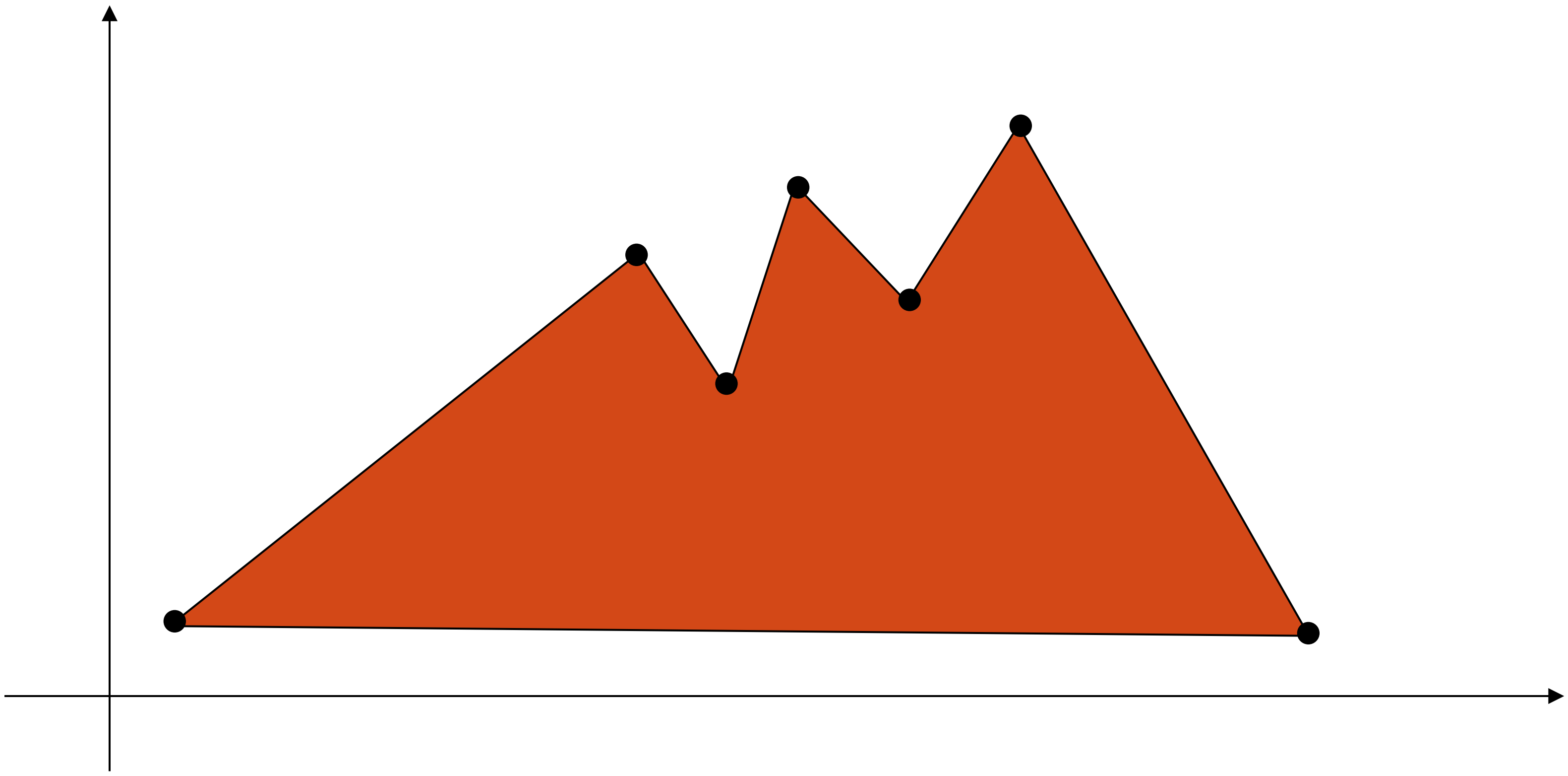
Clipping Algorithm

Week7

Windowing I

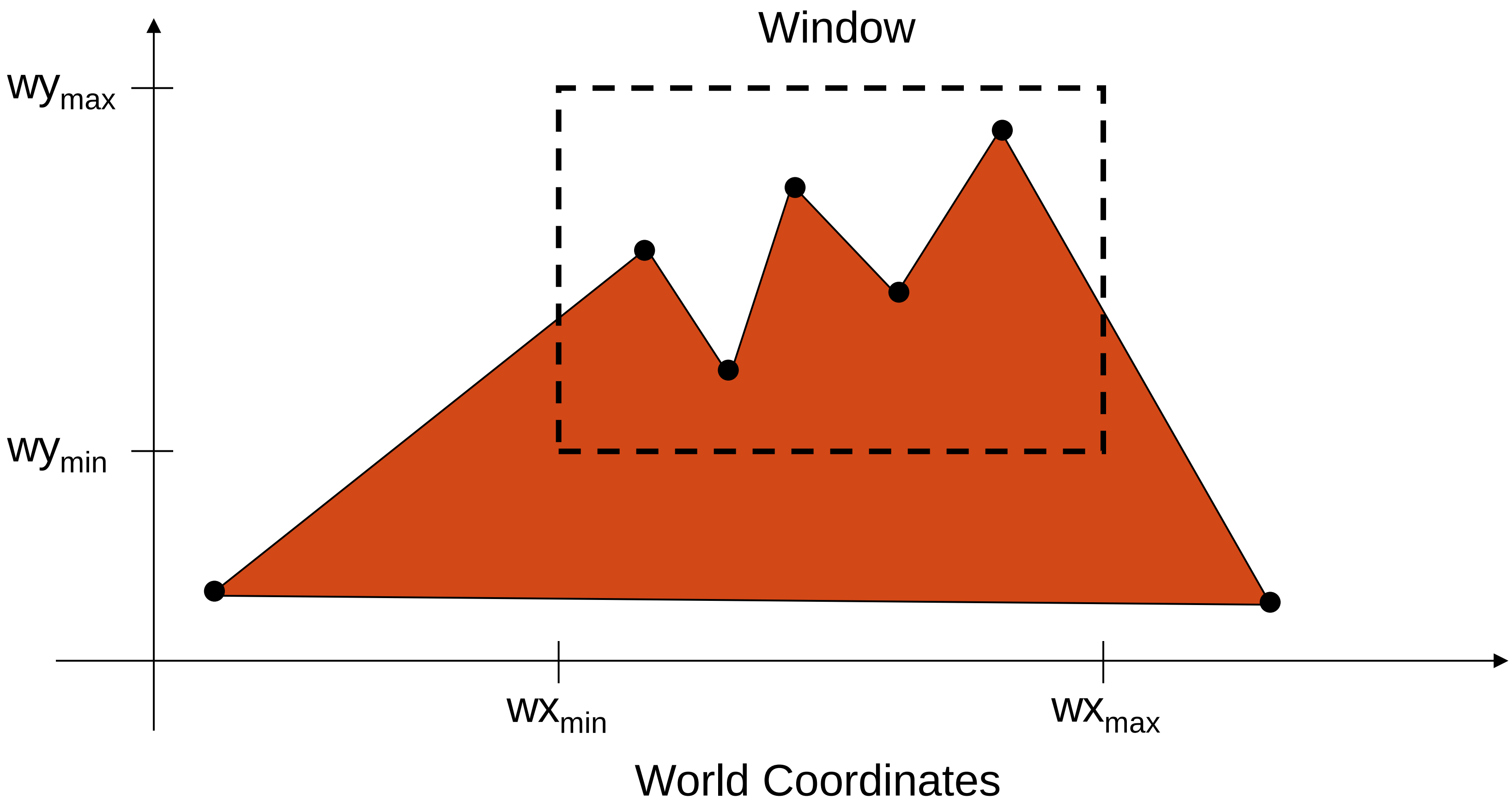
- A scene is made up of a collection of objects specified in world coordinates



