Edge Coherence

CLASSTIME | Page No.
Date / /

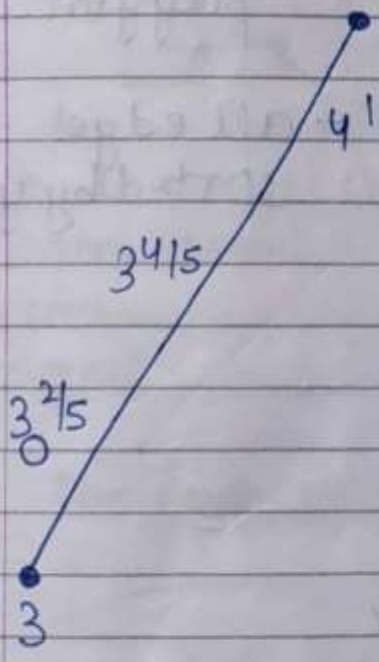
If an edge intersects with a scan line, and slope of edge is m, then successive scan line intersections can be found from

$$x_{i+1} = x_i + \frac{1}{m}$$

(scan line count)

$$\frac{1}{m} = (x_1 - x_0) / (y_1 - y_0)$$

The floating pt. arithmetic can be avoided by storing numerator comparing to the denominator, and incrementing x when it overflows.



$$m = \frac{6-1}{5-3} = \frac{5}{2}$$

$$\frac{1}{m} = \frac{2}{5}$$

$x_{min} = 3$

$\therefore$ Seq $\rightarrow$ $3 + \frac{2}{5} \Rightarrow 3\ 2/5$ (+2/5)

x_i : (3) 4/5, (3) 6/5 = 4 1/5

x_i

num > den.

$\therefore$ rather than adding 1/m we don't increment x value unless num becomes > than den.

$\therefore$ increment x $\rightarrow$ 4 from 3.

$\therefore$ $4\frac{1}{5}$

CLASSTIME | Page No.
Date / /

{ numerator → 2→4→1
denominator → 5. }

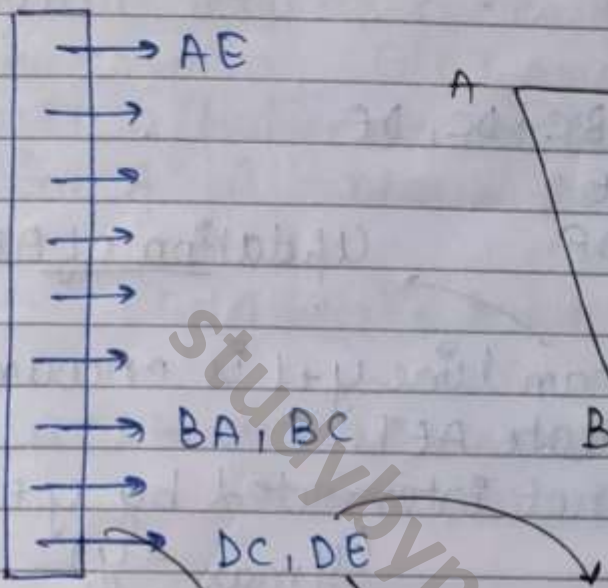
- Algo:

(1) We need a DS to store the polygon.
2 data structures

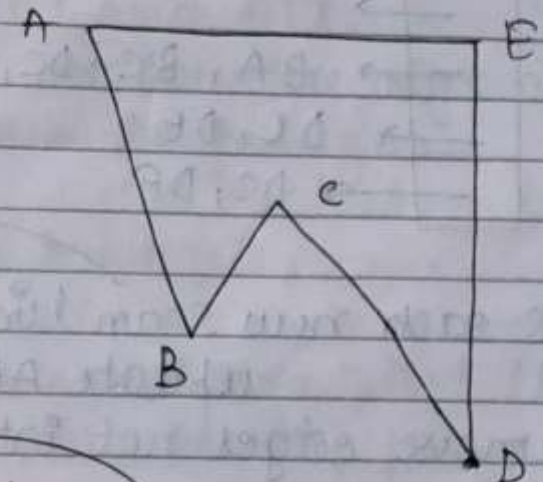
AET
Active edge Table
- Stores info abt all edges which are being currently intersected by scan line.
- Stores edges intersected by scan line y sorted by x coordinate
- Each entry in AET contains ymax coordinate of edge, the x coordinate of intersection pt. and 1/m. → to check whether the previous edge/polygon part is painted or not.

GET
Global edge Table.
- DS which is used to store edges of the polygon.
- all edges sorted by ymin

- GET with all edges sorted by y_{min}
- AET containing
  - edges that intersect current scanline.
  - Their pts. of intersection.
  - Sorted by x coordinate left to right.



GET



Have same y_{min}

y_{min} is on same level for same edges.

(no. of entries = no. of scanlines)

sorting on basis of x coordinate.

- For each edge, we need.

↓

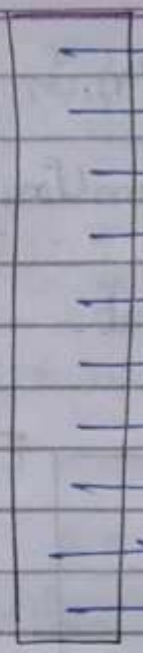
x_{min} and y_{max} and slope of edge.

y_{min} is not req. as we are sorting acc. to that only.

Stored in GET and not AET → Stores x coordinate as pt. of intersection. and (m) and y_{max}.

for calculating x (pt. of intersection)

$x_n = x_0 + 1/m$



{ info. of edges currently intersected by a scan line }.

Updation of AET

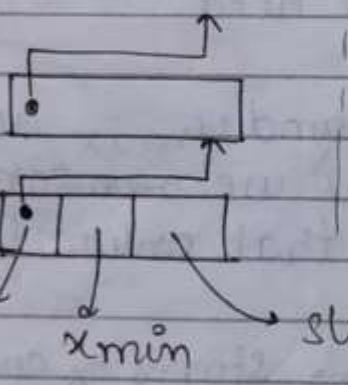
→ As each new scan line $y+1$ is encountered, update AET.

(1) Remove edges not intersected by $y+1$ ($y_{max} = y$)

(2) Add edges intersected by $y+1$ ($y_{min} = y+1$) from GET

(3) Calculate new x intersections. $\{x_{new} = x + \frac{1}{m}\}$

- GET



now fill the AET using values from GET. and we continue algo until we get empty AET.

- Scan line Algorithms

- Find & sort intersections
- Fill spans

} 2 subproblems for polygon filling.

Q (in exam) dry run the polygon filling algo. by creating GET and AET.

Q Given a polygon, follow the algo of scan line & fill colour to polygon.

4th Feb, 19

Odd parity rule

used to decide whether a pixel lies inside / outside the polygon.

* Scan line Algorithm

(1) Set y to the smallest y coordinate that has an entry in GET, that is 'y' for the first non empty bucket.

(2) Initialise AET to be empty.

(3) Repeat until AET and GET are empty.

→ Move from GET bucket y to AET those edges whose $y_{min} = y$ (entering edges)

→ Remove from AET those entries for which $y = y_{max}$ (edges not involved in next scan line) then sort AET on x (made easier because GET is pre-sorted).

→ Fill in desired pixel values on scan line y by using pairs of x coordinates from AET.

→ Increment y by 1 (to coordinate of next scan line).

