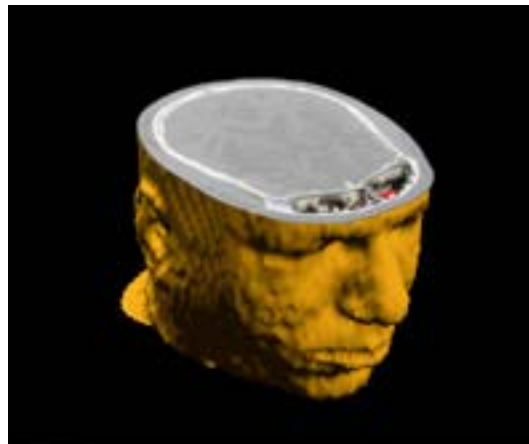
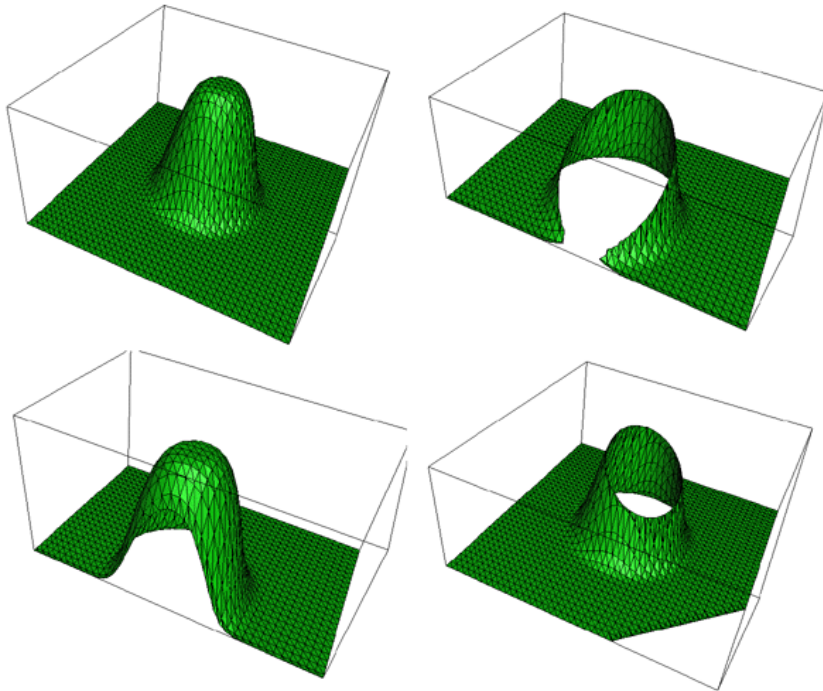


# Clipping

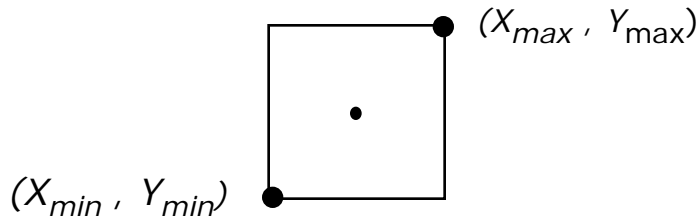
# Clipping

## Chapter 13, Section 10



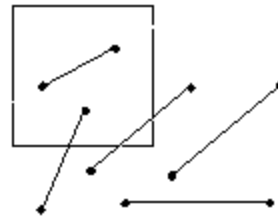
# Line Clipping

- Clipping endpoints



$x_{min} \leq x \leq x_{max}$  **and**  $y_{min} \leq y \leq y_{max}$   $\implies$  point inside

- Endpoint analysis for lines:



- if both endpoints in , do “trivial acceptance”
  - if one endpoint inside, one outside, must clip
  - if both endpoints out, don't know
- Brute force clip: solve simultaneous equations using  $y = mx + b$  for line and four clip edges
    - slope-intercept formula handles infinite lines only
    - doesn't handle vertical lines

