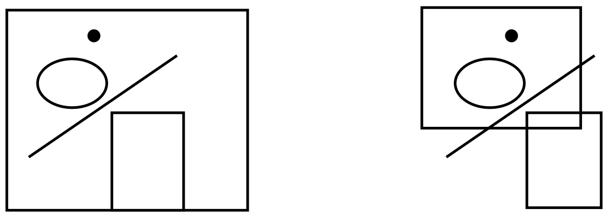
2D Graphics Primitives III

Clipping



Example: user change in window size and Expose event

Redraw entire image?

Redraw clipped image

Specifications of entire scene as lines, circles etc (OpenGL calls)
Scan convert entire scene into a pixmap
Copy portion of pixmap as specified by clipping rectangle

Versus

Specification of entire scene as above
Scan convert entire scene, but write only visible pixels
(Scissoring)

Versus

Specifications of entire scene as above
Specifications of clipped scene
Scan convert from clipped specifications

First is easy, but wastes time and space

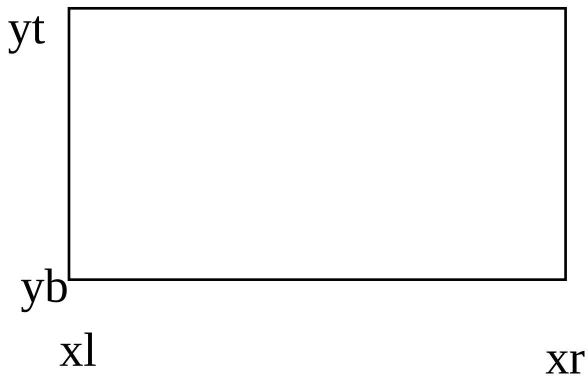
Second may be quite efficient if done in microcode or hardware
Generalizes to arbitrary shape clip regions

Third is often best for points, lines, polygons (simple algorithms)

Define Clip Rectangle (Region)

Start with rectangular clip region (clip-box)

defined by xl, xr, yt, yb

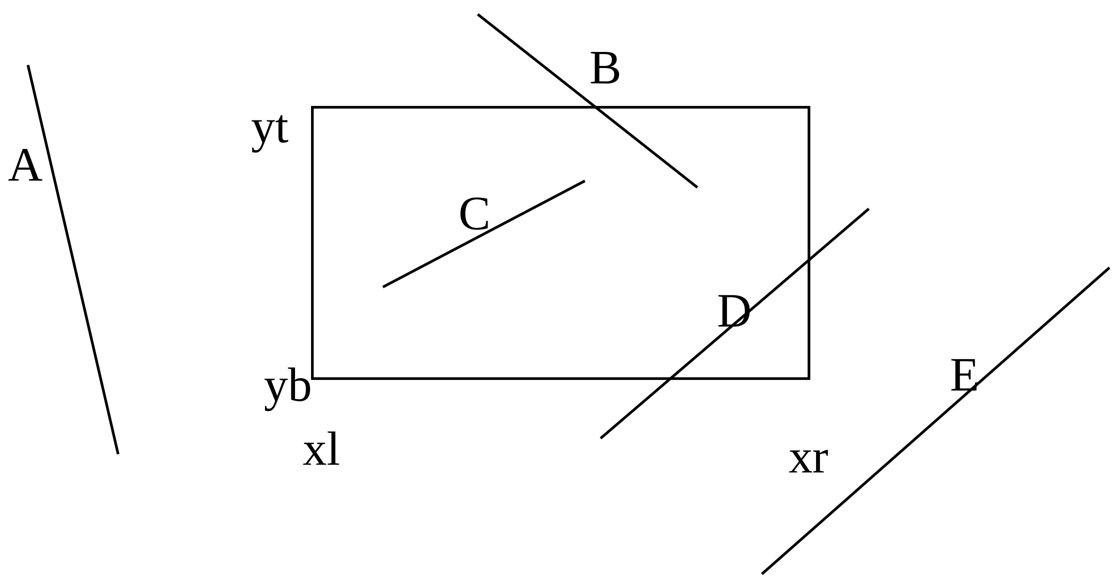


Clipping Points

How determine if point should be displayed?

