Chapter 7

2D Clipping

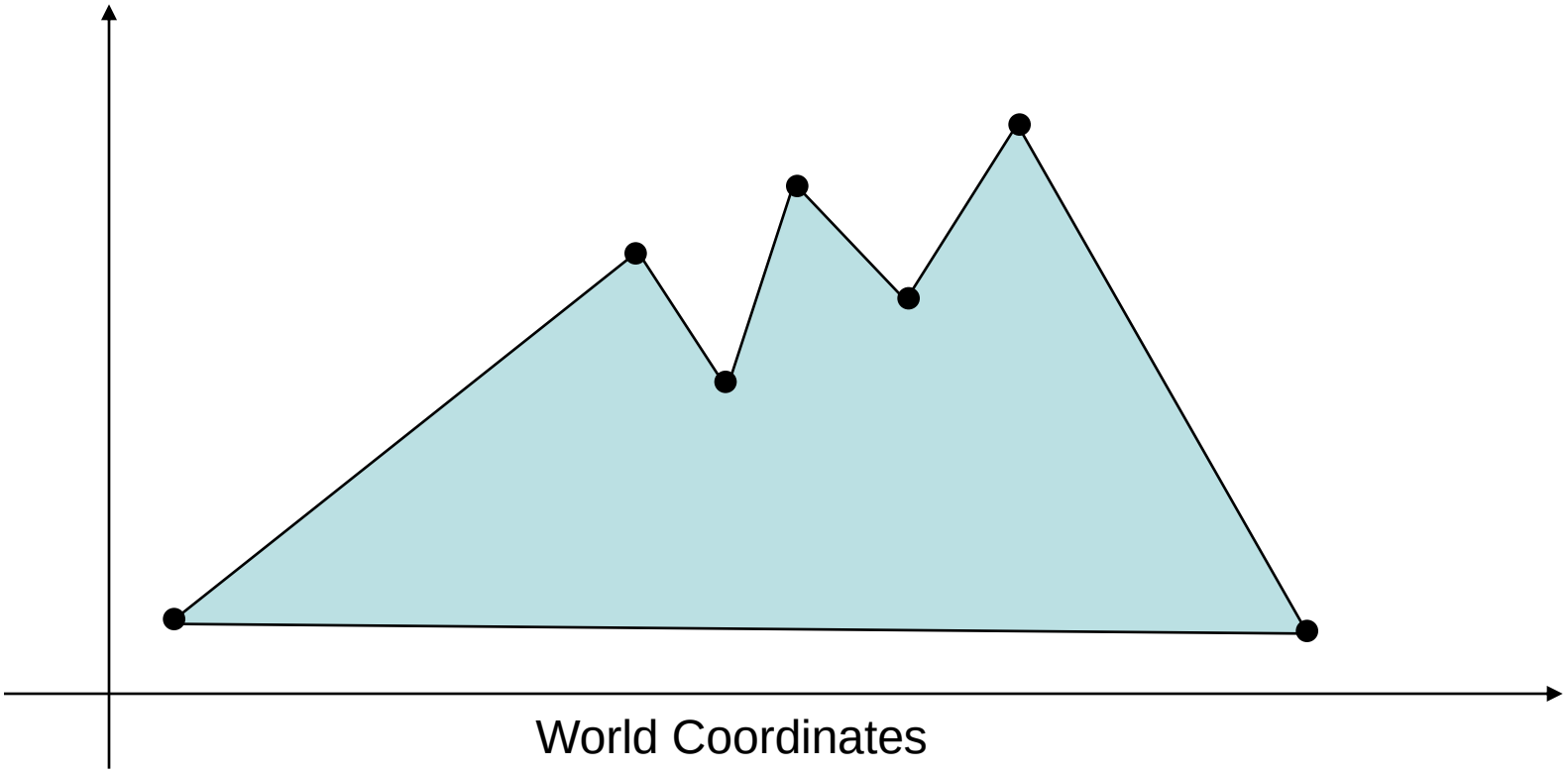
2D Clipping

1. **Introduction**
2. Point Clipping
3. Line Clipping
4. Polygon/Area Clipping
5. Text Clipping
6. Curve Clipping

2D Clipping

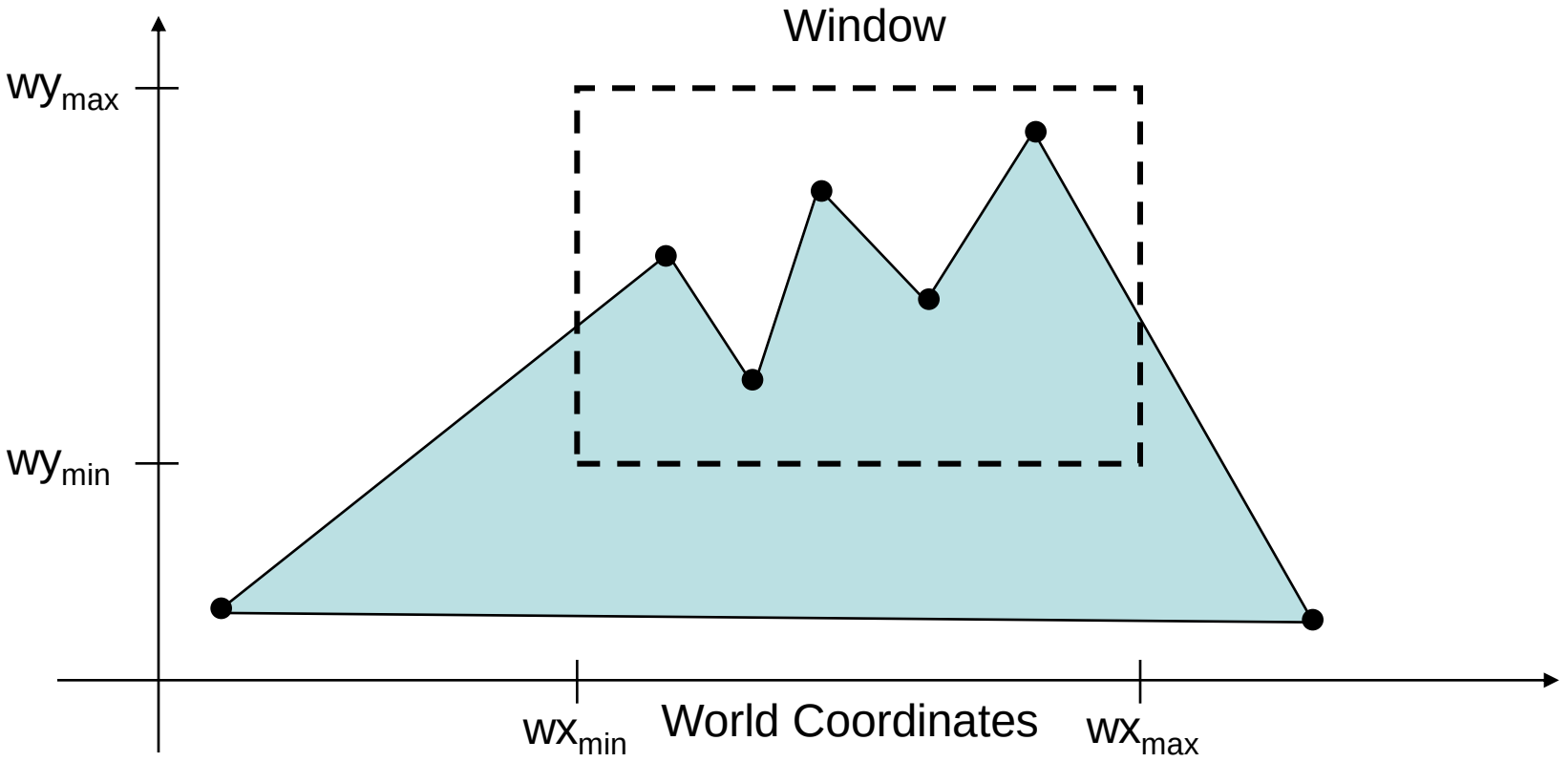
1. Introduction:

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



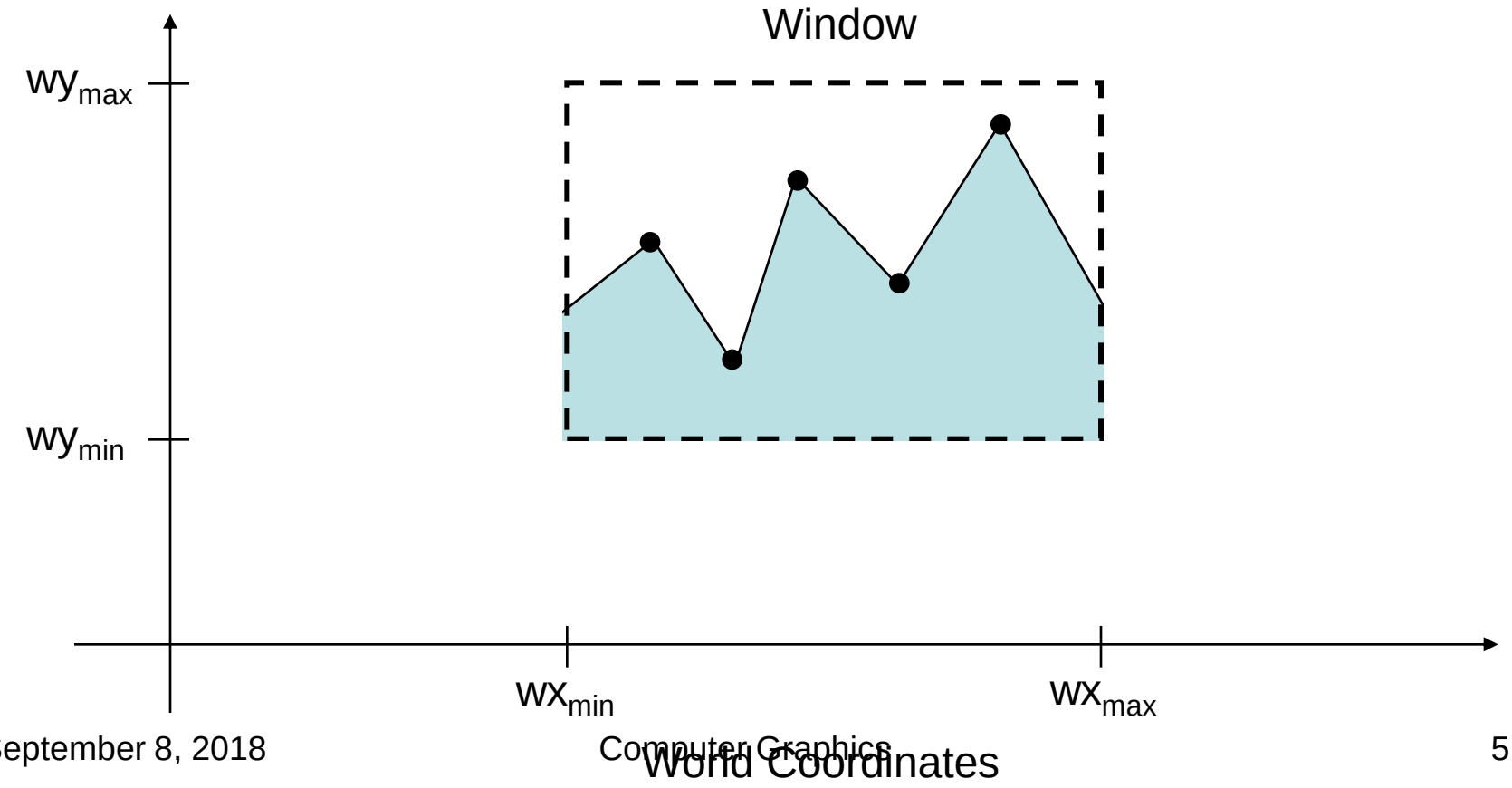
2D Clipping

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



2D Clipping

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



2D Clipping

1.1 Definition:

- ***Clipping*** is the process of determining which elements of the picture lie inside the window and are visible.

- By default, the "***clip window***" is the entire canvas
  - not necessary to draw outside the canvas
  - for some devices, it is damaging (plotters)
  - \
- Sometimes it is convenient to restrict the "clip window" to a smaller portion of the canvas
  - partial canvas redraw for menus, dialog boxes, other obscuration

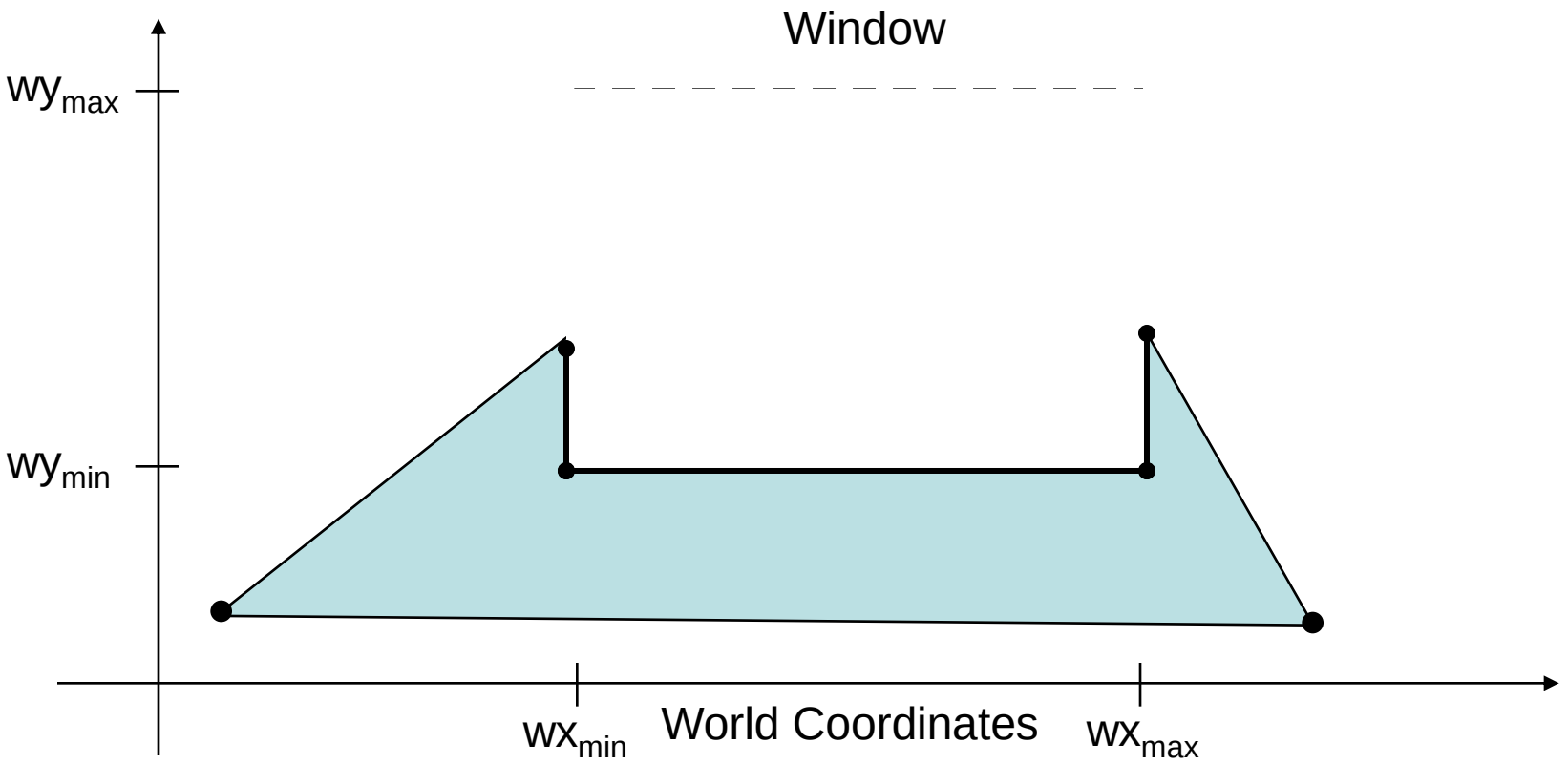
2D Clipping

1.2 Shielding:

- ***Shielding or exterior clipping*** is the reverse operation of clipping where window act as the block used to abstract the view.

- Examples
  - A multi view window system
  - The design of page layouts in advertising or publishing applications or for adding labels or design patterns to picture.
  - Combining graphs, maps o schematics

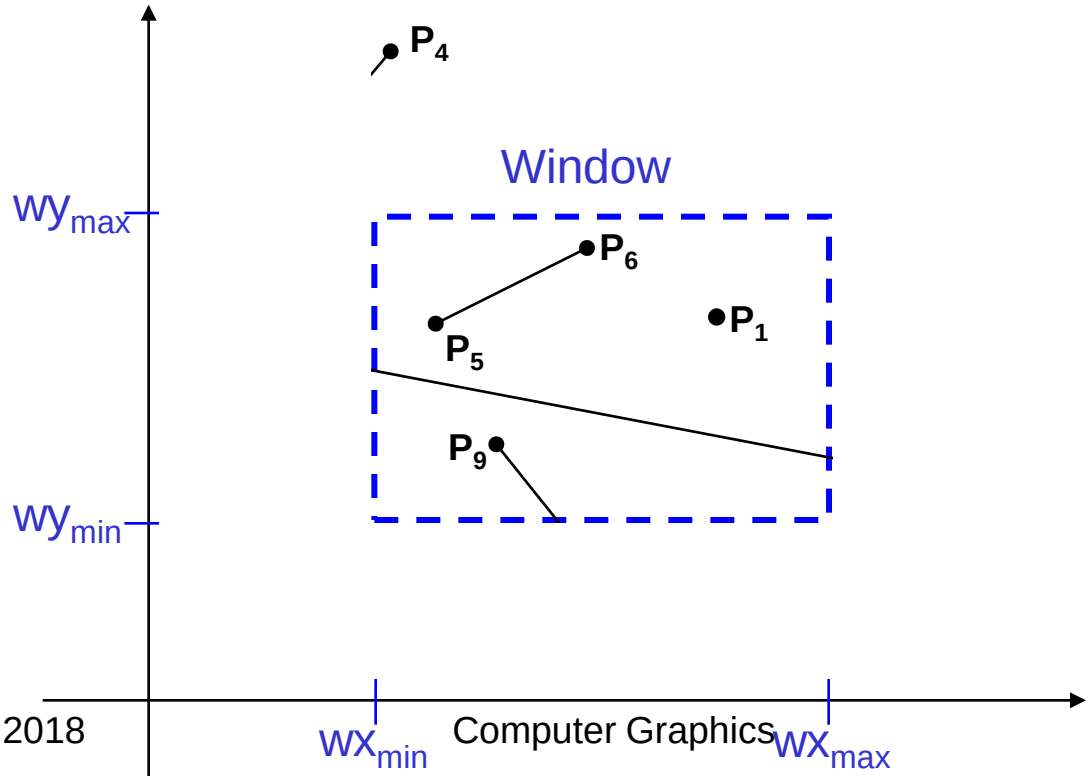
2D Clipping



2D Clipping

1.3 Example:

For the image below consider which lines and points should be kept and which ones should be clipped against the clipping window



2D Clipping

1.4 Applications:

- Extract part of a defined scene for viewing.
- Drawing operations such as erase, copy, move etc.
- Displaying multi view windows.
- Creating objects using solid modeling techniques.
- Anti-aliasing line segments or object boundaries.
- Identify visible surfaces in 3D views.

