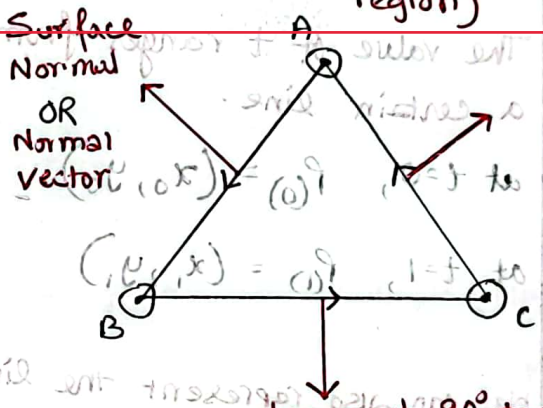
LINE CLIPPING ALGORITHMS:

CYRUS-BECK

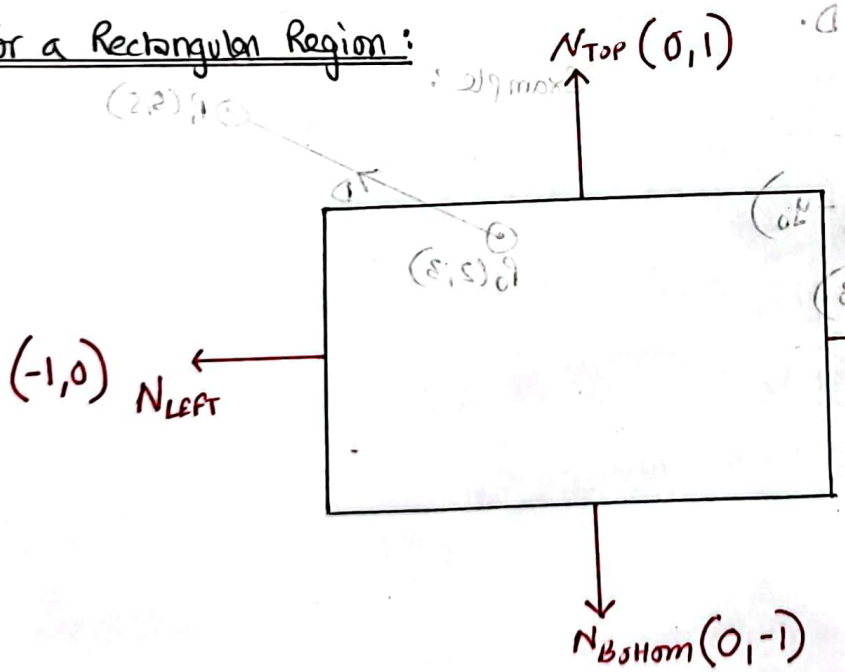
LECTURE 8

- The problem with Cohen-Sutherland was that the clipping region needed to be axis parallel. Which meant that we wouldn't be able to work on anything other than a regular rectangular clipping region.
- But with Cyrus-Beck we could potentially work on any polygon clipping region with n-sides, where $(n \geq 3)$. (Although our main focus will still be on a rectangular clipping region)



(*) Considering the direction of the edge, anything on the left is inside & right is outside.

For a Rectangular Region:



	Values	
N_T	(0,1)	Top
N_B	(0,-1)	Bottom
N_L	(-1,0)	Left
N_R	(1,0)	Right.

(*) N represents a perpendicular unit vector to each of the region.

Note: In Cohen-Sutherland we clipped the line based on the Outcode. Here we will do so based on the value of t.

Parametric equation of a line:

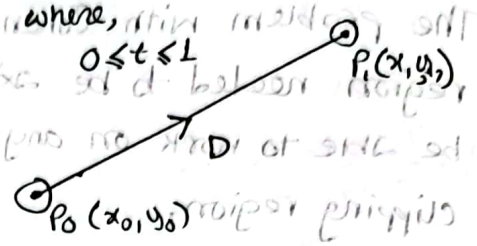
$$P_{(t)} = P_0 + t(P_1 - P_0)$$

$$\Rightarrow P_{(t)} = (x_0, y_0) + t(x_1 - x_0, y_1 - y_0)$$

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

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

where, $0 \le t \le 1$



0 <= t <= 1

t = 0, P(0) = (x0, y0) starting point

t = 1, P(1) = (x1, y1) ending point

The value of t ranges from 0 to 1 for a certain line.