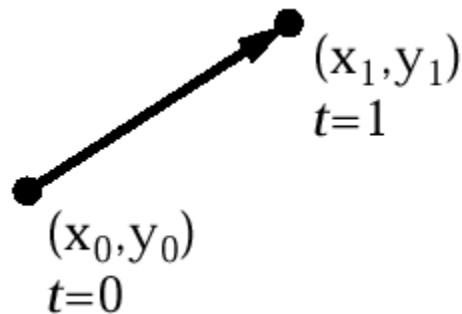
# Parametric Line Formulation For Clipping

- Parametric form for line segment

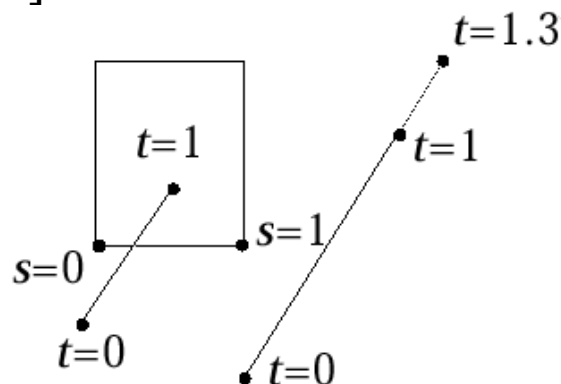
$$X = x_0 + t(x_1 - x_0) \quad 0 \leq t \leq 1$$

$$Y = y_0 + t(y_1 - y_0)$$

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

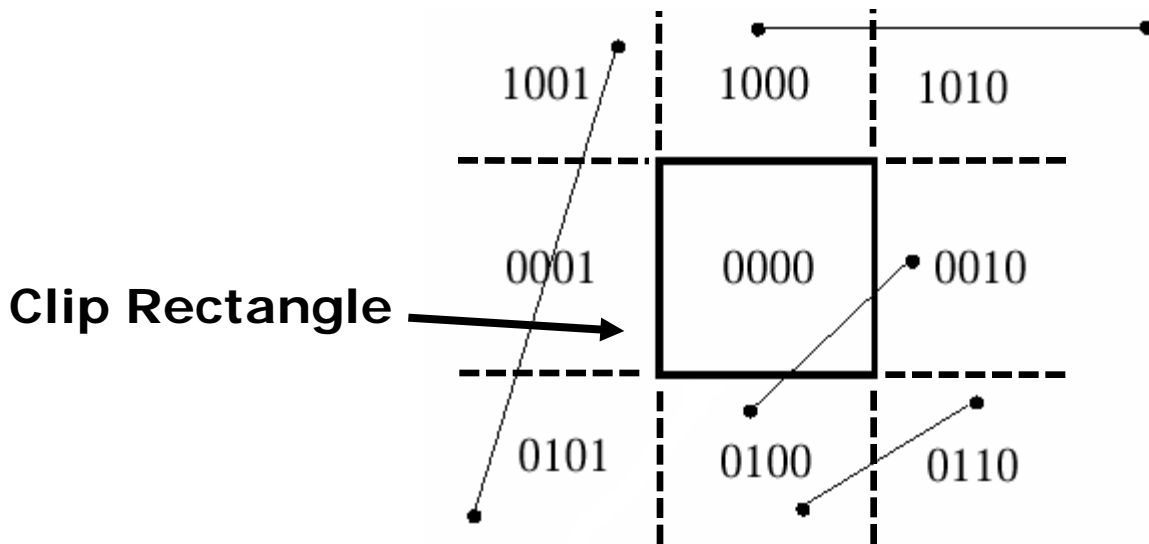


- "true," i.e., interior intersection, if  $s_{edge}$  and  $t_{line}$  in  $[0, 1]$



# Outcodes for Cohen-Sutherland Line Clipping in 2D

- Divide plane into 9 regions
- Compute the sign bit of 4 comparisons between a vertex and an edge
  - $y_{max} - y; y - y_{min}; x_{max} - x; x - x_{min}$
  - point lies inside only if all for sign bits are 0, otherwise exceeds edge



- 4 bit outcode records results of four bounds tests:

**First bit:** outside halfplane of top edge, above top edge  
**Second bit:** outside halfplane of bottom edge, below bottom edge  
**Third bit:** outside halfplane of right edge, to right of right edge  
**Fourth bit:** outside halfplane of left edge, to left of left edge

