Siddharth Sir:

Date: 13 May, 2018
Sunday

Lecture-1

① Clipping (for primitive shapes)

↳ any complex shapes can be represented using primitive shapes.

② Visual Surface Determination (Hidden surfaces/shapes removal)

③ Ray tracing :- (আলোর effect এর ছবি, realistic)

④ Rasterization :- (we draw so many lines, how lines can be drawn)

↳ order (1) এ line draw করা

⑤ Anti-aliasing :- (line এর point গুলো just plot করলে হবে না, আশেপাশের pixel এ color soften করে দেয়, visually pleasing করানোর, sharp change হবে না, point-to-point smooth হবে)

⑥ Illumination :- (source অনুযায়ী object/pixel এর color কেমন হবে)

⑦ Fractals :- (self-repetitive curve)

↳ পাহাড়ের দেয়াল, coastal area.

↳ যতই zoom করি, তত একই লাগে।

⑧ Texture :- (openGL object এর উপর কোনো image embed করা)

↳ sphere এর উপর world map set করা।

⑨ Curve :- (বিভিন্ন complicated curve তৈরি করা যায়)

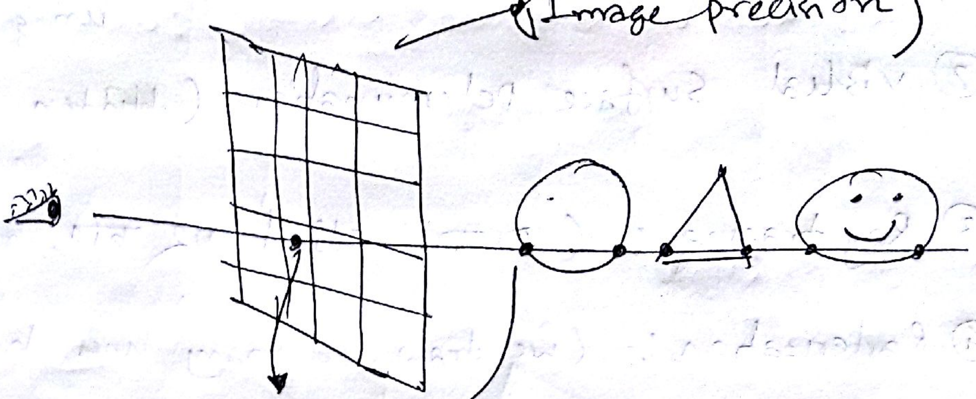
↳ b

⑩ Rendering / Advanced Raytracing (maybe পড়া হবে)

Hidden Surface Removal:- (learn 4 algo's)

2 approaches →

- → Image precision
- → Object precision



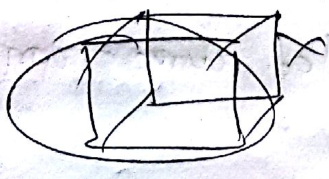
pixel-এর color হবে সেতে সামনের object-এর point-এর color-এর সমান হয় ।

সমস্যা:- অনেক time consuming.

→ যদি object-এর সাথে object-এর relation দেখে আঁকি, তো আমরা তা দিয়ে → object precision.

→ pre-processing:- image/object precision-এর আগে যেটা করতে হয় । near-এর আগে far-এর গুলো আগে দেখা অনেক কাজ নিতে হবে

→ Clipping:-



Algo's :-

- Preprocessing :- Back-face culling.
- Image precision :- • z - buffer
  - Scan-line - Algorithm.
- Object precision :- • Depth sort algo.
  - BSP Tree

Backface Culling : (See slide)

Surface-এর কিছু normal vector তো থাকে।

normal . direction vector > 0 দেখা যাবে।

" < 0 দেখা যাবে না।

এটা কেমন করে হয় ?

↳ inward faced surface গুলো remove করতে হবে না। (See the slide)

Hidden surface removal-এর সবচেয়ে সহজ Algo :-

Painter's Algorithm

↳ কিছুগুলো surface আগে draw করে সামনে আসা গুলো আসবে।

↳ 2টি object overlap করলে এটা কাজ করবে না।

Split করে করতে হবে —

- depth -এর basis-এ sort করি।
- loop হলে resolve করি

} like object precision algo.

→ use of painter's algo: opening multiple window on windows.

<u>Lecture-2</u> :-

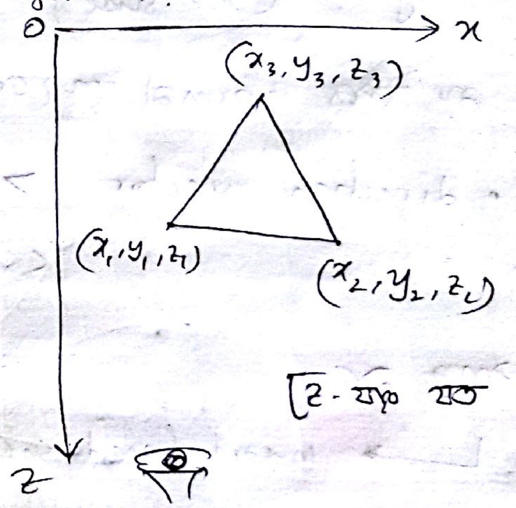
Date: 15 May, 2018
Tuesday

Object Precision : Depth-Sort Algorithm :-

① Sort

② check overlap:

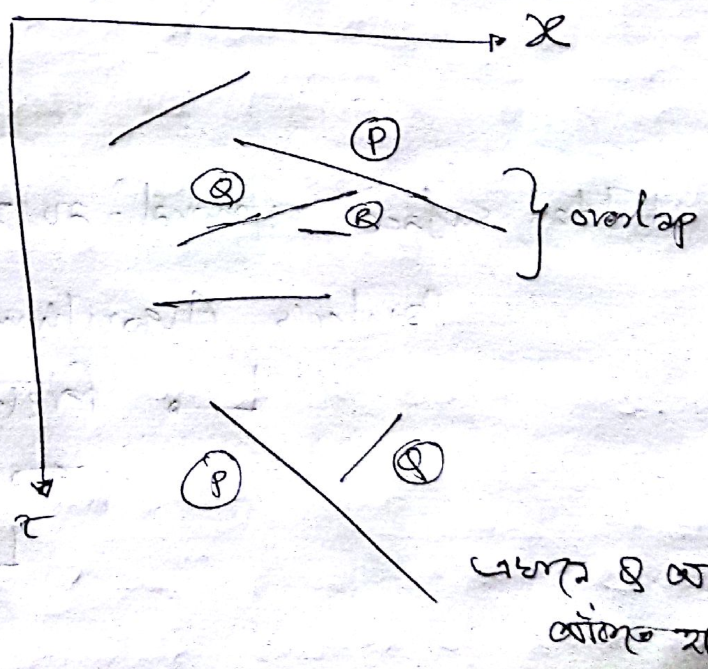
→ একটার min অন্যটার max যদি overlap করে → তাও ambiguous.



[z-অক্ষ যত বড় সে তত দূরে]

how to resolve?

যদি, প্রত্যেক দুটোর (P), বাকিদের সাথে pairwise check করতে হবে।



এখানে Q আগে আঁকতে হবে

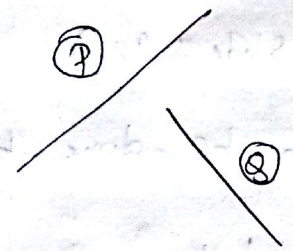
Ask some questions:-

① whether they overlap in x-axis, if not → can draw

② whether they overlap in y-axis, if not → can draw

Scanned by CamScanner

③ Check whether Q is totally on the same side in the direction of looking at P.