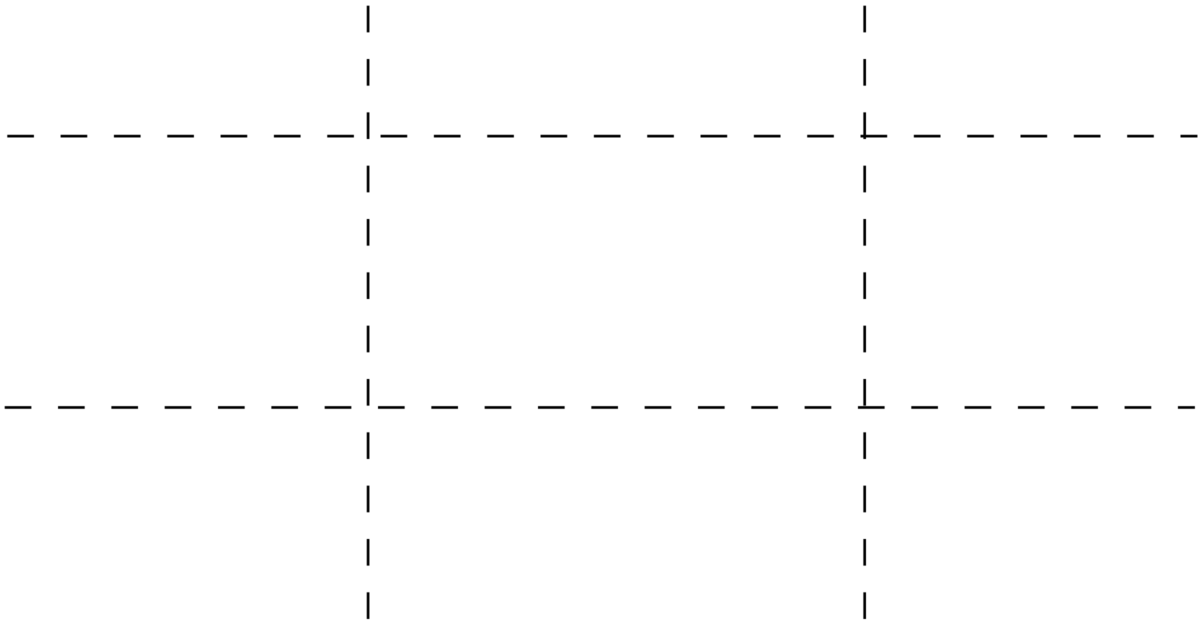
Clipping Lines

More difficult



Which lines easy to clip?

Cohen-Sutherland Line Clipping Algorithm



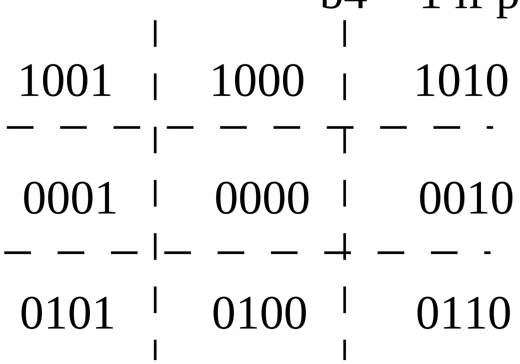
Based on idea that some lines are trivially accepted (entire line drawn)
others trivially rejected (none of line drawn)
others more difficult (maybe clip some, maybe draw nothing)

Look at nine regions of space as divided by the clip-box

Assign 4 bit region code to each region:

b4 b3 b2 b1

b1 = 1 if point is to the left of the left boundary
b2 = 1 if point is to the right of the right boundary
b3 = 1 if point is below bottom boundary
b4 = 1 if point is above top boundary



Find region code of each end of line
(C1, C2)
Use to accept or reject line

eg if both ends are 0000?

what else is easy case?

