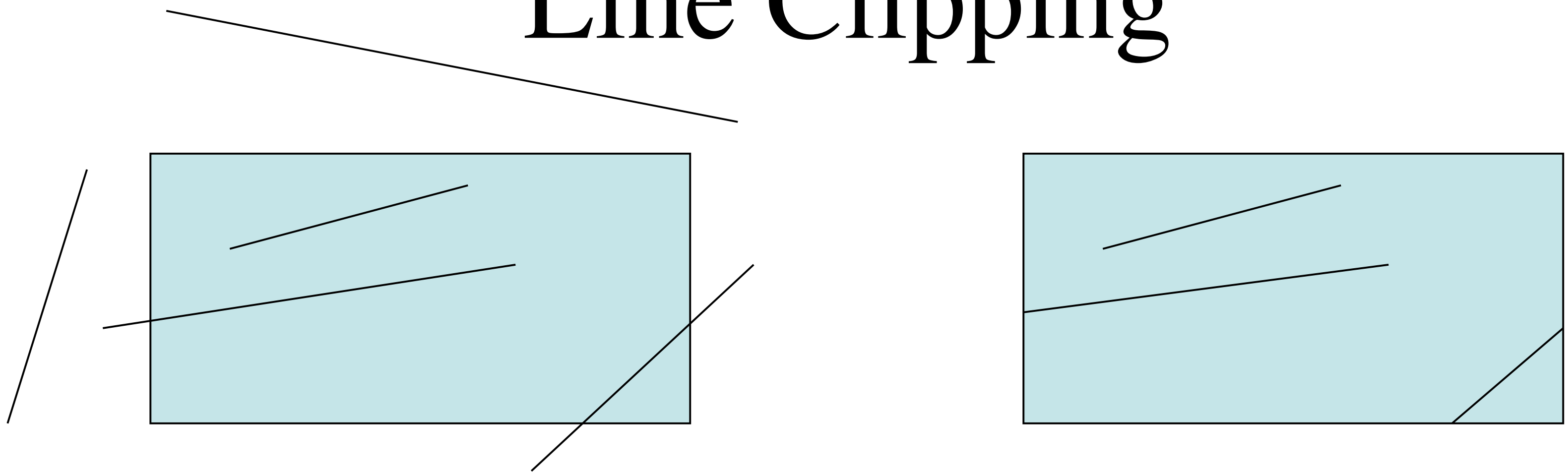
Line Clipping

CS116A
Chris Pollett
Nov. 15, 2004.

Outline

- 2D Line Clipping
- Cohen-Sutherland Line Clipping
- Liang-Barsky Line Clipping
- Nicholl-Lee-Nicholl Clipping

Line Clipping



- Above clipping example shows some possibilities for what can happen to a line when we clip.
- A first step in clipping is to get rid of line segments that do not cross the clipping window at all.
- One can do a first pass at this by doing point tests on endpoints of the line segment. If both points outside any one of the four boundaries then eliminate the line.

Parametric Line Segments and Edge Intersection

- One can represent a line segment with two equations:

  $x = x0 + u(xend - x0)$

  $y = y0 + u(yend - y0)$

- Then one can check if the segment crosses xwmin boundary by plugging xwmin into the x equation and seeing if the value for u is between 0 and 1. If crosses then use this value of u to get a shorter line segment and process against other borders.
- Above idea allows one to do somewhat inefficient clipping

Cohen-Sutherland Line Clipping

- Popular clipping algorithm.
- Each line endpoint is given a four-bit code:
  - Bit 0 -- Left , Bit 1 --Right, Bit 2 -- Bottom, Bit 3 -- Top
- The bit being on indicates point is outside that boundary

More Cohen-Sutherland

- A line segment is completely inside the clipping region if both its codes are 0000. These segments are just saved
- Any segment both of whose endpoints share a 1 in same bit position is outside of the region and are clipped. One can check this by ANDing.
- All other segments must be checked as before to see where intersect