- Lines with  $OC_0 = 0$  and  $OC_1 = 0$  can be *trivially accepted*
- Lines lying entirely in a half plane outside an edge can be *trivially rejected*:  $OC_0 \text{ AND } OC_1 \neq 0$  (i.e., they share an "outside" bit)

# Outcodes for Cohen-Sutherland Line Clipping in 3D

- Very similar to 2D
- Divide volume into 27 regions (Picture a Rubik's cube)
- 6-bit outcode records results of 6 bounds tests

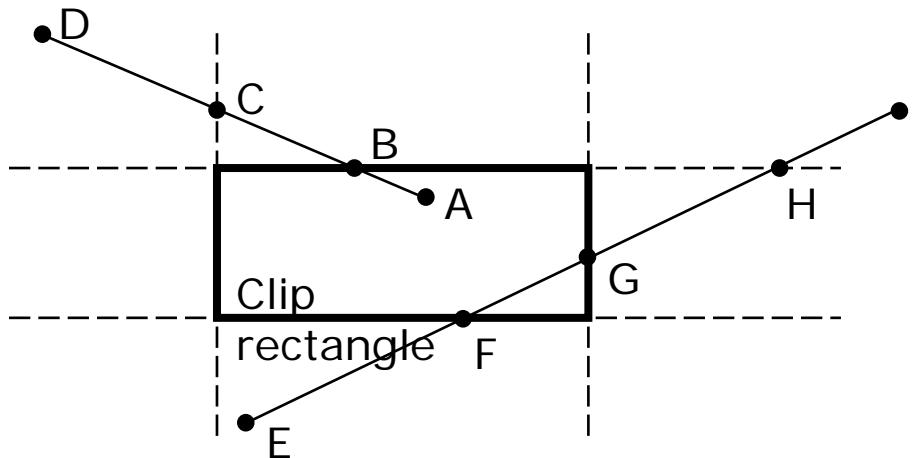
<b>Top plane</b>	<b>Bottom plane</b>	<b>Front plane</b>
<u>001000 (above)</u>	<u>000000 (above)</u>	<u>010000 (in front)</u>
000000 (below)	000100 (below)	000000 (behind)
<b>Left plane</b>	<b>Right plane</b>	<b>Back plane</b>
<u>000001 (to left of)</u>	<u>000000 (to left of)</u>	<u>000000 (in front)</u>
000000 (to right of)	000010 (to right of)	100000 (behind)

- First bit:** outside back plane, behind back plane  
**Second bit:** outside front plane, in front of front plane  
**Third bit:** outside top plane, above top plane  
**Fourth bit:** outside bottom plane, below bottom plane  
**Fifth bit:** outside right plane, to right of right plane  
**Sixth bit:** outside left plane, to left of left plane

- Lines with  $OC_0 = 0$  and  $OC_1 = 0$  can be *trivially accepted*
- Lines lying entirely in a volume on outside of a plane can be *trivially rejected*:  $OC_0$  AND  $OC_1 \neq 0$  (i.e., they share an "outside" bit)

# Cohen-Sutherland Algorithm

- If we can neither trivially reject/accept, divide and conquer
- subdivide line into two segments; then T/A or T/R one or both segments:



- use a clip edge to cut line
- use outcodes to choose edge that is crossed
  - Edges where the two outcodes differ at that particular bit are crossed
- pick an order for checking edges
  - top – bottom – right – left
- compute the intersection point
  - the clip edge fixes either  $x$  or  $y$
  - can substitute into the line equation
- iterate for the newly shortened line
- “extra” clips may happen (e.g., E-I at H)

# Pseudocode for the Cohen-Sutherland Algorithm

- $y = y_0 + \text{slope} * (x - x_0)$  and  $x = x_0 + (1/\text{slope}) * (y - y_0)$

ComputeOutCode(x0, y0, outcode0)

ComputeOutCode(x1, y1, outcode1)