1001	1000	1010
0001	0000	0010
0101	0100	0110

Look at logical operations on the region codes (AND, OR)

A) If C1 OR C2 = 0000, then trivially accept line

B) How trivially reject a line that has both points above top?
C1 AND C2 = 1xxx

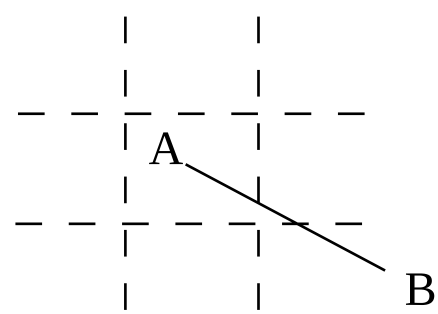
How trivially reject a line below, to right and to left?
C1 AND C2 = ?
C1 AND C2 = ?
C1 AND C2 = ?

How generalize these four cases?
C1 AND C2 not equal 0000

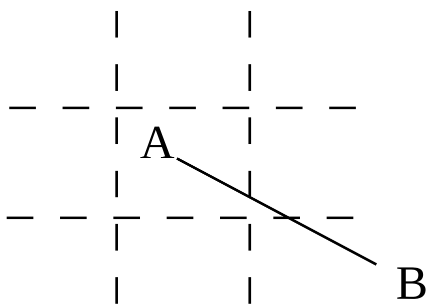
C) Rest are difficult
C1 AND C2 = 0000

Cohen-Sutherland Algorithm

Start with input list of lines (endpoints)
M: While input list is not empty
Find Region codes (C1 and C2) for line
Remove line from input list
If C1 OR C2 = 0000, then add line to output list
Else if C1 AND C2 = 0000, find intersection of line
with an edge (top, bottom, left, right order)
Add intersection point and interior point to input list
End

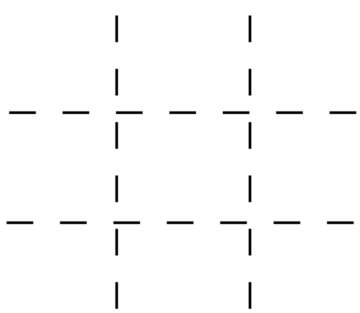


Given the order of testing for intersections, what is a worst case input line for Cohen-Sutherland algorithm?



This required two clips
First required two tests
Second required four tests

Can a line require 4 clips? (How many regions can a line pass through?)



If four clips required,
then how many tests on first clip?
on second?
on third?
on forth?

Cohen-Sutherland not the most efficient algorithm as it can end up doing needless clipping.
Still used widely, since widely known

Cyrus-Beck Parametric Line Clipping Algorithm

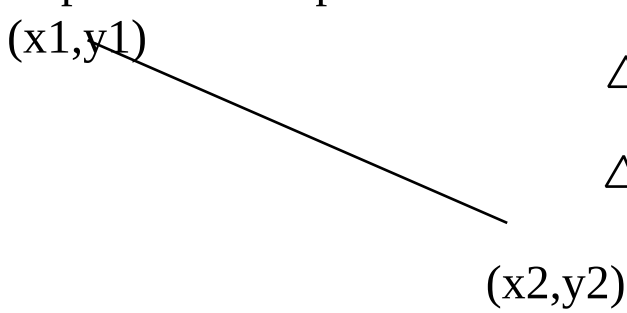
- more efficient
- can clip against convex polygon clip region
- can clip in 3D as well as 2D

Liang-Barsky

- like above, but faster for upright rectangular 2D and 3D regions

Derivaton of Liang-Barsky

Based on parametric representation of line



$\Delta x = x2 - x1$

$\Delta y = y2 - y1$

$x = x1 + \Delta x\, u$
$y = y1 + \Delta y\, u$ $\quad 0 <= u <= 1$

Write clipping equations in parametric form

$xL <= x1 + \Delta xu <= xr$

$yb <= y1 + \Delta yu <= yt$

Rewrite as four inequalities

$u\, p_k <= q_k$, where k = 1, 2, 3, 4

$p_1 = -\Delta x \qquad q_1 = x1 - xL$
$p_2 = \Delta x \qquad q_2 = xr - x1$
$p_3 = -\Delta y \qquad q_3 = y1 - yb$
$p_4 = \Delta y \qquad q_4 = yt - y1$

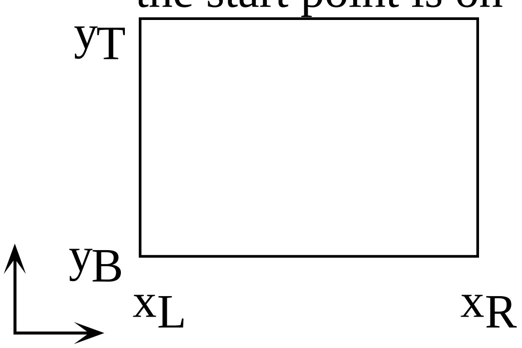
Each value of k corresponds to one boundary:

- $k = 1$ corresponds to left boundary
- $k = 2$ corresponds to the right boundary
- $k = 3$?
- $k = 4$?