World Coordinates

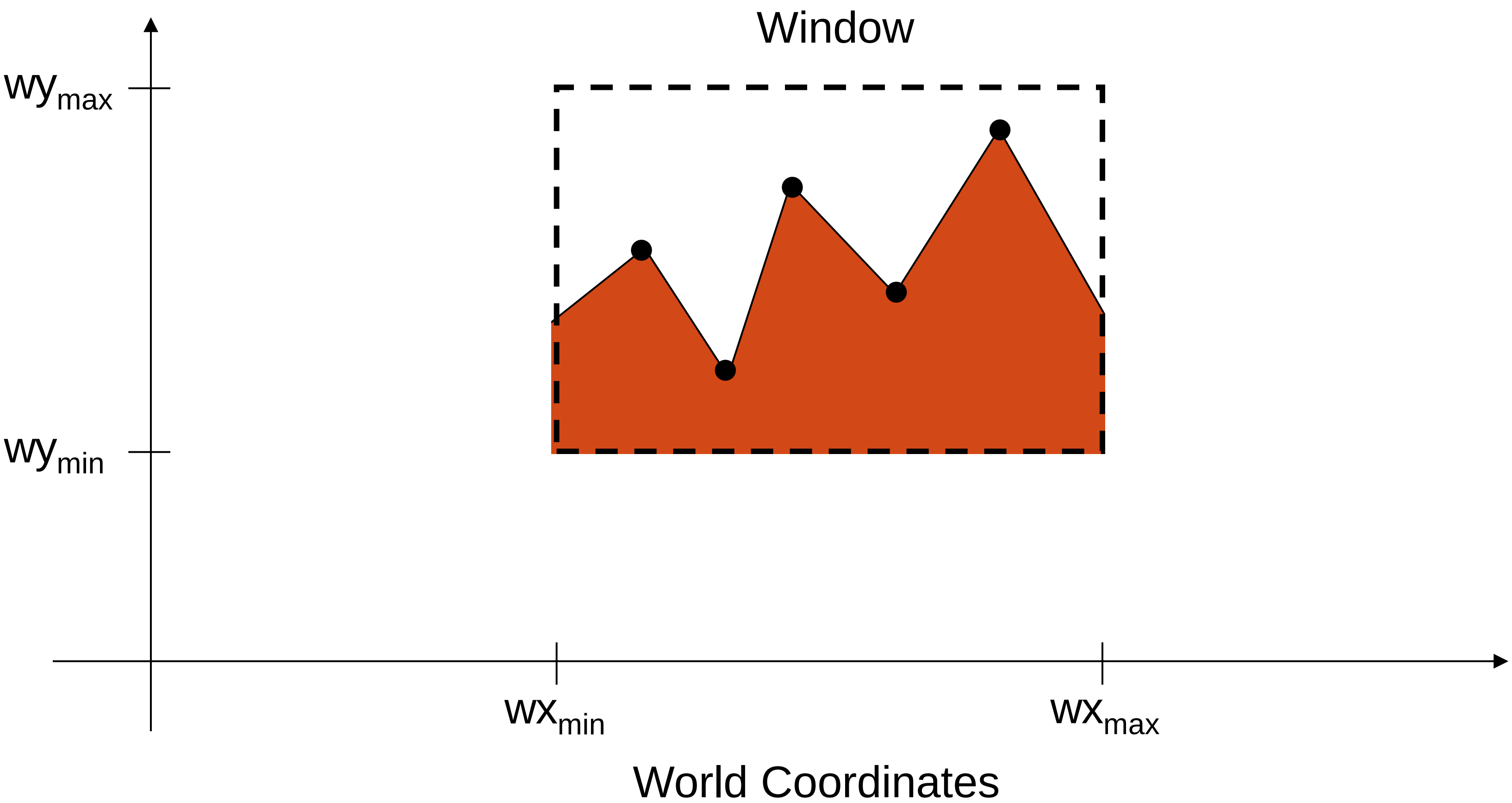
Windowing II

- When we display a scene only those objects within a particular window are displayed