→ For each non vertical edge remaining in AET, update x for new y.
(For vertical edge, x is same)

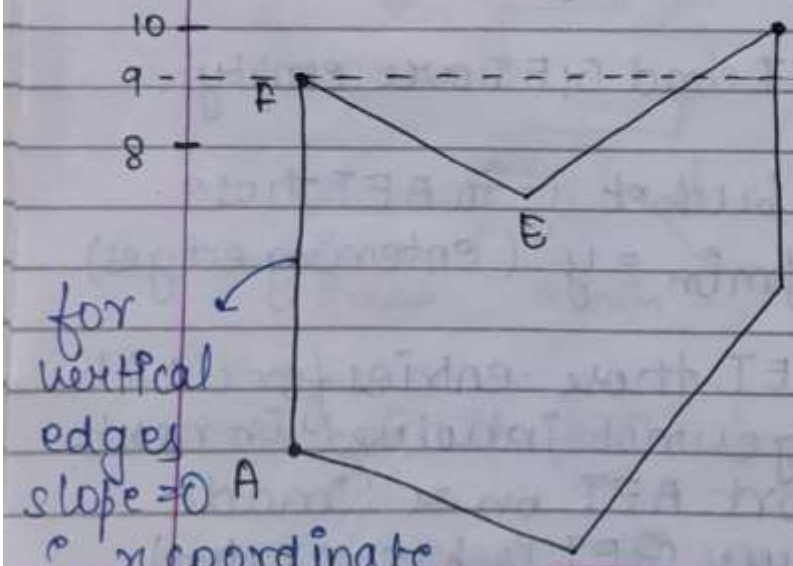
Also, for existing edges in AET, update their x coordinates for finding pt. of intersection with scan line.
(Update x for new value of y)

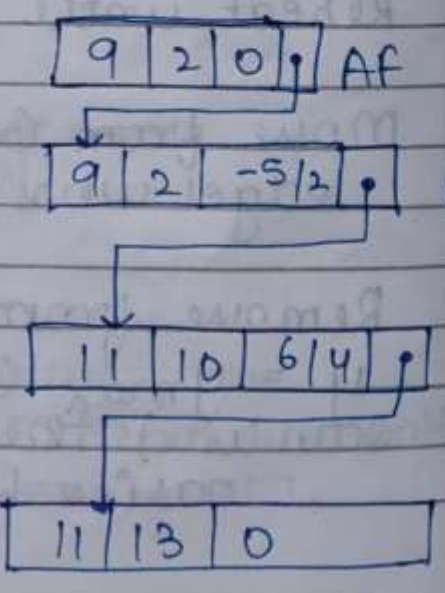
$$x_{new} = x + \frac{1}{m}.$$

→ For GET and AET → Singly linked list as it is updated for every scan line.

array linked list req.

→ We also need to keep a record of edges.





CLASSTIME | Page No.
Date / /

→ We are on scan line 9
Now we see that at scan line we have y_{max} of FE and FA as 9
∴ remove AF and FE.

And if there is an edge with $y_{min} = 9$ then add to AET.

Now increment $y \rightarrow y+1$ i.e 10

scan line no. 10.

Also increment x (pt of intersection)

- Outline :-

(1) Drawing with Thick primitives
(2) Halftone Approximation.
(3) Anti-aliasing

→ One of the attributes to draw a figure / primitive is thickness

In all examples above, we had 1 pixel thickness.

* How to draw a thick primitive

(1) We can do pixel replication to ↑se thickness but will depend upon the slope. (either row-wise or column-wise).

by replicating pixels.

→ Column (Row) Replication :-

(1) | → pixel replication is col. wise

(2) ——— → pixel replication is row wise

But for steep and gradual slope

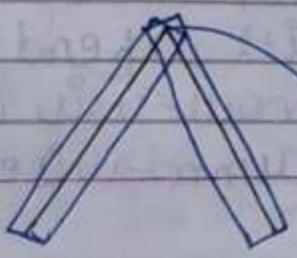
$|m|>1$ → $\frac{dy}{dx}>1$ → replication is column wise.

$|m|<1$ → replication is row wise

- Disadvantages :-

(1) One of the problem with replication is (primitives will not have uniform width) even width. → Pixel replication is better with odd width.

(2) another problem is with ends i.e pixel ends are flattened.



→ gap is encountered since ends gets flattened.

→ Increasing width will enlarge/ Shrink image..

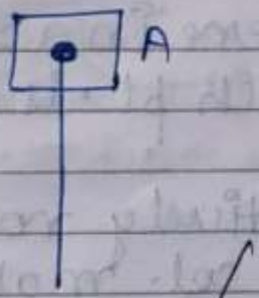
CLASSTIME | Page No.
Date / /

when we join them we get gaps at corners & they need to be filled.

(3) When we draw lines with equal thickness, then perceived thickness is not proper, i.e seen thickness will vary with diff. slopes.
Max. thickness is seen for horizontal & vertical line & the perceived thickness, is generally becomes as $\frac{t}{\sqrt{2}}$ → reduces of a line with a particular slope.

→ 2nd method ↓
A moving pen

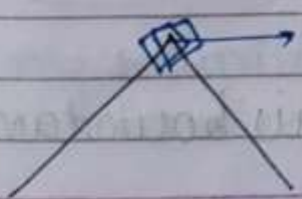
- Replace a pixel with a matrix of pixels.



Suppose a grid of pixels is of 3X3

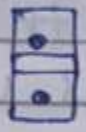
Now in place of point A, it is replaced by a grid of pixels.

The problem of flattened ends gets solved. gap gets filled by pixels.



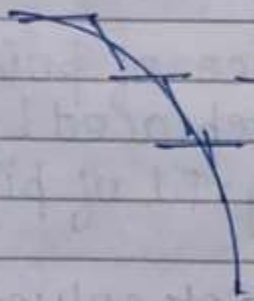
→ Disadvantages -

(1) Pixel repetition.
↓
There is overlapping of pixels.



(2) Here, in this case there is problem of perceived thickness.
↓
Slope with a particular line appears thickest and vertical & horizontal lines appear less thick.
opp. to that of 1st method

(3) For curved figures.
↓
eg - circle (Here in a circle, slope at each pt. is diff.)



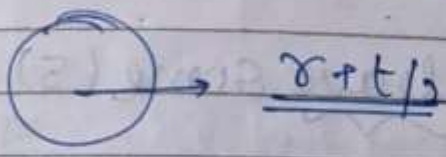
→ alternatively row & col. matrix colouring.

→ 3rd method ↴

Instead of drawing pixels, we ↶ Filling Areas B/w boundaries
fill area above & below the line.

We make a rectangle & fill pts. b/w that rectangle.

eg - To draw a circle of thickness 't' we draw 2 concentric circles

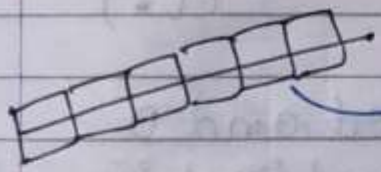


* 4th method

any pixel with thick polylines.

multiple lines

approximate a line with multiple lines.



making non overlapping grids.

→ 2nd attribute of any primitive is "colour"
putpixel (x, y, color).

→ 3rd attribute → style/pattern.

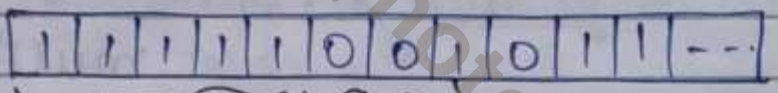
any primitive drawn till now was of solid style

→ For dashed lines (- - - -)
↓
plot pixels and leave some.

→ For dotted lines (. . . . .)
↓
plot a pixel & leave some (5)

leave means don't send them in putpixel, just evaluate them.

- Dashed dotted lines (— . — . —)
↓
Take an array with bit values of 0 & 1's.



5 pixels make a (—) | for spaces | 1 pixel for 1 (.)

1 means pixel is plotted and 0 means no pixel is plotted i.e a blank.

To repeat this procedure, we do modular arithmetic.
↓
(Circular array) → index % maxsize + 1
at last index, after evaluation it again returns to 1st index.

