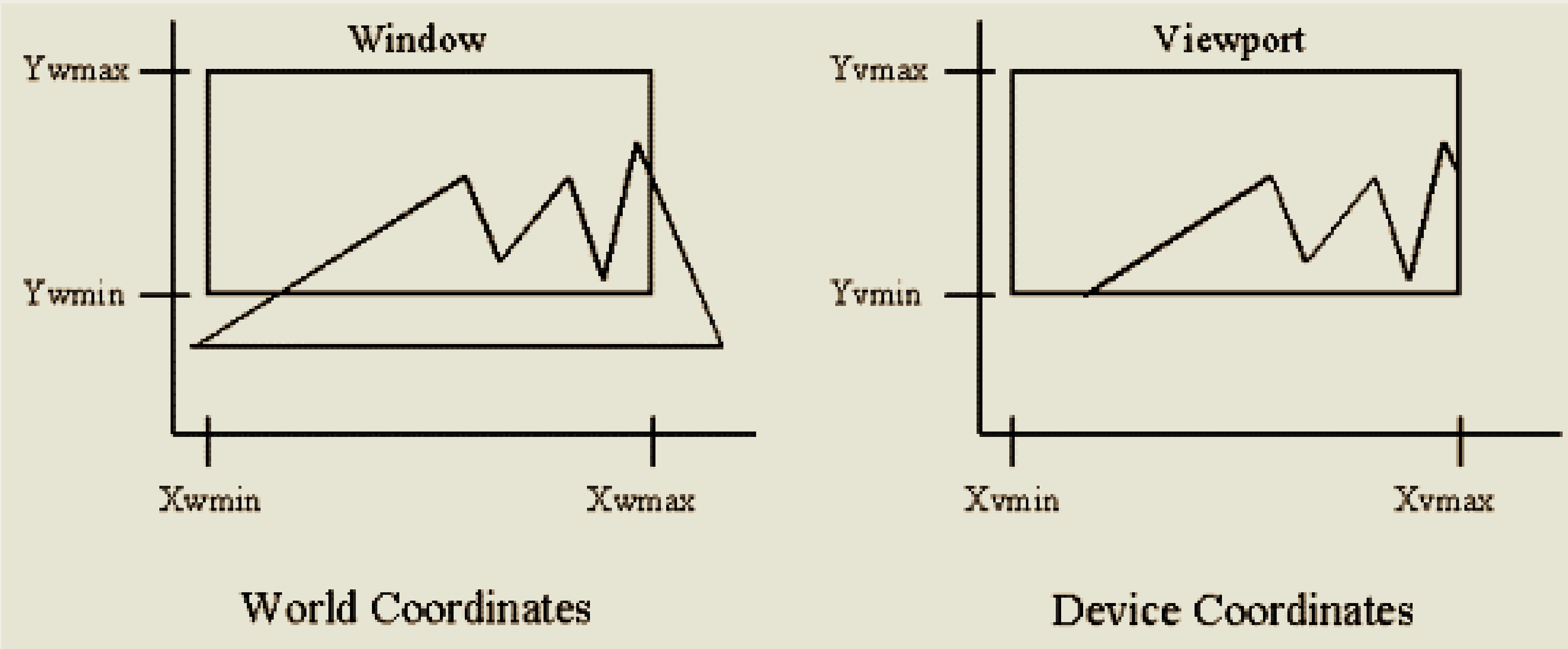
VIEWING & CLIPPING

2D Viewing Transformation

- General Terms:
  - *World Coordinate: It is the cartesian coordinate with respect to which we define the diagram.*
  - *Device Coordinate: It is the screen coordinate where the object is to be displayed.*
  - *Window: Area on world coordinate system selected for display.*
  - *Viewport: It is the area on device coordinate where graphics is to be displayed.*
- Viewing transformation is the process of transforming a 2D world coordinate objects to device coordinates.