If line is parallel to the kth boundary, then

$p_k = ?$

The values of q_k indicate which side of the kth boundary the start point is on



$p_1 = -\Delta x \qquad q_1 = x1 - x_L$

$p_2 = \Delta x \qquad q_2 = x_R - x1$

$p_3 = -\Delta y \qquad q_3 = y1 - y_B$

$p_4 = \Delta y \qquad q_4 = y_T - y1$

if $q_k < 0$, then p1 is outside kth boundary

$q_k >= 0$, the p1 is inside or on the kth boundary

if $p_k < 0$, then line goes from outside to inside the kth boundary

$p_k > 0$, then line goes from inside to outside the kth boundary

if p_k - 0, then the intersection of the line with the kth boundary is at

$$r_k = q_k / p_k$$

For each line we want to find u_1 and u_2 that lie in clip region

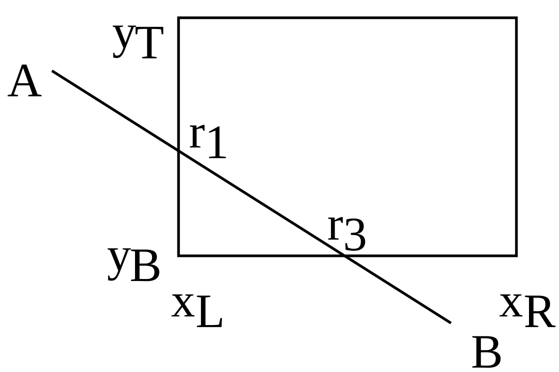
Liang-Barsky Algorithm

For each line segment

- u1 = 0;
- u2 = 1; (We are starting with the original endpoints)
- k =1;
- while still need to clip and k < = 4
  - compute p_k and q_k
  - if $p_k = 0$ and $q_k < 0$, then reject line and stop clipping
  - else
    - if $p_k < 0$,
      - u1 = maximum of u1 and r_k
    - else
      - u2 = minimum of u2 and r_k
  - if u1 > u2
    - reject line and stop clipping
  - k = k + 1;
- end
- if line not rejected, u1 and u2 are end points of clipped line

end

Example:



Line AB

$p_1 = -\Delta x$	$q_1 = x1 - x_L$
$p_2 = \Delta x$	$q_2 = x_R - x1$
$p_3 = -\Delta y$	$q_3 = y1 - y_B$
$p_4 = \Delta y$	$q_4 = y_T - y1$

u1 = 0	u2 = 1
u1 = r_1	u2 = 1
u1 = r_1	u2 = 1
u1 = r_1	u2 = r_3
u1 = r_1	u2 = r_3

$p_1 < 0$	$q_1 < 0$
$p_2 > 0$	$q_2 > 0$
$p_3 > 0$	$q_3 > 0$
$p_4 < 0$	$q_4 > 0$

Liang-Barsky versus Cohen-Sutherland

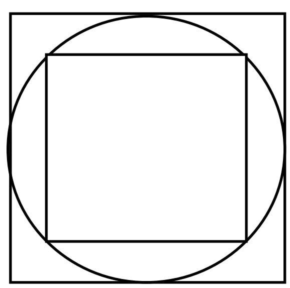
Liang-Barsky computes fewer intersections for a line needing clipping

But doens't have a trivial accept

If most lines can be trivially accepted or rejected,
  Use Cohen-Sutherland
else
  Use Liang-Barsky

Clipping Circles

Can approximate with 2 rectangles for trivial accept and reject



Outer used for?

Inner used for?

Can make better approximations using polygons