FACULTY NAME- PRAVIN KUMAR PANDEY
DEPARTMENT NAME- COMPUTER SCIENCE AND ENGINEERING
SUBJECT- COMPUTER GRAPHICS
SEMESTER- 5
YEAR- 3

UNIT-2

Liang-Barsky Line ClippingAlgorithm

The **Liang-Barsky algorithm** is a line clipping algorithm. This algorithm is more efficient than Cohen–Sutherland line clipping algorithm and can be extended to 3-Dimensional clipping. This algorithm is considered to be the faster parametric line-clipping algorithm. The following concepts are used in this clipping-

- The parametric equation of the line.
- The inequalities describing the range of the clipping window which is used to determine the intersections between the line and the clip window.

The parametric equation of a line can be given by-
$X = x_1 + t(x_2-x_1)$
$Y = y_1 + t(y_2-y_1)$
Where, t is between 0 and 1.

Then, writing the point-clipping conditions in the parametric form:
$xw_{min} <= x_1 + t(x_2-x_1) <= xw_{max}$
$yw_{min} <= y_1 + t(y_2-y_1) <= yw_{max}$

The above 4 inequalities can be expressed as-

$tp_k <= q_k$
Where k = 1, 2, 3, 4 (correspond to the left, right, bottom, and top boundaries, respectively).
The p and q are defined as-
$p_1 = -(x_2-x_1)$, $q_1 = x_1 - xw_{min}$ (Left Boundary)
$p_2 = (x_2-x_1)$, $q_2 = xw_{max} - x_1$ (Right Boundary)
$p_3 = -(y_2-y_1)$, $q_3 = y_1 - yw_{min}$ (Bottom Boundary)
$p_4 = (y_2-y_1)$, $q_4 = yw_{max} - y_1$ (Top Boundary)

1. When the line is parallel to a view window boundary, the p value for that boundary is zero.
2. When $p_k < 0$, as t increase line goes from the outside to inside (entering).
3. When $p_k > 0$, line goes from inside to outside (exiting).
4. When $p_k = 0$ and $q_k < 0$ then line is trivially invisible because it is outside view window.
5. When $p_k = 0$ and $q_k > 0$ then the line is inside the corresponding window boundary.

Using the following conditions, the position of line can be determined:

Condition	Position of Line
$p_k = 0$	parallel to the clipping boundaries

$p_k = 0$ and $q_k < 0$	completely outside the boundary
$p_k = 0$ and $q_k >= 0$	inside the parallel clipping boundary
$p_k < 0$	line proceeds from outside to inside
$p_k > 0$	line proceeds from inside to outside

Parameters t_1 and t_2 can be calculated that define the part of line that lies within the clip rectangle.
When,
$p_k < 0$, maximum(0, q_k/p_k) is taken.
$p_k > 0$, minimum(1, q_k/p_k) is taken.

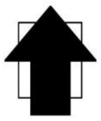
If $t_1 > t_2$, the line is completely outside the clip window and it can be rejected. Otherwise, the endpoints of the clipped line are calculated from the two values of parameter t.

Algorithm –

1. Set t_{min}=0, t_{max}=1.
2. Calculate the values of t (t(left), t(right), t(top), t(bottom)),
   (i) If t < t_{min} ignore that and move to the next edge.
   (ii) else separate the t values as entering or exiting values using the inner product.
   (iii) If t is entering value, set t_{min} = t; if t is existing value, set t_{max} = t.
3. If $t_{min} < t_{max}$, draw a line from ($x_1 + t_{min}(x_2-x_1)$, $y_1 + t_{min}(y_2-y_1)$) to ($x_1 + t_{max}(x_2-x_1)$, $y_1 + t_{max}(y_2-y_1)$).
4. If the line crosses over the window, ($x_1 + t_{min}(x_2-x_1)$, $y_1 + t_{min}(y_2-y_1)$) and ($x_1 + t_{max}(x_2-x_1)$, $y_1 + t_{max}(y_2-y_1)$) are the intersection point of line and edge.

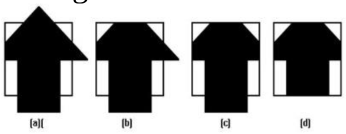
Polygon
A polygon can be defined as a geometric object "consisting of a number of points (called vertices) and an equal number of line segments (called sides), namely a cyclically ordered set of points in a plane, with no three successive points collinear, together with the line segments joining consecutive pairs of the points.

Sutherland Hodgman Polygon Clipping
The Sutherland Hodgman algorithm performs a clipping of a polygon against each window edge in turn. It accepts an ordered sequence of vertices v1, v2, v3, ..., vn and puts out a set of vertices defining the clipped polygon.



This figure represents a polygon (large, solid, upward pointing arrow) before clipping has occurred.

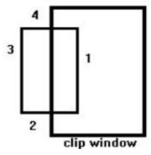
The following figures show how this algorithm works at each edge, clipping the polygon.



- Clipping against the left side of the clip window.
- Clipping against the top side of the clip window.
- Clipping against the right side of the clip window.
- Clipping against the bottom side of the clip window.

Four Types of Edges

As the algorithm goes around the edges of the window, clipping the polygon, it encounters four types of edges. All four edge types are illustrated by the polygon in the following figure. For each edge type zero, one, or two vertices are added to the output list of vertices that define the clipped polygon.



The four types of edges are-

1. Edges that are totally inside the clip window. - add the second inside vertex point
2. Edges that are leaving the clip window. - add the intersection point as a vertex
3. Edges that are entirely outside the clip window. - add nothing to the vertex output list
4. Edges that are entering the clip window. - save the intersection and inside points as vertices

Procedure to Calculate Intersections

Assume that we're clipping a polygon's edge with vertices at (x_1, y_1) and (x_2, y_2) against a clip window with vertices at (x_{min}, y_{min}) and (x_{max}, y_{max}).

The location (IX, IY) of the intersection of the edge with the left side of the window is-

$$IX = x_{min}$$

$$IY = slope*(x_{min}-x_1) + y_1, \text{ where the slope} = (y_2-y_1)/ (x_2-x_1)$$

The location of the intersection of the edge with the right side of the window is-

$$IX = x_{max}$$

$$IY = slope*(x_{max}-x_1) + y_1, \text{ where the slope} = (y_2-y_1)/ (x_2-x_1)$$

The intersection of the polygon's edge with the top side of the window is-

$$IX = x_2 + (y_{max} - y_1) / slope$$

$$IY = y_{max}$$

Finally, the intersection of the edge with the bottom side of the window is-

$$IX = x_1 + (y_{min} - y_1) / slope$$

$$IY = y_{min}$$

Some Problems with This Algorithm

1. This algorithm does not work if the clip window is not convex.
2. If the polygon is not also convex, there may be some dangling edges.

References

[1] https://www.geeksforgeeks.org/liang-barsky-algorithm/
[2] Amrendra N Sinha and Arun D Udai," Computer Graphics", TMH
[3] Udit Agarwal "Computer Graphics", KATSON