Window
Ywmax
Ywmin
Xwmin
Xwmax
World Coordinates
Viewport
Yvmax
Yvmin
Xvmin
Xvmax
Device Coordinates

Window to Viewport Mapping

Approach-1:

- Let (x_w, y_w) be a point on window and (x_v, y_v) be the corresponding point on viewport.
- One Approach is to compute (xv,yv,1) from (xw,yw,1) in terms of translate-> scaling->inverse translate

$$N = \begin{bmatrix} 1 & 0 & -x_{wmin} \\ 0 & 1 & -y_{wmin} \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -x_{vmin} \\ 0 & 1 & -y_{vmin} \\ 0 & 0 & 1 \end{bmatrix}$$

Window to Viewport Mapping

<u>Approach-2:</u>

- Let (x_w, y_w) be a point on window and (x_v, y_v) be the corresponding point on viewport.
- Another approach: We have to calculate the point (x_v, y_v).

Normalized point on window $\left(\frac{x_w - x_{wmin}}{x_{wmax} - x_{wmin}}, \frac{y_w - y_{wmin}}{y_{wmax} - y_{wmin}} \right)$

Normalized point on window $\left(\frac{x_v - x_{vmin}}{x_{vmax} - x_{vmin}}, \frac{y_v - y_{vmin}}{y_{vmax} - y_{vmin}} \right)$

Now the relative position of the object in Window and Viewport are same.

For x coordinate, $\frac{x_w - x_{wmin}}{x_{wmax} - x_{wmin}} = \frac{x_v - x_{vmin}}{x_{vmax} - x_{vmin}}$

For y coordinate, $\frac{y_w - y_{wmin}}{y_{wmax} - y_{wmin}} = \frac{y_v - y_{vmin}}{y_{vmax} - y_{vmin}}$

Window to Viewport Mapping

- After calculating for x and y coordinate, we get,

$x_v = x_{vmin} + (x_w - x_{wmin})\ S_x$

$y_v = y_{vmin} + (y_w - y_{wmin})\ S_y$

Where Sx and Sy are the scaling factors of x and y coordinate respectively.

$$Sx = \frac{x_{vmax} - x_{vmin}}{x_{wmax} - x_{wmin}}$$

$$Sy = \frac{y_{vmax} - y_{vmin}}{y_{wmax} - y_{wmin}}$$

Point Clipping

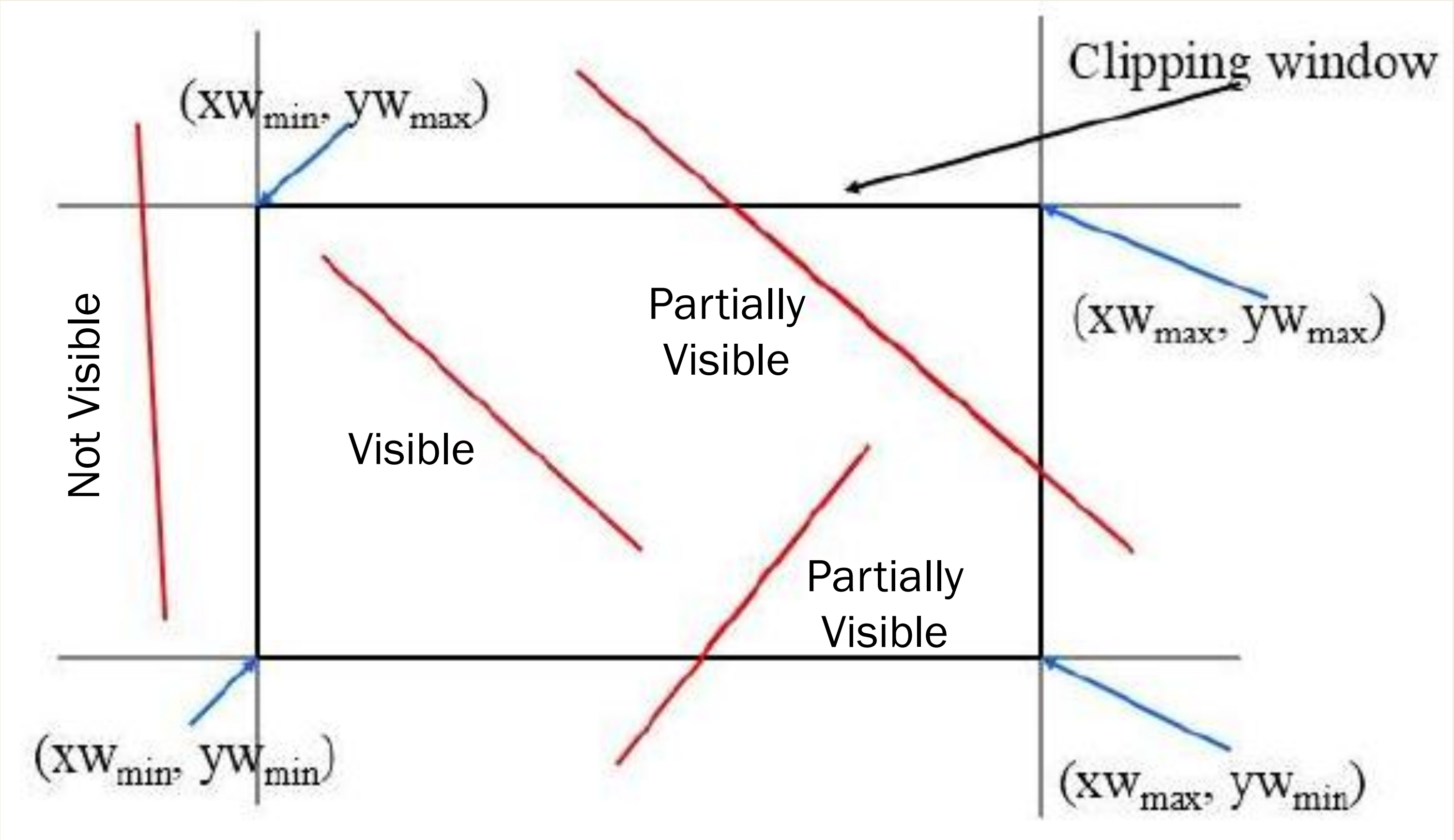
- A point P (x,y) considered inside the window when the following two inequalities evaluate to true

