Q3 → Prove that successive 2D rotation are additive

$$R(\theta_1) R(\theta_2) = R(\theta_1 + \theta_2)$$

× – 2-D – rotation – ×

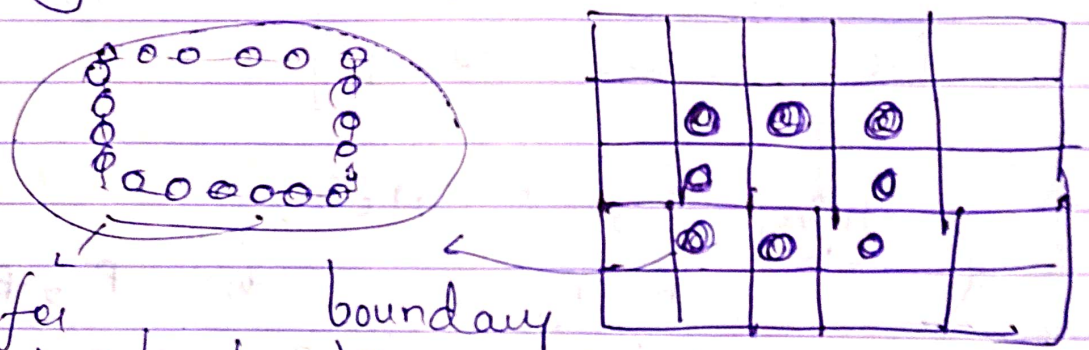
UNIT-II Transformation

Q4 → Prove that successive two translation operations are commutative

(3D + 3D transformation) → नहीं है।

Area Filling :-

① Boundary defined :- region pixels that mark the boundary or outline of the region have a unique color that is not same as the color of the interior pixels. Algorithm that are used to fill boundary-defined region are called boundary fill algorithms.



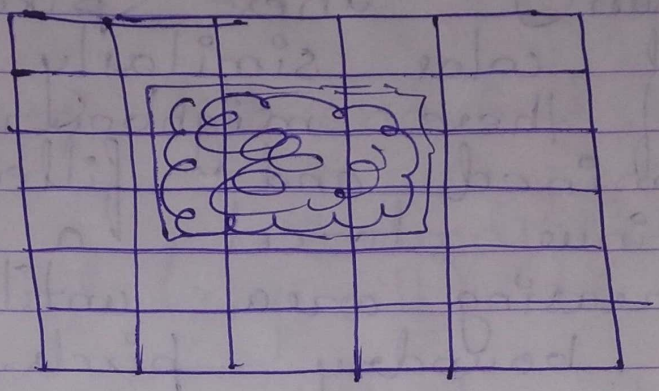
(Not for paper purpose)

② Interior defined :- region is a collection of same color continous pixels

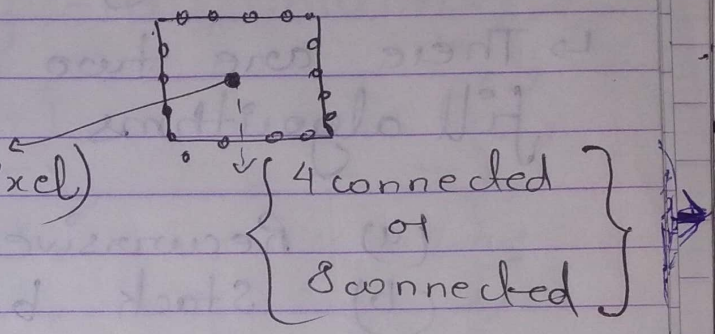
→ pixel exterior to region have different colours.

→ Algorithms that are used to fill interior defined region are called flood fill algorithm:

(recolouring के लिए)



Boundry Fill Algorithm :-



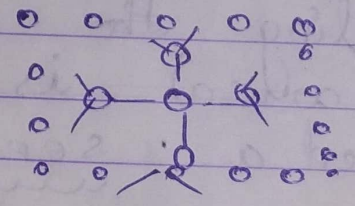
- The Approach to filling ~~the~~ a boundry defined region (area) is to start at a point called a seed inside the area and ~~point~~ paint the area progressively towards the boundry this method is called boundry fill algorithm.
- this procedure accept as input the coordinate of a sample pixel (called the seed pixel) from the area, a fill colour value and a boundary color value.

I/P → (seed pixel co-ordin, boundary color, fill color)

- The seed pixel is first set to fill color while its neighbouring pixels are examined for boundary color, if boundary color is not found than, these pixels are said to fill color similarly. neighbours. of these neighbouring pixels are examined and filled. Thus we continue to set a pixels in an increasing area until we encounter the boundry pixels.

↳ There are two approaches for boundry fill algorithms as follows :-

(a) Recursive approach.
(b) Stack based approach.



(a) Recursive Approach :-

↳ Following is the boundry fill algorithm using 8- connected fill method. It is recursive procedure with fill colour specified by "fill" and boundry color specified by "boundry".

<> Not equal)

Algorithm:-

Input: seed pixel (x, y), fill, boundry:
procedure:-

boundry-fill (x, y, fill, boundry);

if (Get pixel (x, y) <> fill and Get pixel (x, y) <> boundry) then

set pixel (x, y, fill)

boundry-fill (x+1, y, fill, boundy);
boundry-fill (x-1, y, fill, boundy);
boundry-fill (x, y+1, fill, boundy);
boundry-fill (x, y-1, fill, boundy);
boundry-fill (x+1, y+1, fill, boundy);
boundry-fill (x-1, y+1, fill, boundy);
boundry-fill (x-1, y-1, fill, boundy);
boundry-fill (x+1, y-1, fill, boundy);

end of process:-

b) Stack Based Approach:-

↳ Following is the boundry fill algorithm using 4-connected fill method.

↳ It is a stack based approach with fill color specified by "fill" and boundry color specified by "boundry"

Algorithm :-

↳ Input :- Seed Pixel (x,y), fill, boundry.

push pixel (x,y)

while (stack not Empty)

{

pop pixel (x,y)

set pixel (x, y, ~~color~~ fill)

if Getpixel (x+1, y) <> fill and Get pixel (x+1, y) <> boundary then.

{

push pixel (x+1, y).

}

if Getpixel (x-1, y) <> fill and Getpixel (x-1, y) <> boundry

then

push pixel (x-1, y).

if Getpixel (x, y+1) <> fill and Get pixel (x, y+1) <> boundary

then

push pixel (x, y+1).

If Get pixel (x, y-1) <> fill and get pixel (x, y-1) <> bounday.

then

push pixel (x, y-1);

end of while.

~~Interior color, flood fill~~ → recolor.

Flood Fill Algorithm

→ Algorithm used for filling interior defined regions are generally known as Flood Fill Algorithms.

→ Such an algorithm starts with a seed pixel by replacing its existing ~~color~~ color with fill color. Then using the 4 connected or 8 - connected chain, the algorithm sets fill color to other interior pixels.

★ → Assuming that all the pixels in ~~the~~ an interior defined region have same color the flood fill algorithm terminates when no more connected pixels have the old color.

(a) Recursive (b) Stack based

(a) **Recursive Approach** :- Following is the flood - fill algorithm using four connected fill method.

→ It is the recursive approach with old fill color specified by 'old color', 'new color' specified by 'fill color'.