(5th algorithm)

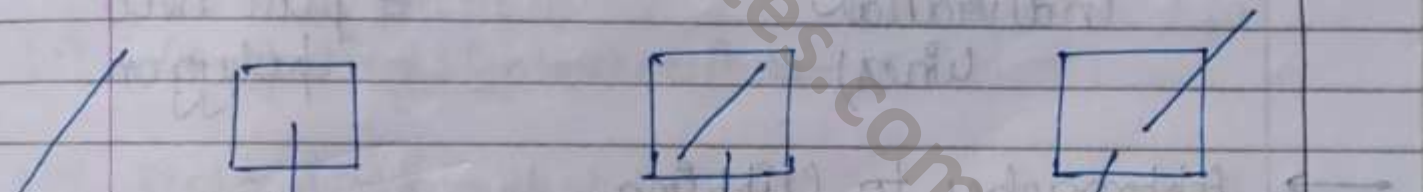
CLASSTIME | Page No.
Date / /

→ For every line, we will now add color, thickness and style.

NOTE: We can implement any pattern using 'BitString'.

6th Feb, 19 Clipping on a Raster Display (3rd algorithm)

- Remove points outside a region of interest
- Show an image in a particular area of screen. (clip area)
- Discard (parts of) primitives outside of window.



To be considered
↓
{ primitive - line.
clip area - window }

- Point clipping -
- Remove pts. outside window. (by comparing coordinates of pt. & clip

$x_c > x_{min}$ and $x_c < x_{max}$ → pt. is inside clip window

window).

- A pt. can either be entirely inside / outside.

- Line Clipping.
  - Remove portion of line segment outside window.

- Polygon Clipping
  - Remove portion of polygon outside window
    - clip each edge for window (not clipping individual lines)
    - find vertices of region lying inside polygon in clip area & plot new polygon.

→ Approaches to Clipping.

- Clipping
  draw primitive & remove portion outside window (first plot entire primitive & then check)
- Sciessoring
  Keep on seeing what portion is inside clip window & plot primitive. (find pts. of intersection of line with clip region)

line completely inside clip window

- scan convert after finding out which portion is inside clip window. & draw only that portion.

Clipping (End pts)

→ Inside $x_{min} <= x <= x_{max}$

$y_{min} <= y <= y_{max}$.

* One of the basic methods to clip line

(1) Computing Intersections

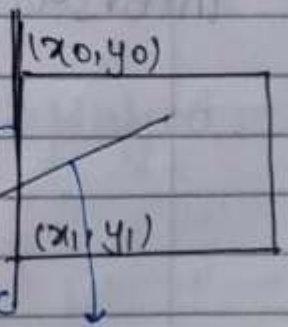
→ Slope intercept form.

→ parametric form

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

(x_1 - end pt.)

parameter t in [0, 1].



If t is b/w 0 & 1

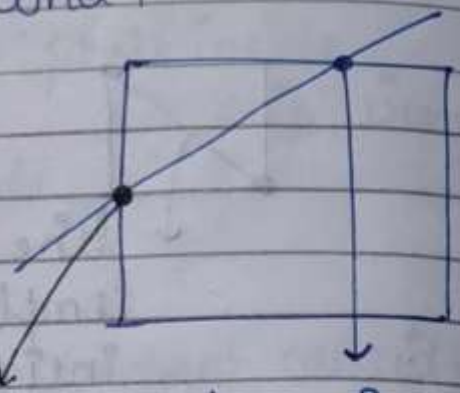
edge representation by parametric form

line representation by param. form.

pt. of intersection is inside/on clip window

1 t for edge and 1 for line.

If t is outside 0 and 1 then, pt. lies outside clip window



Brute force techniques.

Find parameter t b/w line & b/w each edge.

t_{line} = t_{edge} = lies b/w 0 and 1

t_{line} is outside 0 & 1 t_{edge} is inside 0 & 1

max. 2 pts of intersection

choose 2 values of t b/w 0 & 1 → Valid pt. of intersection of x and y.

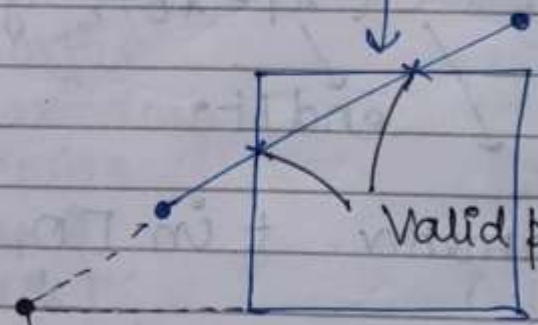
If both end pts. inside window

2 valid pts. of intersection.

If both end pts. outside window

If one end pt. is inside window

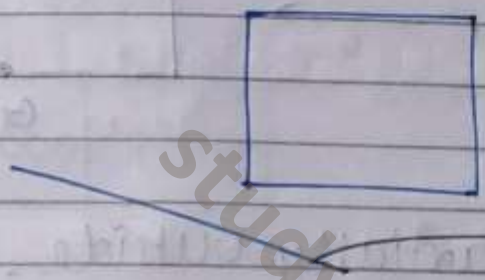
1 valid pt of intersection



This is t value b/w bottom and right edge. But since it is outside 0 and 1 → discard.

- Solve for t edge and t line
  If both lie in [0,1] then real intersection.
- Still need to test for lines ∥ to clip rectangular edges.

Still involves much calculations unnecessary.



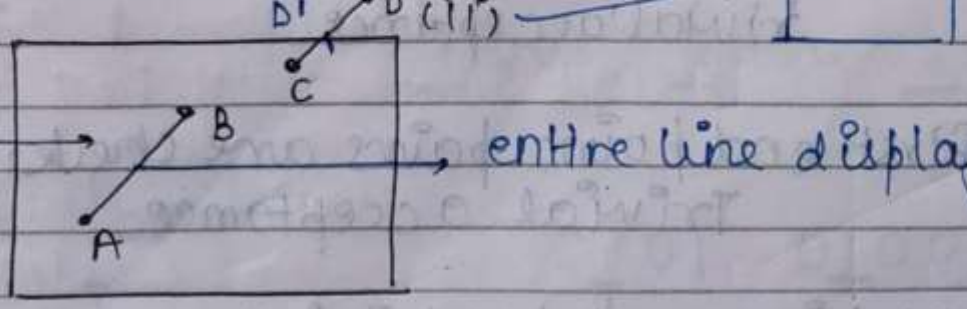
Both end pts. lie outside region but all values of t are calculated & then result is evaluated.

(2) Cohen Sutherland Algo (line)

(3) Sutherland Hodgeman Algo (polygon clipping).

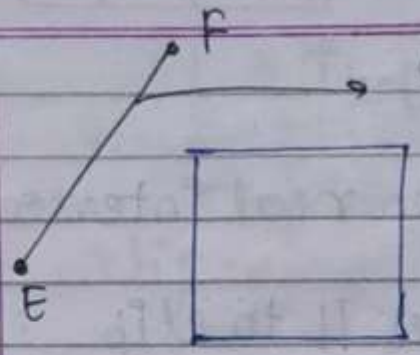
→ Line clipping.

cases to be considered :-



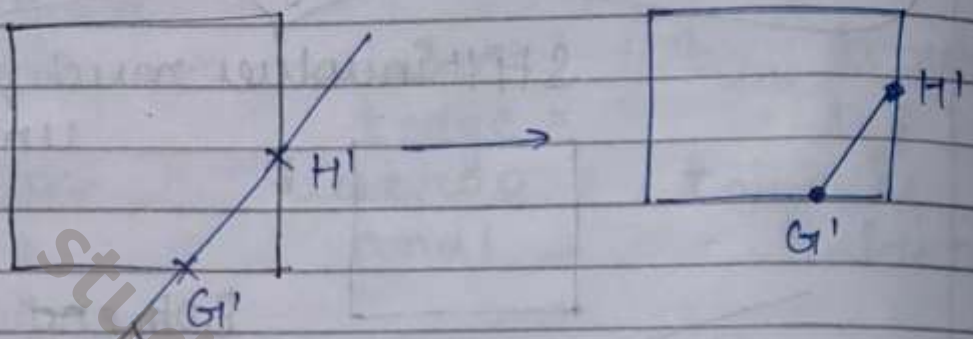
(i) → entire line displayed.

