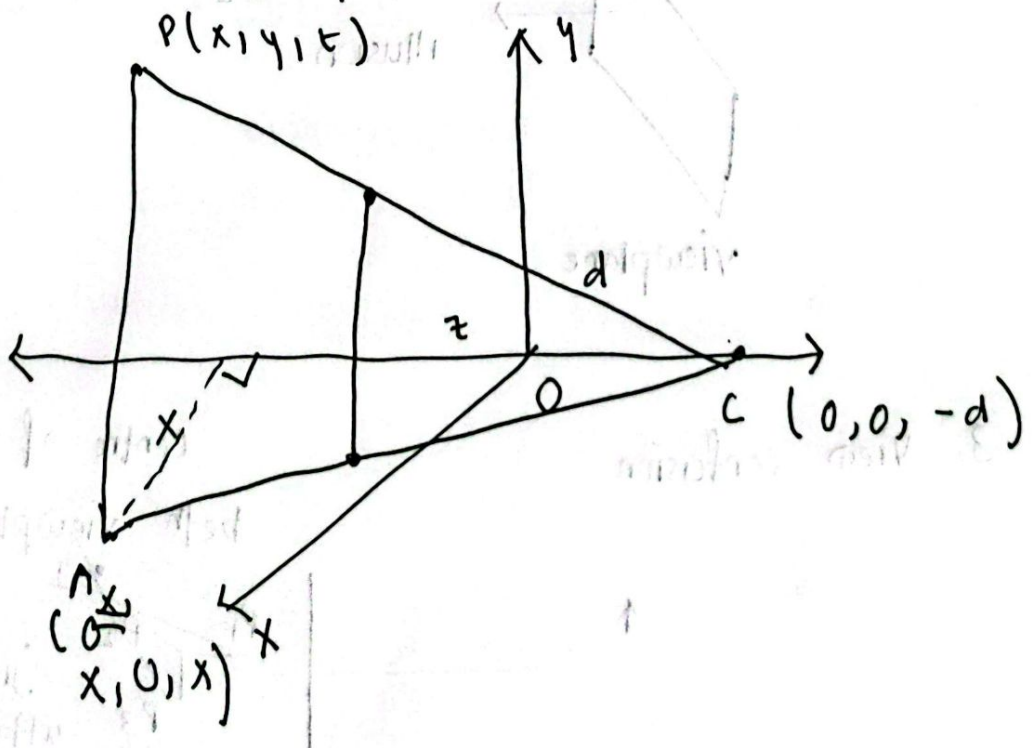
Ex: 2

$$\frac{d}{z+d} = \frac{x'}{x} = \frac{y'}{y}$$

$$x' = \frac{d-x}{z+d}$$ [viewplane in xy, projection from opposite]

$$y' = \frac{d-y}{z+d}$$

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} dx \\ dy \\ 0 \\ z+d \end{pmatrix} = \begin{pmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & d \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

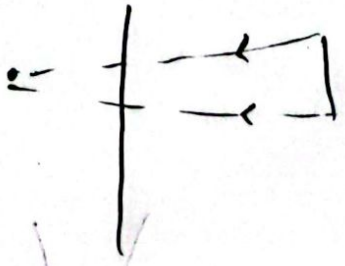


Lecture - 15

Anomalies of perspective projection :

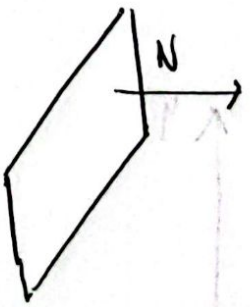
1. perspective foreshortening

↳ kono obj ke dure shorale image choto hoye jay



2. vanishing point

normal er parallel line gula projection hole, mone hobe, at one point line gula mile jacche → illusion

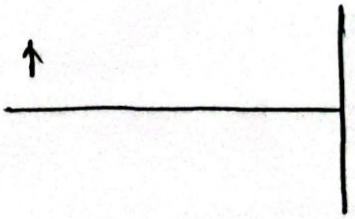


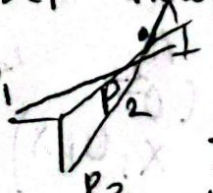
viewplane

3. view confusion

centre of projection

beth viewplane and obj



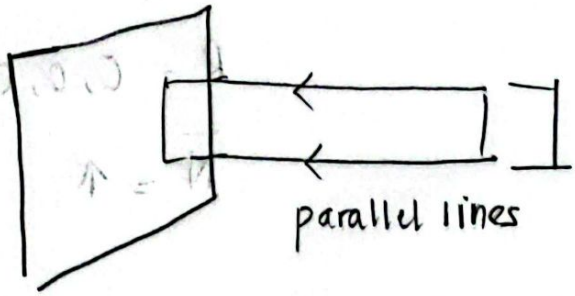


jinishta ulte jache

4. Topological distortion :

The points move further away from each other and the obj looks infinitely large

Parallel projection :



if projection & direction of viewplane are

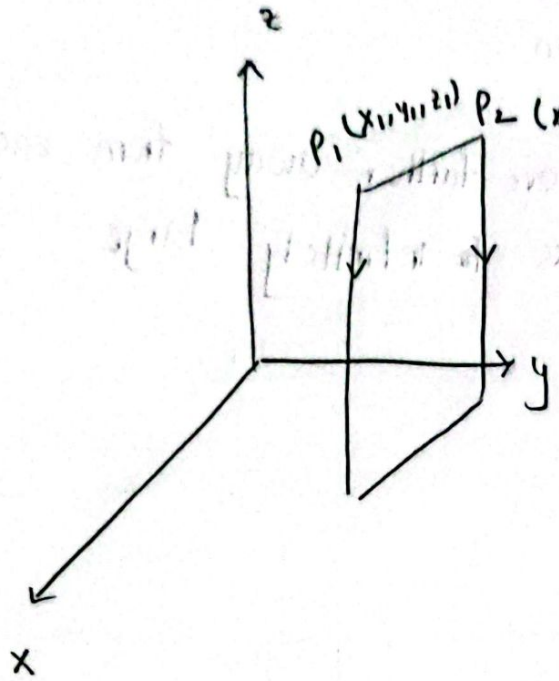
parallel → orthographic

otherwise → oblique (bent)

orthographic

↙ ↓ ↘

isometric dimetric trimetric



$x' = x$

$y' = y$

$z' = 0$

$R = 0, 0, 0$ ref point

$\bar{N} = \uparrow$

- onno kothao plane dekha thakle — calculations