Windowing III

- Because drawing things to a display takes time we *clip* everything outside the window

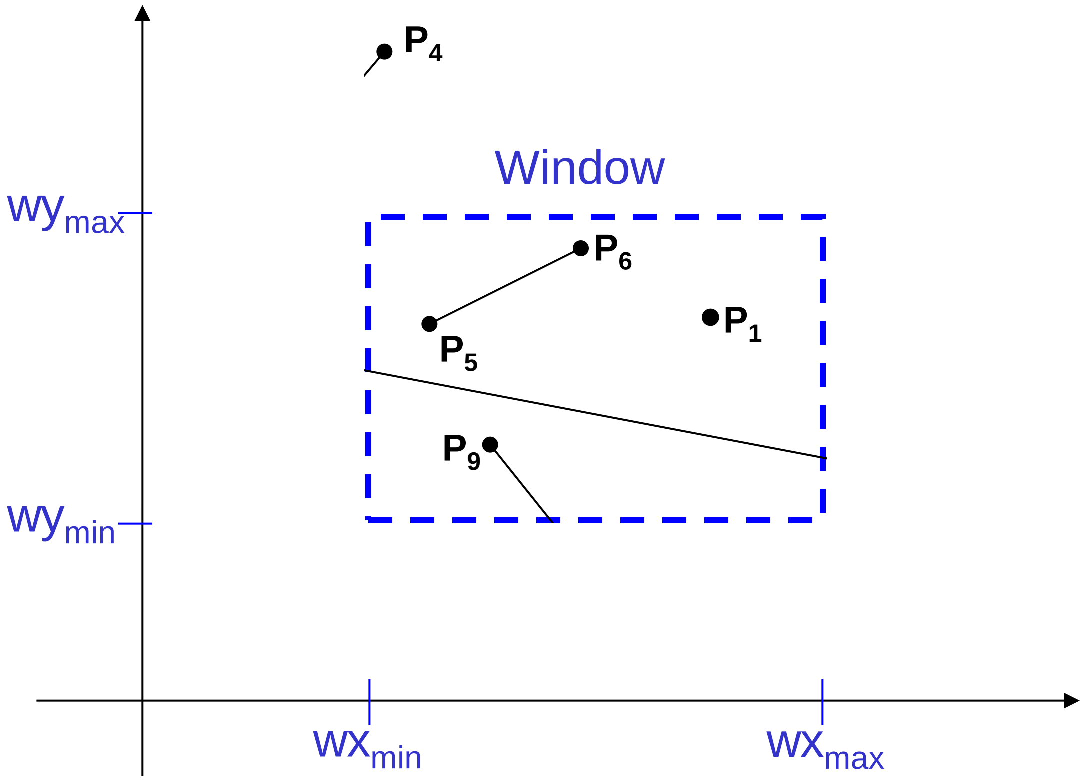


Clipping Operation

- Remove objects, lines, or line segments that are outside the viewing pane.
- Any procedure that are either inside or outside of a specified region of space is referred to as a clipping algorithm.
- The region against which an object is to clipped is called a clip window.