(Check by dotproduct any point of P with Q
↳ must be greater than 0)
↳ can draw (P) before Q.

[Exam: ~~for~~ overlap scenario ~~for~~ ambiguity resolve ~~করতে~~ হবে]

④ totally Q-এর opposite side → P-এর ~~for~~ না।

⑤ ~~দুইটার~~ projection plane-এ overlap ~~হয়~~ না।

If any of the above ⑤ is true, can draw (P) before (Q).

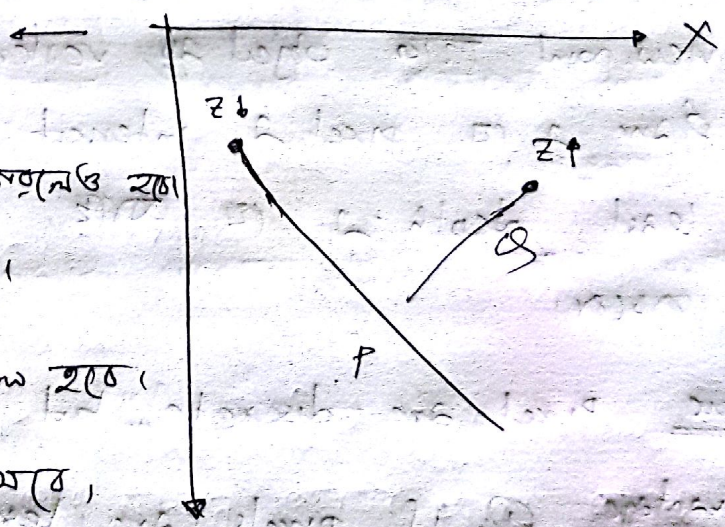
এখানে ~~সবই~~ false.

switch roles. (Q) ^ (P).

①, ② & ④ ~~আলাদা~~ না Check ~~করলেও~~ হবে

③ & ⑤ ~~আলাদা~~ check ~~করতে~~ হবে।

↳ ~~সবসময়~~ Q ~~আগে~~ draw হবে।



[Depth-sort-এ P ~~আগে~~ ~~আসবে~~,
but actually Q ~~আগে~~ draw ~~করতে~~ হবে।]

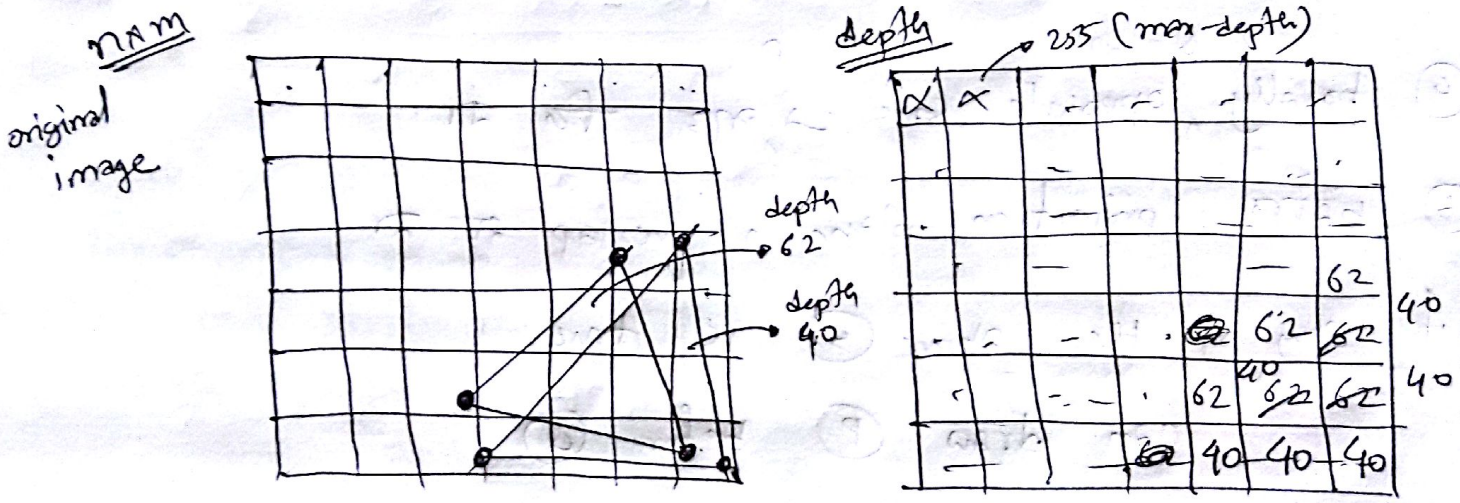
Depth Sort (slide : 42)

(a) ⟶ can be done by above algo. (5 tests)

(b) ⟶ can not be separated in the above process.
↳ need splitting.

(c) ⟶ need splitting.
↳ If switch roles, falls in cycle.

Image Precision : depth Z-buffer Algorithm (assignment-3 ও করতে)



view_point - থেকে object-এর vertex draw করলে projection plane-এ যে pixel-এ intersect করে সেগুলোর মধ্যে যেটার least depth-এ যে রঙ, ওর property তার fill up করবে।

Issue:- Pixel are discrete, but polygon is continuous.

Advantage:- ① If pixels are less, very good.

② we take this z-buffer as reference and can easily simulate motion of a single object.

③ Before hand, আমরা object-এর depth বের করে রাখতে পারি, যেন particular object-কে সরাসরি বাদ দেওয়া যায়।

সমস্যা :-

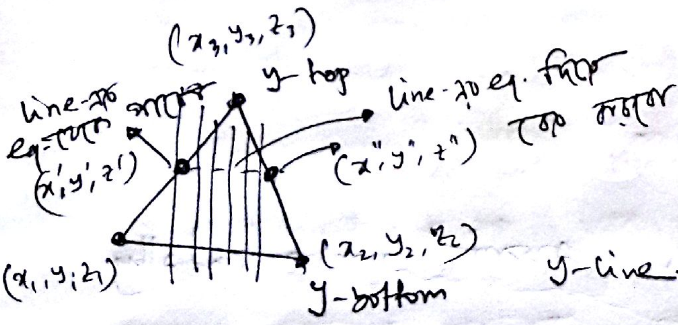
① zoom করলে সব calculation আবার করতে হবে।
  → object precision-এ ফিরে ফিরে এলো।

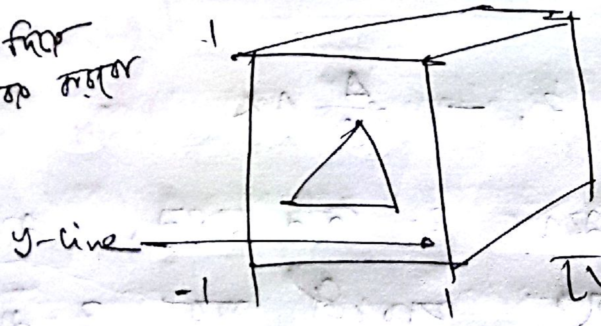
② multiple moving object-এ সমস্যা।

③ memory বেশি খরচ করে।

The present state of assignment :-

Stage 3 → সব point ([-1,1], [-1,1], [-1,1])





[view cube আছে আমাদের]

আমরা y-এর bottom থেকে y-এর top পর্যন্ত y-line ধরে ধরে scan করবো, তারপর x_{left} & x_{right} বের করবো।

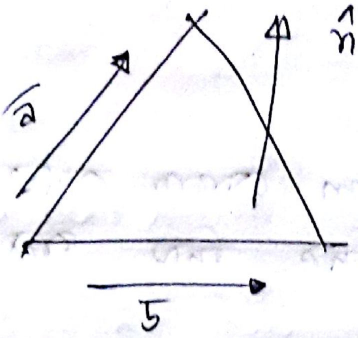
z-কে বের করার জন্য x-বরাবর column scan করবো।

row/column scanner increment হবে। আমরা সব pixel-এ ভাগ করতে চাই ও অনুপাতী।

যেমন : $\frac{1-(-1)}{500} = \frac{2}{500}$ (suppose, row = column = 500)

How to calculate z-value:—

three points given, get the eq. of plane.