2D Clipping

1.6 Levels of clipping:

- Point Clipping
- Line Clipping
- Polygon Clipping
- Area Clipping
- Text Clipping
- Curve Clipping

2D Clipping

1. Introduction
2. **Point Clipping**
3. Line Clipping
4. Polygon/Area Clipping
5. Text Clipping
6. Curve Clipping

Point Clipping

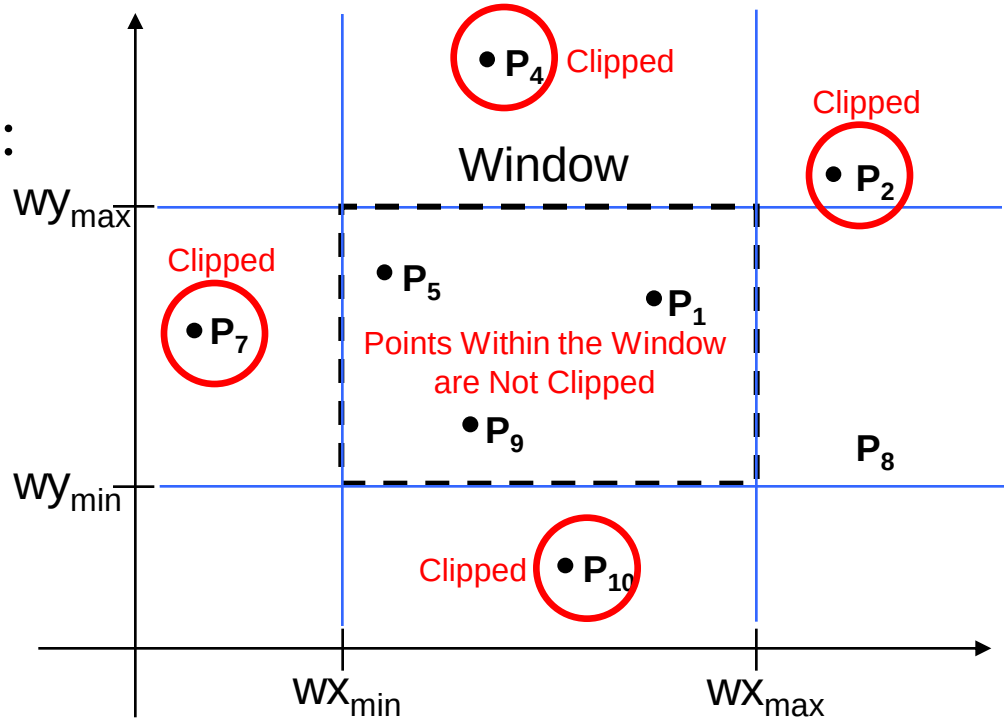
- Simple and Easy
- a point (x,y) is not clipped if:

$$wx_{min} \le x \le wx_{max}$$

&

$$wy_{min} \le y \le wy_{max}$$

- otherwise it is clipped

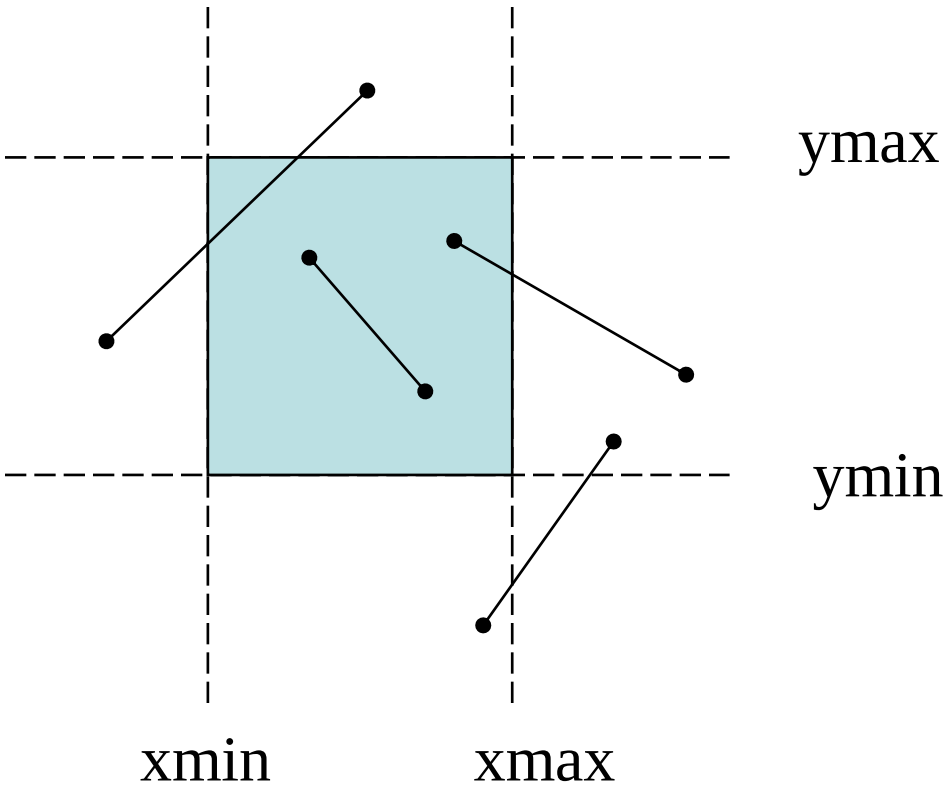


2D Clipping

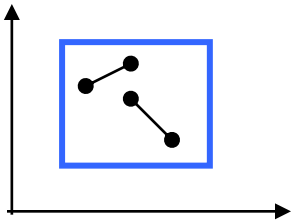
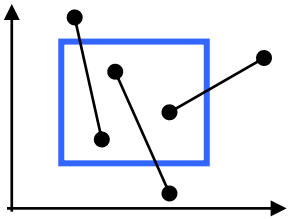
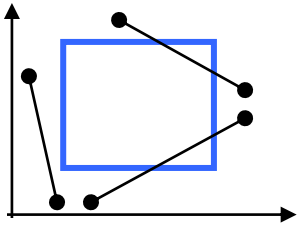
1. Introduction
2. Point Clipping
3. **Line Clipping**
4. Polygon/Area Clipping
5. Text Clipping
6. Curve Clipping

Line Clipping

- It is Harder than point clipping
- We first examine the end-points of each line to see if they are in the window or not
  - Both endpoints inside, line trivially accepted
  - One in and one out, line is partially inside
  - Both outside, might be partially inside
  - What about trivial cases?



Line Clipping

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!	

2D Line Clipping Algorithms

1. Analytical Line Clipping
2. **Cohen Sutherland Line Clipping**
3. Liang Barsky Line Clipping

Cohen-Sutherland Line Clipping

- An efficient line clipping algorithm
- The key advantage of the algorithm is that it vastly reduces the number of line intersections that must be calculated.

Cohen is something of a mystery – can anybody find out who he was?

Dr. Ivan E. Sutherland co-developed the Cohen-Sutherland clipping algorithm. Sutherland is a graphics giant and includes amongst his achievements the invention of the head mounted display.

Cohen-Sutherland Line Clipping

- Two phases Algorithm

Phase I: Identification Phase

All line segments fall into one of the following categories

1. Visible: Both endpoints lies inside
2. Invisible: Line completely lies outside
3. Clipping Candidate: A line neither in category 1 or 2

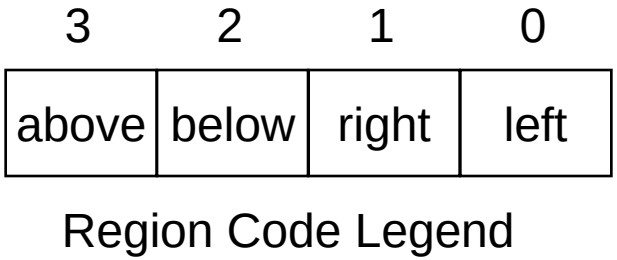
Phase II: Perform Clipping

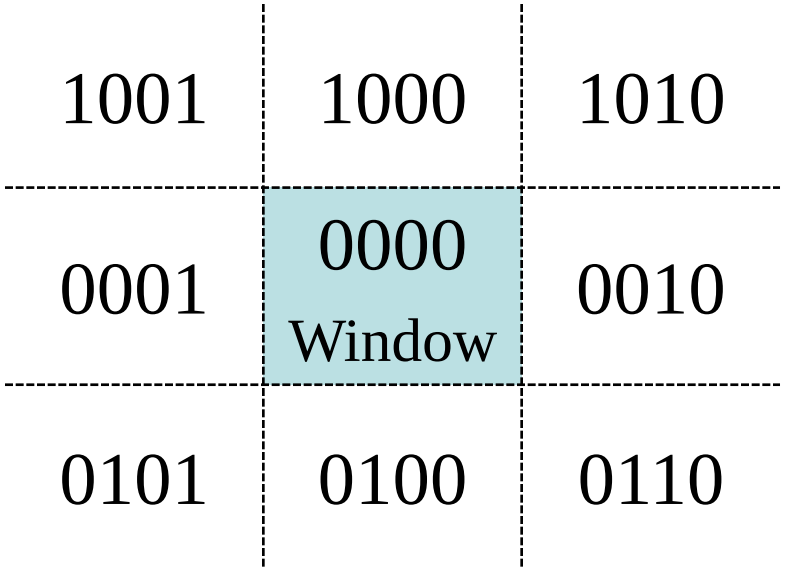
Compute intersection for all lines that are candidate for clipping.