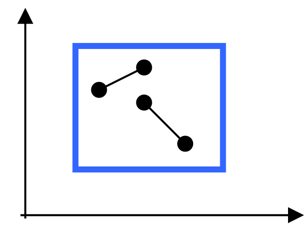
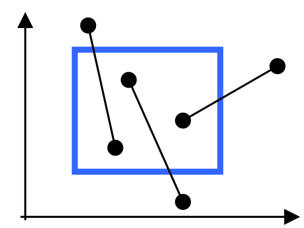
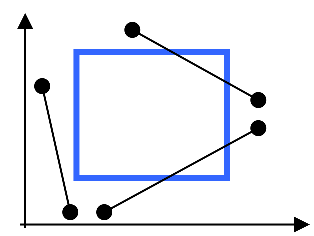
Clipping

- For the image below consider which lines and points should be kept and which ones should be clipped



Line Clipping

- Harder - examine the end-points of each line to see if they are in the window or not

Situation	**Solution**	**Example**
Both end-points inside the window	Don’t clip	
One end-point inside the window, one outside	Must clip	
Both end-points outside the window	Don’t know!	

Cohen-Sutherland Algorithm

- This is one of the oldest and most popular line clipping procedure.
- Reduces the processing by performing initial tests that reduce the number of intersections that must be calculated.
- Every line endpoint is assigned with four digital binary code, called a region code.