Algorithm :-

→ Input :- Seed pixels (x, y), fill color, old - color

↳ Procedure :-

```
Flood Fill (x, y, fill_color, old_color)
if (Getpixel (x, y) == old_color) then
        set pixel (x, y, fill_color).
    Flood Fill (x+1, y, fill_color, old_color)
    Flood Fill (x, y+1, fill_color, old_color)
    Flood Fill (x-1, y, fill_color, old_color)
    Flood Fill (x, y-1, fill color, old_color)
end of procedure.
```

↳ 8 वाले के लिए लिखा लो) :- Stack वाला भी)

Clipping → point (simple)
→ line (" " , smid, 3 और है।)
→ polygon (~~hexa~~ 3, 4 और)

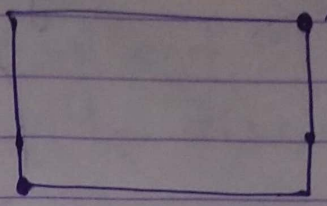
Point clipping :-

↳ point clipping
↳ line clipping
↳ polygon clipping
↳ curve clipping.
↳ Text clipping

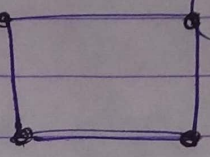
point clipping

The points are said to be interior to the clipping window if.

$x_{wmin} \leq x \leq x_{wmax}$ and
$y_{wmin} \leq y \leq y_{wmax}$



If window size is not given.
Unit size ⟶



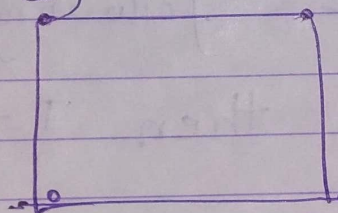
- the quad sign indicates that points on the window boundry are included within the window.

~~Q/1 A clipping window is given by :~~

Line clipping

Q→ ↳ A clipping window is given by A(10,10) B(20,10), C(20,20) and D(10,20) find the visible portion of a line P(5,5), Q,(50,25) inside the window.

Ans B(20,10) C(20,20) A(10,10) D(10,20)



P(5,5), Q(50,25)

↳ Eqn

$$(y-5) = \left(\frac{25-5}{50-5}\right)(x-5)$$

$$y-5 = \frac{20}{45}(x-5)$$

$$\left[y-5 = \frac{4}{9}(x-5)\right]$$

(A) Left edge window → $x = 10$

$$y - 5 = \frac{4}{9}(10 - 5)$$

$$\left(y - 5 = \frac{4 \times 5}{9}\right)$$

check करना ∴ (चारें checks करें)

↳ cohen - Sutherland - Line clipping algorithm

1001	1000	1010
0001	0000	0010
0101	0100	0110

4 bits

T B R L
↳ Left
↳ Right
↳ Bottom
↳ Top.

region code

i) outcode (line के).

↳ Assign the outcode for two end points of line using the following steps.

defining outcode

(i) if $y > w_{ymax}$ then T=1 else T=0

(ii) $y < w_{ymin}$ then B=1 else B=0

(iii) If $x > w_{xmax}$ then R=1 else R=0

(iv) If $x < w_{xmin}$ then L=1 else L=0

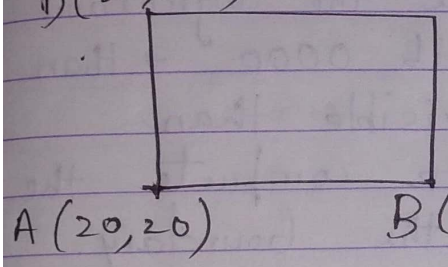
- If line के end points (0000) है तो line is completely visible.
- If outcode of line (both), and end में non zero है → line completly discarded.

Not lie if then partially होगी।)

- Q Given window A(20,20), B(60,20) C(60,50), D(20,50) use cohen sutherland algorithm to find visible portion of the lines

Line1 — P(30,35), Q(50,45) and A(30,55), B(65,75) inside the window.

Line2



1001	1000	1010
0001	0000	0010
0101	0100	0110

Ans →

Now Outcode for the end point of the line:-

TBRL

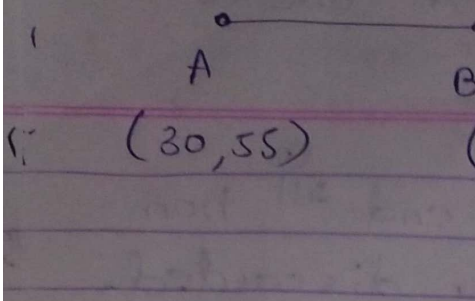
$$\begin{cases} x_{wmax} = 60 \\ x_{wmin} \Rightarrow 20 \end{cases} \qquad \begin{cases} y_{wmax} = 50 \\ y_{wmin} = 20 \end{cases}$$

(30,35) T B R L

Now P → 0 0 0 0 } → So line is completely visible no further investigation

(50,45) Q → 0 0 0 0 }

P Q



	T	B	R	L
A →	1	0	0	0
B →	1	0	1	0
	1	0	0	0

Now if Both are non zero than take logical add

If the logical add. is non zero than line is completely discarded

- If A → zero, B → non-zero → than compute the logical and than if ans is 0000 → than partially visible than we have to compute the intersection point with the boundary.

Q1) Use outcode based line clipping method to clip a line starting from (13, 5) and ending at (17, 11) against the window having its lower left corner at (-8, 4) and Upper right corner at (12, 8)

Q2) Using any of the line clipping algo. to obtain the visible portion of the following line segment.

(i) P_1 (0.5, 0.4) P_2 (1.6, 0.7)

(ii) P_1 (0.4, -1.6) P_2 (0.4, 2.7)

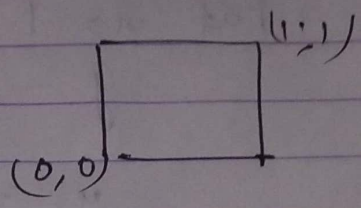
(iii) $P_1(0.4, -1.6)$ $P_2(0.8, -2.7)$

(iv) $P_1(-2, -3)$ $P_2(3, 5)$.

↳ Here, we have to assume

$Y_{wmin} = 1$ $\begin{cases} X_{w\,min} = 0 \\ X_{wmax} = 1 \end{cases}$

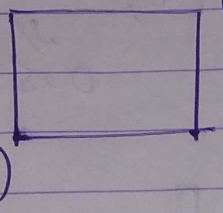
$Y_{wmax} = 1$



(How)

Given

Any (1) :-

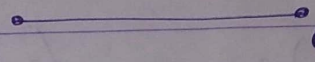


T B R L

P →

Q →

Line

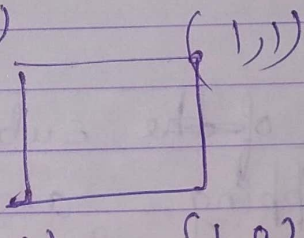


Ans 2(i) $P_1 \rightarrow 0\ 0\ 0\ 0$

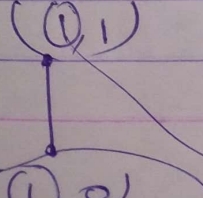
$P_2 \rightarrow 0\ 0\ 1\ 0$

And → $0\ 0\ 0\ 0$

Yes lie



Now Right side से intersect करेंगी।



Line की eqn निकालो P1 & P2 से

Put $x = 1$ then get y.

P_2 eq^n of line ; $y - 0.4 = \frac{0.3}{1.1}(x - 0.5)$

$$y - 0.4 = \frac{3}{11}(x - 0.5)$$

Put $x = 1$

$$y - 0.4 = \frac{3}{11}(0.5)$$

$$y = \frac{0.4}{1.0} + \frac{1.5}{110} \Rightarrow \frac{44 + 15}{110} \Rightarrow \frac{59}{110}$$

$$y \Rightarrow 0.53.$$

point of intersection $(1, 0.54)$ // 0.53.