$\hat{n} = \bar{a} \times \bar{b}$

$ax + by + cz + d = 0$

$\hat{n} = a\hat{i} + b\hat{j} + c\hat{k}$

for any (x, y, z) on plane

calculate $d = ?$

∴ then,
$$z = \frac{D - Ax - By}{C}$$

$$z_1 = \frac{D - A(x + \Delta x) - By}{C}$$

$$z = z_1 - \frac{A}{C} \Delta x$$

∴ অতএব, প্রথম z- বের করে, প্রত্যেক incrementally loop চালিয়ে পরবর্তী সব z-এর মান বের করা সম্ভব।

আবার যখন, y-অক্ষে scan হবে, তখন,

$$z = z_1 - \frac{B}{C} \Delta y.$$

[Assignment এ, incremental / normal calculation যেকোনো একটাই হবে]

সবশেষে, xy-এর জন্য ঐ pixel-এ z value টা রাখবো।

Date: 16 May, 2018
Wednesday

Lecture-3

Assignment 3 :-

input: stage3.txt, config.txt

↓

500 500 → pixel

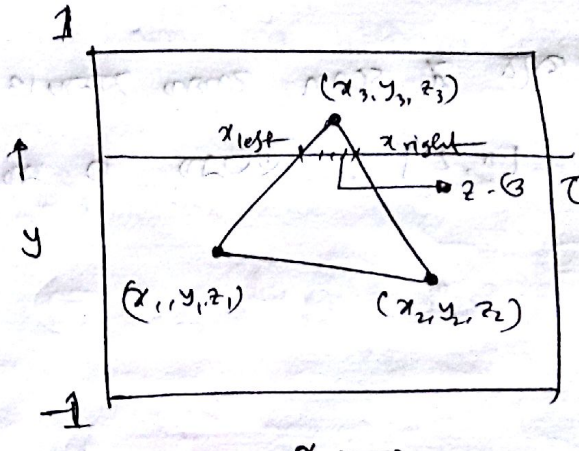
-1 → range x [-1, 1]
-1 → range y [-1, 1]
0 2 → range z [0, 2]
} (new view volume)

pseudo-code:-

$\frac{500 \times 500}{}$

increment: $\frac{\text{range}_y}{500}$

$= \frac{2}{500}$



প্রতিবার y-line scan করার সময়, x-left, x-right বের করবো।

eq: $(x_1, y_1, z_1) + t(x_3 - x_1, y_3 - y_1, z_3 - z_1) = L(t)$

at any point, $y = y_1 + t(y_3 - y_1)$

$$t = \frac{y - y_1}{y_3 - y_1}$$

$$\therefore x = x_1 + \frac{y - y_1}{y_3 - y_1} \times (x_3 - x_1)$$

এভাবে দুইটার line-এর মাঝের দূরত্ব বের করতে হবে,

x_left & x_right এর।

এবং 2 range-এর মধ্যে x-scan হবে, z-buffer করবো।

এখানে y-constant.

আমাদের z-buffer ব্যবহার

↳ new z-value buffer-এর আগের value থেকে ছোট হলে, এর আগের বাদে, z-buffer update হবে।

How to clip?

→ y-scan করার সময় আমাদের check হবে min(topy, ylimit)

একইভাবে z-scan-করার সময়ও check করতে হবে।

z : [0, 2] : অর্থাৎ, 0-এর কম হলে তার বাদ যাবে।

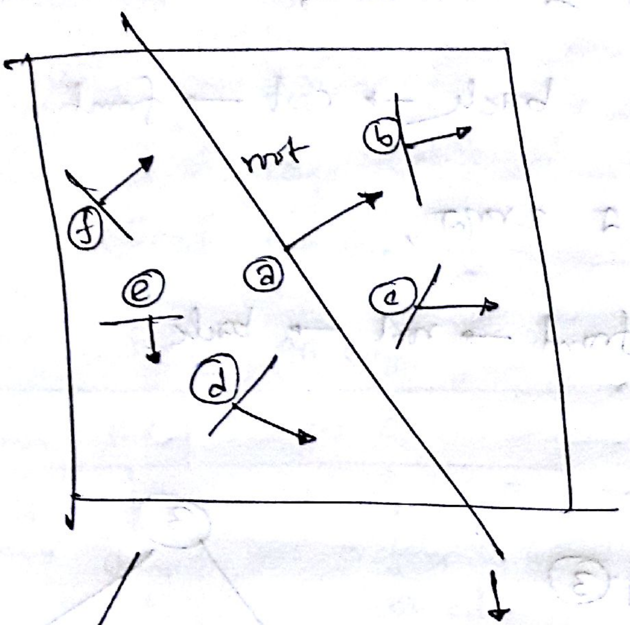
Binary Space Partitioning (BSP) Tree :- (Object precision)

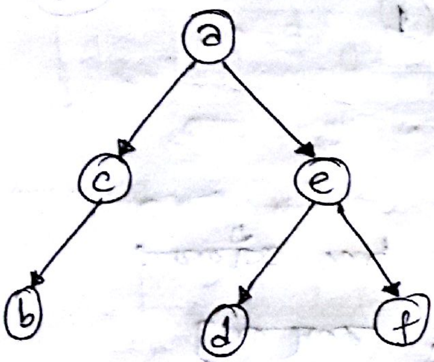
↳ প্রথমে অবজেক্ট sequencing করে ফেলা হয়।

↳ এরপর view-এর সাথে আলাদা আলাদা প্রতিটির rendering করা হয়।

P.T.O.

how it is done ?





choose an arbitrary plane
as root

পিছনের গুলো → back

সামনের গুলো → front

recursively
tree বানাতে থাকবো

যদি কোন root অন্য একটিকে intersect করে
তাহলে যাকে intersect করে,
তাকে front / back দুভাগে ভাগ
করে ফেলতে হবে।

[trick: how to choose root wisely?]

Do in-order traversal & get a front-root-back sequence

[see slide]

Scanned by CamScanner

view root - এ front - এ থাকলে,

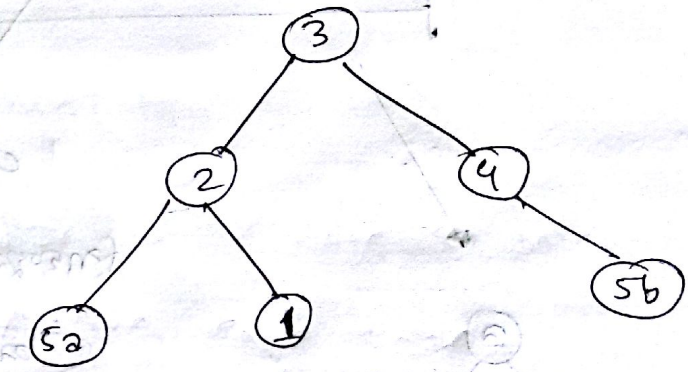
object draw হবে, back → root → front.

view root - এ back - এ থাকলে,

object draw হবে, front → root → back.

example: root 3

view at front of (3)



order:-

(4) (5b) (3) (5a) (2) (1)

