AN EFFICIENT NEW ALGORITHM FOR 2-D LINE CLIPPING ITS DEVELOPMENT AND ANALYSIS
Authors: T. Nicholl, D. T. Lee, and R. Nicholl
Presentation by Marcus McCurdy
**WHAT IS CLIPPING?**

- Basically finding the intersection of two geometric entities.

![Diagram of clipping](image)
OTHER ALGORITHMS

- Cohen-Sutherland
- Cyrus Beck
- Liang-Barsky
- SPY

In all of these algorithms, intersection points not involved in the final output are calculated.
CLIPPING INEFFECTIVENESS

- SPY and CS will calculate IL before rejecting
- LB will compute all 4
MACHINE INDEPENDENT ANALYSIS

- Evaluated efficiency by counting the number of executions of the following operations:
  - Comparisons
  - Addition
  - Subtraction
  - Multiplication
  - Division
COSTS OF OPERATIONS

- Comparison
- Addition
- Subtraction
- Multiplication
- Division
INVISIBLE
Visible
PARTIALLY VISIABLE
9 Regions

- Top left corner
- Top edge
- Top right corner
- Left edge
- Window
- Right edge
- Bottom left corner
- Bottom edge
- Bottom right corner
ALGORITHM

- Characterize P1’s location among the 9 regions (straightforward).
- Based on P1 characterize P2 among appropriate subdivisions (not quite so straightforward).
- Calculate intersection points according to the characterization.
- Only the intersection points required for output are calculated.
Positions of P1 and P2

- P1 is located in the window, an edge region, or a corner region.
- P2 may be at any point on the 2D Cartesian plane.
- Divide all possible positions of P2 into regions each of which corresponds to intersection points at the same boundaries (or none) to be output.
- The following 3 figures show the subdivisions. All other cases are the same as one of them up to a rotation of a multiple of 90° about the origin or a reflection about the line $x = -y$, or a combination of the two.
Fig. 3 Subdivision of the plane when P1 is in the window
Fig. 4 Subdivision of the plane when P1 is in an edge region
Fig. 5 Subdivision of the plane when P1 is in a corner region
GEOMETRIC TRANSFORMATIONS

- P1 may be in any one of 9 regions
- Different procedure could be written to handle each case
- However, writing similar but not identical code may be error-prone
- Exploit symmetry to prevent this
ALGORITHM

- Find where P1 is in relation to the Left and Right boundaries.
- May be beyond left, beyond right, or between the two.
- Beyond left is symmetrical to beyond right up to a rotation of 180° about the origin and they can be handled together.
P1 is beyond left boundary

- If P2 is also beyond LB then line is invisible stop
- Now find if P1 is beyond top boundary, bottom boundary, or between the two.
- Again, beyond top is symmetrical with beyond bottom, reflection about the x-axis.
P1 IS IN TOP LEFT CORNER

- If P2 is above top stop line is invisible
- Compare P2 against the line joining P1 and the top-left corner
- The case where P2 is on one side of this line is symmetrical to the case in which it is on the other side. Handle one of the two cases.
P1 Top left corner

- P2 not beyond the left and to the right of vector from P1 to the top left corner
- If P2 is above the bottom then no more oblique boundaries need to be computed
- However, if P2 is below bottom 3 possibilities
  - Boundary from line joining P1 to bottom left has to be computed no matter which side of right boundary P2 happens to be on. Do this comparison first
  - If left no more oblique boundaries, else more comparison to decide which subdivision P2 is in
P1 IN LEFT EDGE

- If P2 is beyond left then line is invisible
- P2 may be above top, below bottom, or between top and bottom.
- Again above top is symmetric to below bottom
P1 in left edge & P2 beyond bottom

- Comparing boundary joining P1 to the bottom left corner cannot be avoided so do this one first.
P1 BTW LEFT AND RIGHT

- If P1 above top or below bottom case is symmetrical to that of P1 in the left edge region so use the same procedure
P1 IN THE WINDOW

- If P2 is in an edge region then the intersection point can be seen from figure 3
- If P2 is in a corner region then comparison with one oblique boundary is necessary before appropriate intersection calculation is possible
- Use symmetry to reduce different cases
Fig. 3 Subdivision of the plane when P1 is in the window
ANALYSIS

- Fewest divisions among the algorithms. Equal to the number of intersection points for output.
- Fewest comparisons. About 1/3 CS and about 1/2 LB.
- CS requires largest number of comparison in most cases
- With large window dominating case is when P2 is in window. CS does few set operations, and so LB is slowest
- Small window dominating cases are the four corner regions. CS is slower in this case,
<table>
<thead>
<tr>
<th>CS</th>
<th>LB</th>
<th>NLN</th>
</tr>
</thead>
<tbody>
<tr>
<td>25.5</td>
<td>20</td>
<td>8</td>
</tr>
<tr>
<td>1.5</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>4.5</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>1.5</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>1.5</td>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CS</th>
<th>LB</th>
<th>NLN</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>20</td>
<td>7.5</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>

**COMPARISONS**

<table>
<thead>
<tr>
<th>CS</th>
<th>LB</th>
<th>NLN</th>
<th>P1</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>20</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>6</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>4</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

**FIG. 6** Analysis of the three algorithms when P1 is in the window
FIG. 7 Analysis of the three algorithms when
P1 is in an edge region
FIG. 8 Analysis of the three algorithms when
P1 is in a corner region