$$x_{min} \leq x \leq x_{max} \text{ and } y_{min} \leq y \leq y_{max}$$

Where x_{min} , x_{max} , y_{min} and y_{min} define clipping window.

Cohen Sutherland Line Clipping Algorithm

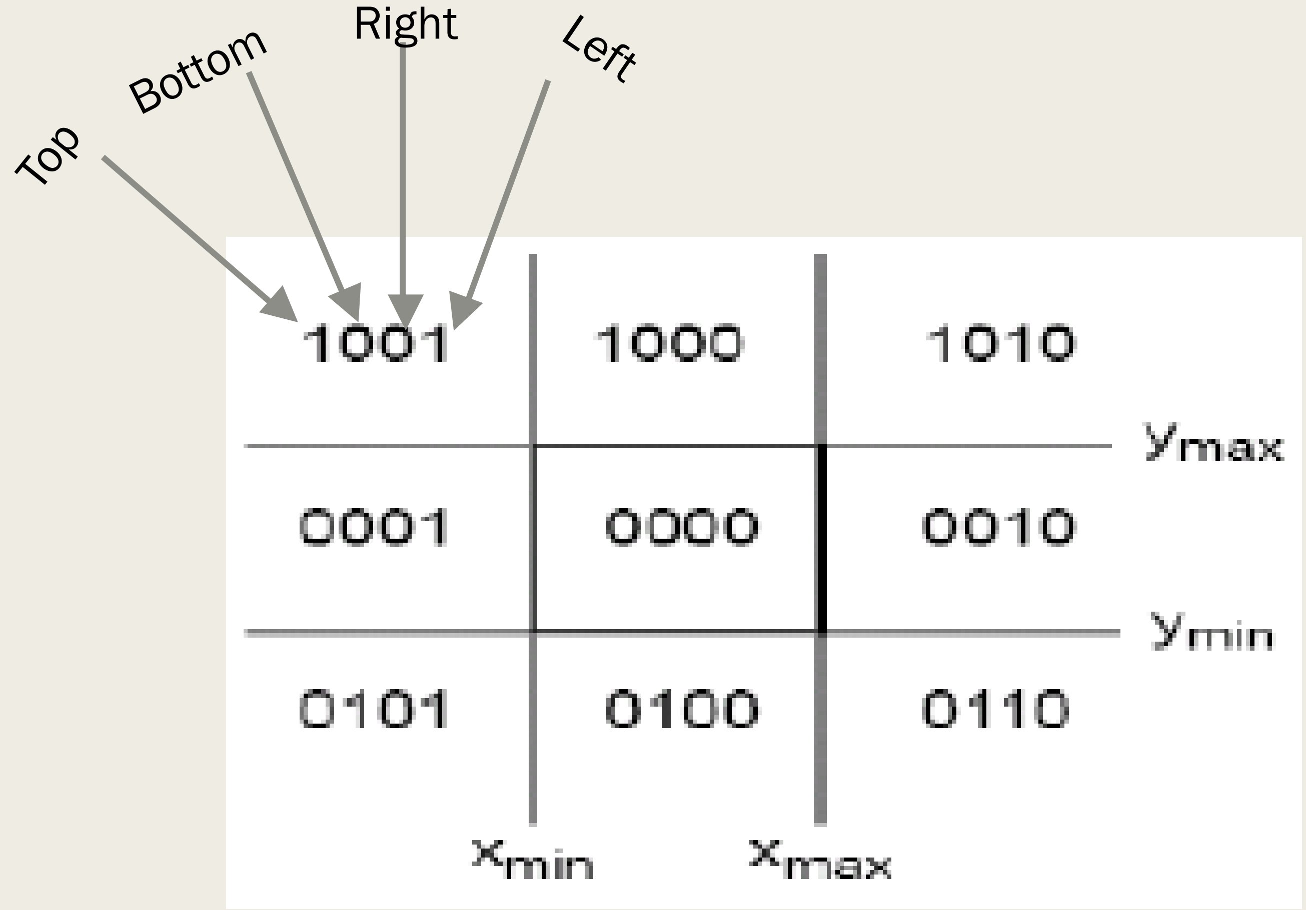
- Line clipping is the process of removing line or portions of lines that lies outside the clipping window.
- A line can be
  - *Visible: if both end points of the line inside the clip window.*
  - *Not visible: if the line lies completely outside the window.*
  - *Clipping candidate (Partially visible): Line intersects the clip window at one or more points.*