[Class Test → : scenario থাকবে, arbitary tree আঁকবে, view point অনুযায়ী drawing বলবে]

↳ এতে backface-culling করলেও চলে

Scan-line Algorithm (Image Precision):-

↳ [Ref: Folley's book; ch-3, 15]

z-buffer → : একই pixel, একাধিক object থাকে তখন depth update হয়।

↳ (সমস্যা)

scan-line → : একবার pixel update হলে আর না।

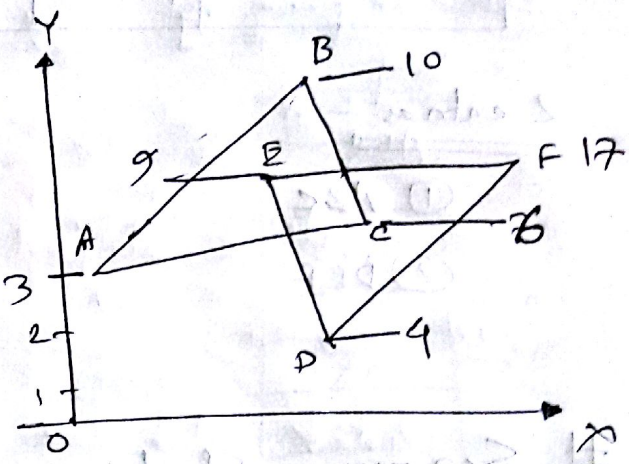
↳ সবশেষেই update finalize হয়ে যাবে।

Different data structures required :-

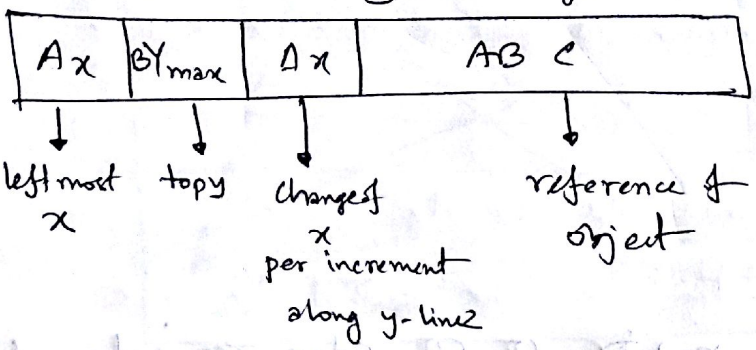
At first, two nodes :-

① at edge table.

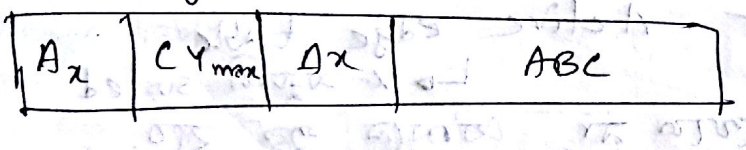
② at polygon table.



Edge table : Entry : AB edge

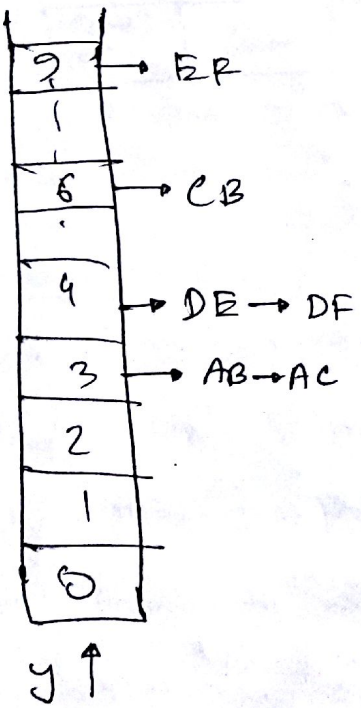


AC edge



→ then we have to do bucket sort on edge table.

↳ buckets are each value of Y_0 to Y_{max} along y-axis.



edge-এর y-bottom অনুযায়ী bucket-এ রাখতে।

↓

multiple entry হলে, x-এর lower value যার সে যেটা আগে আসবে।

Polygon table:—

eqn	color	bool

2 entries:—

① ABC

② DEF

Scanning Starts:—

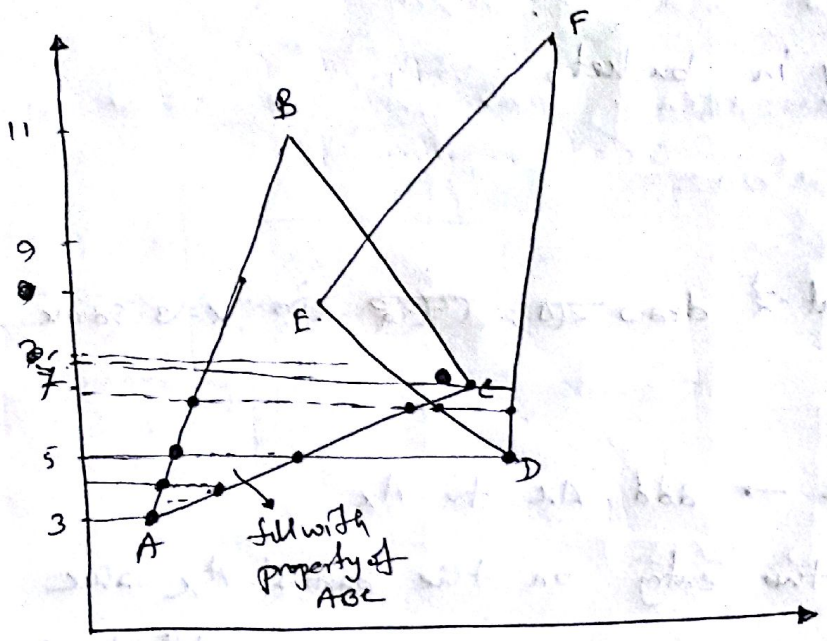
row-scan চলার সময়ে :- row তে যে যে edge-কে intersect করে, তার সমস্ত - Active edge table.
↳ x-অনুযায়ী sorted

↳ প্রথমে bottom থেকে না, প্রথমে শুরু হবে, bucket-এ lowest যে edge entry আছে সেখান থেকে।

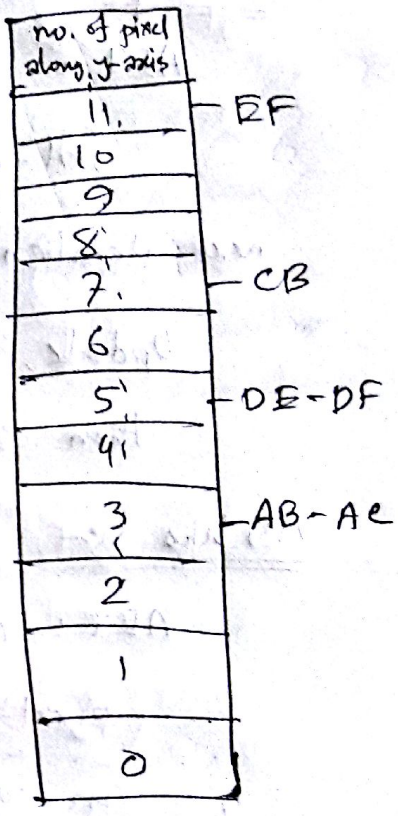
Lecture - 4 :-

Date : 20 May, 2018
Sunday

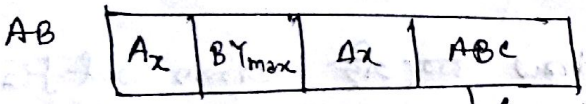
Image precision : Scan-line Algorithm



edge table :- (in bucket sort)

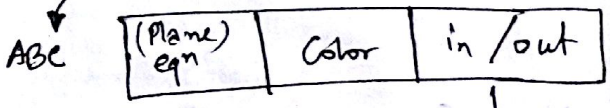


I) edge table :-



(keep the polygon table entry pointer)

II) Polygon table :-



true if inside ABC
false if outside ABC

Explanation :-

A_x : left most x-value at current scan-line

By_{max} :

Δx : $m = \frac{\Delta y}{\Delta x}$, here, $\Delta y = 1$, so, only Δx is stored to get next point.