Cohen-Sutherland Line Clipping

Phase I: Identification Phase: World space is divided into regions based on the window boundaries

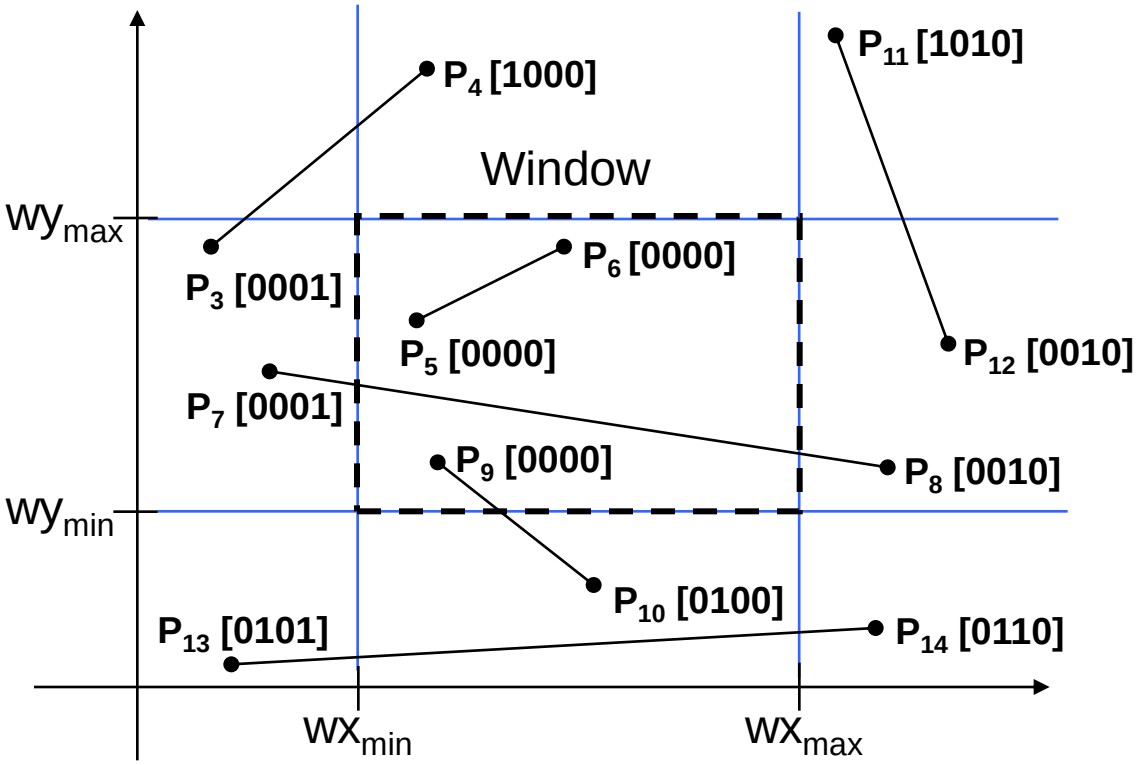
- Each region has a unique four bit region code
- Region codes indicate the position of the regions with respect to the window





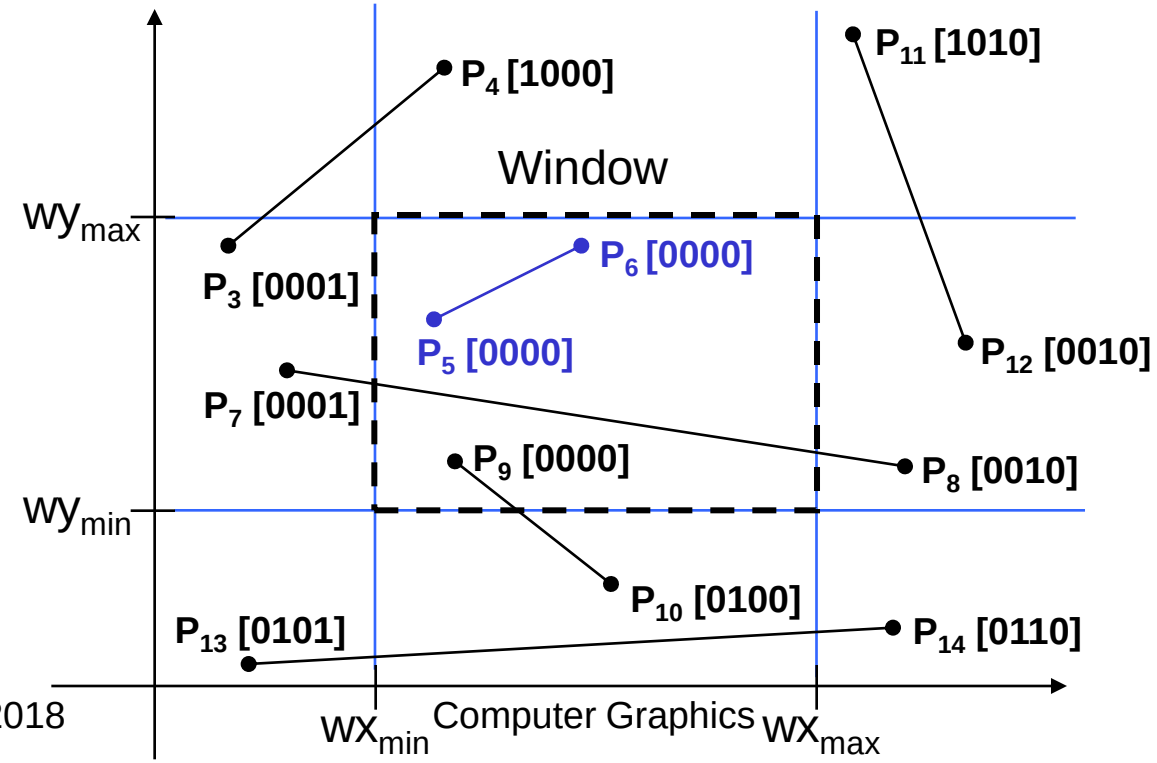
Cohen-Sutherland Line Clipping

Every end-point is labelled with the appropriate region code



Cohen-Sutherland Line Clipping

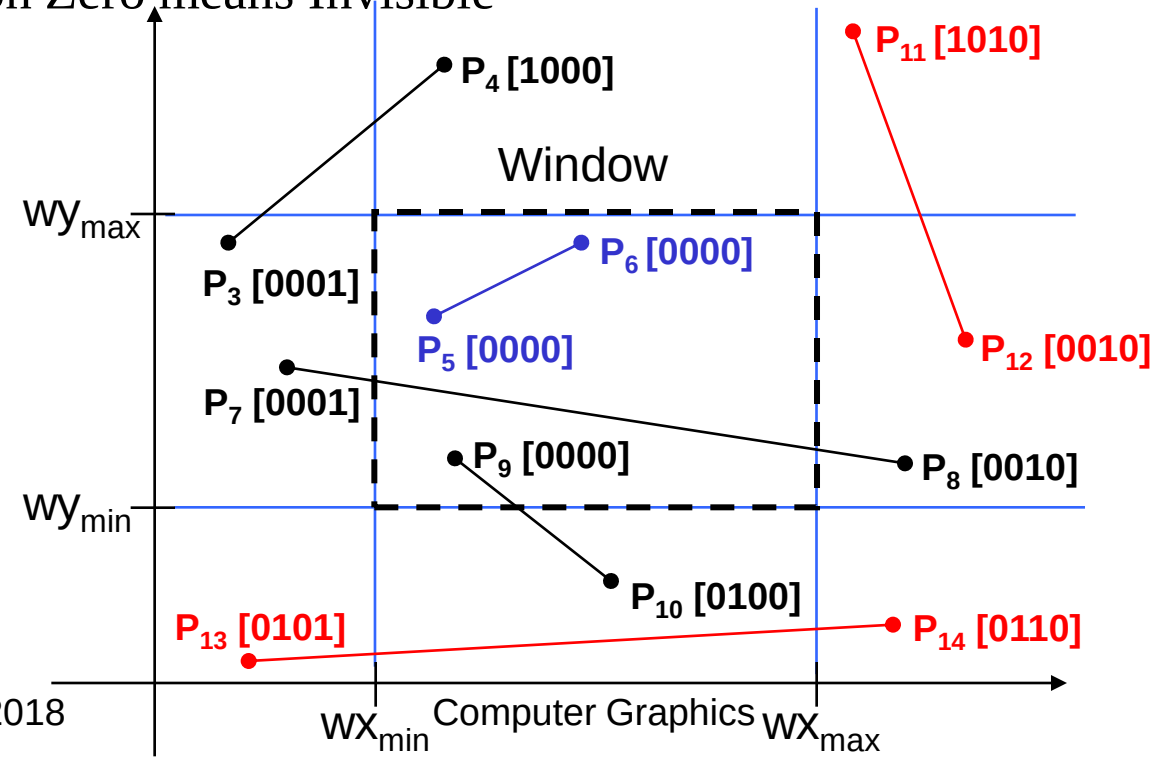
Visible Lines: Lines completely contained within the window boundaries have region code [0000] for both end-points so are not clipped



Cohen-Sutherland Line Clipping

Invisible Lines: Any line with a common set bit in the region codes of both end-points can be clipped completely