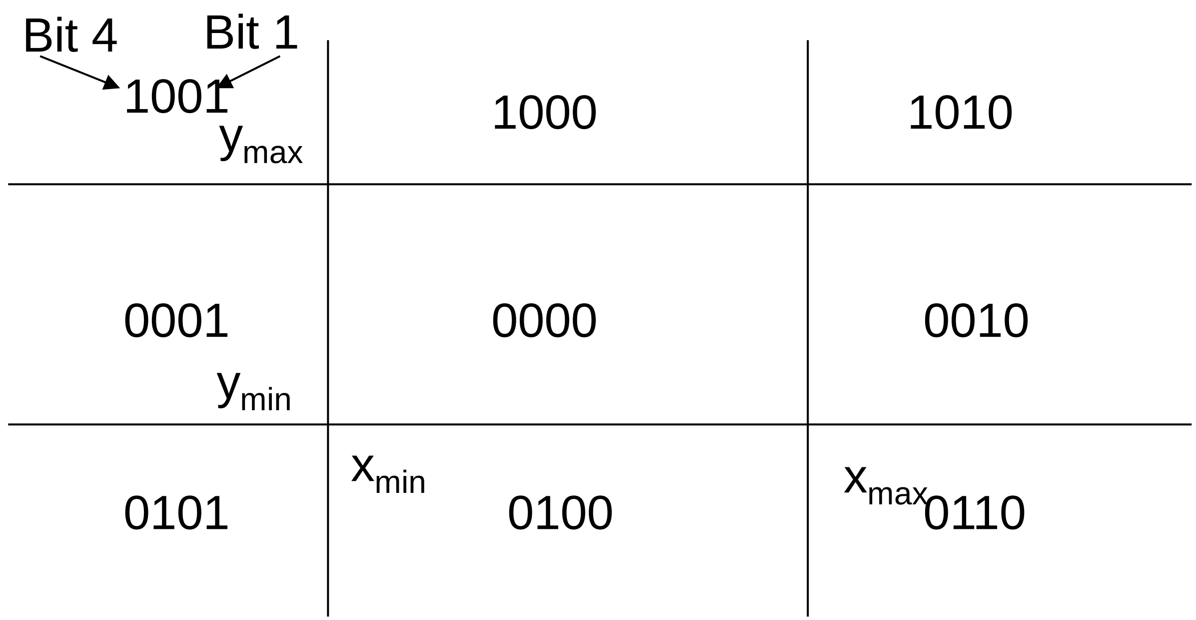
Defining Outcodes

- For each endpoint, define an outcode

$b_1b_2b_3b_4$

b_1 :left
b_2:right
b_3:below
B_4:above

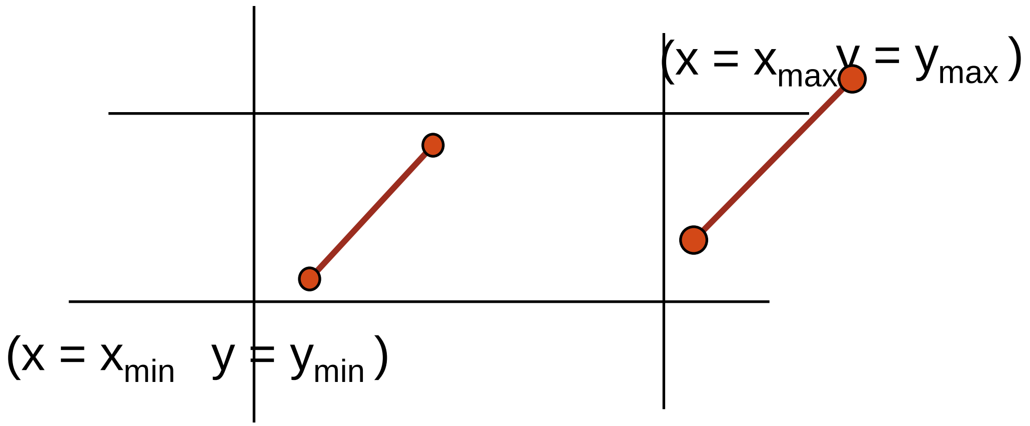
Above 1 if $y>y_{max}$
Below1 if $y<y_{min}$
Right 1 if $x>x_{max}$
Left1 if $x<x_{min}$



Outcodes divide space into 9 regions

The Cases

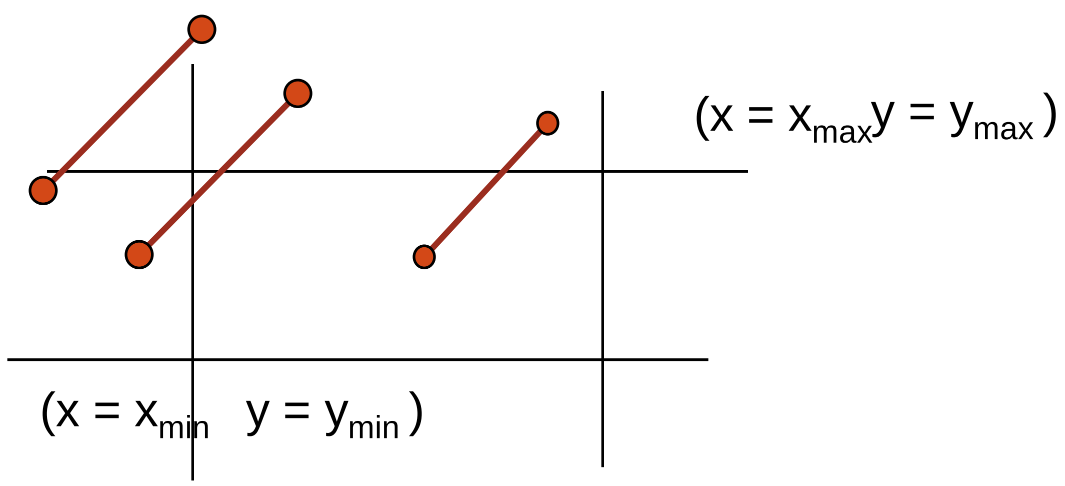
- Case 1: both endpoints of line segment inside window boundary have a region code 0000 for both endpoints.
  - Draw (trivially accept) line segment as is