Some Question → Group photo

*Cyrus - Brek. line Clipping algorithm :- *

(दूसरी वाली में window rectangle होती है।)

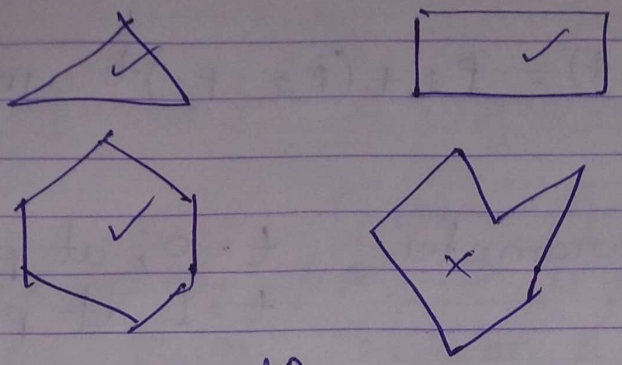
Advantage → Work on rectangular as well as non-rectangle window (eg; Δ)

Points :-

- Mid point of the subdivision and cohen sutherland line clipping assume that the clipping window is a regular rectangle.
- These algorithm are a not applicable for non rectangular clipping window
  → prev algo में point थे।

Now u Cyrus - Brek line clipping in points :-

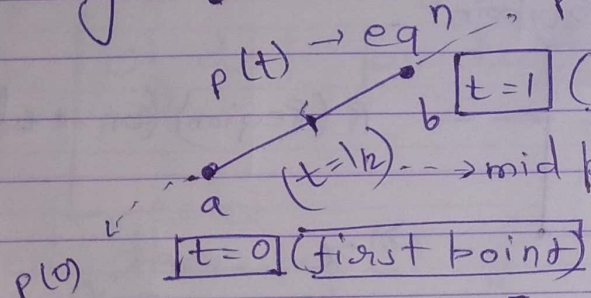
↳ Cyrus and Brek have developed a generalised line clipping algorithm which is applicable to arbitary convex region.



✓ → convex (accepted)

✗ → concave (rejected).

- This algorithm uses a parametic eqⁿ of a line segment to find the intersection points of a line with the clipping edge. ⇒



$P_1 = a$

$P_2 = b$

eqⁿ ← $[P(t) = P_1 + (P_2 - P_1)t]$

$P(t) = a + (b-a)t$

At $t=0$ ↳ $(P(0) = a]$ → first point

At $t=1$ $P(t) = a + b - a = b$ (last point)

At $t = 1/2$ $P(1/2) = a + \frac{b-a}{2} = \frac{b+a}{2}$

midpoint.

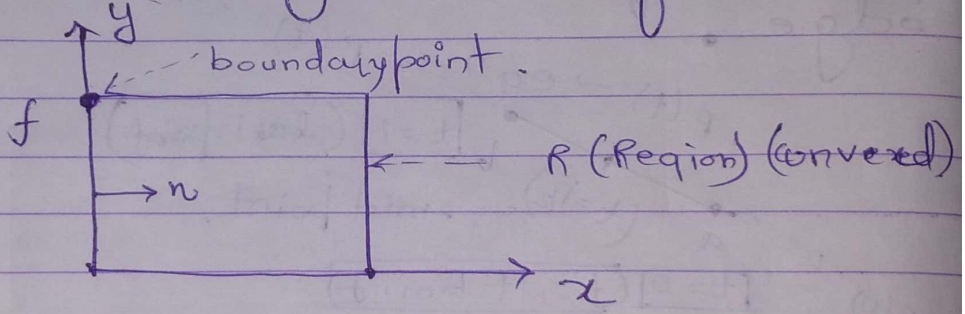
- by varying value of t we will get the points of lines.

- The parametric equation of line segment from P_1 to P_2 is P:

$$P(t) = P_1 + (P_2 - P_1)t \quad \text{where } 0 \le t \le 1$$

where t is a parameter $t = 0$, at P_1.
$t = 1$ at P_2.

- Consider a convex clipping region R, f is a boundry point of a boundary region R and n is an inner normal for one of its boundaryes.

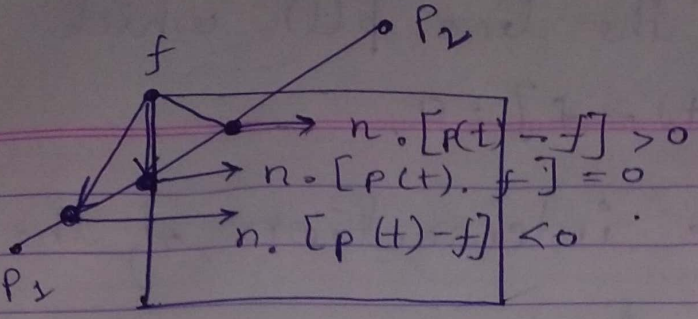


. इसमें reference point होंगे जो हर boundary को cover करेगा।

. (और हर boundary के लिए एक normal होगा।)

- then we can distinguish in which region a point lie by looking at a value of a dot product.

$$n \cdot [P(t) - f]$$ as shown in figure below:-



- $n.[p(t)-f]$
  - $p(t)$ → reference point of line data structure vector.
  - f → vector of edge direction.

- If dot product is negative i.e.,

$$n.[p(t)-f]<0$$

① then the vector $p(t)-f$ is pointed away from the interior region R.

② If dot product is zero i.e.,

$$n.[p(t)-f]=0$$

then $p(t)-f$ is pointed parallel to the plane contained f and perpendicular to the normal.

③ If dot product is +ve i.e.,

$$n.[p(t)-f]>0.$$

then ~~then~~ vector $p(t)-f$ is pointed towards the interior R.

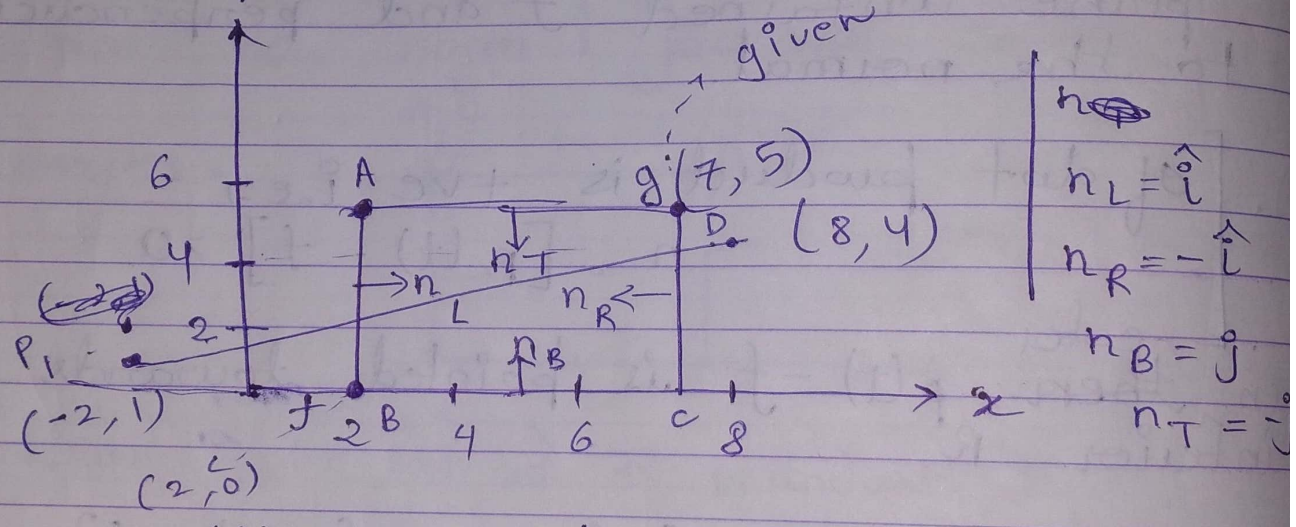
(for intersection point find :- $n.[p(t)-f]=0$
find $t=?$
then put in eqn get.)