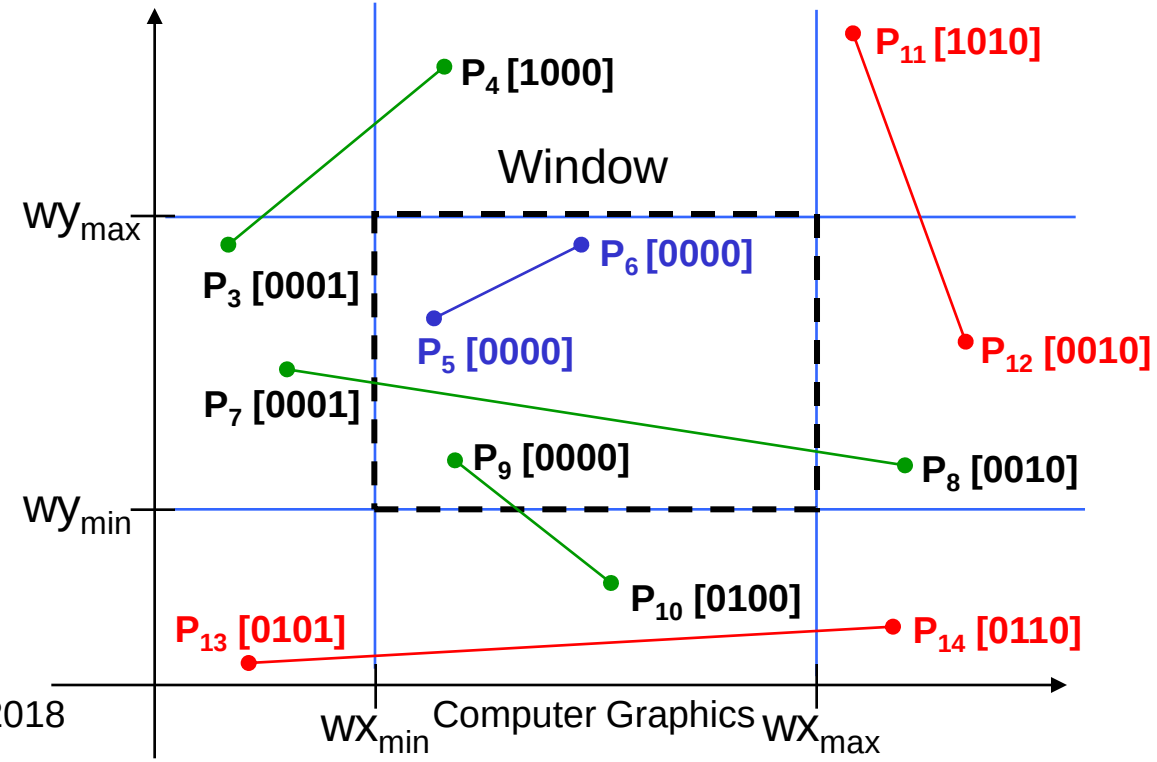
- The AND operation can efficiently check this
- Non Zero means Invisible



Cohen-Sutherland Line Clipping

Clipping Candidates: Lines that cannot be identified as completely inside or outside the window may or may not cross the window interior. These lines are processed in Phase II.

- If AND operation result in 0 the line is candidate for clipping



Cohen-Sutherland Line Clipping

Assigning Codes

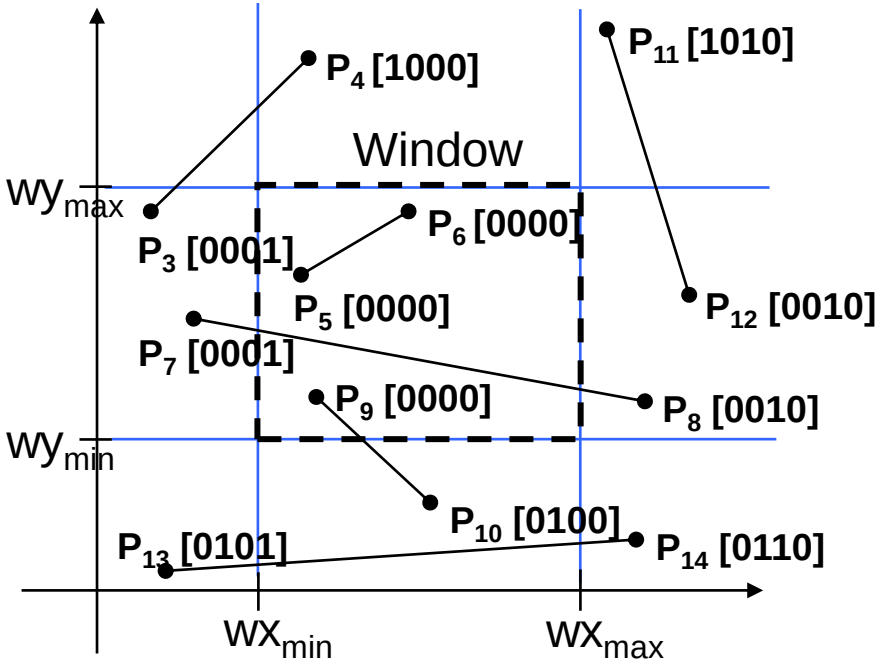
– Let point (x,y) is be given code $b_3b_2b_1b_0$:

bit 3 = 1 if $wy_{max} - y < 0$

bit 2 = 1 if $y - wy_{min} < 0$

bit 1 = 1 if $wx_{max} - x < 0$

bit 0 = 1 if $x - wx_{min} < 0$



Cohen-Sutherland Clipping Algorithm

Phase II: Clipping Phase: Lines that are in category 3 are now processed as follows:

- Compare an end-point outside the window to a boundary (choose any order in which to consider boundaries e.g. left, right, bottom, top) and determine how much can be discarded
- If the remainder of the line is entirely inside or outside the window, retain it or clip it respectively
- Otherwise, compare the remainder of the line against the other window boundaries
- Continue until the line is either discarded or a segment inside the window is found

Cohen-Sutherland Line Clipping

- Intersection points with the window boundaries are calculated using the line-equation parameters
  - Consider a line with the end-points (x_1, y_1) and (x_2, y_2)
  - The y-coordinate of an intersection with a vertical window boundary can be calculated using:

    $$y = y_1 + m\,(x_{boundary} - x_1)$$

    where $x_{boundary}$ can be set to either wx_{min} or wx_{max}
  - The x-coordinate of an intersection with a horizontal window boundary can be calculated using:

    $$x = x_1 + (y_{\text{boundary}} - y_1) / m$$

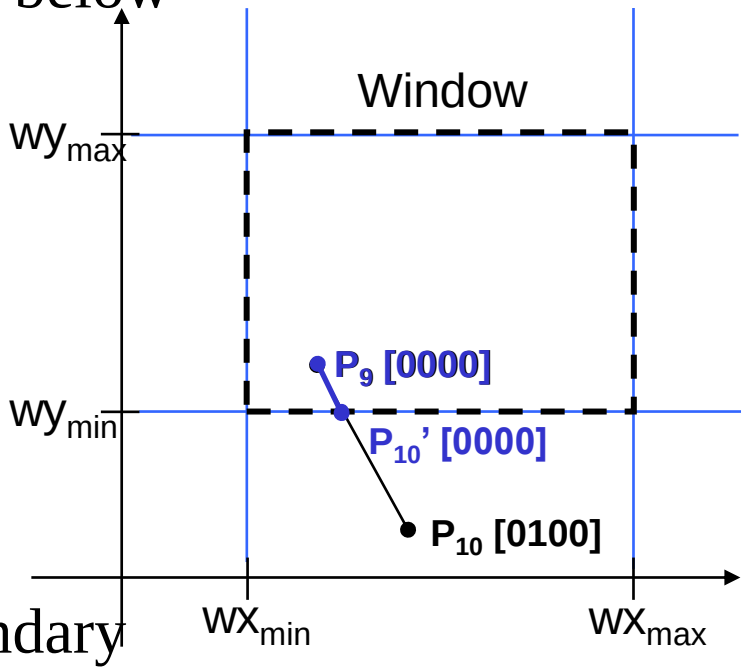
    where $y_{boundary}$ can be set to either wy_{min} or wy_{max}

Cohen-Sutherland Line Clipping

- We can use the region codes to determine which window boundaries should be considered for intersection
  - To check if a line crosses a particular boundary we compare the appropriate bits in the region codes of its end-points
  - If one of these is a 1 and the other is a 0 then the line crosses the boundary.