(iii)



→ whole line rejected.

(iv)



→ partially outside.

→ Cohen Sutherland algo

(1) performs initial tests on a line to determine whether intersections calculations can be avoided.

If initially we can predict whether line is

accepted — trivial acceptance

rejected.

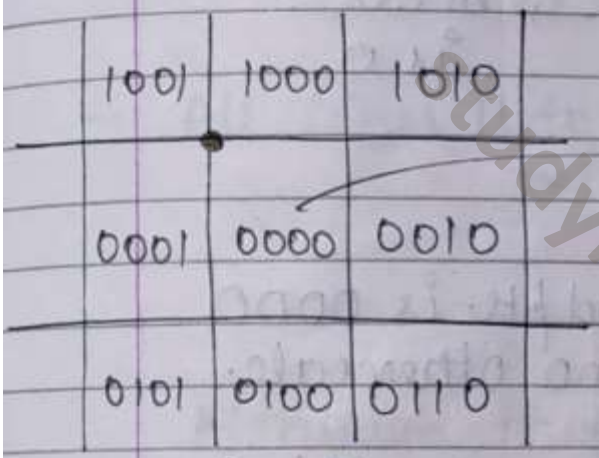
→ First end point pairs are checked for Trivial acceptance.

→ If line can't be trivially accepted, region checks are done for Trivial

rejection.

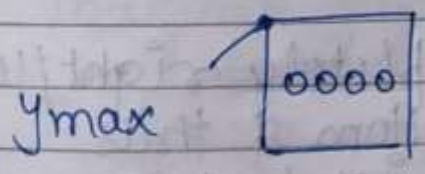
→ If line can neither trivially accepted or rejected, it is divided into 2 segments at a clip edge. (9 diff. regions)

We assign each region a region code / bit code
min 4 bits are req. to identify a region.



If line intersects at all bits 0000 then line is inside region.

0 bit is set
bit 1 → $y < y_{min}$; $y > y_{max}$
bit 2 : $x > x_{max}$
bit 3 : $x < x_{min}$.



Now $y > y_{max}$

1001 1000 1010

0101 0100 0110

0001

for bit 3
$x < x_{min}$ ✓

decided on basis of this whether line is trivially accepted/ rejected

$x \not> x_{max}$ → bit 2 becomes 0

$x \not< y_{min}$ → bit 1 will be set to 0

$x \not> y_{max}$ → bit 0 is 0.

Trivially accepted ↓

code of endpts. is 0000.
draw line & no other calc.

- A line is trivially accepted if outcodes of both endpts are zero

- A line is trivially rejected if bitwise AND of outcodes of endpts. is not zero.

either line is completely right/left/top/bottom & thus atleast there will be 1 common bit

(1001)

(0101)

totally towards left

1 bit common.

```
1001
0101
0001   → 1 bit set i.e non zero value.
```

```
1001
0100
0      → no common bit i.e zero value.
```

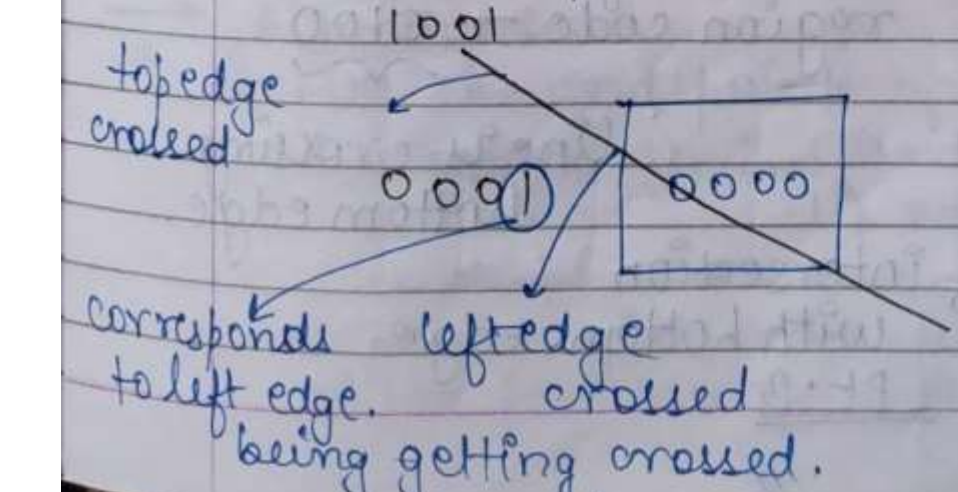
- All lines ∥ to clip window will be rejected.

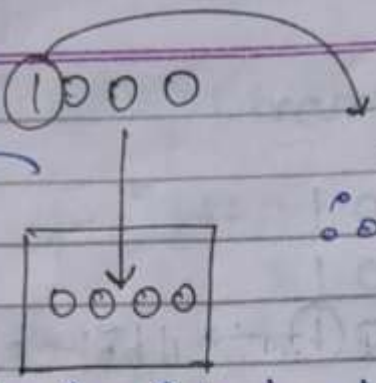
acc. to above logic.

Although there are certain cases, where line segments are rejected but after calculations & not directly trivially rejected.

Disadvantage.

- A key property of outcode is that bits that are set in non zero outcode corresponds to the edges crossed.





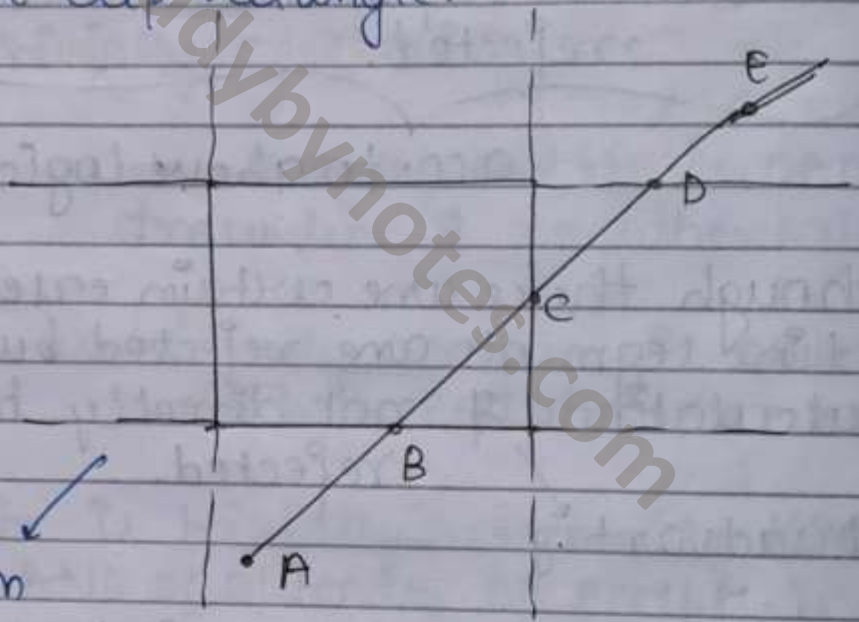
top edge
∴ intersects / crosses top edge.

only to check when a line comes from outside to inside for pt. of intersection

We don't check for inside → outside as bits will be 0000.