**repeat**

    check for trivial reject or trivial accept

    pick the point that is outside the clip rectangle

**if TOP then**

$x = x_0 + (x_1 - x_0) * (y_{\max} - y_0) / (y_1 - y_0); y = y_{\max};$

**else if BOTTOM then**

$x = x_0 + (x_1 - x_0) * (y_{\min} - y_0) / (y_1 - y_0); y = y_{\min};$

**else if RIGHT then**

$y = y_0 + (y_1 - y_0) * (x_{\max} - x_0) / (x_1 - x_0); x = x_{\max};$

**else if LEFT then**

$y = y_0 + (y_1 - y_0) * (x_{\min} - x_0) / (x_1 - x_0); x = x_{\min};$

**if (x0, y0 is the outer point) then**

$x_0 = x; y_0 = y; \text{ComputeOutCode}(x_0, y_0, \text{outcode}_0)$

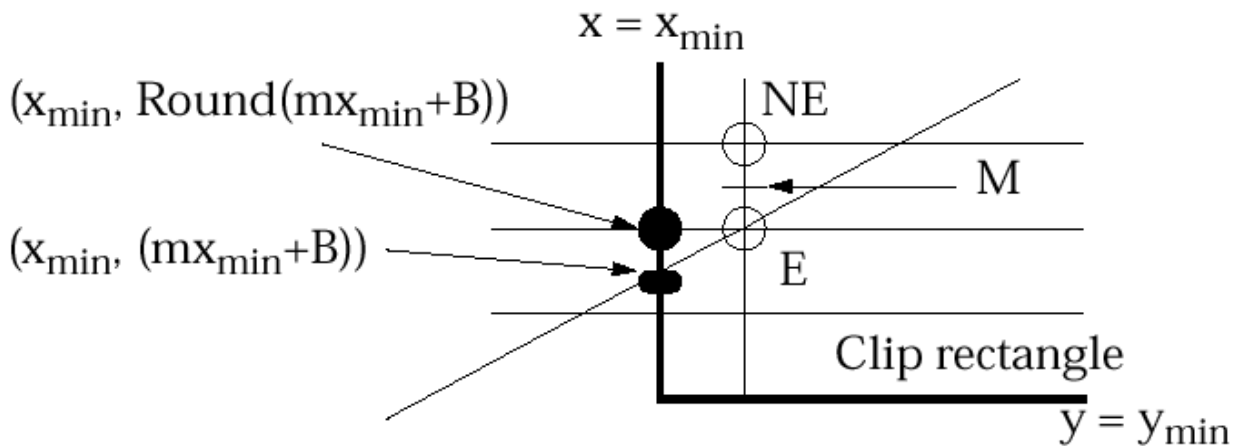
**else**

$x_1 = x; y_1 = y; \text{ComputeOutCode}(x_1, y_1, \text{outcode}_1)$

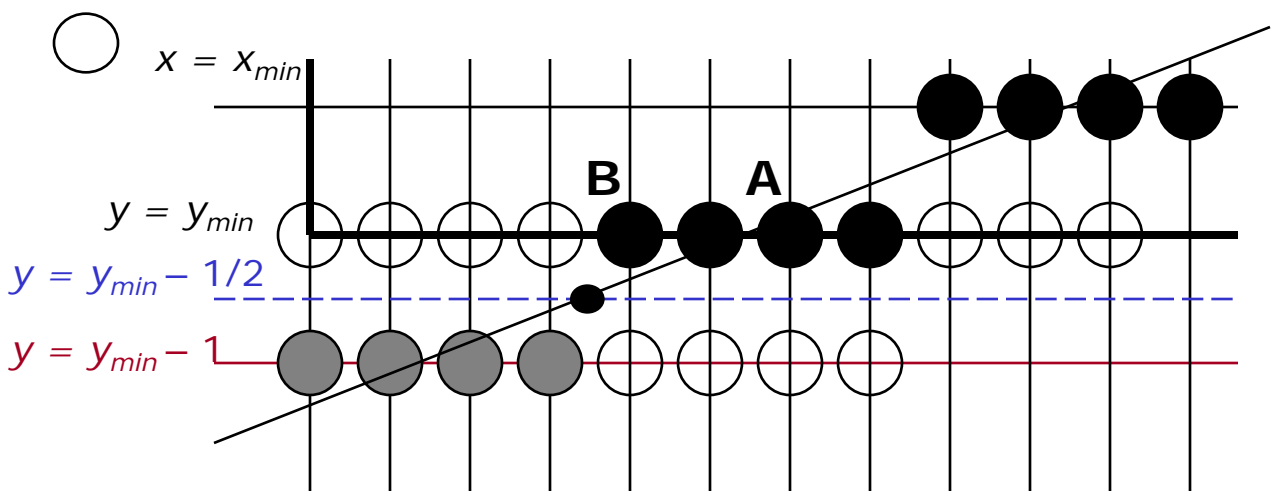
**until** done

# Scan Conversion after Clipping

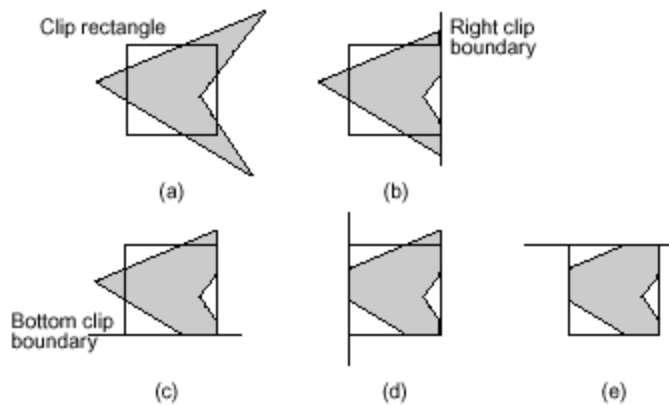
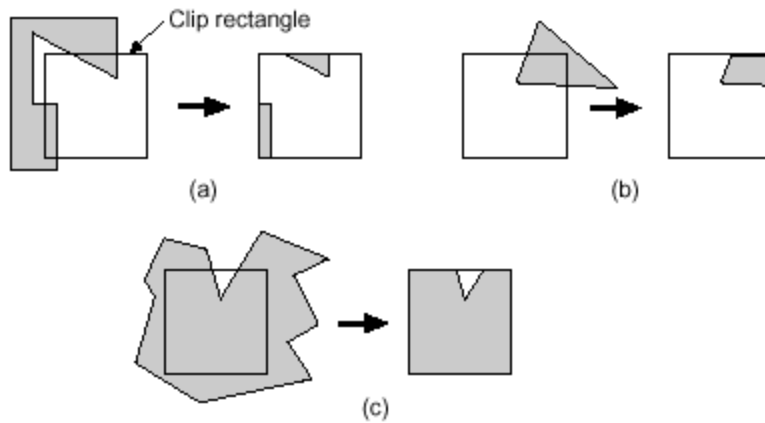
- a) Don't round and then scan convert:  
 calculate decision variable based on pixel chosen on left edge



- b) Horizontal edge problem:  
 clipping/rounding produces pixel A; to get pixel B, round up  $x$  of the intersection of line with  $y = y_{min} - 1/2$  and pick pixel above:



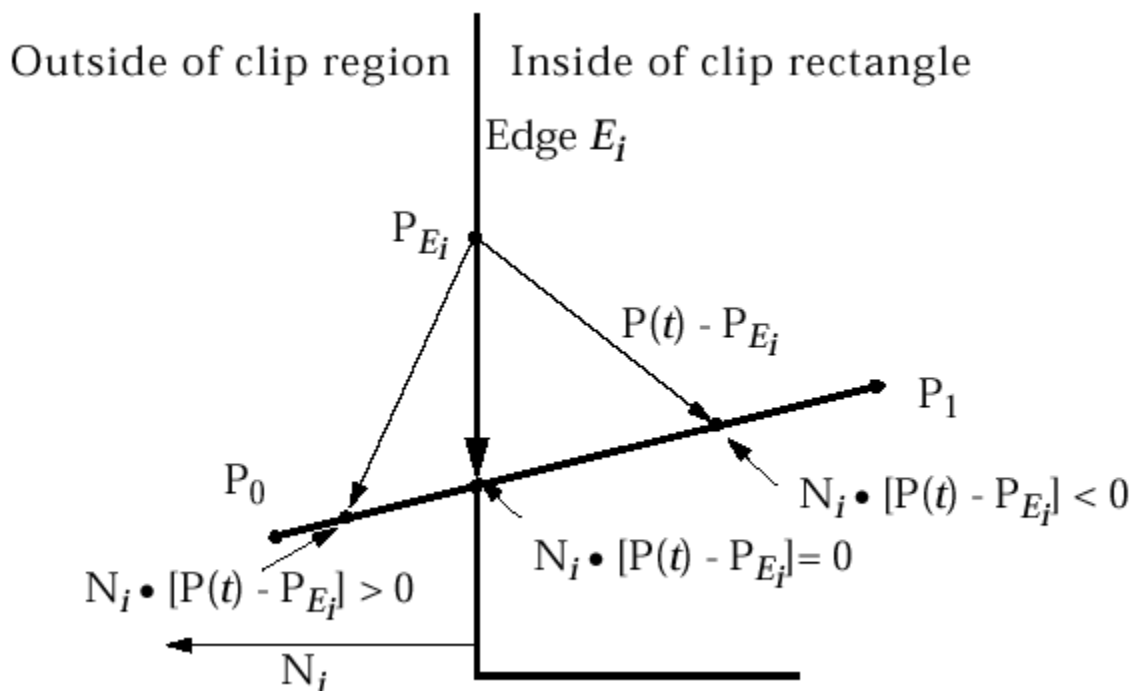
# Sutherland-Hodgman Polygon Clipping



# Cyrus-Beck/Liang-Barsky Parametric Line Clipping-1

- Use parametric line formulation  

$$P(t) = P_0 + (P_1 - P_0)t$$
- Determine where the line intersects the infinite line formed by each edge by solving for  $t$  4 times. Decide which of these intersections actually occur on the rectangle



- For any point  $P_{E_i}$  on edge  $E_i$

## C-B/L-B Param. Line Clipping-2

Now solve for the value of  $t$  at the intersection of  $P_0 P_1$  with the edge  $E_i$ :

$$N_i \cdot [P(t) - P_{E_i}] = 0$$

First, substitute for  $P(t)$ :

$$N_i \cdot [P_0 + (P_1 - P_0)t - P_{E_i}] = 0$$

Next, group terms and distribute dot product:

$$N_i \cdot [P_0 - P_{E_i}] + N_i \cdot [P_1 - P_0]t = 0$$

Let  $D$  be the vector from  $P_0$  to  $P_1 = (P_1 - P_0)$ , and solve for  $t$ :

$$t = \frac{N_i \cdot [P_0 - P_{E_i}]}{-N_i \cdot D}$$

Note that this gives a valid value of  $t$  only if the denominator of the expression is nonzero.

For this to be true, it must be the case that:

$N_i \neq 0$  (that is, the normal should not be 0;  
this could occur only as a mistake)

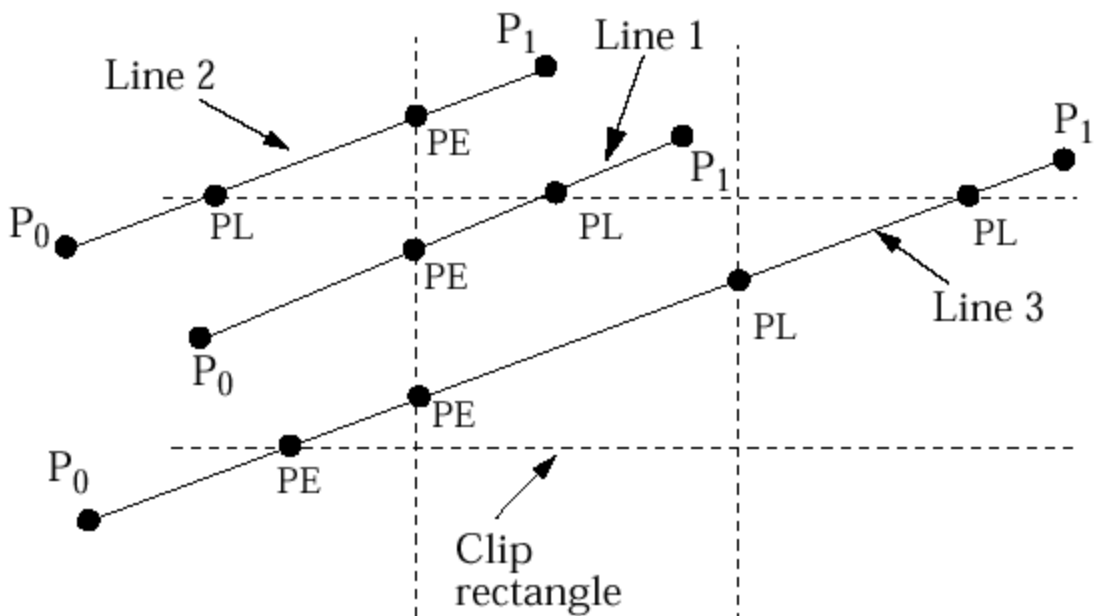
$D \neq 0$  (that is,  $P_1 \neq P_0$ )

$N_i \cdot D \neq 0$  (edge  $E_i$  and line  $D$  are not parallel; if they are, no intersection).