Clipping window
(xw_{min}, yw_{max})
(xw_{max}, yw_{max})
(xw_{min}, yw_{min})
(xw_{max}, yw_{min})
Not Visible
Visible
Partially Visible
Partially Visible

Cohen Sutherland Line Clipping Algorithm

- Clipping Process
  - *Identify lines which intersects the clipping window.*
  - *Perform clipping.*
- Finding out the category of a line
  - *Assign a 4 bit region code to 9 region of clipping window (each end point of the line).*
    - These 4 bits represent the top, bottom, right and left.
  - *Determine whether a line is visible, not visible or clipping candidate.*



First bit set 1: Point lies above (top) window $y>y_{max}$
Second bit set 1: Point lies below(bottom) window $y<y_{min}$
Third bit set 1: Point lies right of window $x>x_{max}$
Fourth bit set 1: Point lies left of window $x<x_{min}$

The sequence for reading the code's bits is TBRL (Top, Bottom, Right, Left) [left to right sequence (Bit 4, bit 3, bit 2, bit 1)]

Cohen Sutherland Line Clipping Algorithm

- Line is trivially accepted (visible) if both end points of the region code is 0000.
- Line is trivially rejected (not visible) if bitwise AND of the region codes is not 0000.
- Line is partially accepted (clipping candidate) if bitwise AND of the region codes is 0000.

Cohen Sutherland Line Clipping Algorithm

- Step 1: Assign a region code for two end points of a given line.
- Step 2: If both end points have a region code 000 then accept the line.
- Step 3: Else, perform the logical AND operation for both region codes.
  - *If the result is not 0000, then reject the line.*
- Else line is partially inside.
  - *Determine the intersecting edge of clipping windows as follows:*
    - Assume that the intersection point is (x_i, y_i).
    - Calculate $m=(y_2-y_1)/(x_2-x_1)$

Cohen Sutherland Line Clipping Algorithm

- If bit 1 is '1' line intersects with left boundary of rectangle window

  $y_i=y_1+m(x_i-x_1)$ Where $x_i=x_{min}$
- If bit 2 is '1' line intersects with right boundary of rectangle window

  $y_i=y_1+m(x_i-x_1)$ Where $x_i=x_{max}$
- If bit 3 is '1' line intersects with bottom boundary of rectangle window

  $x_i=x_1+(y_i-y_1)/m$ Where $y_i=y_{min}$
- If bit 4 is '1' line intersects with top boundary of rectangle window

  $x_i=x_1+(y_i-y_1)/m$ Where $y_i=y_{max}$

- Replace the end point with the intersection point .
- Update the region code and recategorized the line.
- Repeat the process until all clipped lines are being accepted.