2 t's clip edge & not clip rectangle.

eg



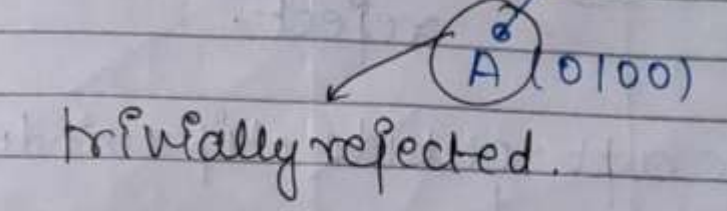
Start from a pt outside clip region

Start from A
region code → 0100

line is crossing bottom edge.

Find pt. of intersection of line with bottom edge
Pt. B

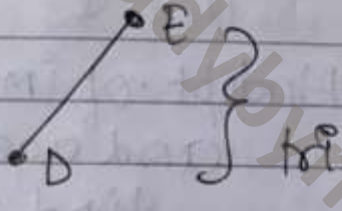
Now find region code of B. (0000)

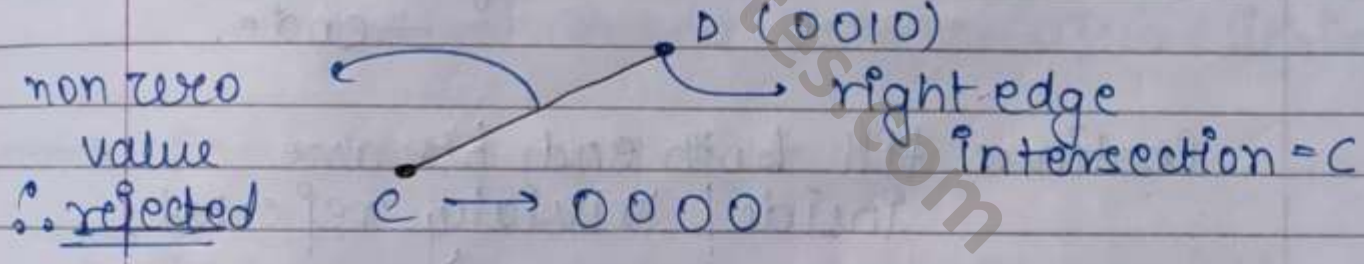


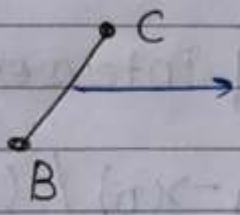
Now consider E → (1010)

crossing right or top edge.

We need to see for algo. whether checking is for top first or right







→ Algorithm :-

Input the end points of line & coordinates (xmin, ymin) and (xmax, ymax) of clip region.

Compute region codes for endpts.

check for trivial acceptance

→ (both codes = 0000)

If $code_1 \& code_2 \neq 0$

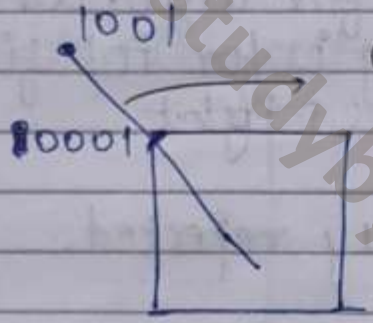
reject.

Take a pt. outside clip window.

↓

& code of pt. to top (1000)

1 will be there in AND only if the pt. crosses top.



crosses top edge.

Find pt. of intersection and again find region code.

Continue till both end pts. are inside / trivially rejected.

For finding pt. of intersection

Top → $x = x_0 + (x_1 - x_0) / (y_1 - y_0) \cdot (y_{max} - y_0)$

↳ We have y, find x.

Right → $y = y_0 + (y_1 - y_0) / (x_1 - x_0) * (x_{max} - x_0)$

Bottom → $x = x_0 + (x_1 - x_0) / (y_1 - y_0) \cdot (y_{min} - y_0)$

left → $y = y_0 + (y_1 - y_0) / (x_1 - x_0) * (x_{min} - x_0)$.

↓

We have x to find y

- Polygon clipping ↓

(1) apply Cohens Southerland on all line segments.

OR

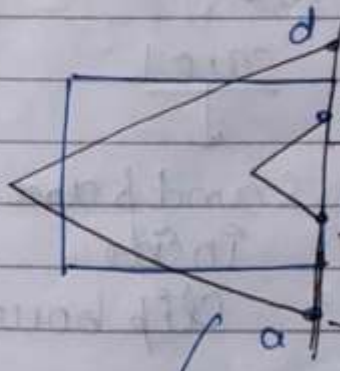
(2) Sutherland Hodgemen Algo.

↓

→ for a concave polygon, we will get more than 1 polygon after clipping.

→ for convex → only 1 polygon

- In this algo, we clip polygon against each edge seperately.



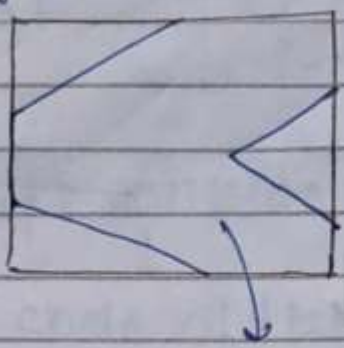
clipping by right boundary

polygon rejected on RHS.

addition of new vertices after right clipping

(a, b, c, d)

Clipping from bottom → left → top.

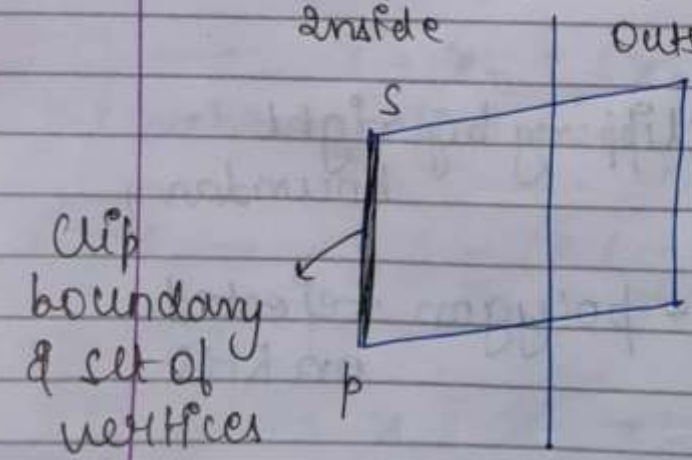


resultant fig.

(1) Initial set of vertices
(2) First passed through one clip boundary, and we remove all pts. outside boundary
(3) new vertices introduced at every stage
(4) again check against boundary.

Repeat it & final set of vertices to be plotted.

4 Cases of polygon filling ↓

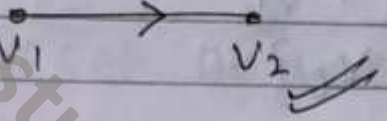


case 1
↓
s and p are inside clip boundary.

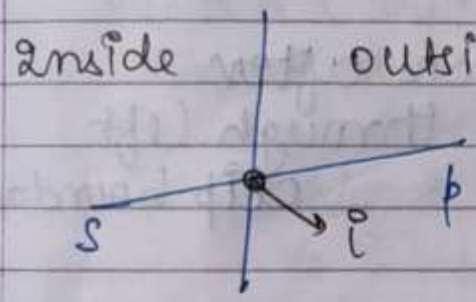
for top edge →

y of s and p must be < ymax.

Case 1 → If both vertices are inside clip boundary, then in O/P set we only include 2nd pt. / vertex & not first as polygon is closed so 1st pt. will be included afterwards.