at $t=0$, $P_{(0)} = (x_0, y_0)$

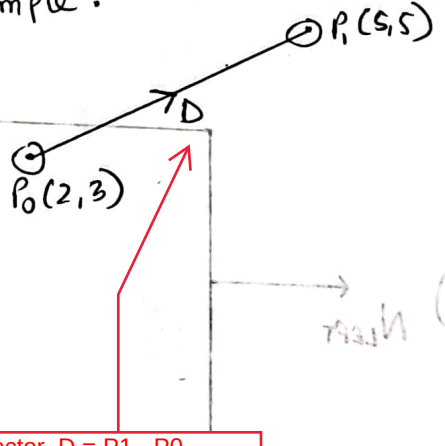
at $t=1$, $P_{(1)} = (x_1, y_1)$

We can also represent the line P_0 to P_1 using a vector D.

$$D = P_1 - P_0$$

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

Example:



$$\Rightarrow D = (5-2, 5-3)$$

$$\Rightarrow D = (3, 2)$$

OR

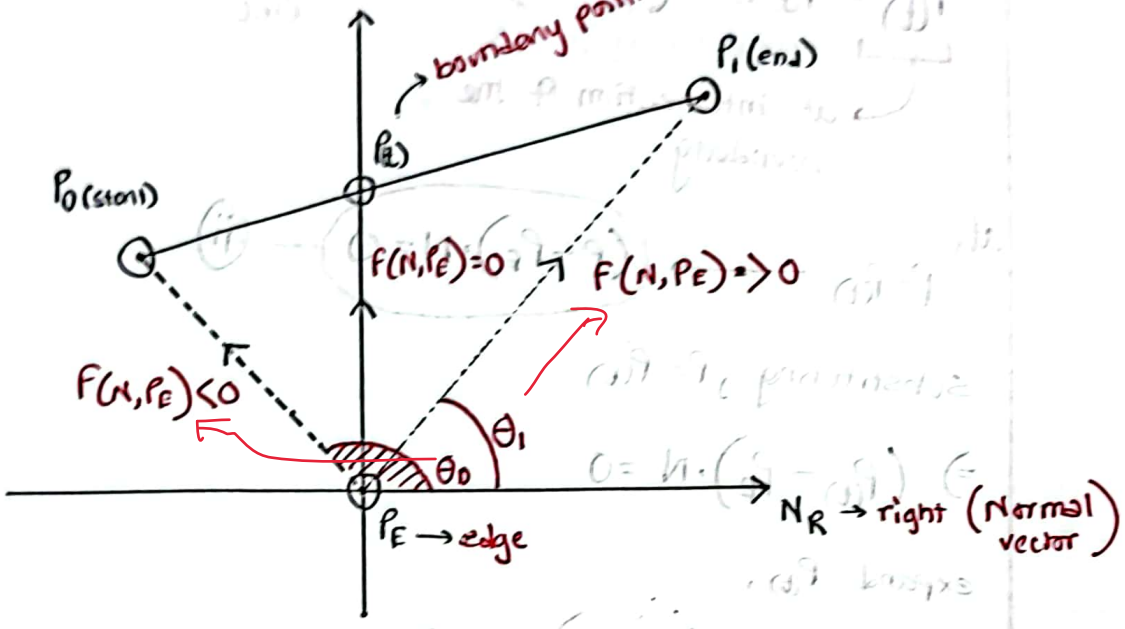
$$\Rightarrow D = 3i + 2j$$

vector, D = P1 - P0

$F(N, P_E) = N \cdot (P - P_E)$ → dot product.

Equation for intersection

θ = Angle between the vectors $\&\ N_R$



So, $N \cdot (P - P_E)$ has 3 possible values; +ve → leaving the frame

↓ dot product between two vectors can be represented with "cos" of the angle between the vectors.

−ve → entering the frame

0 → on the intersection/boundary.

