COMPUTER GRAPHICS (RCS-603)

Nicholl-Lee-Nicholl (NLN) Line Clipping & Line Clipping Using Non-Rectangular Clip Windows

Date: 28/03/2020

Nicholl-Lee-Nicholl (NLN) Line Clipping

- By creating more regions around the clip window, this algorithm avoids multiple clipping of an individual line segment.
- In Cohen-Sutherland, for example, multiple intersections may be calculated along the path of a single line before an intersection on the clipping rectangle is located or the line is completely rejected.
- The extra intersection calculations in Cohen-Sutherland method are eliminated in NLN algorithm by carrying out more region testing before intersection positions are calculated.
- Compared to Cohen-Sutherland and Liang-Barsky algorithms, NLN algorithm performs fewer comparisons and divisions.
- The trade-off is that NLN can only be applied to 2D clipping, whereas, both previously discussed algorithms are easily extended to 3D scenes.

Nicholl-Lee-Nicholl (NLN) Line Clipping

- For a line with end points $\mathbf{P_1}$ and $\mathbf{P_2}$, we first determine the position of point $\mathbf{P_1}$ for the nine possible regions relative to the clipping rectangle. Only the three regions shown in the given figure need to be considered.
- If $\mathbf{P_1}$ lies in any one of the other six regions, we can move it to one of the three regions using symmetry transformation.
- For example, the region directly above the clip window can be transformed to the region left of the clip window using a reflection about the line y=-x, or we could use a **90°** counterclockwise rotation.

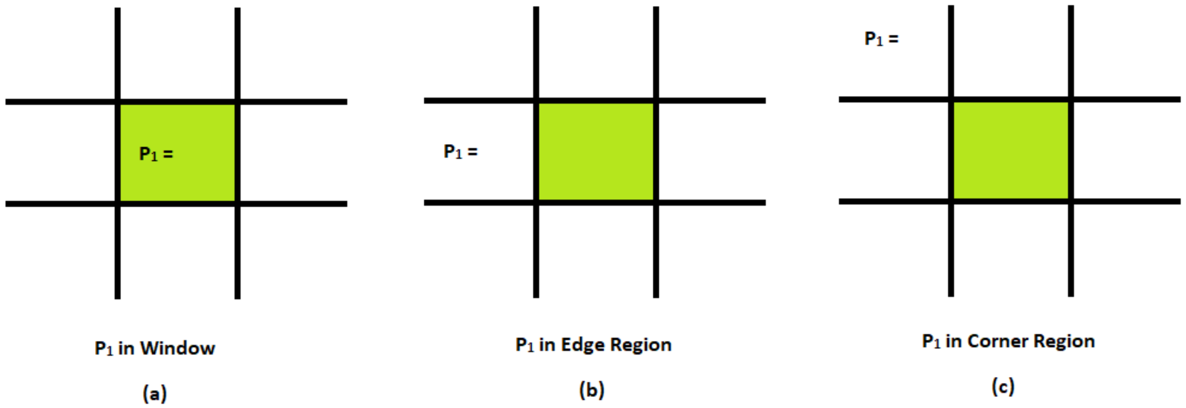


Fig: Three possible positions for a line end point P_1 in the NLN line clipping algorithm.

Nicholl-Lee-Nicholl (NLN) Line Clipping

- Next we determine the position of $\mathbf{P_2}$ relative to $\mathbf{P_2}$. To do this, we create some new regions in the plane, depending on the location of $\mathbf{P_1}$.
- Boundaries of the new regions are half-infinite line segments that start at the position of $\mathbf{P_1}$ and pass through the window corners.
- If $\mathbf{P_1}$ **is inside the clip window and** $\mathbf{P_2}$ is outside, we set up four regions as shown.
- The intersection with the appropriate window boundary is then carried out, depending on which one of the four regions (L, T, R, or B) contains $\mathbf{P_2}$.
- If both $\mathbf{P_1}$ and $\mathbf{P_2}$ are inside the clipping rectangle, we simply save the entire line.

Nicholl-Lee-Nicholl (NLN) Line Clipping

- If $\mathbf{P_1}$ is in the region to the left of the window, we set up four regions, L, LT, LR, and LB.
- These four regions determine a unique boundary for the line segment. For example, if the $\mathbf{P_1}$ is in region L, we clip the line at the left boundary and save the line segment from this intersection point to $\mathbf{P_2}$.
- But if $\mathbf{P_2}$ is in region LT, we save th eline segment from the left window boundary to the top boundary.
- If $\mathbf{P_2}$ is not in any of the four regions, L, LT, LR, or LB, the entire line is clipped.
- For the third case, when $\mathbf{P_1}$ is to the left and above the clip window, we use the clipping regions. In this case, we have two possibilities shown, depending on the position of $\mathbf{P_1}$ relative to the top left corner of the window.