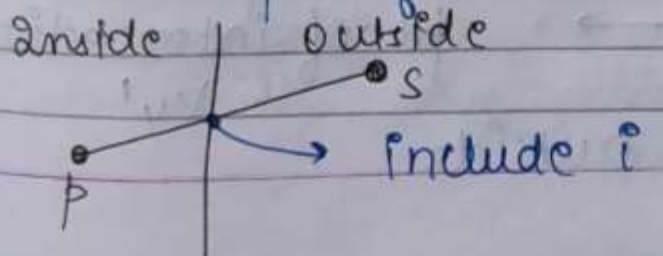
Case 2 → If out of 2, one is inside and other is outside, then O/P set of vertices will contain only pt. of intersection.



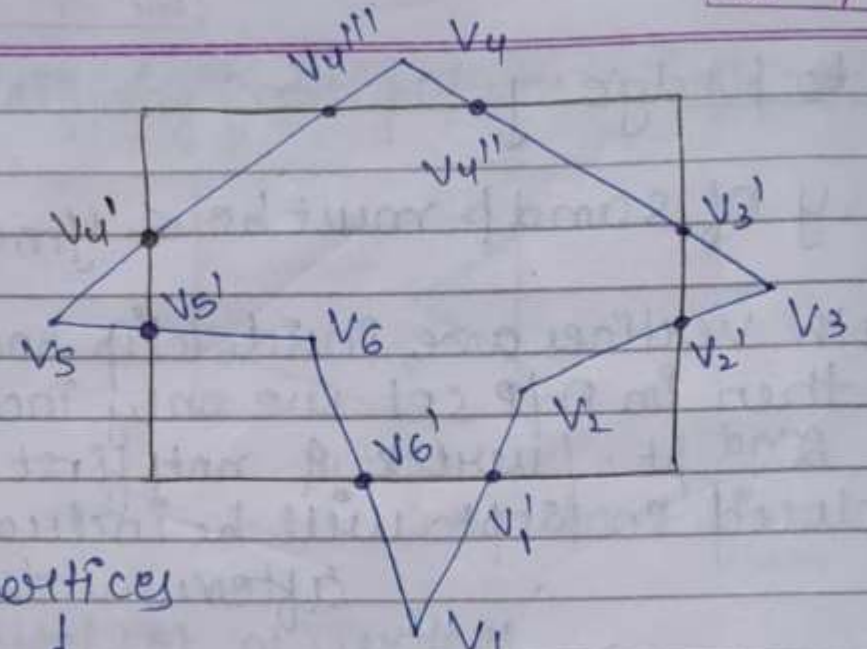
Case 3 - If both s & p are outside clip boundary, both are rejected in O/P set.

Case 4 → If 1st vertex is outside & 2nd is inside, then include 2 vertices in O/P set (Reverse of case 2)

pt. of intersection & 2nd vertex.



eg:-

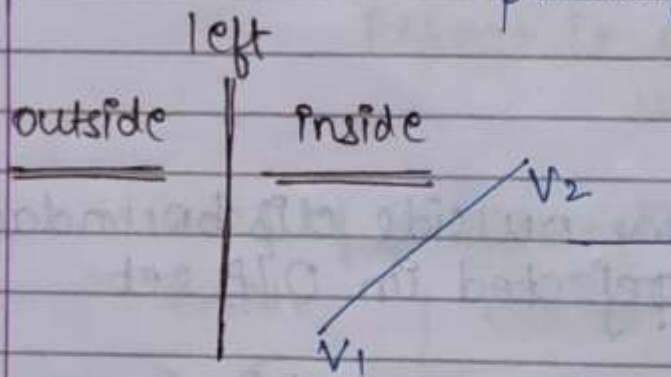


o/p set of vertices

$\{V_1, V_2, V_3, V_4, V_5, V_6\}$

Pass set of vertices to one of the clip boundary

→ left ↓ o/p set of vertices after passing through left clip boundary



→ both inside V_2 ≅

V_2 & V_3 → V_3 ≅

V_3 & V_4 → V_4 ≅

V_4 & V_5 → pt. of intersection V_4'

V_4 ↙ inside, V_5 ↘ outside

V_5 and V_6 → pt. of intersection + 2nd pt.
↳ inside
outside
V_5' and V_6.

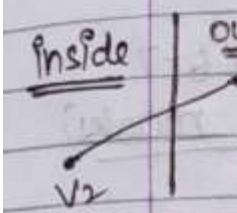
V_6 & V_1 → both inside (V_1)

not included in case 1

$V = \{ V_2, V_3, V_4, V_4', V_5', V_6, V_1 \}$.

↓

I/P set of vertices for 2nd boundary right



V_2 & V_3 → V_2' (pt. of intersection)
V_3 & V_4 → V_3' and V_4
V_4 and V_4' → V_4' (both inside)
V_4' & V_5' → V_5'
V_5' & V_6 → V_6
V_6 & V_1 → V_1
V_1 & V_2 → V_2

$\{ V_2', V_3', V_4, V_4', V_5', V_6, V_1, V_2 \}$

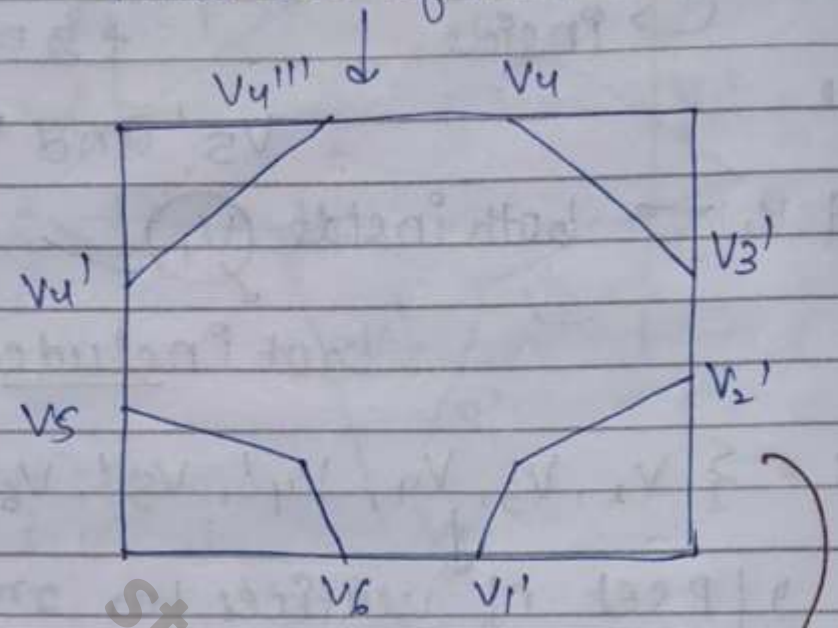
Now for top ↓

$\{ V_3', V_4'', V_4''', V_4', V_5', V_6, V_1, V_2, V_2' \}$

Now for bottom ↓

$\{ V_4'', V_4''', V_4', V_5', V_6, V_6', V_1', V_2, V_2', V_3' \}$

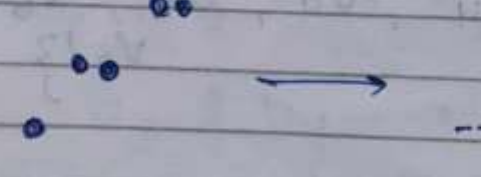
final set of vertices



Q (2m exam) either dry run algo + algo writing (6-7 marks)

11 Feb 19

Aliasing and Antialiasing.

- Aliasing - name given to jagged effect of lines.
- Jagged effect caused by finite set of pixels when change rows get step.
- Increase resolution steps (jaggies) get less steep.
-  increase in resolution.
- double resolution → quadruple memory, bandwidth, and scan conversion time.