Cohen Sutherland Line Clipping Algorithm

Limitation:

- It can be used only on a rectangular clip window.
- Unnecessary clipping is done.
- Different Clipping order may take less iterations to finish.

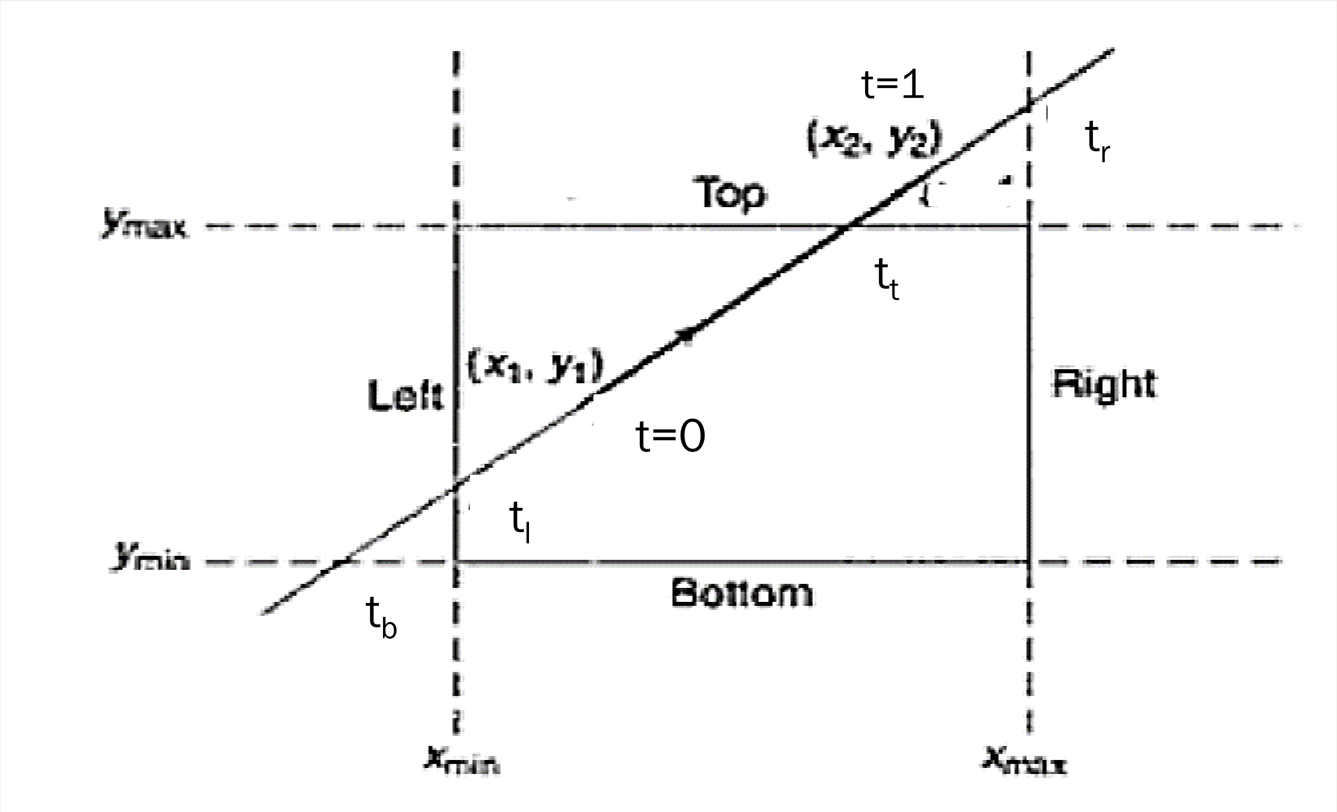
Liang-Barsky Line Clipping Algorithm

- It is more efficient than Cohen Sutherland algorithm.
- The concepts that used in this clipping are:
  - *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.*
- Let assume, we have two endpoints of a line (x_1,y_1) and (x_2,y_2)
- Parametric equation of a line with its infinite extension can be given by

$$x=x_1+t\ (x_2-x_1) = x_1+t\ \Delta x$$

$$y=y_1+t\ (y_2-y_1) = y_1+t\ \Delta x$$

Where t is between 0 and 1.