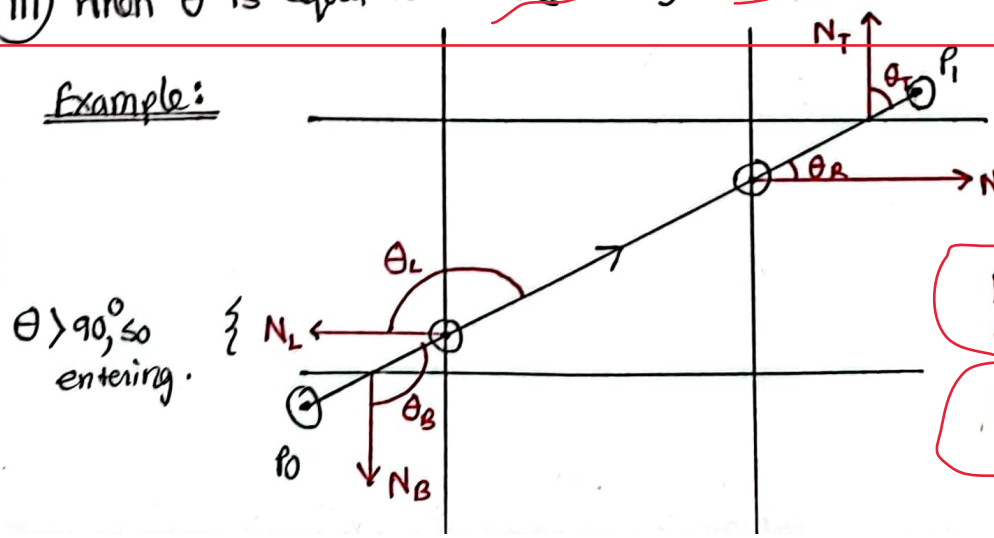
Considerations:

(i) When θ is less than 90° $(\theta < 90^\circ)$, $(P - P_E) \cdot N > 0$, {Leaving the frame}

(ii) When θ is greater than 90° $(\theta > 90^\circ)$, $(P - P_E) \cdot N < 0$, {Entering the frame}

(iii) When θ is equal to 90° $(\theta = 90^\circ)$, $(P - P_E) \cdot N = 0$, {On the boundary}

Example:



$\theta > 90^\circ$, so entering.

$\theta < 90^\circ$, so leaving

$D \cdot N_R > 0$, $D \cdot N_T > 0$] leaving

$D \cdot N_L < 0$, $D \cdot N_B < 0$] entering

We know,

$$P_{(t)} = P_0 + t\,(P_1 - P_0) \quad \cdots (i)$$ Parametric eqn of a line.

$P_{(t)}$ → at intersection of the boundary

at, $P = P_{(t)}$ ⟶ $(P - P_E) \cdot N = 0$ ---- (ii)

Substituting, $P = P_{(t)}$

$\Rightarrow (P_{(t)} - P_E) \cdot N = 0$

expand $P_{(t)}$,

$\Rightarrow (P_0 + t(P_1 - P_0) - P_E) \cdot N = 0$

$\Rightarrow N \cdot P_0 + N \cdot t \cdot P_1 - N \cdot t \cdot P_0 - N \cdot P_E = 0$

→ take this term to the RHS

$\Rightarrow N \cdot t \cdot P_1 - N \cdot t \cdot P_0 = N \cdot P_E - N \cdot P_0$

$\Rightarrow t \cdot (N \cdot P_1 - N \cdot P_0) = N \cdot (P_E - P_0)$

$\Rightarrow t = \dfrac{N \cdot (P_E + P_0)}{N \cdot (P_1 - P_0)}$ → This equation will be used to find the t of line & the edge.

(*) List of All possible boundaries:

	Boundary	N	$N \cdot (P_E - P_0)$	$N \cdot (P_1 - P_0)$	t
(1)	Top where, $y_E = y_{max}$	$(0,1)$	$= (0,1) \cdot ((x_E - x_0), (y_E - y_0))$ $= (x_E - x_0) \cdot 0 + (y_E - y_0) \cdot 1$ $= (y_{max} - y_0)$	$= (x_1 - x_0) \cdot 0 + (y_1 - y_0) \cdot 1$ $= (y_1 - y_0)$	$= \frac{y_{max} - y_0}{y_1 - y_0}$
(2)	RIGHT $x_E = x_{max}$	$(1,0)$	$= (x_{max} - x_0) \cdot 1 + (y_E - y_0) \cdot 0$ $= (x_{max} - x_0)$	$= (x_1 - x_0)$	$= \frac{x_{max} - x_0}{x_1 - x_0}$
(3)	BOTTOM $y_E = y_{min}$	$(0,-1)$	$= -1 (y_{min} - y_0)$ ↳ creates no effect	$= -1 (y_1 - y_0)$	$= \frac{y_{min} - y_0}{y_1 - y_0}$
(4)	LEFT $x_E = x_{min}$	$(-1,0)$	$= -1 (x_{min} - x_0)$ ↳ creates no effect	$= -1 (x_1 - x_0)$	$= \frac{x_{min} - x_0}{x_1 - x_0}$