CLASSTIME | Page No.
Date / /

Major drawback of Raster Graphics.
↓
Jagged lines

- The lines images is not smooth. This effect is Staircase problem or jagging.
This is called 'aliasing'.
↙
Sol^n to these problems is 'anti aliasing' (reduces defect)

{This defect can't be removed but only reduced}

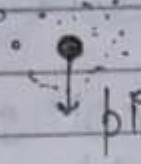
→ One simple sol^n → (1) Increase resolution so that many pixels will be plotted in a particular area.
But this actually doesn't reduce aliasing.

• 2 more techniques used are ↓ (based on area sampling)
Area sampling
Weighted / unweighted.

We see the pixel to be plotted covers how much area or how much it is close to the line.

- In this, we plot extra pixels but they are

plotted with variable intensity / reduced intensity, depending on how much area of pixel is being covered by line.



Sharpness of staircase effect / jaggies will increase through area sampling

- Unweighted Area sampling -
  considering a pixel, every part of it contributes to intensity of pixel.

→ If line covers whole of pixel, we plot that pixel with full intensity.

↓ every area of pixel contributes equally to intensity of pixel.

- Weighted Area Sampling -
  The area at centre covered will be plotted with more intensity
  eg - 2 pixels

more intensity ✓ ○ → covers 1/4 at centre

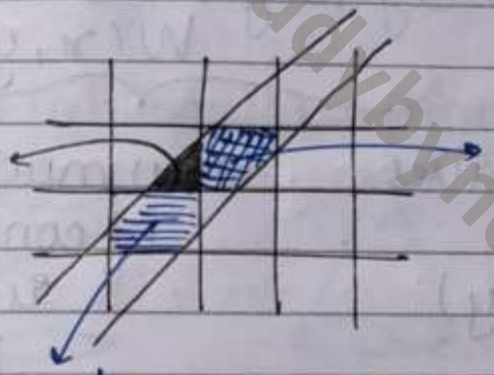
○ → covers 1/4 at side.

* Unweighted Area Sampling :-
  - Assume background white - lines black
  - Recogonize that primitive has non zero width.

(thinnest line - 1 pixel)

- Consider line as thin rectangle.
  covers diff. pixels to diff extent.
  squares.

- In most cases should not set a single pixel black.
  - Set intensity of pixel differently for each pixel covered.
  - Only horizontal & vertical lines effect only by 1 pixel per row.



area covered by a pixel.

plotted with less intensity, as area covered is less.

plotted with more intensity as area overlapped is more.

{If a pixel is away from line → less intensity.}

- works for non horizontal & vertical lines.

Geometry of pixels.
non overlapping squaretiles
lines contributes to intensity ∝ area of pixels.

Properties :-

(1) Intensity decreases with increasing dist. from pixel to edge.

(2) Primitives dont influence pixels that they dont intersect

(3) Equal areas of pixels contribute equal intensity.

Unweighted Area Sampling

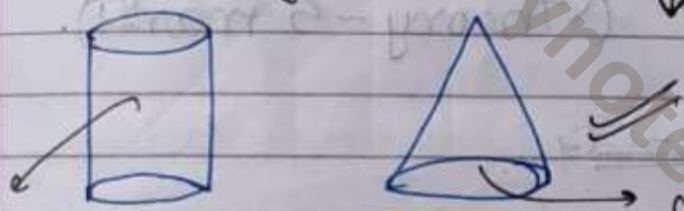
considering weighing function $W(x, y)$

- height at (x, y) gives weight for area dA at (x, y) how much amt. of contribution is given by a pixel.
- unweighted - graph of W is box i.e a cube i.e weight is constant.
- Thus, whatever part we may consider, we only see area covered & not which part is covered.

* Weighted Area Sampling
- equal areas may not contribute equally & every part also is not contributing equally.
- change 3rd property of Unweighted Area Sampling

(areas closer to pixel centre contribute more)

- Need to change "geometry" of pixel to preserve 2nd property. □□ → ◯◯
(pixel is circle larger than square and they are overlapping).
- If line intersects circle it contributes.
- Terms weighted & unweighted come from idea of weighing function.
- Weighing function → weighing func. is (in form of a cone).



each part contributes equally.

→ centre part contributes max. than sides

Weighing function W(x,y)
- choose simplest graph with height proportional to distance.
- graph of W is circular cone.
- choose radius as distance b/w pixel centres.

(Amount of area as well as which part is covered matters)

Consequences of weighting
↓
not necessary that equal area

pixels are plotted with equal intensity.

→ Gupta-Sproull Antialiasing

Whenever we want to plot a pixel, we plot its neighbouring pixels also but with variable intensity depending on how far the pixel is from line.

(calculating perpendicular distances)

Q (2m exam) Aliasing & antialiasing. (Theory - 5 marks).

Theory - CG

- Display (Video Display Device)
  - CRT (most commonly used)
  - Other popular display types
    - Liquid Crystal Display
    - Plasma Display
    - Field Emission Displays
    - LED's
    - 3D display devices

Cathode Ray Tube

(1) cathode rays emitted by electron gun.
(2) focussing & deflection.
(3) when electron beams contact screen phosphor emits light.
(4) light fades redraw req. in a small period (refresh)

all values of pixel before coming to CRT are in digital values.

→ $x_{coord.}$, $y_{coor.}$, intensity.

The intensity is fed as a voltage into CRT
Before feeding into CRT, digital values are converted to analogue.

→ Accelerating anode – focuses e^- in a single line.
and focussing anode. (↑)

→ $x_{co.}$, $y_{co.}$ → given as I/P to deflection plates. They deflect e^- beam or hit e^- beam to particular pixel position.

(deflection plates – horizontal & vertical).

→ When e^- atom hits phosphor screen, the energy is raised & is released, emitting light.
(energy released ∝ voltage).

→ But this light will be available for shorter period of time.

This is persistence.

Thus pixel needs to be refreshed again & again.

→ To ↑se Speed, we depends on frames/sec. have interlaced monitors.

→ Control grid → controles power of e⁻ fired.

• CRT Types

(1) DVST - Direct View Storage Tubes.

no need for refresh, pictures stored as permanent change on phosphor screen.

(2) Calligraphic refresh CRT - line drawing or vector random scan, need refreshing, image is stored in form of line drawing commands.

vector scan.

(3) Raster-scan (point by point refreshing)

CLASSTIME | Page No.
Date / /

• Color displays -

(1) Beam penetration method :
special phosphors emitted diff. colors of different intensity of e^-

(2) Shadow mask displays.

- Inside section of CRT is coated with red (outer layer) and green (inner layer) phosphors.

If e^- are slow they penetrate only outer layer thus giving red light.

If e^- moves fast, they give green light.

The e^- speeds can be adjusted in a way by combination of red & green. i.e yellow & orange.

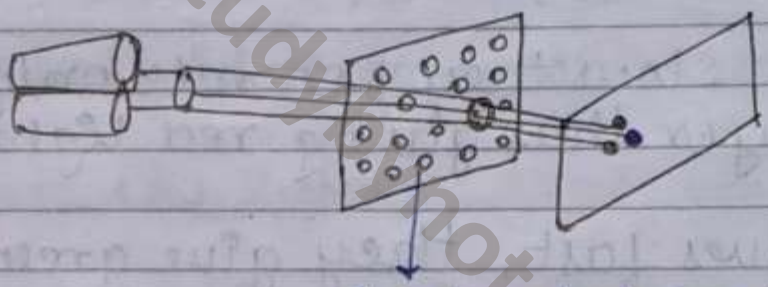
quality of image is diminished.

only 4 / limited colors available.
use in vector scan.

The screen has 3 phosphor colored dots which emits colors red, green & blue.