- If the point f lies in boundary plane or edge for which n is an inner normal

then that point on the ... satisfies $n \cdot [p(t) - f] = 0$.

Condition is the intersection of the line with the boundy edge.

Q1 Consider the line from $P_1(-2, 1)$ to $P_2(8, 4)$ clipped to the rectangular region as shown in below. the line P_1, P_2 intersect the window calculate the intersection point.



$$P(t) = P_1 + (P_2 - P_1)t$$

$$P(t) = [-2, 1] + [10, 3]\,t$$

$(8 - (-2)) \quad (4 - 1)$

$$P(t) = (10t - 2)\hat{i} - 2\hat{i} + 1\hat{j} + 10t\hat{i} + 3t\hat{j}$$

$$P(t) = (10t - 2)\,\hat{i} + (3t + 1)\hat{j}$$

$\hookrightarrow 0 \le t \le 1$

Let take refrence $pt = (2, 0) \rightarrow f$

$$P(t) - f \Rightarrow (10t - 2 - 2)\hat{i} + (3t + 1 + 0)\hat{j}$$

$$P(t) - f = (10t - 4)\hat{i} + (3t + 1)\hat{j}$$

Now, $n_L \cdot [P(t) - f] = 0$

$$\hat{i} \cdot [(10t - 4)\hat{i} + (3t + 1)\hat{j}] = 10t - 4$$

$$\Rightarrow 10t - 4 = 0$$

$$\left(t = \frac{4}{10} = \frac{2}{5}\right)$$

$$[t \Rightarrow 0.4]$$

Put $t =$ in $P(t)$ 2

$$P(t) = \left(10 \times \frac{2}{5} - 2\right)\hat{i} + \left(3 \times \frac{2}{5} + 1\right)\hat{j}$$

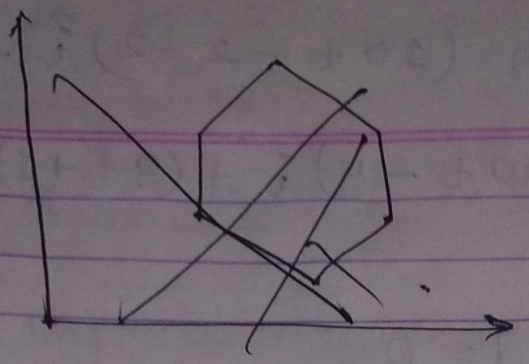
$$P(t) = 2\hat{i} + \frac{11}{5}\hat{j}$$

Point $\left(2, \frac{11}{5}\right) = (2, 2.2)$

★

- Now for f refrence point intersection point.
  - on line AB & BC
- Now for g refrence point intersection point
  - on line AD & DC.

?



↳ Q4

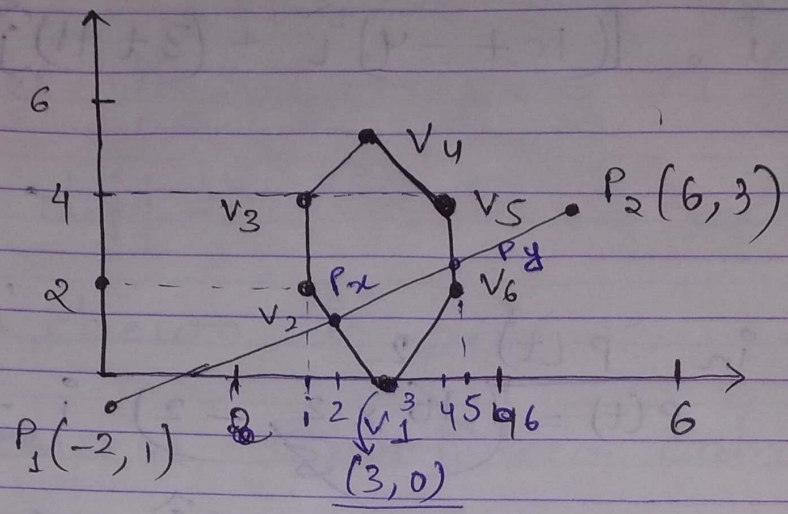


Figure above shown the hexagonal clipping window the line $P_1(-2,1)$ to $P_2(6,3)$ is to be clipped to this window.

- Find the intersection points.

Ans → $i = 1, \ldots 6$

$n_i [P(t) - f]$

Normal vector :-

→ $(V_1, V_2)(i, j)$ (x increased, y increase) $(i + j)$

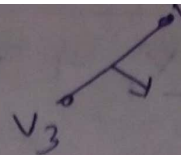
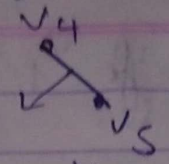
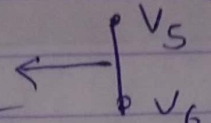
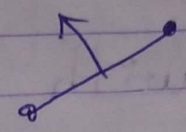
V_1 — V_2

→ $V_3 \rightarrow V_2$ $(1, 0)$ → i (constant)

Reference point

→ $(3, 0)$

→ $(1, 4)$

- V_3, V_4 $(1, -1)$  $\rightarrow (i - j)$ $(1, 4)$
- $V_4\ V_5$ $(-1, -1)$ ..  $(-i - j)$ $(5, 4)$
- $V_5\ V_6$ $(-1, 0)$  $(0 - i)$ $(5, 4)$
  Constant
- $V_6\ V_1$ $(-1, 1)$  $(-i, j)$ $(3, 0)$

Solve

Ans $\rightarrow$ $\begin{pmatrix} (1.2, 1.8) \rightarrow P_x \\ (5, 2.75) \rightarrow P_y \end{pmatrix}$

- **Liang - Barsky Line Clipping algorithms:**
  - more efficient than → Cyrus - Brick line.
    ↳ work same on parametric eqⁿ of line.

$$\left.\begin{aligned} x &= x_1 + t\Delta x \\ y &= y_1 + t\Delta y \end{aligned}\right\} (0 \le t \le 1)$$

where $\begin{pmatrix} \Delta x = x_2 - x_1 \\ \Delta y = y_2 - y_1 \end{pmatrix}$

- point clipping condition for Liang Barsky approach in the parametric for ~~are~~ are as follows:-

$$x_{wmin} \le x_1 + t\Delta x \le x_{wmax}$$

$$y_{wmin} \le y_1 + t\Delta y \le y_{wmax}$$

- Liang Barsky express these ~~following~~ four inequalities with two parameter p and q as follows:-

$$t_{pi} \le q_i \quad \text{for } i = 1, 2, 3, 4$$

where parameter p & q are defined as

#formula learn it

$p_1 \Rightarrow -\Delta x \quad q_1 \Rightarrow x_1 - x_{wmin}$

$p_2 \Rightarrow \Delta x \quad , q_2 = x_{wmax} - x_1$

$p_3 \Rightarrow -\Delta y \quad , q_3 \Rightarrow y_1 - y_{wmin}$

$p_4 \Rightarrow \Delta y \quad , q_4 \Rightarrow y_{wmax} - y_1$

- Following observation can be easily made from above definitions of parameter p & q

→ if $p_1 = 0$, line is parallel to left clipping boundry.

→ if $p_2 = 0$, line is parallel to right clipping boundry.

→ if $p_3 = 0$, Line is parallel to bottom clipping boundary.

→ if $p_4 = 0$, Line is parallel to top clipping boundry.