t=1
(x2, y2)
tr
Top
ymax
tt
(x1, y1)
Left
Right
t=0
tl
ymin
Bottom
tb
xmin
xmax

Liang-Barsky Line Clipping Algorithm

- Point (x,y) inside the clipping window,

  $x_{min} \leq x_1 + t\,\Delta x \leq x_{max}$

  $y_{min} \leq y_1 + t\,\Delta x \leq y_{max}$

- The above 4 inequalities can be expressed as,

  $tp_k \leq q_k$,

Where k=1,2,34 (corresponding to the left, right, bottom, and top boundaries respectively)

- The p and q are defined as,

$p_1 = -\Delta x$ $q_1 = x_1 - x_{min}$ (left boundary)

$p_2 = \Delta x$ $q_2 = x_{max} - x_1$ (right boundary)

$p_3 = -\Delta y$ $q_3 = y_1 - y_{min}$ (bottom boundary)

$p_4 = \Delta y$ $q_4 = y_{max} - y_1$ (top boundary)

Liang-Barsky Line Clipping Algorithm

- When the line is parallel to view window boundary, the p value for that boundary is 0.
  - *When $p_k<0$, line goes from the outside to inside (entering).*
  - *When $p_k>0$, line goes from the inside to outside (exiting).*
  - *When $p_k=0$ and $q_k<0$, completely outside the boundary.*
  - *When $p_k<0$ and $q_k>0$, line is inside the corresponding window boundary.*