ABC : Belongs to which polygon. (keep the pointer to polygon table entry)

Plane(eqn) : keep 4 values, $ax + by + cz + d = 0$

these 4 values.

Algorithm :-

Initially, Aactive Edge Table (AET) = ϕ

Start scan from 1st entry in bucket;

in loop:

AET = AB, AC

first -এ যেই point-এ draw হচ্ছে, সেখানে যেহেতু x-এর same

next iteration-এ,

Update, A_x value → add Δx to A_x.

then sort the two entry on the basis of A_x value left to right.

At line 5:-

AET : AB, AC, DE, DF.

pixel-টা polygon-এর কয়টি draw করেছি, তারপর out flag on করে দিতে হবে।

(sort after adding new entries)

At line 7: we reach y_{max} of AC edge; so remove it,

Again add CB as new entry.

∴ AET: AB, DE, CB, DF.

AB-তে যেতে যেতে ABC আর DEF overlap করলে

কোনটা দিয়ে fill up করবে?

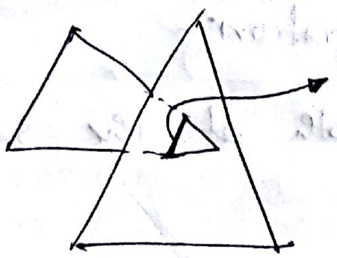
→ Check ABC & DEF's plane eqn :-

get whose z-value is nearer.

যেটার মান বেশি সেটার out flag চালু হবে,

আর যেটার দিয়ে fill হবে সেটার in flag on হবে।

penetrating plane হলে এই Algo খাটে না।



এটা solve করতে হলে, আমাদের dummy edge তৈরী করে দিতে হবে।

কিভাবে তৈরী করবো?

find the intersection of two planes.

1) $n_1 \times n_2$ gives a vector parallel to intersection line.

2) Get a point on the intersection of the planes.

আমাদের Assignment-1 এ কখনো penetrating polygon case হবেনা।

Lecture-5 :-

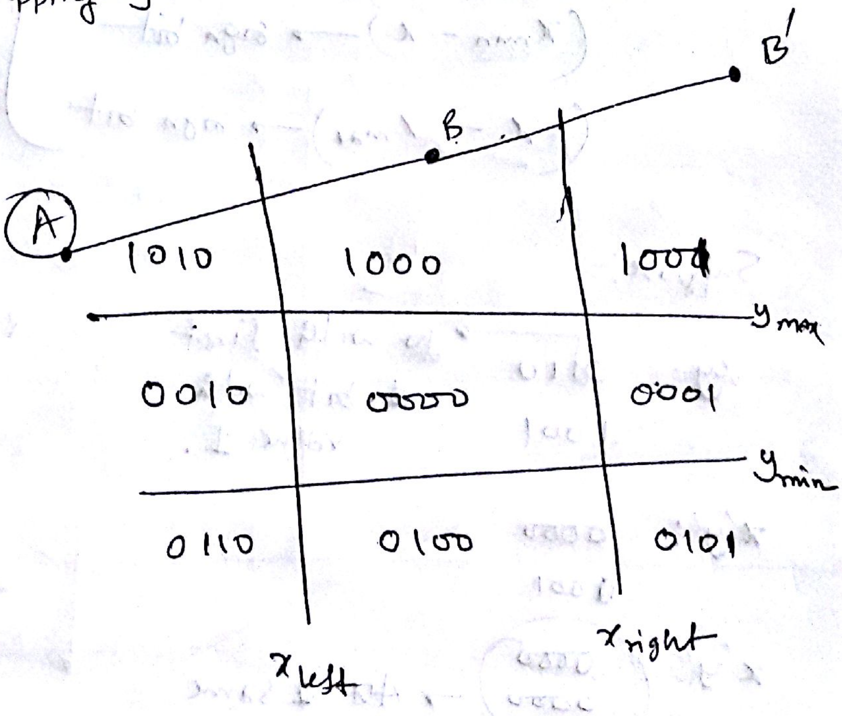
Date: 22 May, 2018 Tuesday

(i) Cohen-Sutherland

(ii) Cyrus-beck parametric line clipping } 2D - 3D

Polygon-clipping :-

Sutherland hodgeman's



Region গুলোতে binary value assign bitwise.

A: এর জন্য

y_{max} - এর উপরে 1st bit 1

y_{min} - এর নিচে না 2nd bit 0

x_{left} - এর left-এ 3rd bit 1

x_{right} " right-এ না 4th bit 0

Scanned by CamScanner

we shall assign value to two terminal points and do logical "and" operation;-

If get a "1" this means both at the same side at test bit position.

```
A :   1010
B' :  1001
-----------
A.B' : 1000
```

(ymax -ve sign)

how to get the bit value;-

use the sign bit of subtraction;-

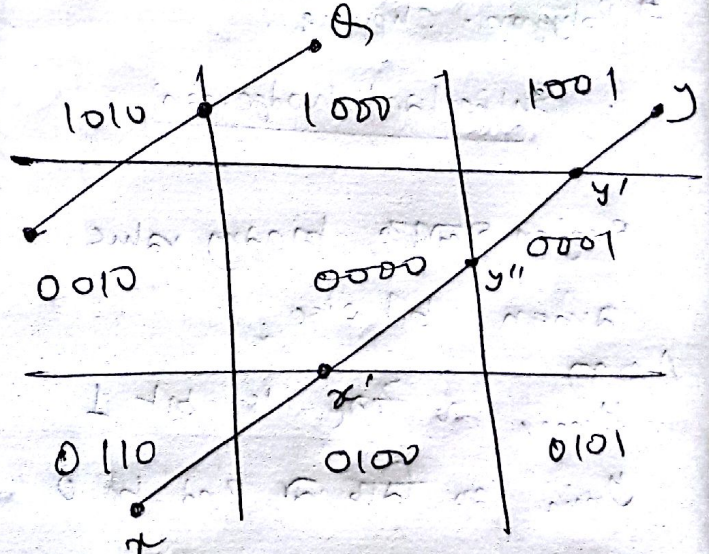
	A
$(y - y_{max})$ → sign bit	1
$(y_{min} - y)$ → sign bit	0
$(x_{min} - x)$ → sign bit	1
$(x - x_{max})$ → sign bit	0

Suppose;-

xy : 0110 → go with first 1 bit with value 1.
1001

x(y') : 0000
0001

x'y'' : 0000
0000 → All 0 same
trivially acceptable



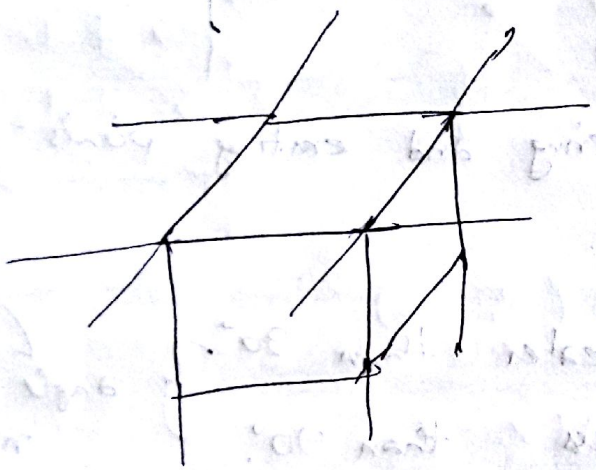
Suppose:-

PQ : ~~0000~~ 0010
1000

P'Q : 1000
1000 } trivially acceptable.

this does not require 4 checks in all cases.

