**LOGIC FOR NICHOLL-LEE-NICHOLL LINE CLIPPING:**

One of the problems common to both the Cohen-Sutherland and the Liang-Barsky algorithm is that more intersections are computed than necessary. For example, consider

Figure 3.6 again where we are clipping line segment [**P**1,**P**2] against the window.

The Cohen-Sutherland algorithm will compute the intersection of the segment with the top boundary at t4 even though the segment is later rejected. The Liang-Barsky algorithm will actually compute allthe parameter values corresponding to the intersection of the line with the window. Avoiding many of these wasted computations is what the Nicholl-Lee-Nicholl line-clipping algorithm ([NiLN87]) is all about

The authors also make a detailed analysis of the deficiencies of the Cohen-Sutherland and Liang-Barsky algorithms. Their final algorithm is much faster than either of these. It is not really much more complicated conceptually, but involves many cases. We

Describe one basic case below.

Assume that we want to clip a segment [**P**1, **P2**] against a window. The determination of the exact edges, if any, that one needs to intersect, reduces, using symmetry, to an analysis of the three possible positions of **P**1 shown in Figure 3.8. The cases are

(1) **P**1 is in the window (Figure 3.8(a)),

(2) **P**1 is in a “corner region” (Figure 3.8(b)), or

(3) **P**1 is in an “edge region” (Figure 3.8(c)).

For each of these cases one determines the regions with the property that no matter where in the region the second point **P**2 is, the segment will have to be intersected with the **same** boundaries of the window. These regions are also indicated in Figure 3.8. As one can see, these regions are determined by drawing the rays from **P**1 through the four corners of the window. The following abbreviations were used:

T – ray intersects top boundary

LT – ray intersects left and top boundary

L – ray intersects left boundary

LR – ray intersects left and right boundary

B – ray intersects bottom boundary

LB – ray intersects left and bottom boundary

R – ray intersects right boundary

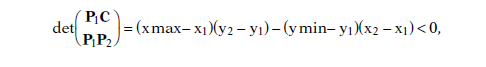
TR – ray intersects top and right boundary

TB – ray intersects top and bottom boundary

For example, suppose that the segment [**P**1,**P**2] is as shown in

Figure 3.8(c). Here are the computations one has to perform. Let **P**i = (xi,yi) and let **C** = (xmax,ymin) be the corner of the window also indicated in Figure 3.8(c). After checking that y2 < ymin,

we must determine whether the vector **P**1**P**2 is above or below the vector **P**1**C**. This reduces to determining whether the ordered basis (**P**1**C**,**P**1**P**2) determines the standard orientation of the plane or not. Since



**P**1**P**2 is below **P**1**C**. We now know that we will have to compute the intersection of

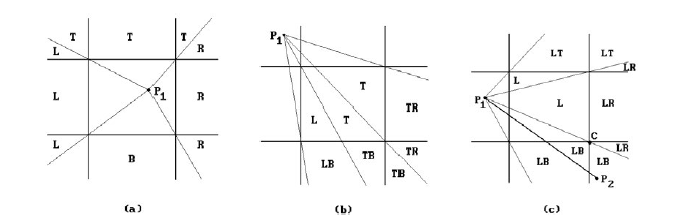
[**P**1,**P**2] with both the left and bottom boundary of the window.

Algorithm 3.2.4.1 is an abstract program for the Nicholl-Lee-Nicholl algorithm in

the edge region case (**P**1 in the region shown in Figure 3.8(c)). We assume a window

[xmin,xmax]\* [ymin,ymax].

Here is the diagram for figure 3.8,



ABSTRACT PROGRAM:

**procedure** LeftEdgeRegionCase (**ref real** x1, y1, x2, y2; **ref boolean** visible)

**begin**

**real** dx, dy;

**if** x2 < xmin

**then** visible := **false**

**else if** y2 < ymin

**then** LeftBottom (xmin,ymin,xmax,ymax,x1,y1,x2,y2,visible)

**else if** y2 > ymax

**then**

**begin**

{ Use symmetry to reduce to LeftBottom case }

y1 := -y1; y2 := -y2; { reflect about x-axis }

LeftBottom (xmin,-ymax,xmax, -ymin,x1,y1,x2,y2,visible);

y1 := -y1; y2 := -y2; { reflect back }

**end**

**else**

**begin**

dx := x2 - x1; dy := y2 - y1;

**if** x2 > xmax **then**

**begin**

y2 := y1 + dy\*(xmax - x1)/dx; x2 := xmax;

**end**;

y1 := y1 + dy\*(xmin - x1)/dx; x1 := xmin;

visible := **true**;

**end**

**end**;

**procedure** LeftBottom ( **real** xmin, ymin, xmax, ymax;

**ref real** x1, y1, x2, y2; **ref boolean** visible)

**begin**

**real** dx, dy, a, b, c;

dx := x2 - x1; dy := y2 - y1;

a := (xmin - x1)\*dy; b := (ymin - y1)\*dx;

**if** b > a

**then** visible := **false** { (x2,y2) is below ray from (x1,y1) to bottom left corner }

**else**

**begin**

visible := **true**;

**if** x2 < xmax

**then**

**begin** x2 := x1 + b/dy; y2 := ymin; **end**

**else**

**begin**

c := (xmax - x1)\*dy;

**if** b > c

**then** { (x2,y2) is between rays from (x1,y1) to

bottom left and right corner }

**begin** x2 := x1 + b/dy; y2 := ymin; **end**

**else**

**begin** y2 := y1 + c/dx; x2 := xmax; **end**

**end**;

**end**;

y1 := y1 + a/dx; x1 := xmin;

**end**;