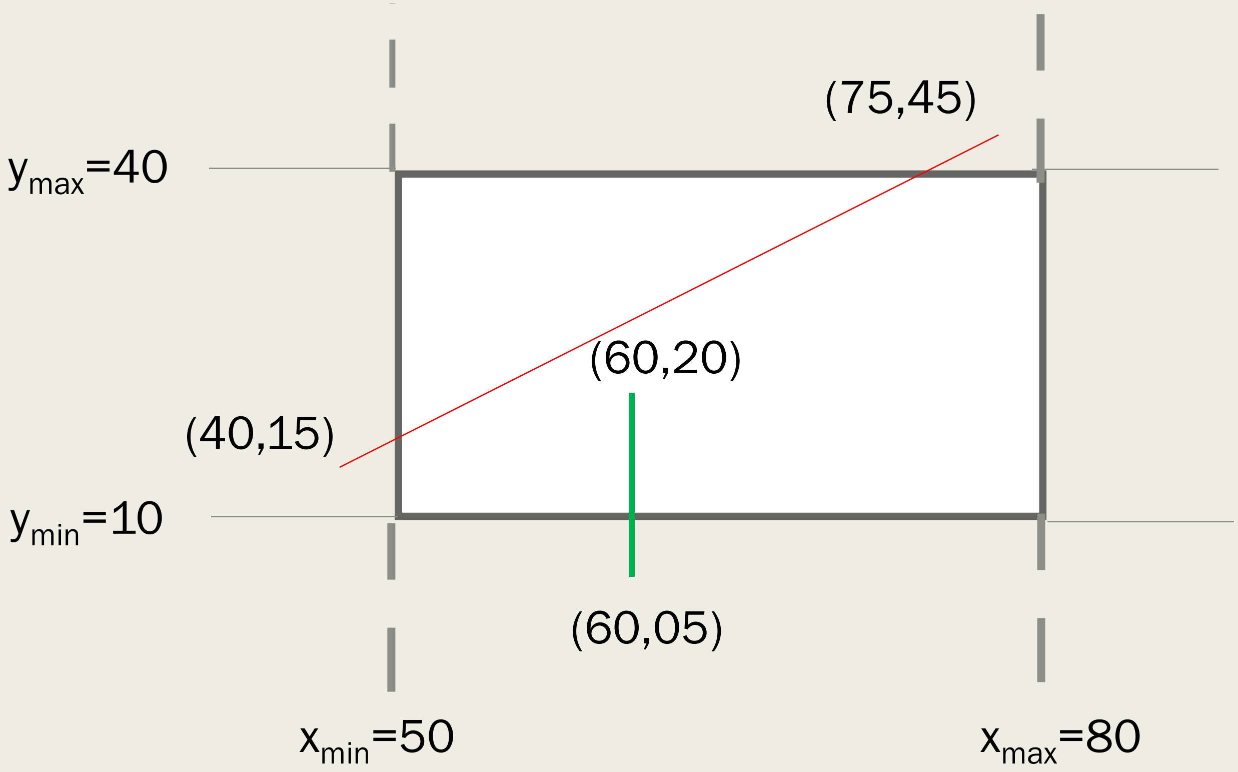
Liang-Barsky Line Clipping Algorithm

Pseudocode:

- Initialize the line intersection parameters t_1=0 and t_2=1.
- If p_k=0 and q_k<0 for any k, eliminate the line and stop. Otherwise proceed to the next step.
- For all k such that p_k<0 , calculate t_1=max(0, q_k/p_k).
- For all k such that p_k>0 , calculate t_2=min(1, q_k/p_k).
- If t_1>t_2, eliminate the line since it is completely outside the clipping window. Otherwise draw a line from ($x_1+t_1\Delta x$, $y_1+ t_1\Delta y$) to ($x_2+t_2\Delta x$, $y_2+ t_2\Delta y$).
- If the line crosses over the window, from ($x_1+t_1\Delta x$, $y_1+ t_1\Delta y$) and ($x_2+t_2\Delta x$, $y_2+ t_2\Delta y$) are the intersection point of the line and edge.

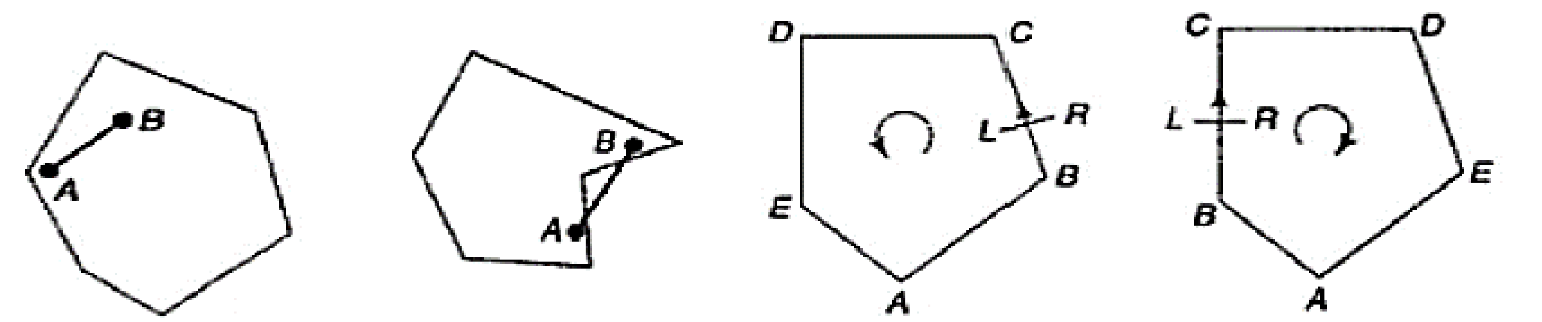
Practice Problem (Line Clipping)

Use the Cohen Sutherland/ Liang-Barsky Algorithm to clip the line in the following figure.