The algorithm checks these conditions.

## C-B/L-B Param. Line Clipping-3

- Eliminate  $t$ 's outside  $[0,1]$  on the line
- Which remaining  $t$ 's produce interior intersections?
- Can't just take the innermost  $t$  values!



- Move from  $P_0$  to  $P_1$ ; for a given edge, just before crossing: if  $N_i \cdot D < 0 \Rightarrow$  Potentially Entering (PE), if  $N_i \cdot D > 0 \Rightarrow$  Potentially Leaving (PL)
- Pick inner PE, PL pair:  $t_E$  for  $P_{PE}$  with max  $t$ ,  $t_L$  for  $P_{PL}$  with min  $t$ , and  $t_E > 0$ ,  $t_L < 1$ .
- If  $t_L < t_E$ , no intersection

# Pseudocode for Cyrus-Beck/ Liang-Barsky Line Clipping Algorithm

```

Pre-calculate  $N_i$  and select  $P_{E_i}$  for each edge;
for each line segment to be clipped
  if  $P_1 = P_0$  then
    line is degenerate so clip as a point;
  else
    begin
       $t_E = 0$ ;  $t_L = 1$ ;
      for each candidate intersection with a clip edge
        if  $N_i \cdot D \neq 0$  then {Ignore edges parallel to line}
          begin
            calculate  $t$ ; {of line and clip edge intersection}
            use sign of  $N_i \cdot D$  to categorize as PE or PL;
            if PE then  $t_E = \max(t_E, t)$ ;
            if PL then  $t_L = \min(t_L, t)$ ;
          end
        if  $t_E > t_L$  then
          return nil
        else
          return  $P(t_E)$  and  $P(t_L)$  as true clip intersections
      end

```

# Calculations for Parametric Line Clipping for Upright Clip Rectangle (1/2)

- $D = P_1 - P_0 = (x_1 - x_0, y_1 - y_0)$
- Leave  $P_{E_i}$  as an arbitrary point on the clip edge; it's a free variable and drops out

## Calculations for Parametric Line Clipping Algorithm

Clip Edge <sub>i</sub>	Normal N <sub>i</sub>	P <sub>E<sub>i</sub></sub>	P <sub>0</sub> -P <sub>E<sub>i</sub></sub>	$t = \frac{N_i \cdot (P_0 - P_{E_i})}{-N_i \cdot D}$
left: $x = x_{\min}$	(-1,0)	( $x_{\min}, y$ )	( $x_0 - x_{\min}, y_0 - y$ )	$\frac{-(x_0 - x_{\min})}{(x_1 - x_0)}$
right: $x = x_{\max}$	(1,0)	( $x_{\max}, y$ )	( $x_0 - x_{\max}, y_0 - y$ )	$\frac{-(x_0 - x_{\max})}{(x_1 - x_0)}$
bottom: $y = y_{\min}$	(0,-1)	( $x, y_{\min}$ )	( $x_0 - x, y_0 - y_{\min}$ )	$\frac{-(y_0 - y_{\min})}{(y_1 - y_0)}$
top: $y = y_{\max}$	(0,1)	( $x, y_{\max}$ )	( $x_0 - x, y_0 - y_{\max}$ )	$\frac{-(y_0 - y_{\max})}{(y_1 - y_0)}$

# Calculations for Parametric Line Clipping for Upright Clip Rectangle (2/2)

- Examine  $t$ :
  - Numerator is just the directed distance to an edge; sign corresponds to OC
  - Denominator is just the horizontal or vertical projection of the line,  $dx$  or  $dy$ ; sign determines PE or PL for a given edge
  - Ratio is constant of proportionality: “how far over” from  $P_0$  to  $P_1$  intersection is relative to  $dx$  or  $dy$