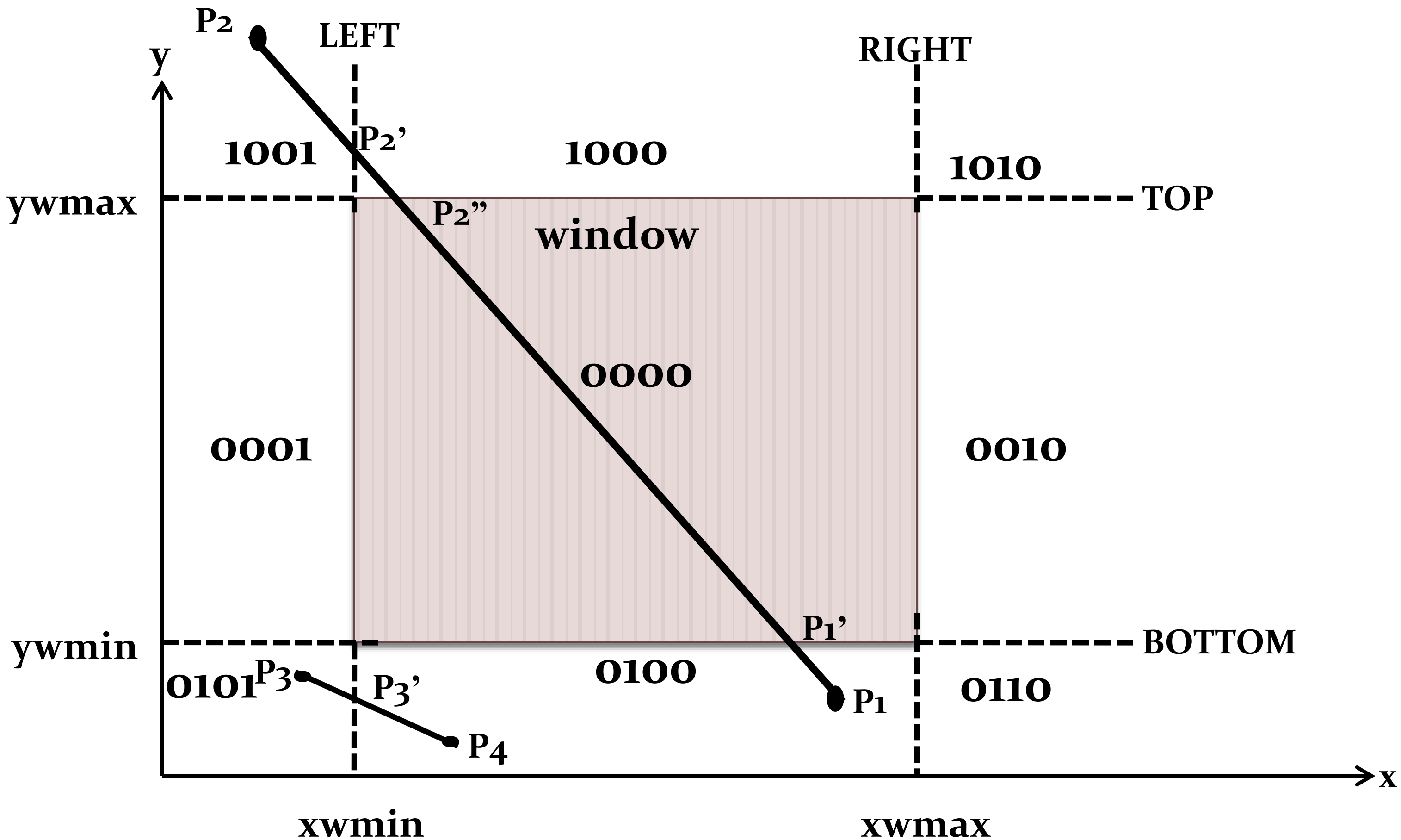
- Case 2: both endpoints outside all lines and on same side of a line
  - Discard (trivially reject) the line segment

The Cases

- Case 3: One endpoint inside, one outside
  - Must do at least one intersection
- Case 4: Both outside
  - May have part inside
  - Must do at least one intersection



Intersection Calculation and Clipping



Intersection Point

- Intersection point with a clipping boundary can be calculated using the slope-intercept from of the line equation.
- The y coordinates of the intersection point with a vertical boundary can be obtained with the calculation

$$y=y_1+m(x-x_1)$$

$x=xw_{min}$ or xw_{max}

$M=(y_2-y_1)/(x_2-x_1)$.