Follow the class lecture for the solution

Polygon Clipping



Convex polygon Concave polygon Positive orientation Negative orientation

Sutherland-Hodgeman Polygon Clipping Algorithm

- The main concept of the algorithm
  - *Consider each edge e of clipping area and do following:*
    - Clip given polygon against e.
- How to clip against an edge of clipping area?
  - *The edge of clipping area is extended infinitely to create to boundary and all the vertices are clipped using this boundary.*
  - *The new list of vertices generated is passed to the next edge of the clip polygon in anti clock direction until all the edge have been used.*

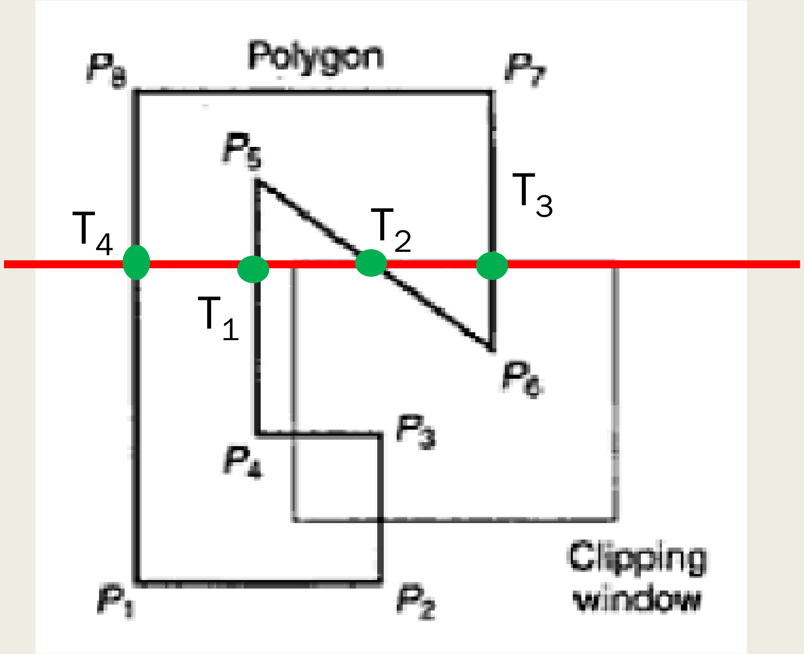
Sutherland-Hodgeman Polygon Clipping Algorithm

- There are possible cases for any given edge of given polygon against current clipping edge e.
  - *Both vertices are inside:*
    - Add only the second vertex (destination) to the output list.
  - *First vertex is outside while second one is inside:*
    - Add intersection point and second vertex to the output list.
  - *First vertex is inside while second one is outside:*
    - Add only intersection point of the edge with clipping boundary in the output list.
  - *Both vertices are outside:*
    - No vertices are added to the output list.

Sutherland-Hodgeman Polygon Clipping Algorithm

Clipping against top boundary: Starting vertex P_1

Edge	Intersection Point	Output List	Remarks
P_1-P_2	-	P_2	Both in
P_2-P_3	-	P_3	Both in
P_3-P_4	-	P_4	Both in
P_4-P_5	T_1	T_1	In->out
P_5-P_6	T_2	T_2, P_6	Out->in
P_6-P_7	T_3	T_3	In->out
P_7-P_8	-	-	Both out
P_8-P_1	T_4	T_4,P_1	Out->in

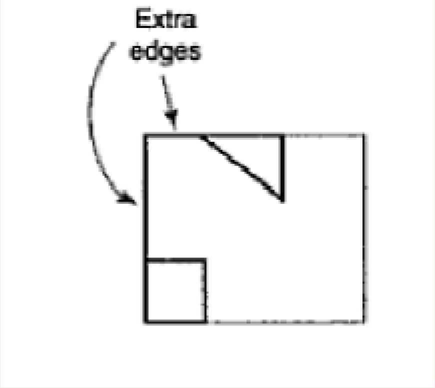


For full simulation of the problem, go through the class recording

Sutherland-Hodgeman Polygon Clipping Algorithm

Limitations:

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



Practice Problems

- Book: Computer Graphics (Schaums Series)-2nd edition.

Solved Problem: 5.1-5.12, 5.15,5.16

- Book: Computer Graphics: Principles and Practice- 2nd Edition, Foley, van Dam, Feiner, Hughes, Chapter 5
- Self Study : Weiler Atherton Polygon Clipping Algorithm

Camera Effect & Double Buffering

- Panning
- Zooming
- Double Buffering

THE HARDEST PART OF ENDING IS STARTING AGAIN.