→ if $p_i = 0$, Line is parallel to one of clipping boundry corresponding to the value of i.

if $q_i < 0$ line is completely outside the boundry and can be eliminated.

if $q_i \geq 0$ line is inside the clipping boundary.

(यहाँ बार बार calculate होगा है।)

if $p_i < 0$, line proceeds from outside to inside of the clipping boundary.

if $p_i > 0$, Line proceed from inside to outside of the clipping boundary.

इसी को case में हमे check करना पड़ेगा। (line की)

- Therefore, for non zero value of p_i the line crosses the clipping boundry and we have find parameter t.
- The parameter t for any clipping boundry i can be given as :-

$$t = \frac{q_i}{p_i}$$ ——→ as studied earlier $t_{pi} \leq q_i$

$(i = 1, 2, 3, 4)$

↳ Liang Barsky calculate two value of parameter of 't' (t_1 and t_2) that define a part of line that lies within the clipped rectangle.

↳ the value of t_1 is determined by checking the rectangle edges for which the line proceed from the outside to inside. (i.e $P<0$).

↳ the value of t_1 is taken as the largest value amongst various value of intersection with all edges

↳ On the other hand the value of t_2 is determined by checking the edges for which the line process from inside to the outside.

↳ ($P>0$) the minimum of calculated value is taken as a value of t_2.

↳ If $t_1 > t_2$ the line is completely outside the clipping window and can be rejected.

↳ Otherwise the value of t_1 and t_2 are substituted in the parametric equation to get the end point of the clipped line.

Liang Barsky → procedure आएगा → !! △ !!

Q → $P_1(10, 10)$ $P_2(60, 30)$

Window given :- $(x_{wmin}, y_{wmin}) = (15, 15)$

$(x_{wmax}, y_{wmax}) = (25, 25)$

Ans → $p_1 = -\Delta x = -50 \rightarrow (60 - 10)$

$q_1 = x_1 - x_{wmin} \Rightarrow (10 - 15) = -5$

$t_1 = \frac{q_1}{p_1} = \frac{-5}{-50} = 0.1$

$p_2 = \Delta x = (60 - 10) = 50$

$q_2 \Rightarrow x_{wmax} - x_1 \rightarrow (25 - 10) = 15$

$t_2 = \frac{q_2}{p_2} = \frac{15}{50} = 0.3$

$p_3 = -\Delta y = -(30 - 10) = -20$

$q_3 \Rightarrow y_1 - y_{1wmin} \Rightarrow -5$

$t_3 \Rightarrow 0.25$

$p_4 \Rightarrow 20$

$q_4 = 15$ → $t_4 = 0.75$

Now where $p_i < 0$ → p_1 & p_3 → go to

$t_1 = \max(0.1, 0.25) = 0.25$

($0.1 \rightarrow t_1$, $0.25 \rightarrow t_3$)

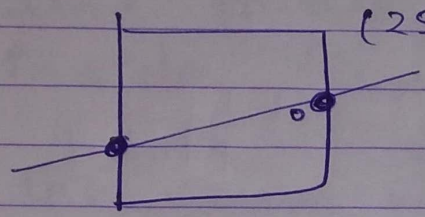
Now where, $P > 0$ → (P_2, P_4) → $t_2 = \min(0.3, 0.75)$

$t_2 = 0.3$

Now last condⁿ check:-

$t_1 > t_2$ } → false

So clipping now points



$XX_1 = X_1 + t_1 \Delta x$

$= 10 + .25 \times 50$

$= 22.5$

$YY_1 \Rightarrow Y_1 + t_1 \Delta y$

$= 10 + .25 \times 20 = 15$

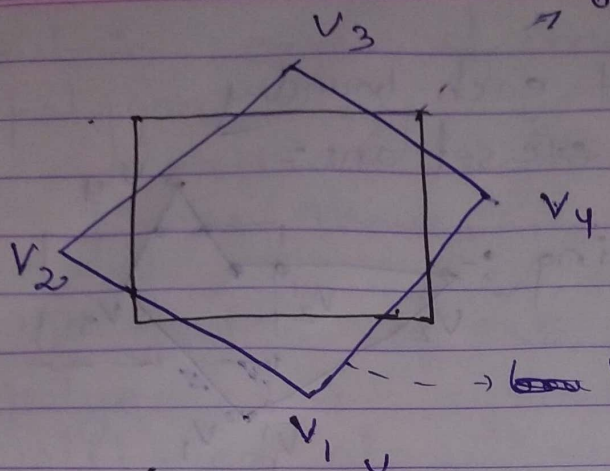
$XX_2 \Rightarrow X_1 + t_2 \Delta x$

$= 10 + .3 \times 30 = 25$

$YY_2 \Rightarrow Y_1 + t_2 \Delta y$.

$= 10 + .3 \times 20 = 16$.

Sutheland - Hodgeman polygon clipping algorithm :-

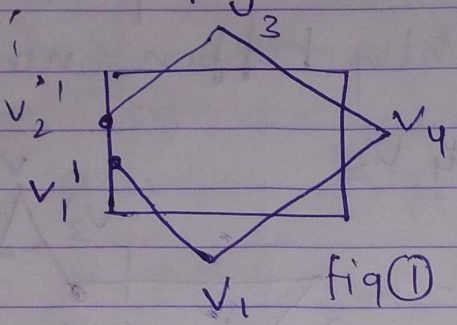


→ boundry

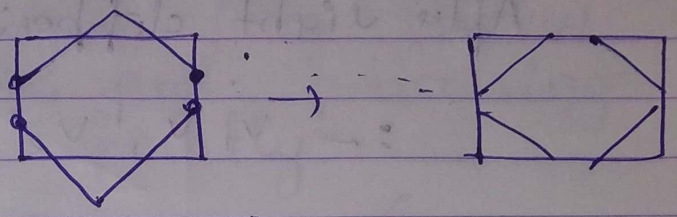
→ polygon

Procedure, left, right, bottom, top boundary.

- Left window ($V_1, V_2, V_3 V_4$) and now clipping new vertex set fig① is passed to right window → fig②

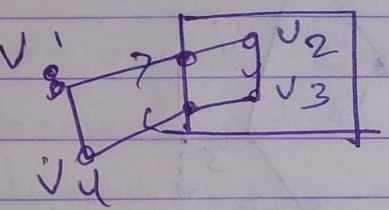


fig①



- newly generated vertex will passed to next window clipper.

Ex:

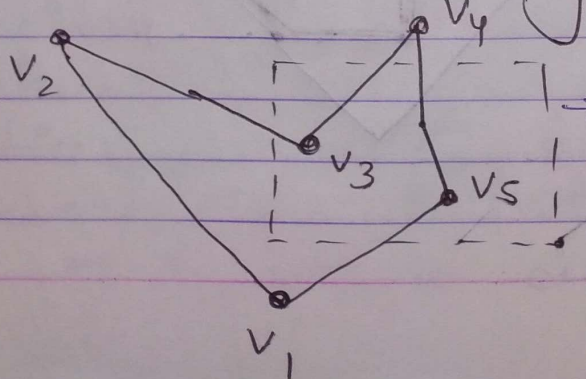


out to in :-

① save → intersection, inside vertex.

Numerical (Question → procedure + Question (note what to do))

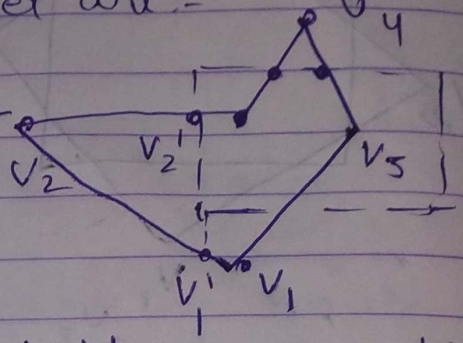
- For a polygon and clipping algorithm window shown in fig. below; give the list of vertices after each boundry clipping.