- The x coordinates of the intersection point with a horizontal boundary can be obtained with the

$$x=x_1+(y-y_1)/m$$

$Y=yw_{min}$ or yw_{max}

Do check with the condition: xwmin <=x<= xwmax &&yw_{min} <=y<= yw_{max}

Problem

- Let ABCD be the rectangular window with A(4,4), B(10,4), C (10,8,) and D(4,8). Find the region codes for endpoint and use Cohen Sutherland algorithm to clip the line P_1P_2 with P_1 (7,9) P_2(11,4).

x_{min}, $y=y_1+m(x-x_1)$ (Left)

x_{max}, $y=y_1+m(x-x_1)$ (Right)

y_{min}, $x=x_1+ (y-y_1)/m$ (Below)

y_{max}, $x=x_1+(y-y_1)/m$ (Top)

Efficiency

- In many applications, the clipping window is small relative to the size of the entire data base
  - Most line segments are outside one or more side of the window and can be eliminated based on their outcodes
- Inefficiency when code has to be reexecuted for line segments that must be shortened in more than one step

Algorithm

Step 1: Assign a region code for each endpoints.

Step 2: If both endpoints have a region code **0000** then accept this line.

Step 3: Else, perform the logical **AND** operation for both region codes.

Step 3.1: If the result is not **0000**, then reject the line.

Step 3.2: Else you need clipping.

Step 3.2.1: Choose an endpoint of the line that is outside the window.

Step 3.2.2: Find the intersection point at the window boundary (base on region code).

Step 3.2.3: Replace endpoint with the intersection point and update the region code.

Step 3.2.4: Repeat step 2 until we find a clipped line either trivially accepted or trivially rejected.

Step-4: Repeat step 1 for other lines.

Liang – Barsky clipping

- In computer graphics, the **Liang–Barsky algorithm** (named after You-Dong Liang and Brian A. Barsky) is a line clipping algorithm.
- This algorithm is significantly more efficient than Cohen–Sutherland.
- It can be extended to 3 Dimensional clipping.
- The algorithm is considered to be the faster parametric line clipping algorithm with the following concepts.
  - 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 idea of the Liang-Barsky clipping algorithm is to do as much testing as possible before computing line intersections. The parametric form of a straight line:

$$x = x_0 + u(x_1 - x_0) = x_0 + u\Delta x$$
$$y = y_0 + u(y_1 - y_0) = y_0 + u\Delta y$$

- A point is in the clip window, if

$$x_{\min} \le x_0 + u\Delta x \le x_{\max}$$

$$y_{\min} \le y_0 + u\Delta y \le y_{\max},$$

which can be expressed as the 4 inequalities

$$up_k \le q_k, \quad k = 1, 2, 3, 4.$$

where

$$p_1 = -\Delta x,\ q_1 = x_0 - x_{\min} \text{ (left)}$$
$$p_2 = \Delta x,\ q_2 = x_{\max} - x_0 \text{ (right)}$$
$$p_3 = -\Delta y,\ q_3 = y_0 - y_{\min} \text{ (bottom)}$$
$$p_4 = \Delta y,\ q_4 = y_{\max} - y_0 \text{ (top)}$$

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

- Here

For each line, calculate u_1 and u_2. For u_1, look at boundaries for which $p_k < 0$ (outside -> in). Take u_1 to be the largest among $\left(0, \frac{q_k}{p_k}\right)$

. For u_2, look at boundaries for which $p_k > 0$ (inside -> out). Take u_2 to be the minimum of $\left(1, \frac{q_k}{p_k}\right)$.

If $u_1 > u_2$, the line is outside and therefore rejected.

Problem

- Let ABCD be the Rectangular window with A(5,5), B(9,5), C(9,9) and D(5,9). Use Liang Barsky algorithm to clip the line P_0P_1 with $P_0(4,12)$ $P_1(8,8)$