Cohen-Sutherland Line Clipping

Example1: Consider the line P_9 to P_{10} below

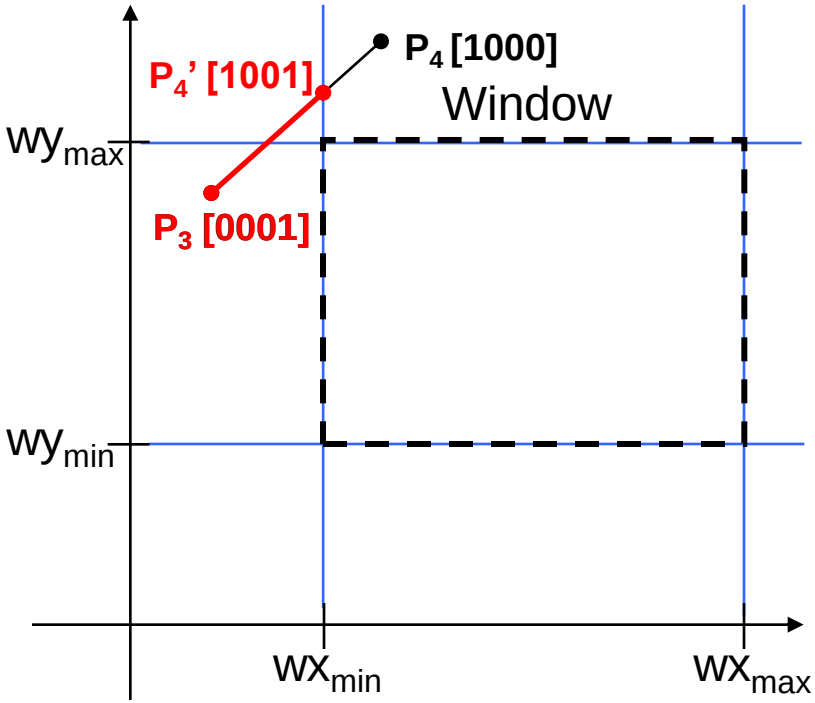


- Start at P_{10}
- From the region codes of the two end-points we know the line doesn't cross the left or right boundary
- Calculate the intersection of the line with the bottom boundary to generate point P_{10}'
- The line P_9 to P_{10}' is completely inside the window so is retained

Cohen-Sutherland Line Clipping

Example 2: Consider the line P_3 to P_4 below

- Start at P_4
- From the region codes of the two end-points we know the line crosses the left boundary so calculate the intersection point to generate P_4'

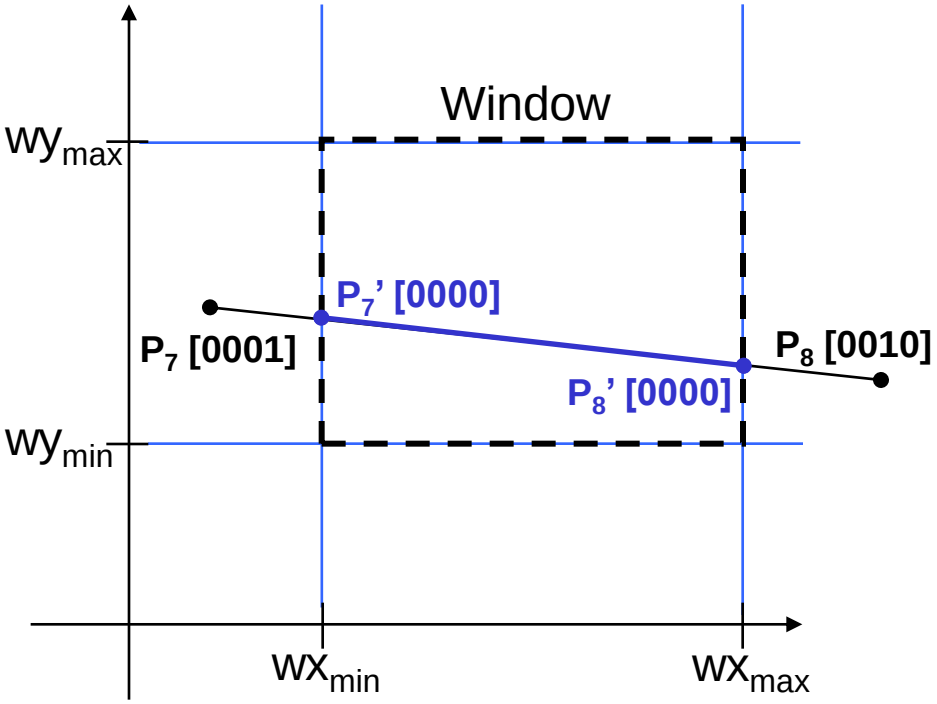


- The line P_3 to P_4' is completely outside the window so is clipped

Cohen-Sutherland Line Clipping

Example 3: Consider the line P_7 to P_8 below

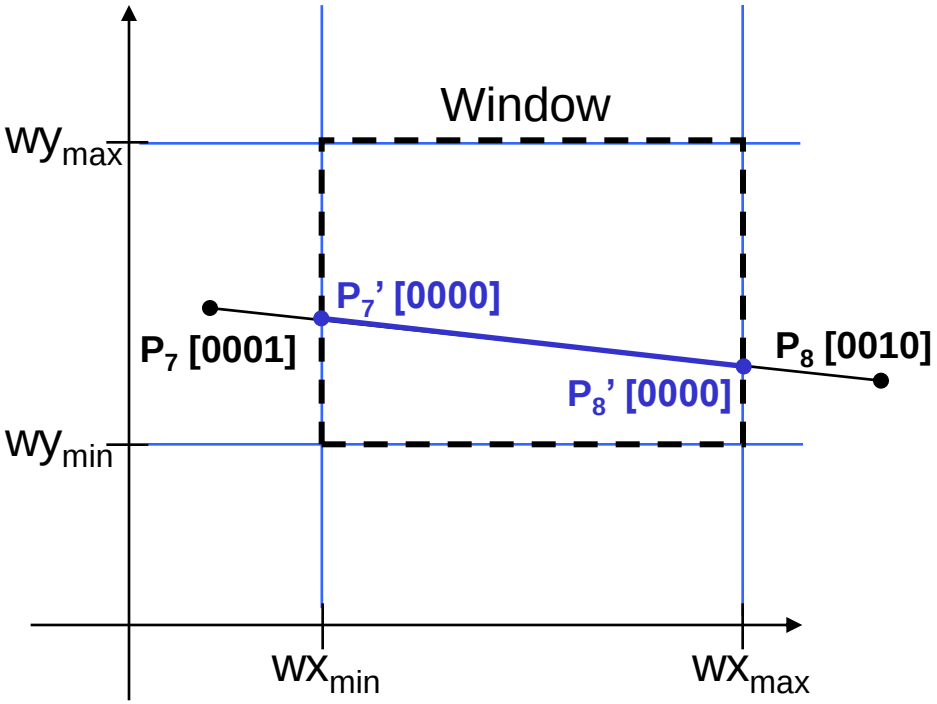
- Start at P_7
- From the two region codes of the two end-points we know the line crosses the left boundary so calculate the intersection point to generate P_7'



Cohen-Sutherland Line Clipping

Example 4: Consider the line P_7' to P_8

- Start at P_8
- Calculate the intersection with the right boundary to generate P_8'
- P_7' to P_8' is inside the window so is retained