3D extension:-



এখানে দিয়ে হবে,
plane-এর কোন
দিকে সেটা অনুপাতী।
এবার cube-এর ক্ষেত্রে কাজ
হবে।

↳ তারপর এটা just; exam-এ
2D-ই আসবে।

(ii) Cyrus-beck parametric line clipping :-

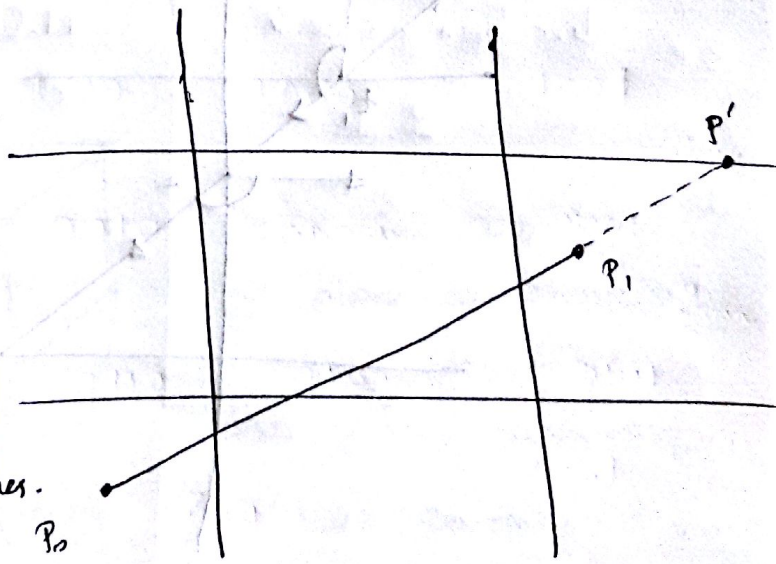
$P(t) = P_0 + t(P_1 - P_0)$

$t = [0, 1]$

we get 4 t's for 4 intersecting lines.

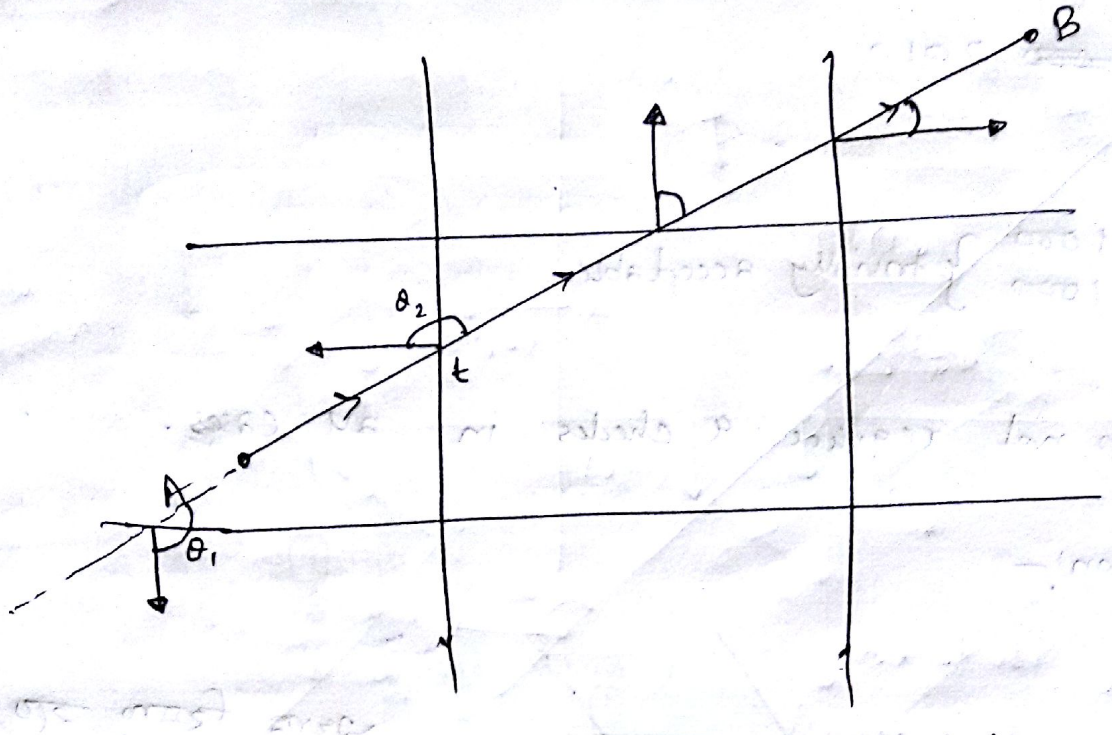
[for parallel lines, we get 2 t's]

↳ consider only non-parallel lines.



P'-এ t-এর value 1 ↑

∵ P_0P_1-এর অবস্থান on right direction.

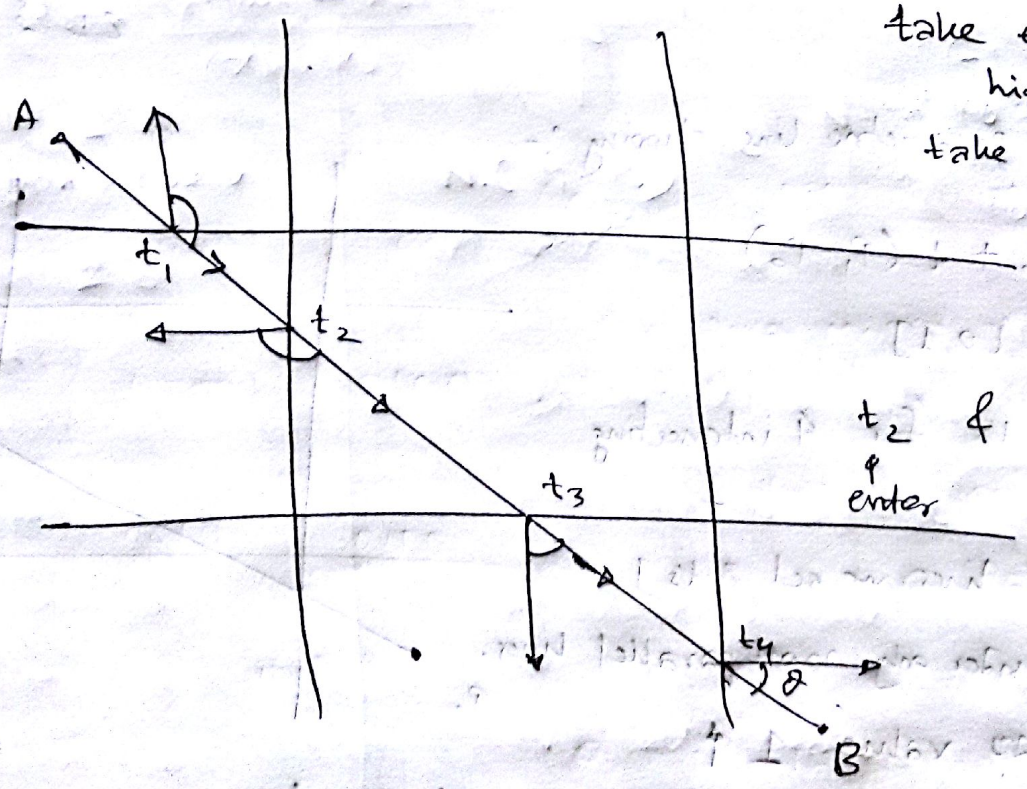


get normal at entering and exiting points and angle between them.