Any original polygon vertices are:
v_1, v_2, v_3, v_4, v_5

After clipping against each boundy new vertex ~~ste~~ set are:-

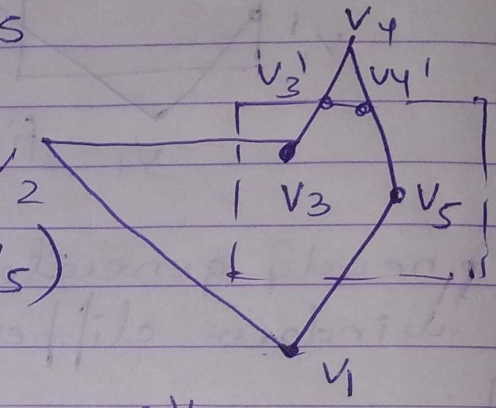
↳ After left clipping :-



$v_1\ v_1'\ v_2'\ v_3\ v_4\ v_5$

After right clipper:- (Nothing happen no cutting)

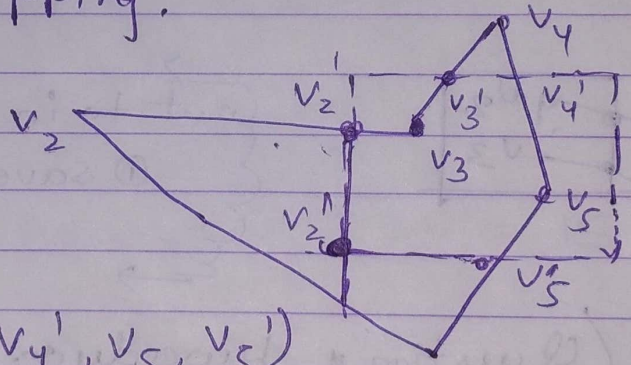
:- $v_1\ v_1'\ v_2'\ v_3\ v_4\ v_5$



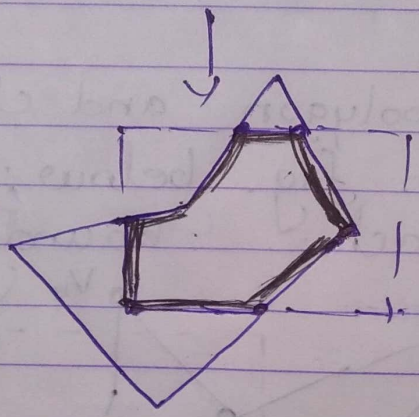
- After top clipping :-

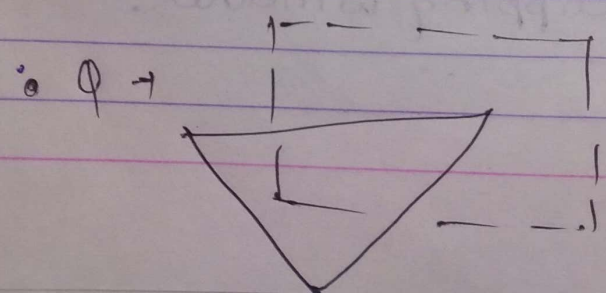
$(v_1\ v_1'\ v_2'\ v_3\ v_3'\ v_4'\ v_5)$

After bottom clipping.



$(v_2'', v_2', v_3, v_3', v_4', v_5, v_5')$





→ The sutherland- Hooyeman polygon clipping algorithm clips convex polygon correctly but in case of concave polygons, clipped polygon may be displayed with extraneous lines as shown in figure above.

→ The problem of extraneous lines for concave polygons in sutherland - Hodeiman polyon clipping algorithm can be solved by seperating concave polygon into two or more convex polygons and processing each convex polygon seperately:-

Weiler & Atherton polygon clipping:- { particular problem solve करती }

(o in to out जैसे होगा then go clockwise or follow the polygon..)

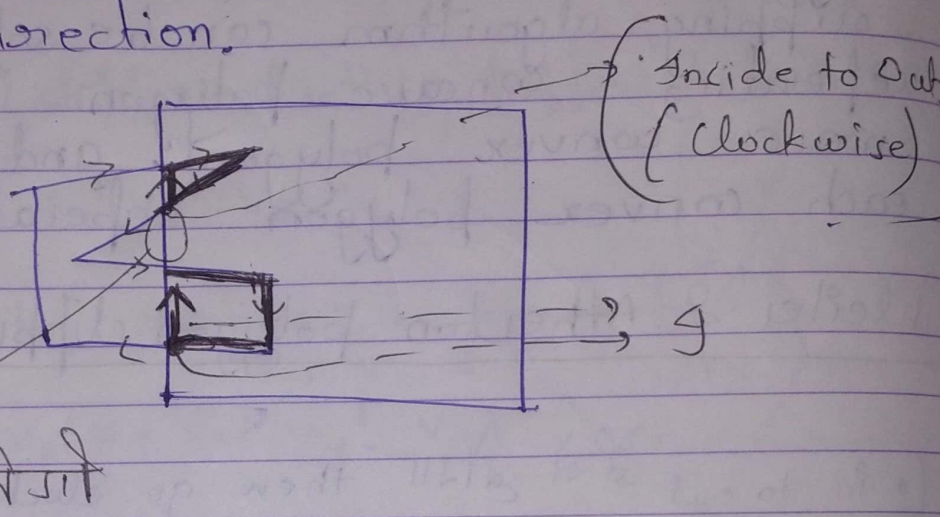
↳ The vertex processing procedure for window boundries are modified so that concave polygon are displayed correctly.

↳ The basic idea in this algorithm is that instead of always proceeding around polygon edges as vertices are process. we sometimes want to follow the window boundries; which path we follow depends on the polygon processing direction (clockwise or counter clockwise) and whether the polygon vertices correctly being processed represents an outside to inside pair or inside to outside pair for.

- For clockwise processing of polygon vertices we use the following rules:-

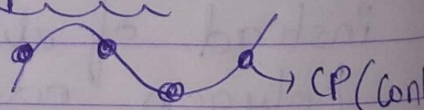
↳ For an outside to inside pair of vertices follow the polygon boundry.

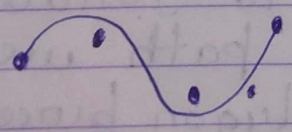
↳ For an inside to outside pair of vertices follow the window boundry in a clockwise direction.



Curve :- Interpolation & Approximation

- Spline representation :- • piece of curve.
- control point से curve पास हो तो Interpolation.



↳  ⇒ curve ⇒ Interpolate at start & end point and approx at middle CPs.

- CP are used to control the shape of curve.