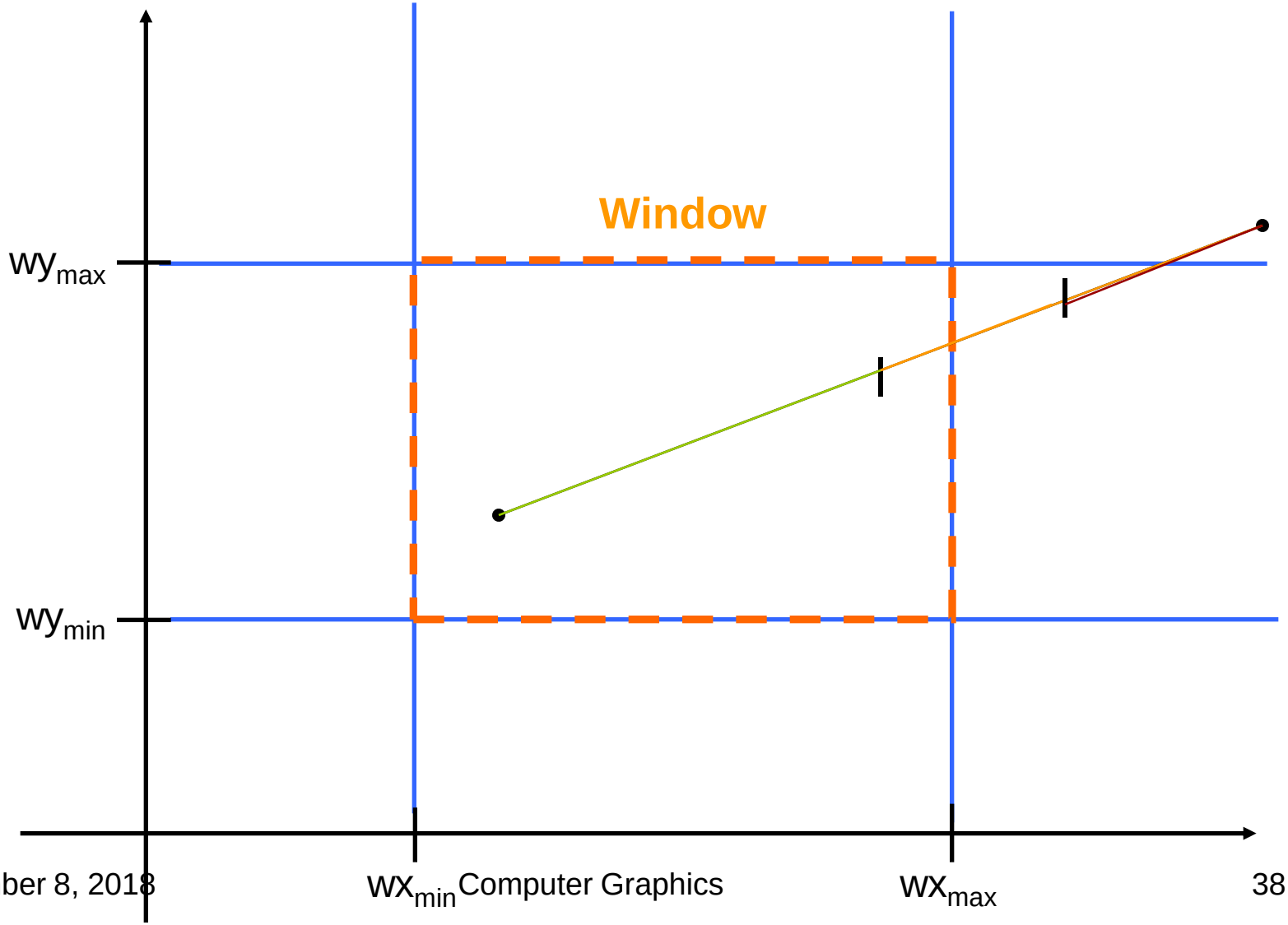


Cohen-Sutherland Line Clipping

Mid-Point Subdivision Method

- Algorithm
  1. Initialise the list of lines to all lines
  2. Classify lines as in ***Phase I***
     i. Assign 4 point bit codes to both end points $a_3a_2a_1a_0$ and $b_3b_2b_1b_0$
     ii. If ($a_3a_2a_1a_0 = b_3b_2b_1b_0 = 0$)Line in category 1
     iii. If ($a_3a_2a_1a_0$)AND ($b_3b_2b_1b_0$) # 0) Line in category 2
     iv. If ($a_3a_2a_1a_0$)AND ($b_3b_2b_1b_0$) = 0) Line in category 3
  3. Display all lines from the list in category 1 and remove;
  4. Delete all lines from the list in category 2 as they are invisible;
  5. Divide all lines of category 3 are into two smaller segments at mid-point (x_m,y_m) where $x_m = (x_1 + x_2)/2$ *and* $y_m = (y_1 + y_2)/2$
  6. Remove the original line from list and enter its two newly created segments.
  7. Repeat step 2-5 until list is null.

Cohen-Sutherland Line Clipping



Cohen-Sutherland Line Clipping

Mid-Point Subdivision Method

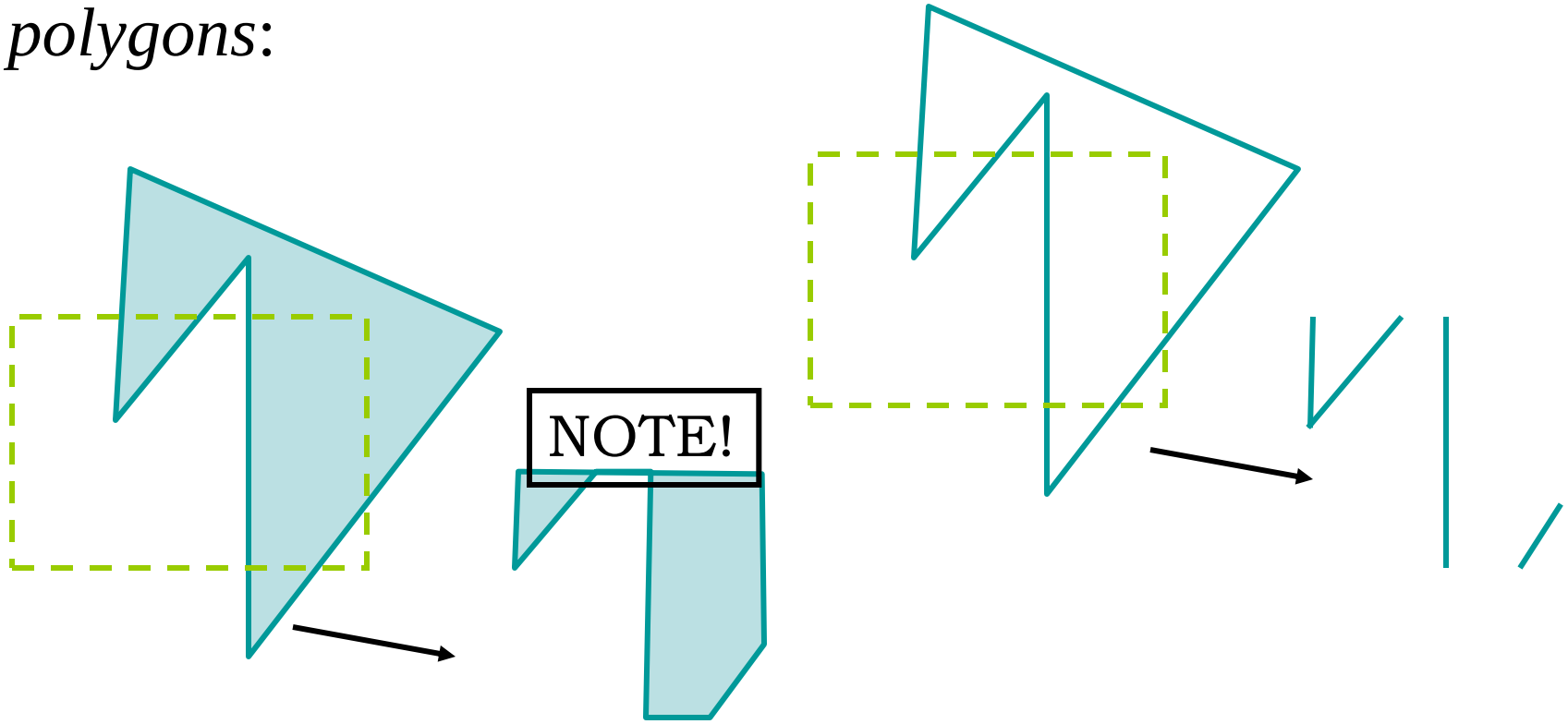
- Integer Version
- Fast as Division by 2 can be performed by simple shift right operation
- For NxN max dimension of line number of subdivisions required $\log_2 N$.
- Thus a 1024x1024 raster display require just 10 subdivisions……

2D Clipping

1. Introduction
2. Point Clipping
3. Line Clipping
4. **Polygon / Area Clipping**
5. Text Clipping
6. Curve Clipping

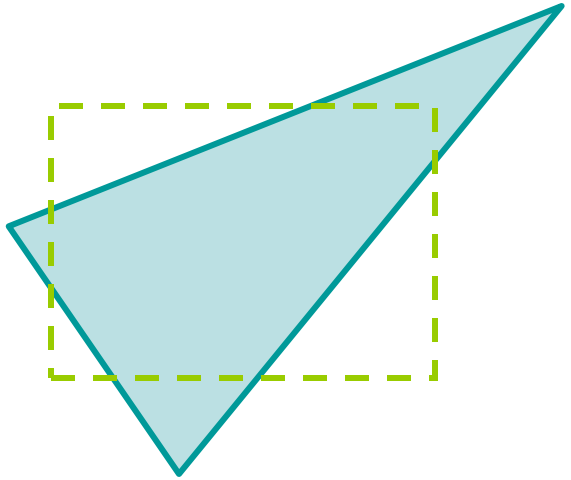
Polygon Clipping

- Note the difference between clipping *lines* and *polygons*:



Polygon Clipping

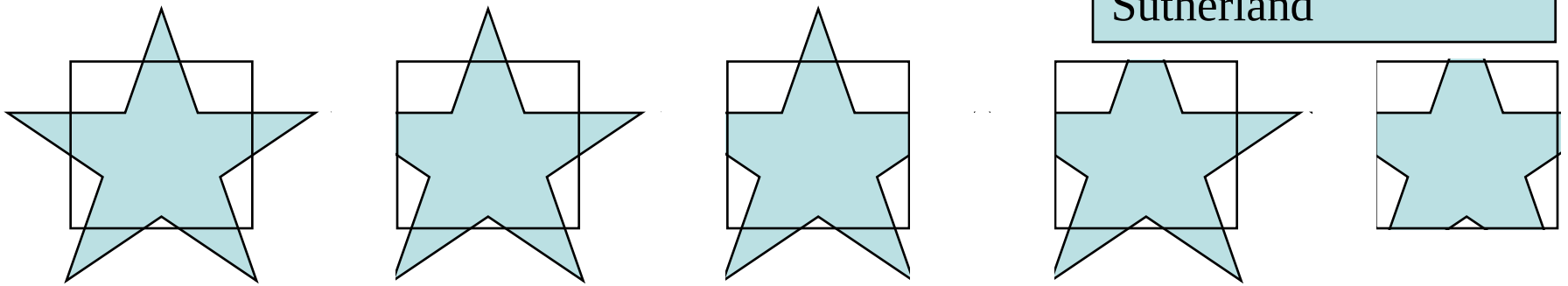
- Some difficulties:
  - Maintaining correct inside/outside
  - Variable number of vertices
  - Handle screen corners correctly



Sutherland-Hodgman Area Clipping

- A technique for clipping areas developed by Sutherland & Hodgman
- Put simply the polygon is clipped by comparing it against each boundary in turn

Sutherland turns up again. This time with Gary Hodgman with whom he worked at the first ever graphics company Evans & Sutherland