Nicholl-Lee-Nicholl (NLN) Line Clipping

- If **P_2** is in one of the regions T, L, TR, TB, LR, or LB, this determines a unique clip-window edge for the intersection calculations. Otherwise, the entire line is rejected.
- To determine the region in which **P_1** is located, we compare the slope of the line to the slopes of the boundaries of the clip regions.

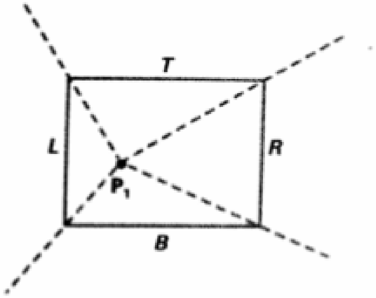


Figure 6-11
The four clipping regions used in the NLN algorithm when P_1 is inside the clip window and P_2 is outside.

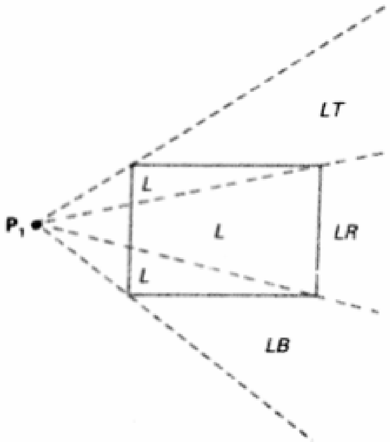


Figure 6-12
The four clipping regions used in the NLN algorithm when P_1 is directly left of the clip window.

For example, if $\mathbf{P_1}$ is left of the clipping rectangle, then $\mathbf{P_2}$ is in regions LT if:

$$\text{slope } \overline{\mathbf{P}_1\mathbf{P}_{TR}} < \text{slope } \overline{\mathbf{P}_1\mathbf{P}_2} < \text{slope } \overline{\mathbf{P}_1\mathbf{P}_{TL}} \tag{6-14}$$

or

$$\frac{y_T - y_1}{x_R - x_1} < \frac{y_2 - y_1}{x_2 - x_1} < \frac{y_T - y_1}{x_L - x_1} \tag{6-15}$$

And we clip the entire line if

$$(y_T - y_1)(x_2 - x_1) < (x_L - x_1)(y_2 - y_1) \tag{6-16}$$

The coordinate difference and product calculations used in the slope tests are saved and also used in the intersection calculations. From the parametric equations

$$x = x_1 + (x_2 - x_1)u$$

$$y = y_1 + (y_2 - y_1)u$$

an x-intersection position on the left window boundary is $x = x_L$, with $u = (x_L - x_1)/(x_2 - x_1)$, so that the y-intersection position is

$$y = y_1 + \frac{y_2 - y_1}{x_2 - x_1}(x_L - x_1)$$

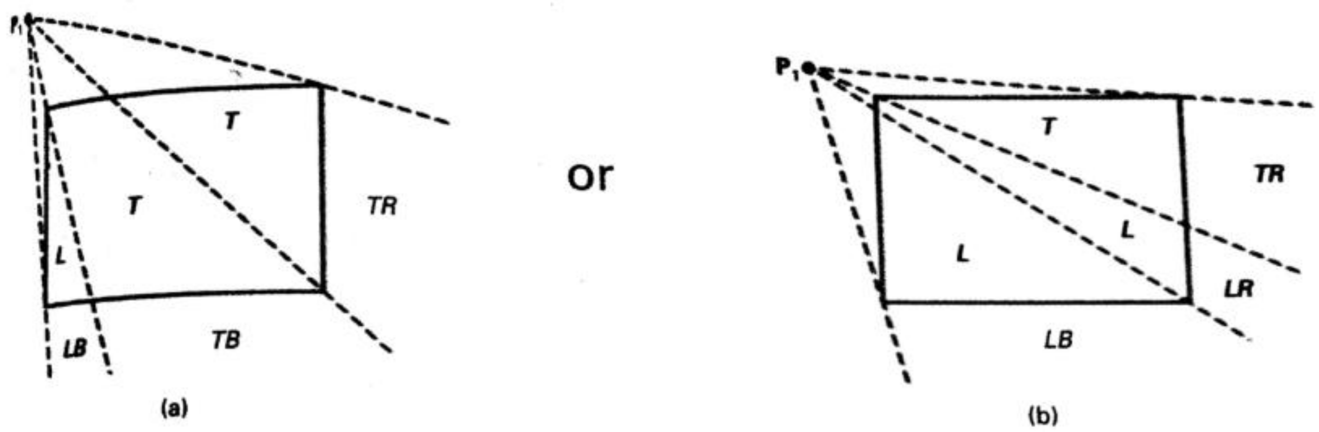


Figure 6-13
The two possible sets of clipping regions used in the NLN algorithm when P_1 is above and to the left of the clip window.