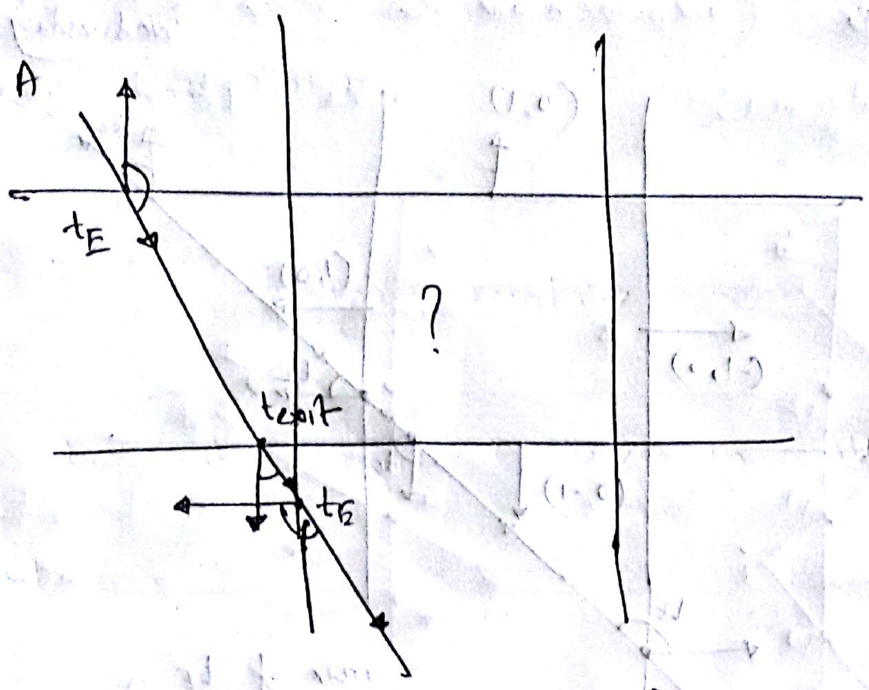
entering → $\angle\theta$ greater than 90°
exiting → $\angle\theta$ less than 90°.

} angle between normal & vector. (get by dot product)

take entering → higher t,
take exiting → lower 't'.



t_2 & t_3
t_2 ↑ enter
t_3 ↑ exit



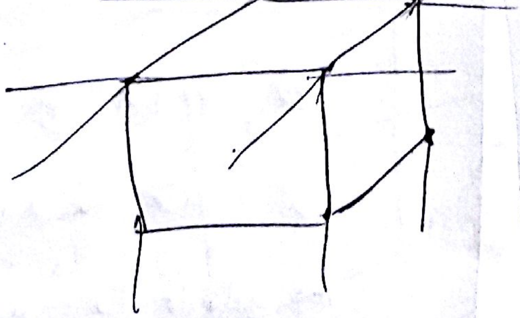
Actually region-এর ভেতর দিয়েই যায় না।

কখন বাদ দিবো?

↳ যদি entering-এর max t is less than exiting-এর min t, that line does not enter the region.

[# Clipping - actually built-in আছেই। but আমাদের assignment-এ এটা ব্যবহার না।]

for 3D:-
(syllabus-এ নেই)



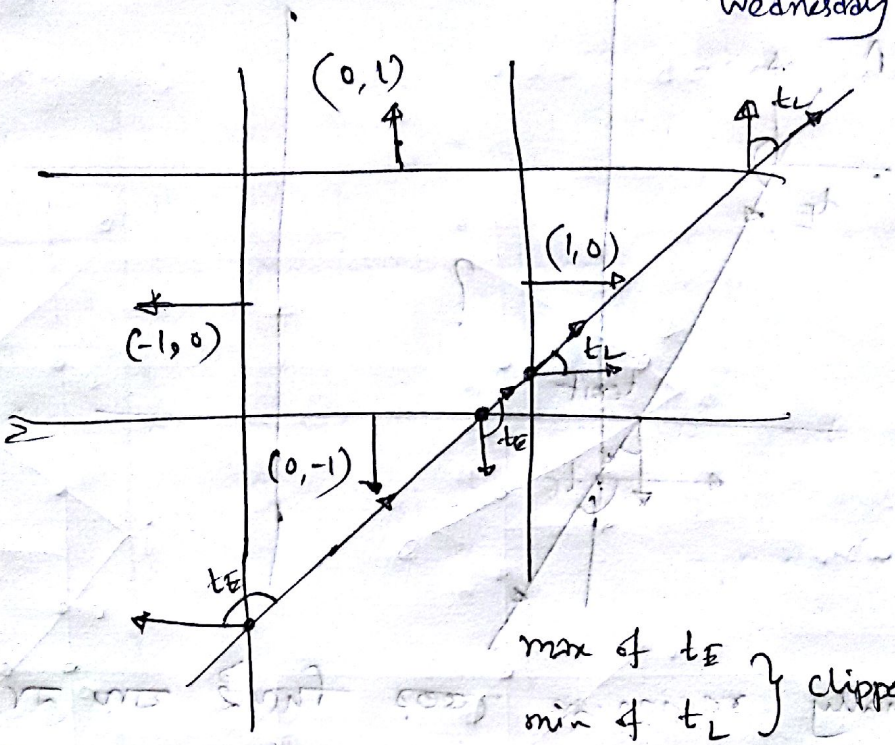
৩টা line সব plane-এর সাথেই cut করে।
তখন 6টা করতে হবে।

(we are done) { max ← 3 ~~max~~ : entering
min ← 3 : exiting

Date: May 23, 2018
Wednesday

Lecture-6 :-

(if no rotation)