(*) $(P_{0_E} - P_{0_0}) \cdot N = (x_{0_E} - x_{0_0}) \cdot Nx + (y_{0_E} - y_{0_0}) \cdot Ny$

(*) $(P_1 - P_E) \cdot N = (x_1 - x_0) \cdot Nx + (y_1 - y_0) \cdot Ny$

(*) **Explanation for the derivation of t**

$(P_E - P_0) \cdot N$;

for Top, $N = (0,1)$ & $P_E = (x_E, y_{max})$

$$\Rightarrow \left[(x_E, y_{max}) - (x_0, y_0)\right] \cdot (0,1)$$

$$\Rightarrow (x_E - x_0) \cdot 0 + (y_{max} - y_0) \cdot 1$$

$$\Rightarrow (y_{max} - y_0) \quad //$$

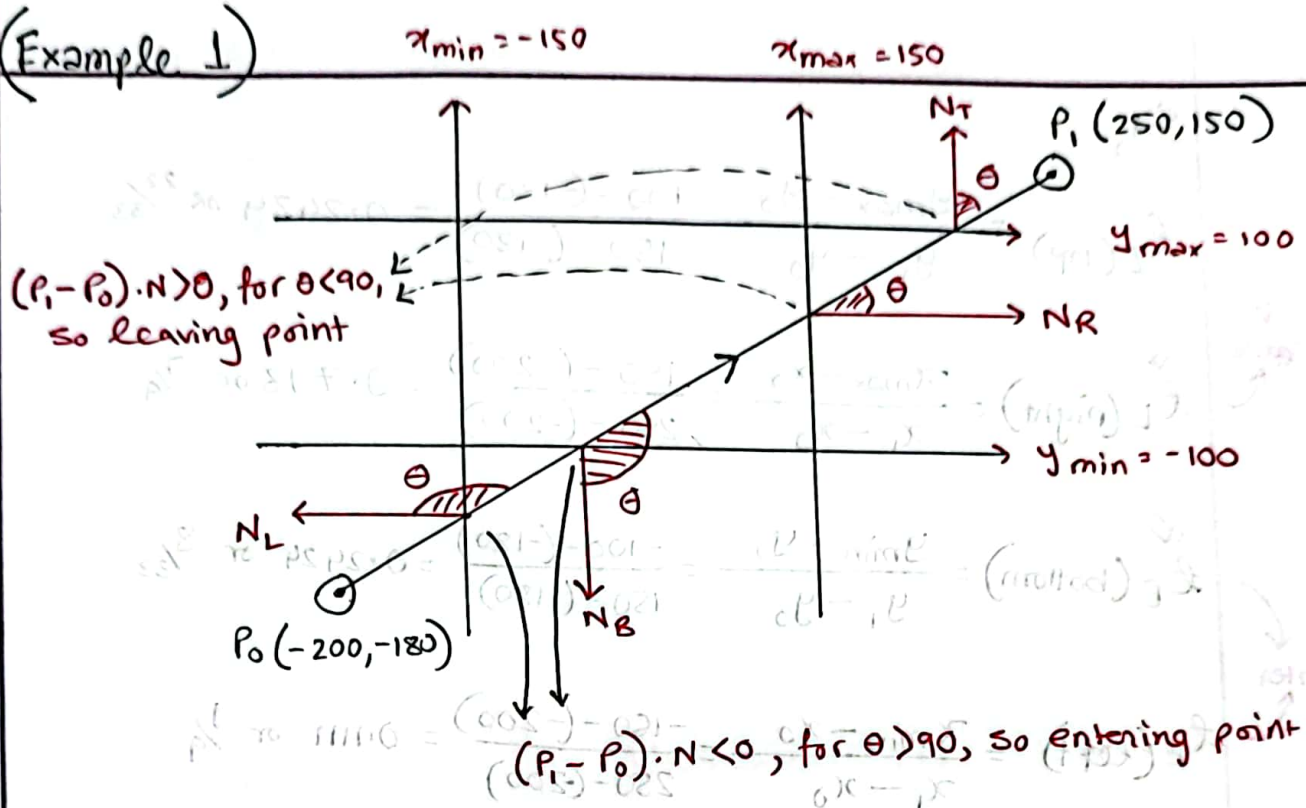
Also, $(P_1 - P_0) \cdot N \Rightarrow (x_1 - x_0) \cdot 0 + (y_1 - y_0) \cdot 1$