And an intersection position on the top boundary has $y = y_T$ and $u = (y_T - y_1)/(y_2 - y_1)$, with

$$x = x_1 + \frac{x_2 - x_1}{y_2 - y_1}(y_T - y_1) \qquad (6\text{-}18)$$

Line Clipping Using Non-Rectangular Clip Windows

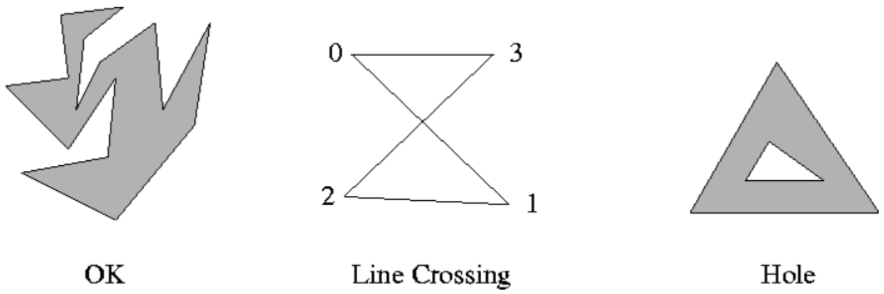
- I some applications, it's often necessary to clip lines against arbitrary shaped polygons. Algorithms based on parametric line equations, such as the Liang-Barsky method can be extended easily to convex polygon windows.
- We can do this by modifying the algorithm to include parametric equations for the boundaries of the clip region.
- Preliminary screening of line segments can be accomplished by processing lines against the coordinate extents of the clipping polygon. For concave polygon-clipping regions, we can still apply these parametric clipping procedures if we first split the concave polygon into a set of convex polygons.

Line Clipping Using Non-Rectangular Clip Windows

- Circles or other curved boundary clipping regions are also possible but less commonly used. Clipping algorithms for these are slower because intersection calculations involve nonlinear curve equations.
- At the first step, lines can be clipped against the bounding rectangle (coordinate extents) of the curved clipping region.
- Lines that can be identified as completely outside the bounding rectangle are discarded.
- To identify, inside lines, we can calculate the distance of the line endpoints from the circle center. If the square of this distance for both endpoints of a line I sless than or equal to the radius squared, we can save the entire line.
- The remaining lines are then processed through the intersection calculations, which must solve simultaneous circle-line equations.

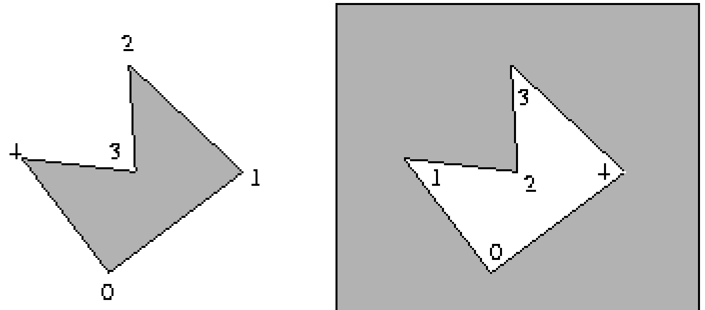
Polygons

- Need an area primitive
- Simple polygon:
  - Planar set of ordered points, V_0,,…, V_{n-1}, (sometimes we repeat V_0 at end of the list)
  - No holes
  - No line crossing



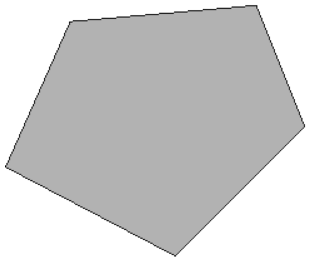
Polygons

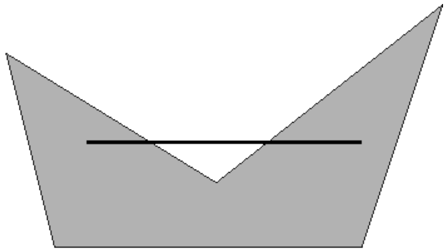
- Normally defined using interior and exterior points (ordered in counter-clockwise order)
- To the “left” as we traverse inside



Polygons

- Two types: Convex and Concave (We prefer *Convex* polygons to *Concave* polygons)
- Polygon is convex if for any two points inside the polygon, the line segment joining these two points is also inside.
- Convex polygons "behave" better when shading, etc
- Although convex polygons normally "well behaved" under affine transformation, affine transformations may introduce degeneracies
- Example: Orthographic projection may project entire polygon to a line segment.





THANK YOU