Liang-Barsky Line Clipping

- Consider:

  x = x0 + u*dx

  y = y0 + u*dy where dx = xend-x0 and dy =yend-y0
- Want values:

  xwmin <= x0 + u*dx <= xwmax

  ywmin <= y0 + u*dx <= ywmax
- Can rewrite these conditions as: u*p_k <= q_k where k=1,2,3,4 and p_1 = -dx, p_2 = dx, p_3 = -dy, p_4 = dy and q_1 = x0 - xwmin, q_2 = xwmax -x0, q_3 = y0 - ywmin, q_4 = ywmax - y0

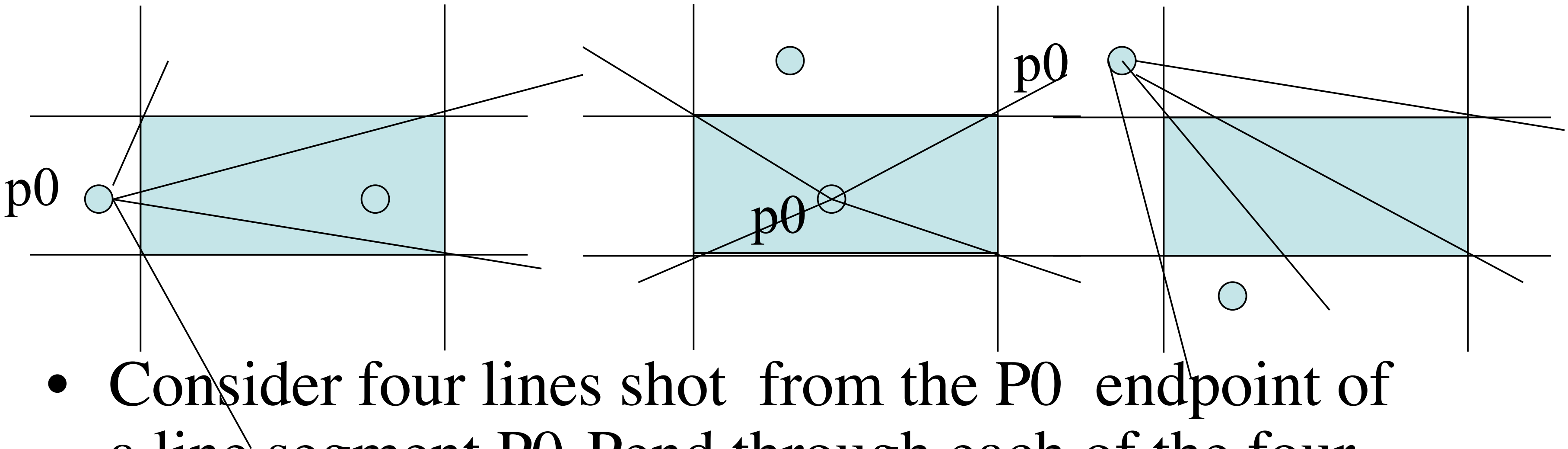
More Liang-Barsky

- Note if $p_k = 0$ for any k line must be parallel to one of the boundaries and problem is easy.
- Note if $p_k < 0$ line proceeds from inside to outside given boundary following u until $u*p_k = q_k$. If $p_k > 0$ line proceeds from outside to inside
- For k such that $p_k < 0$ we compute $r_k = q_k/p_k$. Let u1 = max of these r_k and 0.
- For k such that $p_k > 0$ we compute $r_k = q_k/p_k$ again. Let u2 = min of these r_k and 1.
- If $u_1 > u_2$ then the line is outside the clipping window. Otherwise, u1 and u2 can be used to get intersection

Nicholl-Lee-Nicholl Line Clipping

- Does the least number of comparisons and divisions.
- Unlike other two doesn’t extend well to 3D.
- The algorithm:
  - Does a region testing like C-S to see if line segment can be easily accepted or rejected
  - If not, we set up additional regions to do testing.

More NLN Clipping



- Consider four lines shot from the P0 endpoint of a line segment P0-Pend through each of the four corners of clipping region.
- Determine which of these four new regions Pend lives in by comparing slopes of P0Pend with those of four other lines.
- Now use the at most two boundary edges to do clipping

Line Clipping using NonRectangular Polygon Clip Windows

- Can add additional edges to a concave clipping regions to make it into a set of convex ones.
- Then can use an extension of Liang-Barsky to clip in these convex regions