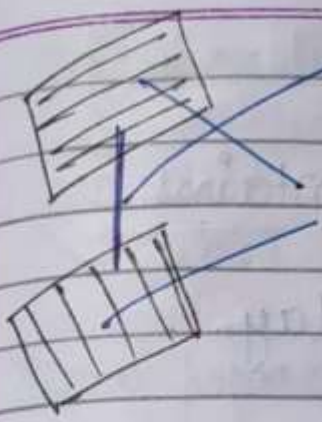
- We use 3 e^- guns for emitting colors red green, blue

- Thus voltage of e^- is such that colors appear in combination of rgb.
- For white color – all 3 beam hit with equal intensity.
- For black – all 3 beams with no intensity.
- For red – e^- gun for red is max, e^- gun for green & blue is 0.



perforated mask (shadow mask).

* LCD Displays :-
- No tube & e^- beams.
- A crystalline material is present which is known as "Nematic".
- Blocking / unblocking of light through polarized crystals.
- A matrix of LC cells one for each pixel.
- no refresh unless screen changes. Colors 3 cells per pixel.



contains nematic crystalline material.

polarizers are in ⊥ direction. To pass the light, we need to twist light.

role of crystalline molecules (nematic molecules).

used to glow a pixel position.

The pixels/pts. that don't twist their direction don't get plotted.

* Plasma Panel Displays :-

- use of neon gas b/w 2 plates. (sandwiching neon gas b/w 2 plates).
- Each plate is coated with conductive print. Print on one plate contains vertical conductive lines and other has horizontal conductive lines.
- Together 2 plates form a grid.
- Gas at intersection glows, creating a point of light / pixel.
- can be seen as collection of very small neon bulbs.

Q (in exam) I/P & O/P devices. / Working of CRT,

CLASS TEST DATE No.
Date / /

* Light Emitting Diodes

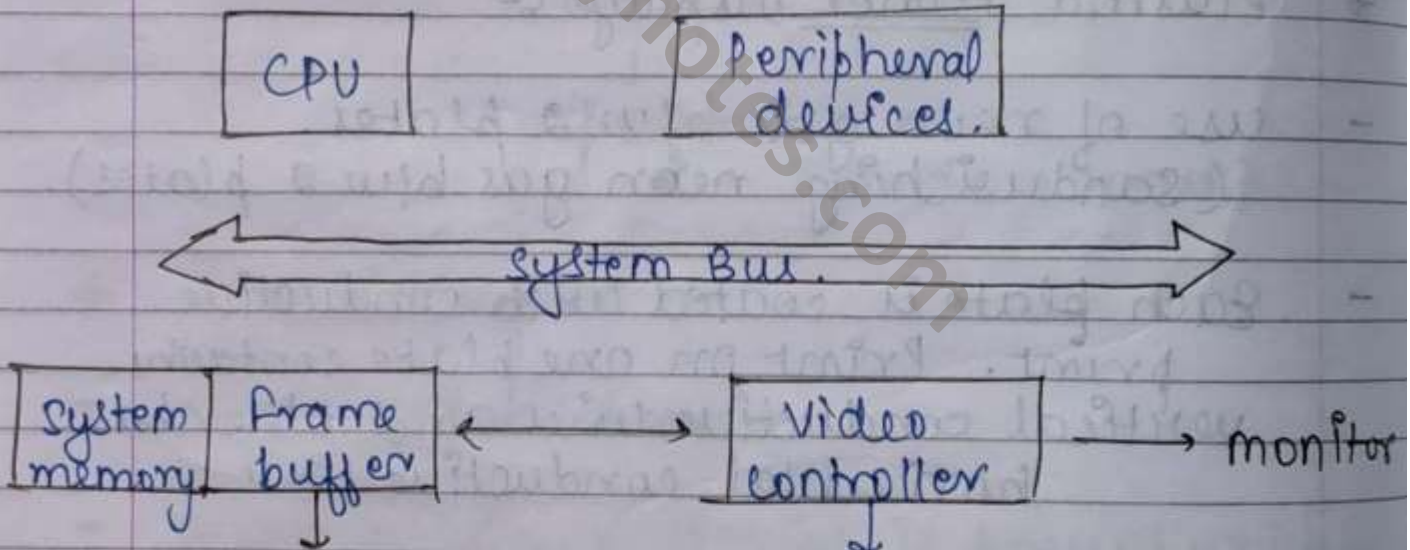
→ cross section of 2 plates contains LED's.

→ Similar str. as plasma display.

* Simple Raster Display Systems

frame buffer : Stored pixel map of screen.

Video controller just refreshes frame buffer on monitor periodically.



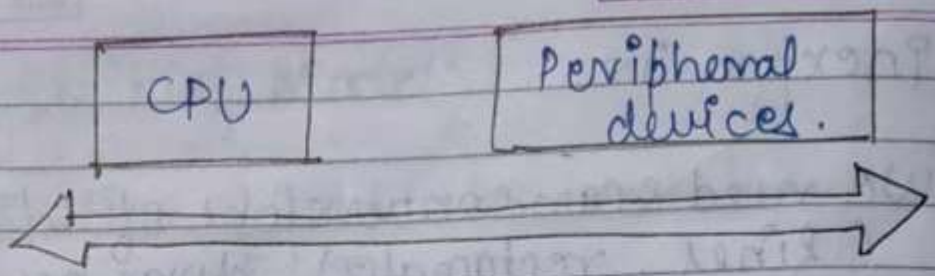
- Inexpensive
- We need scan conversions of O/P primitives (lines, rectangles) done by CPU.
- Slow process
- As refresh cycle ↑es, memory cycle used by video controller.

↓

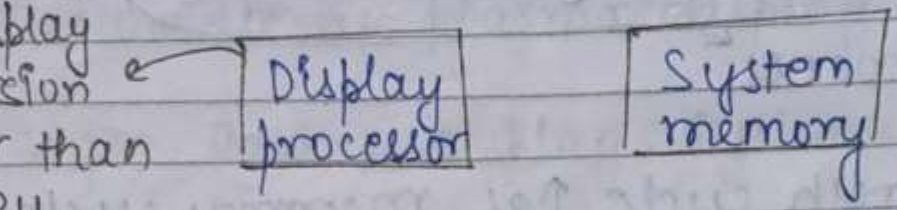
* Graphics Display processor → digitizes a picture defination.

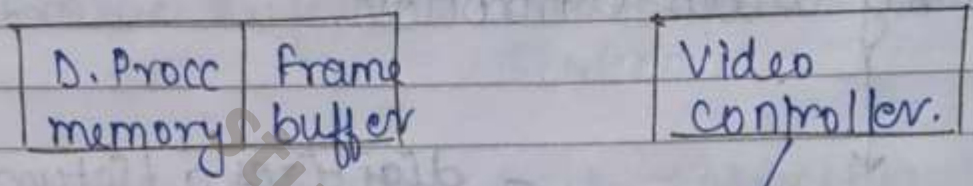
- A processor is added along with CPU which does all graphics work.
- System becomes faster.
- Purpose of display processor is to free CPU from graphics chores. A seperate display processor memory area can be provided.
- also called graphics coprocessor.

Co



(Scan) does display conversion rather than CPU.





doesn't directly access System memory

* CG softwares.
- Line drawing.
- Piecewise polynomial (spline) curves.
- Implicit surfaces

* Algos.

(1) Transformation → modifying a figure.
(2) Clipping
(3) Hidden surface removal → to display & identify all visible parts remove all hidden parts.
(4) Rasterization.

Included in Translation. { Scaling – enlarging a figure
Rotation – different orientation
Reflection – mirror image of figures.
3-D viewing / projections.

CLASSTIME | Page No.
Date / /

- Picking - Select 3D obj. by clicking an I/P device over a pixel location.
- Shading & illumination - show image with a particular effect
  (lighting/shading/material)
- Animation - movement by rendering sequence of frames.

Co