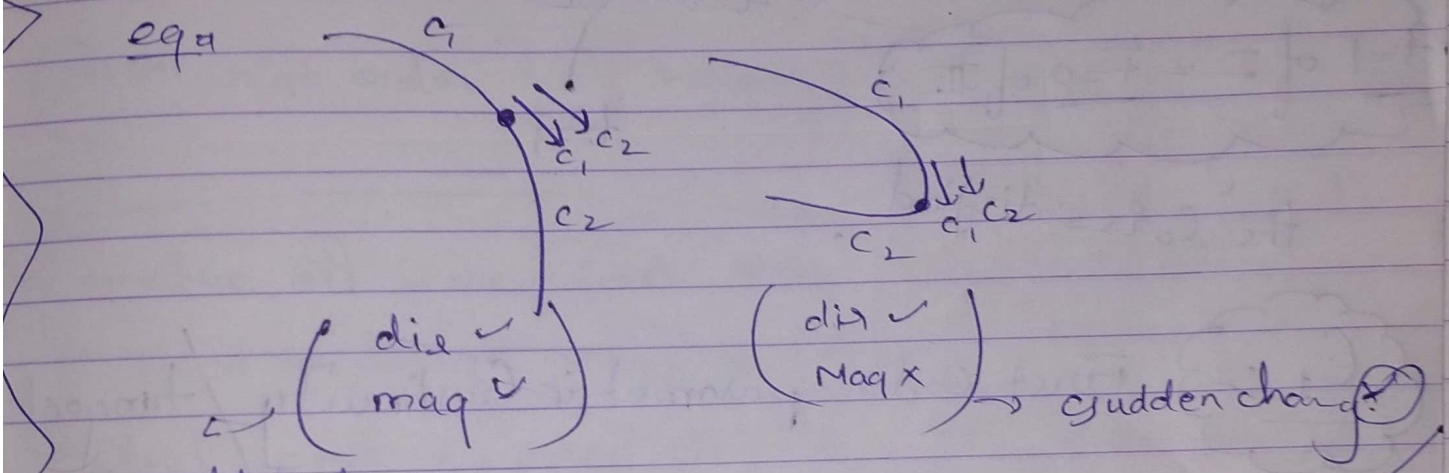
• तो अब Joint point will help for joining the two curves.

→ यहां पर continuity का concept आएगा।

• जितना derivate निकाल पाएंगे is point पर, - उतनी smooth होगी।

* Continuity * → ① Parametric continuity
→ ② Geometric

① direction same माँगता है } → that means
• magnitude " " " " " " } sudden change नहीं होगा।



② In second we only check direction

Parametic Continuity (C)

zero-order parametric continuity
→ C_0

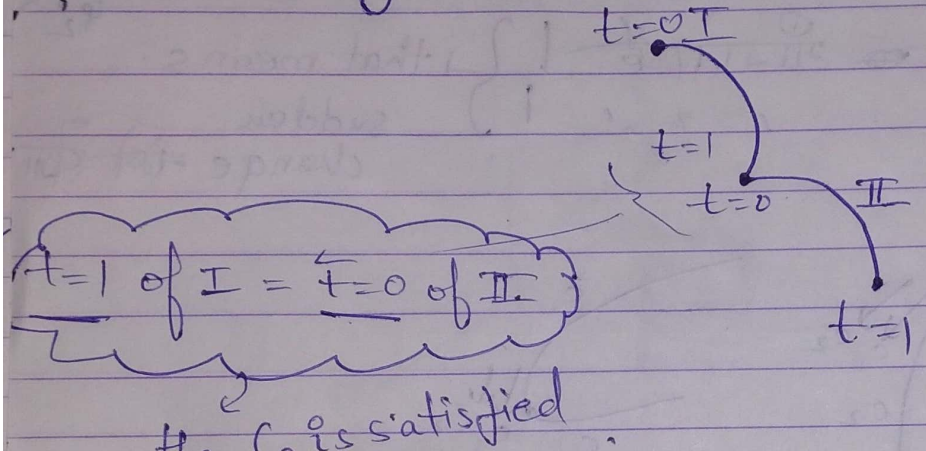
→ C_1 (First order parametric continuity).

C_2

C_0 :- ⊛ point / position continuity.

⊛ zero order parametric continuity means simply that curves meet.

⊛ This kind of continuity exists when two curves ~~shapes~~ shares a common end point without regard to the shape of curve at the joining point.

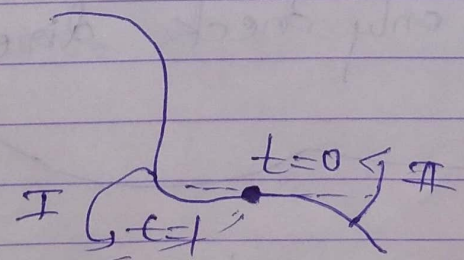


the C_0 is satisfied.

C_1 :- First Order parametric Continuity / tangent continuity

or

C^1 (both same)



2 ⊙ 1st order

~~derivative~~

• If the tangent vector ~~of~~ of two connected curve segment are equal (i.e; their direction & magnitude are equal) at the joining point than the curve has first order parameter continuity i.e (C_1)

$\frac{d}{d()}$ of $t=1$ of I $=$ $\frac{d}{d()}$ of $t=0$ of II

(2) :-

Second - order parametric continuity

- curvature continuity.
- When 1st and 2nd ~~parametric~~ order parameter derivates of curves section are equal at joining point called second order curve parametrics continuity.

$\left(\frac{d^2}{d()} = \rho \right.$ curvature)

(जितना high order उतना smooth.)

Geometric भी same just mag. different होगा।

- Used to design complex shape.
- Civil work (speed बनाने में parametric).

Two Curves :-

★

Bezier Curve :-

(*) (If n+1 Control Point then degree of curve = n)

$0 \le t \le 1$ ∴

- 4 Blended fun (सभी का योग 1)

$t = 0 \rightarrow$ 1st fun. $\rightarrow$ 100% (1)
↳ other 2, 3, 4 $\rightarrow$ 0

$t = 1 \rightarrow$ 4th fun $\rightarrow$ 100% (1)
↳ (1, 2, 3 $\rightarrow$ 0)

and $0 < t < 1$
↳ 1, 2, 3, 4
↳ partially सब contibute करेंगे।)

Points :-

(*) Given a set of n+1 control points P_0, P_1, P_2 upto some P_n a parametric Bezier curve segment, that will fit to those points is defined by

$$P(t) = \sum_{i=0}^{n} P_i B_{i,n}(t) \quad (0 \le t \le 1)$$

↳ (1)]

P_i → Control, $B_{i,n}(t)$ → Blending function.

where $B_{i,n}(t)$ are the Bezier Blending functions also known as Bernstien basis functions defined as follows :-

$$B_{i,n}(t) = C(n,i)\, t^i (1-t)^{n-i} \quad \text{(2)}$$

$$C(n,i) = \frac{n!}{i!\,(n-i)!} = {}^nC_i$$

from eq^{n} (1)

$$P(t) = P_0 B_{0,n}(t) + P_1 B_{1,n}(t) + P_2 B_{2,n}(t) + \cdots + P_n B_{n,n}(t) \quad \text{(III)}$$

(1) → recursive function ⇒ degree 3 Bezier curve, ⇒ तो 3+1 → control point

eg $\sum_{i=0}^{3} B_{0,3}, B_{1,3}, B_{2,3}, B_{3,3}$ — 4 func.

on degree के लिए exact तो नहीं निकाल सकते ⇒ तो we will ⊗ cubic → matrix → then directly by multiplying by C.P control points then we will get Bezier curve points.