max of t_E
min of t_L
} clipped portion

how to get the t's :-

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

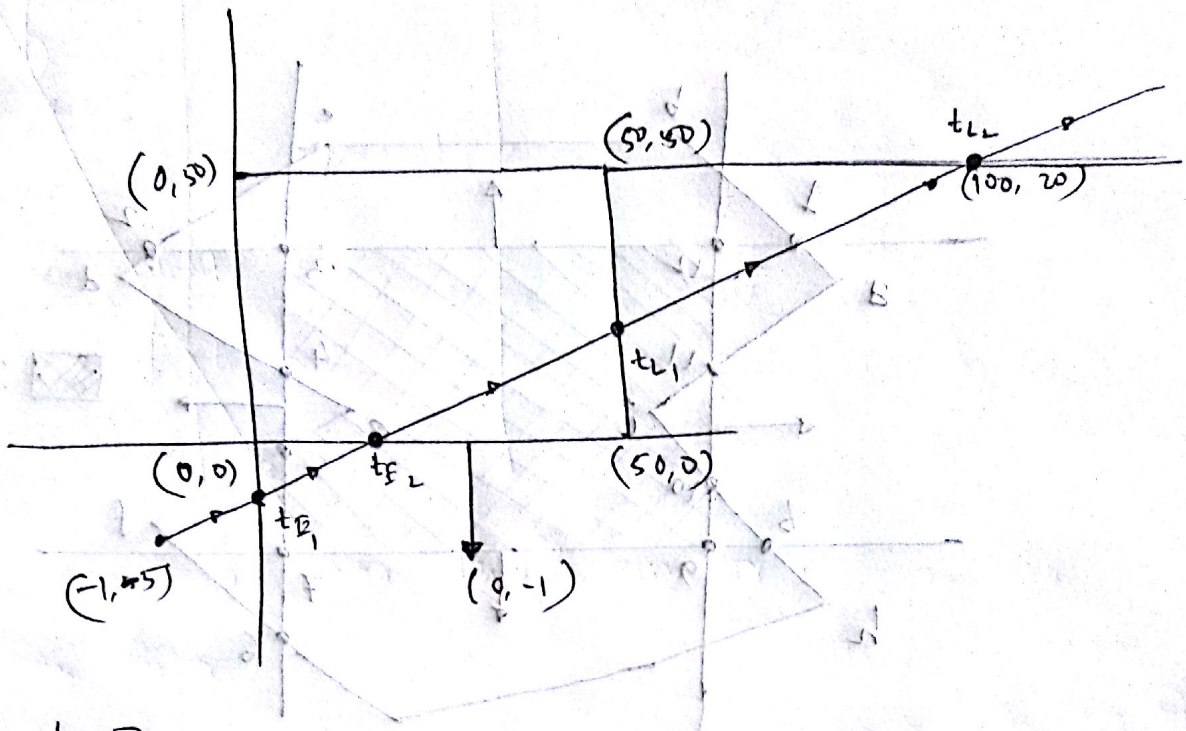
$$x_{min} = x_0 + t(x_1 - x_0)$$

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

$N = \begin{cases} (-1,0) \\ (0,-1) \\ (1,0) \\ (0,1) \end{cases}$

direction check :- $N.(P_1 - P_0)$

Example question :-



t_1 = - - - t_{E_1} } max (t_E)

t_2 = - - - t_{E_2} }

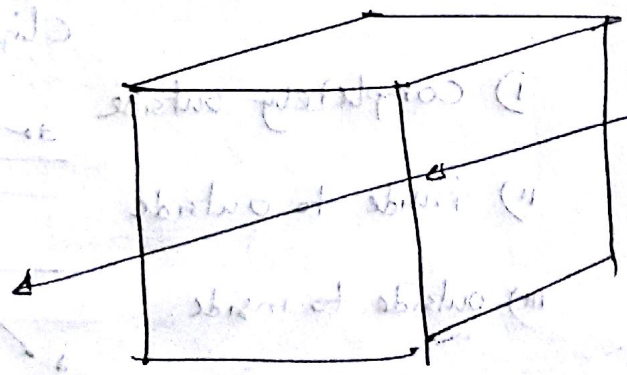
t_3 = - - - t_{L_1} } min (t_L)

t_4 = - - - t_{L_2} }

[See the cyrus-beck pseudocode from book]

↳ very easy.

in 3D :- there will be 6 t's



(cuts plane - 6 planes)

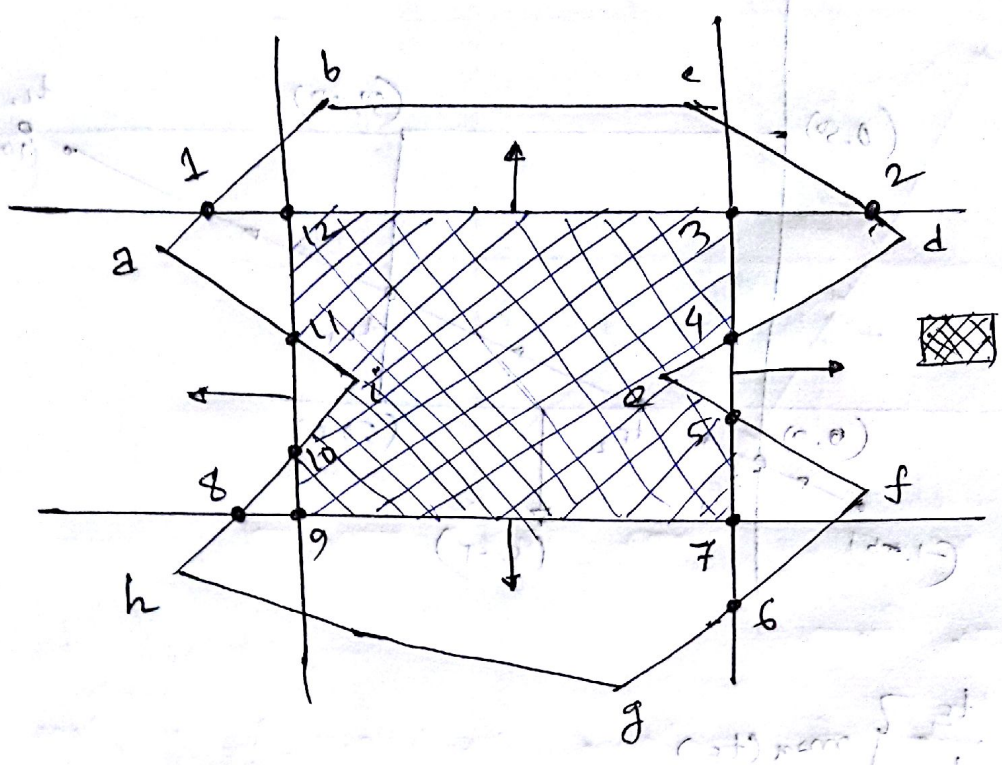
(a line intersects all 6 planes)

↳ (not in syllabus)

↳ 3D.

Polygon Clipping Algo:— (one problem like this in exam)

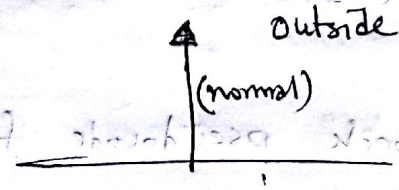
Sutherland Hodgman :- order: abcdefghi



প্রতিটা line-কে আলাদাভাবে clip করে তারপর সবগুলোকে merge করতে হবে।

take edges in order (top clip line)

ab	in-out	1
bc	out-out	∅
cd	out-in	2, d
de	in-in	e
ef	in-in	f
fg	in-in	g
gh	in-in	h
hi	in-in	i
ia	in-in	a



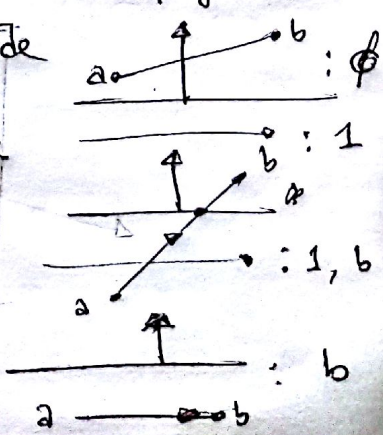
A line can be in 4 states w.r.t. clipping edge.

i) completely outside

ii) inside to outside

iii) outside to inside.

iv) full inside :



now order:-

1 2 d e f g h i a

w.r. to right :-

12	in-out	3
2d	out-out	∅
de	out-in	4, e
ef	in-out	5
fg	out-in	6, g
gh	in-in	h
hi	in-in	i
ia	in-in	a
a1	in-in	1

new order:-

3 4 e 5 6 gh i a 1

w.r. to bottom :-

34	in-in	4
4e	in-in	e
e5	in-in	5
56	in-out	7
6g	out-out	∅
gh	out-out	∅
hi	out-in	8, i
ia	in-in	a
a1	in-in	1
13	in-in	3
~~a1~~	~~in-in~~	~~1~~

new order:-

4 e 5 7 8 i a 13

After w.r. to left :-

e 5 7 9 10 i 11 12 3 4