LESSON 10:
CYRUS–BECK ALGORITHM AND COHEN–SUTHERLAND ALGORITHM

Today'sTopics

- Cyrus beck algorithm.

Cyrus-Beck Techniques (1978): A Parametric Line Clipping Algorithm

Cohen-Sutherland algorithm can only trivially accept or reject lines within the given bounding volume; it cannot calculate the exact intersection point. But, the parametric line clipping algorithm can calculate the value of the parameter t , where the two lines intersect. This can be easily understood by looking at the following picture and pseudo code:

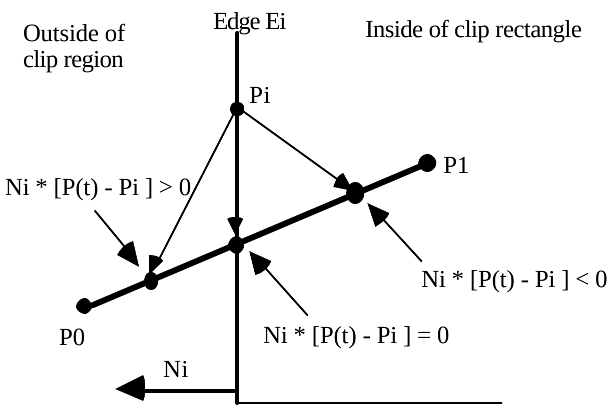


Figure 1: Dot Product for 3 points outside, inside, and on the boundary of the clip region.

The line is parametrically represented by P(t) = P0 + t (P1 - P0)

%% Pseudo Code for Cyrus Beck Parametric Line-Clipping Algorithm

```
{
        precalculate Ni and select a Pi for each edge Ei
        for (each line segment to be clipped) {
                if (P1 = P0)
                        line is degenerate so clip as a point;
                else {
                        D = P1 - P0;
                        te = 0;
                        tl  = 1;
                        for (each candidate intersection with
a clip edge) {
                  if (Ni * D # 0)  {
                        t =  - { Ni * [P0 - Pi] }  /  (Ni * D)
                              if (Ni * D > 0)
                                    tl = min (tl, t);
                                 else
                                 te = max (te, t);
                                    }
                                 }

                                 if (te > tl)
                                    return nil;
                                 else
      return P(te) and P(tl) as true clip intersection points;
                              }
                  }
}
```

The Cyrus-Beck Algorithm

The basic idea of the algorithm (Cyrus-Beck) is as follows:

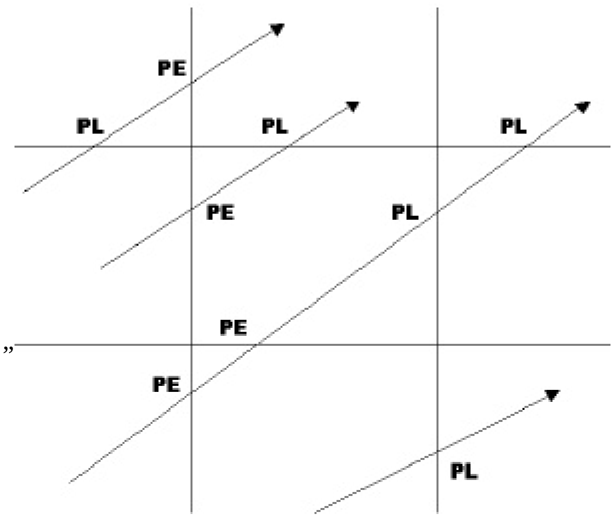
The line to be clipped is expressed using its parametric representation. For each edge of the clipping polygon, we are given a point P_e on the edge, and an outward-pointing normal N. (The vertices of the clipping polygon are traversed in the counterclockwise direction.) The objective is to find the t values where the line enters and leaves the polygon (t_E and t_L), or to determine that the line lies entirely outside the polygon.

t_E is initialized to 0; t_L is initialized to 1.

The t values of the intersection points between the line and the clip edges are determined.

For Each t Value:

Classify it as "potentially entering" (PE) or "potentially leaving" (PL). It is potentially entering if P_0P_1 is (roughly) in the direction opposite to the normal; that is, if $(P_1 - P_0)$ l $N < 0$. (Note that this is the denominator of the expression used to compute t.) It is potentially leaving if $(P_1 - P_0)$ l $N > 0$, indicating that the line P_0P_1 is pointing (roughly) in the same direction as the normal.



for each value of t {

if the line is PE at that intersection point {

 if $t > t_L$ then the line lies entirely outside the clip polygon, so it can be rejected;

 else $t_E = \max(t,t_E)$;

}

else if the line is PL at that intersection point {

 if $t < t_E$ then the line lies entirely outside the clip polygon, so it can be rejected;

 else $t_L = \min(t,t_L)$;

}

}

if the line has not been rejected, then t_E and t_L define the endpoints of the clipped line.

The Liang-Barsky version of the algorithm recognizes athat if the clipping polygon is an upright polygon bounded by x_{xmin}, x_{max}, y_{min}, and y_{max}, the calculations can be simplified. The normal vectors are (1,0), (0,1),(-1,0), and (0,-1). The points P_e can be chosen as $(x_{max},0)$, $(0,y_{max})$,$(x_{min},0)$, and $(0,y_{min})$. The values of $(P_1 - P_0)$ l N are $(x_1\text{-}x_0)$, $(y_1\text{-}y_0)$, $(x_0\text{-}x_1)$, and $y_0\text{-}y_1)$. The t values at the intersection points are $(x_{max}\text{-}x_0)/(x_1\text{-}x_0)$, $(y_{max}x\text{-}y_0)/(y_1\text{-}y_0$ $(x_0 - x_{min})/(x_0\text{-}x_1)$, and $(y_0\text{-}y_{min})/(y_0\text{-}y_1)$.

References:

- Computer Graphics, C Edition, Hearn D. & Baker, M.P. (Prentice-Hall)
- Computer Graphics Principles and Practice, Foley J.D., van Dam A., Feiner S.K., & Hughes J.F., (Addison-Wesley).
- Computer Graphics, Hill F.S. (Macmillan).
- Fundamentals of Three-Dimensional Computer Graphics, Second Edition, Watt A., (Addison-Wesley
- http://research.microsoft.com/~hollasch/cgindex/index.html
- http://graphics.csail.mit.edu/classes/6.837/F98

Notes