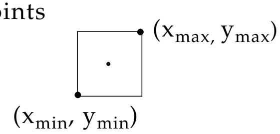
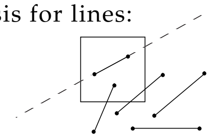
Line Clipping

- Clipping endpoints



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

- Endpoint analysis for lines:



  - if both endpoints in, can do "trivial acceptance"
  - if one endpoint is 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

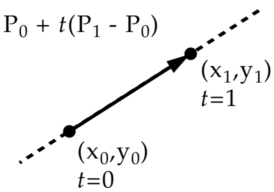
Parametric Line Formulation for Clipping

- Parametric form for line segment

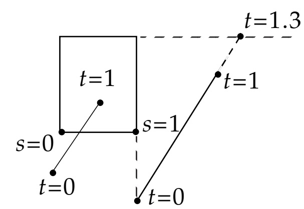
$x = x_0 + t(x_1 - x_0) \quad 0 \leq t \leq 1$

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

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

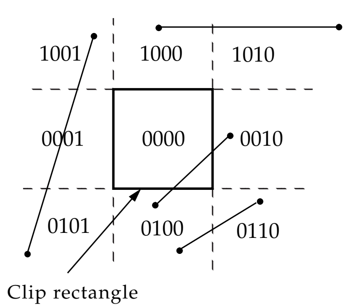


- "true", i.e., interior intersection, if $s_{\mathbf{edge}}$ and $t_{\mathbf{line}}$ in [0,1]



Outcodes for Cohen-Sutherland Line Clipping

- Divide plane into 9 regions



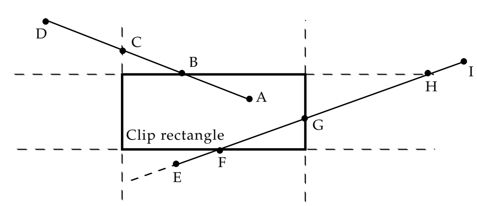
- 4 bit outcode records results of four bounds tests:

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 on outside of an edge can be *trivially rejected*: OC_0 & $OC_1 \neq 0$ (i.e., they share an "outside" bit)

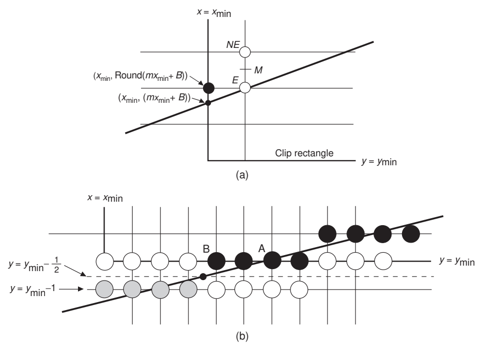
Cohen-Sutherland Algorithm

- If can neither trivial reject nor accept, divide and conquer: subdivide line into two segments, then can T/A or T/R one or both segments:
  - Use a clip edge to cut the line
  - Use outcodes to choose edge that is crossed: if outcodes differ in the edge's bit, endpoints must straddle that edge
  - Pick an order for checking edges: top - bottom - right - left
  - Compute the intersection point; the clip edge fixes either x or y, can be substituted into the line equation
  - Iterate for the newly shortened line
  - "extra" clips may happen, e.g., EI at H



Scan Conversion after Clipping

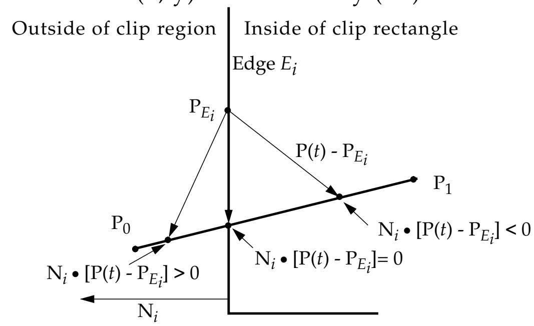
a) don't round and then scan convert, (a)!



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

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

- Use parametric line formulation
  $P(t) = P_0 + (P_1 - P_0)t$
- Find the four t's for the four clip edges, then decide which form true intersections and calculate (x, y) for those only (≤ 2)



- For any point P_{E_i} on edge E_i

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

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

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

First, substitute for $P(t)$:

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

Next, group terms and distribute the dot product:

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

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

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

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)

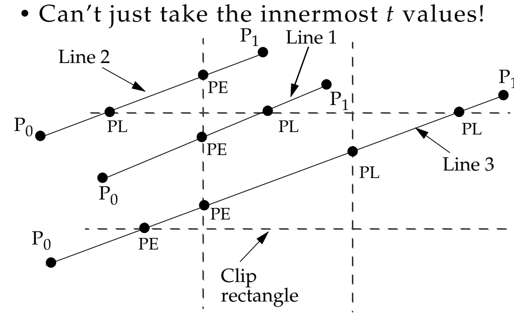
$D \neq 0$ (that is, $P_1 \neq P_0$)

$N_i \bullet D \neq 0$ (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 $N_i \bullet D < 0 \Rightarrow$ Potentially Entering (PE) if $N_i \bullet D > 0 \Rightarrow$ 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

precalculate N_i *and select a* 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 clip edge*
        if $Ni \bullet D \neq 0$ **then** {Ignore edges parallel to line}
          begin
            calculate t; {of line $\cap$ clip edge}
            use sign of $N_i \bullet D$ *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 {else}

Calculations for Parametric Line Clipping for Upright Clip Rectangle

- $D = P_1 - P_0 = (x_1 - x_0, y_1 - y_0)$
- 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$_i$	Normal N_i	P_{E_i}	$P_o - P_{E_i}$	$t = \frac{N_i \cdot (P_0 - P_{E_i})}{-N_i \cdot D}$
left: $x = x_{min}$	$(-1, 0)$	(x_{min}, y)	$(x_0 - x_{min}, y_0 - y)$	$\frac{-(x_o - x_{min})}{(x_1 - x_o)}$
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 (cont)

- 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

Sutherland-Hodgeman Polygon Clipping