$$\Rightarrow (y_1 - y_0) \quad //$$

Hence,

$$t = \frac{y_{max} - y_0}{y_1 - y_0}$$

(Example 1)



Here,

(t_E) the entering t's are $\rightarrow \{t_{left} \text{ & } t_{bottom}\}$

(t_L) the leaving t's are $\rightarrow \{t_{right} \text{ & } t_{top}\}$

Imp (*) Cyrus beck says that the line has to be drawn from $t_E(max)$ to $t_L(min)$.

So,

$t_{L(top)} = \frac{y_{max} - y_0}{y_1 - y_0} = \frac{100-(-180)}{150-(-180)} = 0.8484$ or $^{28}/_{33}$

t_L leave

$t_{L(right)} = \frac{x_{max} - x_0}{x_1 - x_0} = \frac{150-(-200)}{250-(-200)} = 0.778$ or $^{7}/_{9}$

$t_{E(bottom)} = \frac{y_{min} - y_0}{y_1 - y_0} = \frac{-100-(-180)}{150-(-180)} = 0.2424$ or $^{8}/_{33}$

t_E enter

$t_{E(left)} = \frac{x_{min} - x_0}{x_1 - x_0} = \frac{-150-(-200)}{250-(-200)} = 0.1111$ or $^{1}/_{9}$

(*) drawn from $t_{E(max)}$ to $t_{L(min)}$

to

$t = {}^{8}/_{33}$, $t_{E(max)} \longrightarrow (-90.91, -100)$

$t = {}^{7}/_{9}$, $t_{L(min)} \longrightarrow (150, 76.67)$

drawn

Example 2 (Type of sum to expect in exam)

Q) Clip Region (-100, -120) to (150, 200)
Draw the line (clipped) for $P_0(-125, 260)$ to (195, -140)

ANS: Region → (-100, -120) to (150, 200)

$x_{min} = -100$ | $y_{min} = -120$
$x_{max} = 150$ | $y_{max} = 200$

line $\begin{bmatrix} x_0 = -125 & x_1 = 195 \\ y_0 = 260 & y_1 = -140 \end{bmatrix}$

for vector D,

$$D = P_1 - P_0$$
$$= (x_1 - x_0, y_1 - y_0)$$
$$= (320, -400)$$

Process -
1) Determine P0 (xmin, ymin) and P1(xmax, ymax)

2) Vector, D = P1 - P0

3) Find NL*D for all four corner of screen
if NL*D < 0 , Entering point:
if NL*D > 0, leaving point

4) Get t for all four corners
t top = (ymax - y0) / (y1 - y0)
t bottom = (ymin - y0) / (y1 - y0)
t right = (xmax - x0) / (x1 - x0)
t left = (xmin - x0) / (x1 - x0)

5) For two entering point, select tmax as entering point
For two leaving point, select tmin as leaving point

(1) Solving for left :

$$N_L \cdot D = (-1, 0) \cdot (320, -400) = -320 < 0, \text{ Entering}$$

$$\therefore t_{left} = \frac{x_{min} - x_0}{x_1 - x_0} = \frac{-100 - (-125)}{195 - (-125)} = 0.078 \text{ or } 5/64$$

(2) Solving for right :

$$N_r \cdot D = (1, 0) \cdot (320, -400) = 320 > 0, \text{ Leaving}$$

$$\therefore t_{Right} = \frac{x_{max} - x_0}{x_1 - x_0} = \frac{150 + 125}{195 + 125} = 0.8594 \text{ or } 55/64$$

③ Solving for top :

$N_T \cdot D = (0,1) \cdot (320,-400) = -400 < 0$, Entering

$$t_{top} = \frac{y_{max} - y_0}{y_1 - y_0} = \frac{200 - 260}{-140 - 260} = 0.15 \text{ or } 3/20$$