Now Put (2) in III :-

$$P(t) = P_0 \{C(n,0) \cdot t^0 (1-t)^n\} + P_1 \{C(n,1) \cdot t^1 (1-t)^{n-1}\} + \cdots$$

((• The degree of Bazier Curve segment is one less than the number of control Points.)) used

• The Bezier curve in general passes through the first control point P_0 and last control point P_n as @ $P(t=0) = P_0$

and $P(t=1) = P_1$

↳ the Bezier curve of degree $3^{\nearrow n}$ $(n=3)$, We find ~~four~~ ~~the~~ that four $^{\nearrow n+3}$ control points are required to specify a cubic bezier curve segment Hence $P(t) = \sum_{i=0}^{3} P_i B_{i,3}(t)$, $0 \le t \le 1$

uCOP

$$P(t) = P_0 B_{0,3}(t) + P_1 B_{1,3}(t) + P_2 B_{2,3}(t) + P_3 B_{3,3}(t) \quad - (i)$$

where

- $B_{0,3}(t) = C(n,0) \cdot t^0 (1-t)^n \rightarrow (1-t)^n$ ($^n C_0 = 1$)
- $B_{1,3}(t) = \left\{ \frac{3}{1\ 2} \right\} t^1 (1-t)^2 = 3t(1-t)^2$
- $B_{2,3}(t) = {}^3C_2\, t^2 (1-t) = 3t^2(1-t)$
- $B_{3,3}(t) = {}^3C_3\, t^3 (1-t)^0 = t^3$

Computing Blending function.

we can clearly see $\nearrow$ if $t=0$ then $B_{0,3}(t)=1$

if $t=1$, then $B_{3,3}(t)=1$

Now put in (i) :-

$$P(t) = (1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t^2(1-t)P_2 + t^3 P_3$$

↳ eqⁿ of cubic Bezire Curve.

now in matrix form to represent this eqⁿ.

$$P(t) = (-t^3 + 3t^2 - 3t + 1)P_0 + (3t^3 - 6t^2 + 3t)P_1 + (-3t^3 + 3t^2)P_2 + t^3 P_3.$$

↓ simplifying

$P(t) = [F][B]$ ---- → Two parts splits.

$[T][M][B]$

M = coefficient denot
t = parametrice
B → P_0, P_1 .. coeffi

$$B = \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}$$

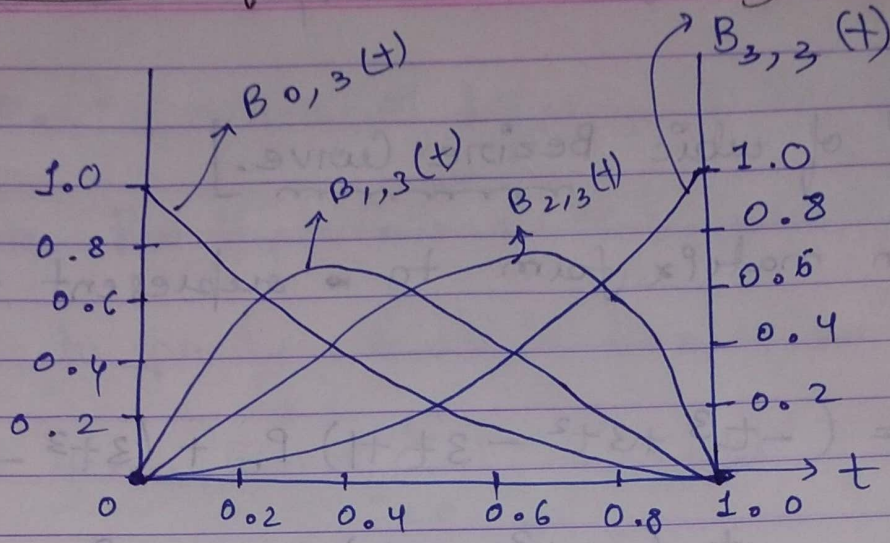
$\therefore -t^3 + 3t^2 - 3t + 1$

$$M = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

$T = [t^3, t^2, t, 1]$.

Matrix representation.

How to Plot :- Plots of Bezier blending function :-



- At $t=0$, $B_{0,3}(t) = 1$, All other '0'.
  At $t=1$, $B_{3,3}(t) = 1$, All other '0'' }

from $0 < t < 1$ } All B (func^n) have some value partially.

- so all will contibute some value in $0<t<1$.

↳ none of the Bezier function is negative anywhere in $(0 \le t \le 1)$.

↳ the sum of ~~blending function~~ value of all blending function i.e.,

$$\sum_{i=0}^{3} B_{i,3}(t) \text{ is unity everywhere}$$

★

- It can be proved that for any Bezier blending function.

$B_{i,n}(t)$ the maximum value occurs at $t = i/n$

for $B_{0,3}(t)$ = max at $t = 0/3 = 0$ at $t=0$

$B_{1,3}(t)$ = max at $1/3 = 0.33$

$B_{2,3}(t) = \max$ at $\frac{2}{3} = \frac{0.66}{t}$, a

$B_{3,3}(t) = \max = \frac{3}{3} = t = 1$

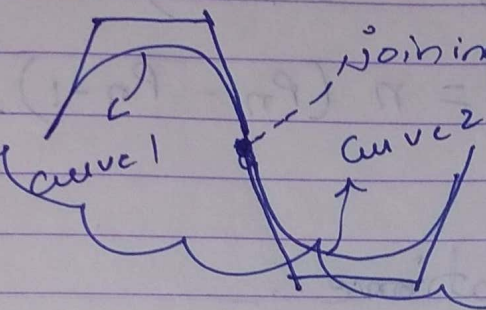
ideq :-

Tangent $\Rightarrow$ $\frac{d}{d}$ (~~B(t)~~ eqn) = 0 (eq = $E'(t)$)

Put $E'(t=0)$ $E' = (t=1)$

1st eqn 2nd eqn

Use → given two curves then join both curve smoothly.



Tangent On Beizele Curve :-

$P(t) = (-t^3 + 3t^2 - 3t + 1) P_0 + (3t^3 - 6t^2 + 3t) P_1 +$

$(-3t^3 + 3t^2) P_2 + (t^3) P_3.$

Find $P'(t)$:-

$P'(t) = (-3t^2 + 6t - 3) P_0 + (9t^2 - 12t + 3) P_1$

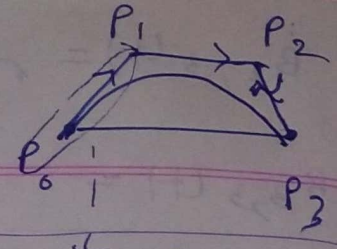
$+ (-9t^2 + 6t) P_2 + (3t^2) P_3$ —(a)

two points ~~पर~~ tangents $t = 0$ & $t = 1$.

Put $t=0$ in (a) stating point :-

$$P'(0) = (-3)\,P_0 + 3P_1$$

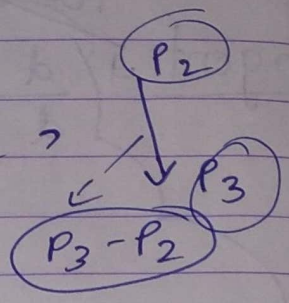
$$P'(0) \Rightarrow 3\,(P_1 - P_0)$$



$P_1 - P_0$

$$P'(1) = -3P_2 + 3P_3$$

$$P'(1) = 3(P_3 - P_2)$$



Now for n-degree curve:-

n = degree

$$\begin{cases} P'(0) = n\,(P_1 - P_0) \\ P'(1) = n\,(P_n - P_{n-1}) \end{cases} \rightarrow \text{Generalise}$$

↳ Now :- 2nd Derivative :-

$$\begin{cases} P''(0) = 6\,(P_0 - 2P_1 + P_2) \\ P''(1) = 6\,(P_3 - 2P_2 + P_1) \end{cases}$$

~~double~~ ← for cubic

↳ General form :- (n-degree)

$$\begin{cases} P''(0) = (n)(n-1)(P_0 - 2P_1 + P_2) \\ \qquad \Rightarrow n(n-1)\{(P_2 - P_1) - (P_1 - P_0)\} \\ P''(1) = n(n-1)\{P_n - 2P_{n-1} + P_{n-2}\} \\ \Rightarrow n(n-1)\{(P_{n-2} - P_{n-1}) - (P_{n-1} - P_n)\} \end{cases}$$