④ Solving for Bottom :

$N_B \cdot D = (0,-1) \cdot (320,-400) = 400 > 0$, Leaving

$$t_{Bottom} = \frac{y_{min} - y_0}{y_1 - y_0} = \frac{-120 - 260}{-140 - 260} = 0.95 \text{ or } 19/20$$

⑤ Now,

$t_{E(max)} = t_{top} = 0.15 = 3/20$

$t_{L(min)} = t_{right} = 55/64$

Draw $(t_{top} \longrightarrow t_{right})$, $(t_{E(max)} \longrightarrow t_{L(min)})$

So, at $t = t_{E(max)}$

$$x = -125 + \left(\frac{3}{20} \cdot (195 + 125)\right) = -77$$

$y = y_{max} = 200$

So, $(x_0, y_0) = (-77, 200)$ → new

at $t = t_{L(min)}$

$x = x_{max} = 150$

$y = 260 + \left(\frac{55}{64} \cdot (-140 - 260)\right)$

$= -83.75$

Ans

So, $(x_1, y_0) = (150, -83.75)$

Solve at Home :

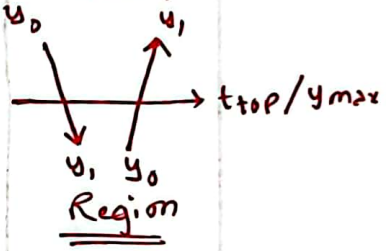
Q 1) Clip Region (-60, -50) to (60, 50)
Line (-100, 60) to (110, -70)

Q 2) Clip Region (-120, -100) to (120, 100)
Line (-135, -200) to (140, 200)

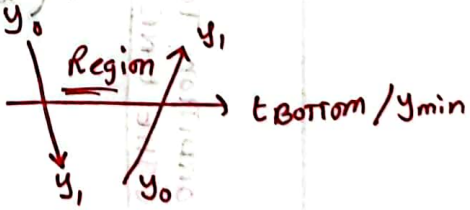
Pseudo Code

```
void Cyrus_Beck (int x0, int y0, int x1, int y1)
    float tEmax = 0.0, tLmin = 1.0, t;
    ymax = ...., ymin = ...., xmax = ...., xmin = ....
For TOP
    t = (ymax - y0) / (y1 - y0)
    if (y1 > y0) { // Leaving
        if (t < tLmin) {
            tLmin = t
        }
    } else { // Entering
        if (t > tEmax) {
            tEmax = t
        }
    }
FOR BOTTOM
    t = (ymin - y0) / (y1 - y0)
    if (y1 > y0) { // entering
        if (t > tEmax) {
            tEmax = t
        }
    } else { // leaving
        if (t < tLmin) {
            tLmin = t
        }
    }
```

Example :

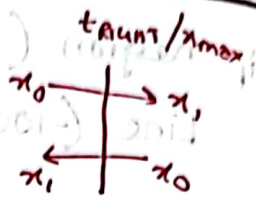


Example :



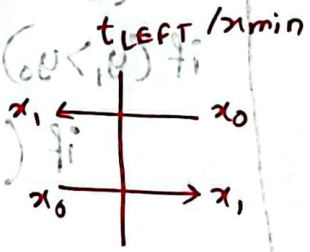
FOR RIGHT

```
t = (xmax - x0)/(x1 - x0)
if (x1 > x0) { // leaving
    if (t < tLMIN){
        tLMIN = t
    }
}
else { // entering
    if (t > tEMAX){
        tEMAX = t
    }
}
```



FOR LEFT

```
t = (xmin - x0)/(x1 - x0)
if (x1 > x0) { // entering
    if (t > tEMAX){
        tEMAX = t
    }
}
else { // leaving
    if (t < tLMIN){
        tLMIN = t
    }
}
```



FINAL CONDITION TO DRAW THE LINE

```
if (tLMIN >= tEMAX){ // only acceptable condition
    newP0 = Point(tEMAX)
    newP0,1 = Point(tLMIN)
    drawline(newP0.x, newP0.y, newP1.x, newP1.y)
}
```

(*)

```
POI   new P0, new P1

POI   Point( float t )
        POI.x = P0.x + t (P1.x - P0.x)
        POI.y = P0.y + t (P1.y - P0.y)
        return POI
}
```