Original Area | Clip Left | Clip Right | Clip Top | Clip Bottom

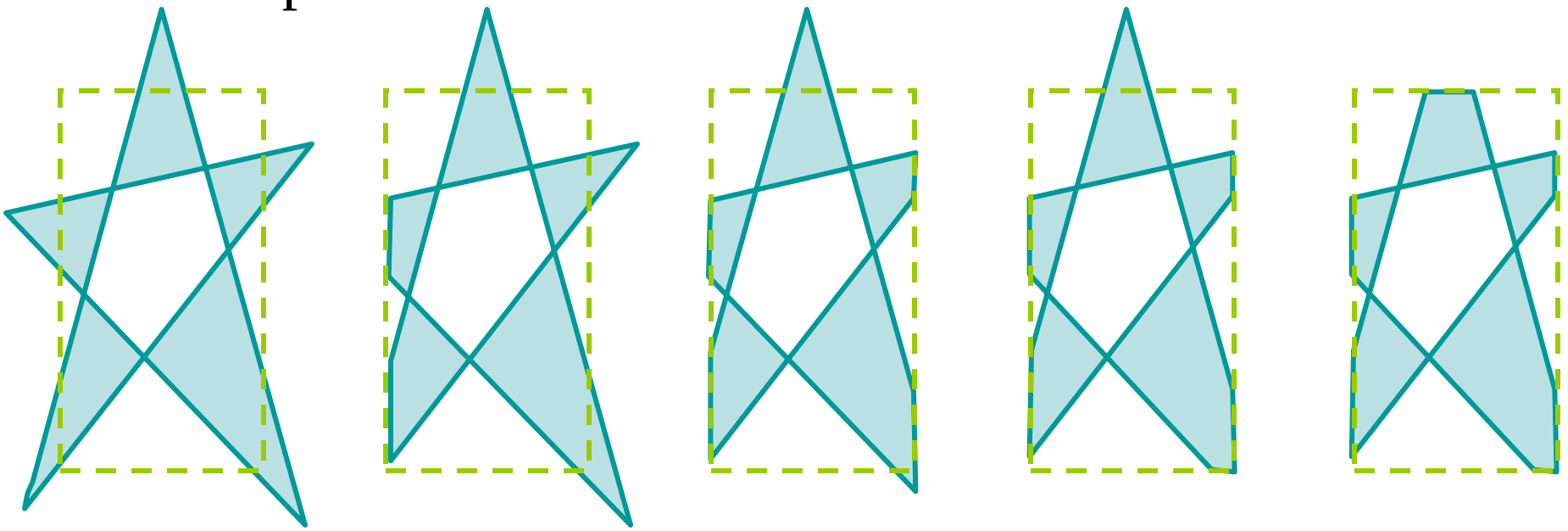
Sutherland-Hodgeman Polygon Clipping

1. Basic Concept:

- Simplify via separation
- Clip whole polygon against one edge
  - Repeat with output for other 3 edges
  - Similar for 3D
- You can create intermediate vertices that get thrown out

Sutherland-Hodgeman Polygon Clipping

- Example



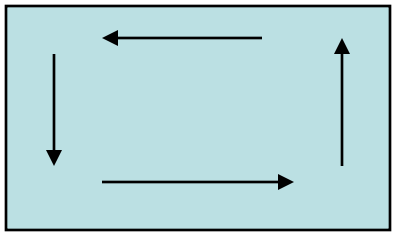
Start Left Right Bottom Top

Note that the point one of the points added when clipping on the right gets removed when we clip with bottom

Sutherland-Hodgeman Polygon Clipping

2. Algorithm:

Let (P_1, P_2,…. P_N) be the vertex list of the Polygon to be clipped and E be the edge of *+vely oriented, convex clipping window*.



We clip each edge of the polygon in turn against each window edge E, forming a new polygon whose vertices are determined as follows:

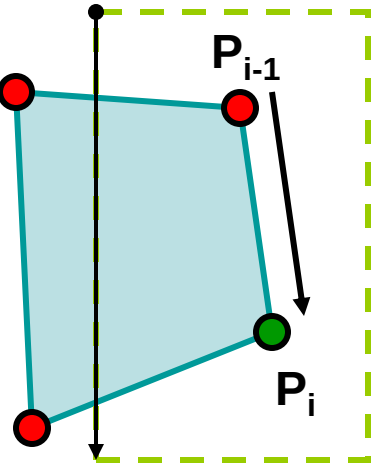
Sutherland-Hodgeman Polygon Clipping

Four cases

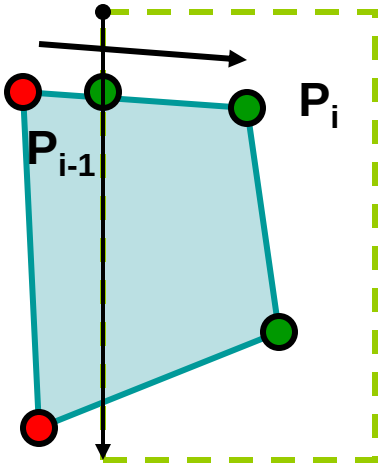
1. ***Inside:*** If both P_{i-1} and P_i are to the left of window edge vertex then P_i is placed on the output vertex list.
2. ***Entering:*** If P_{i-1} is to the right of window edge and P_i is to the left of window edge vertex then intersection (I) of P_{i-1} P_i with edge E and P_i are placed on the output vertex list.
3. ***Leaving:*** If P_{i-1} is to the left of window edge and P_i is to the right of window edge vertex then only intersection (I) of P_{i-1} P_i with edge E is placed on the output vertex list.
4. ***Outside:*** If both P_{i-1} and P_i are to the right of window edge nothing is placed on the output vertex list.

Sutherland-Hodgeman Polygon Clipping

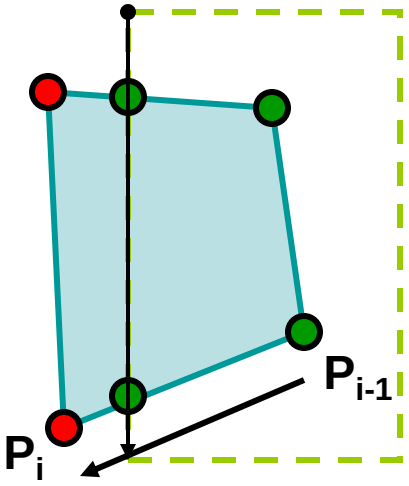
Creating New Vertex List



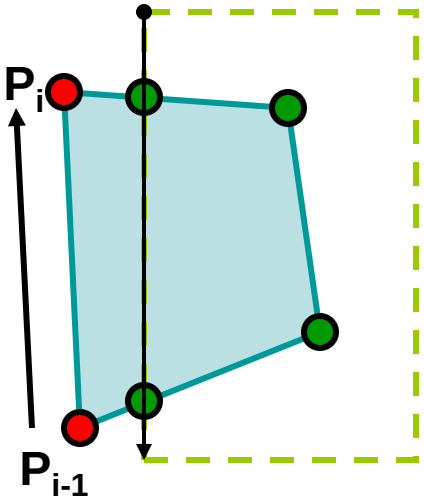
in → in
save ending vert
Inside
(1 output)



out → in
save new clip vert
and ending vert
Entering
(2 outputs)



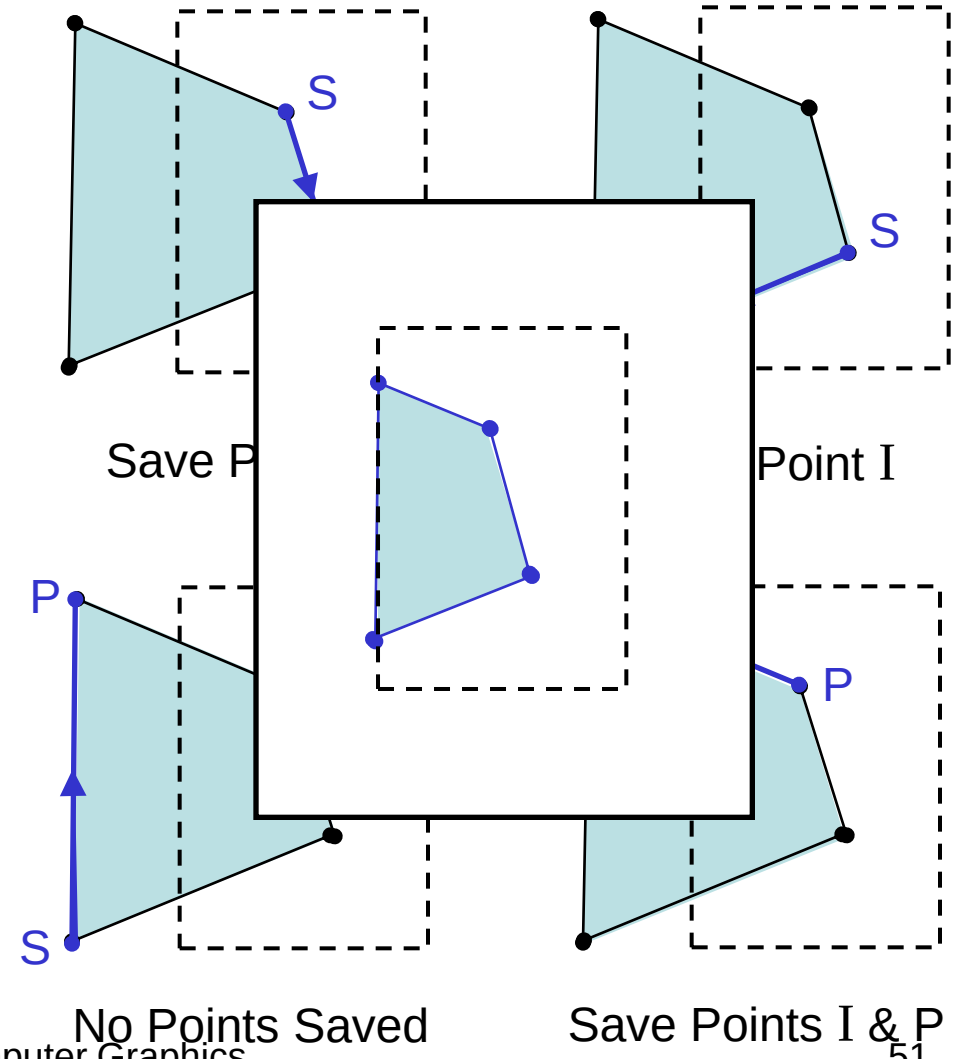
in → out
save new clip vert
Leaving
(1 output)



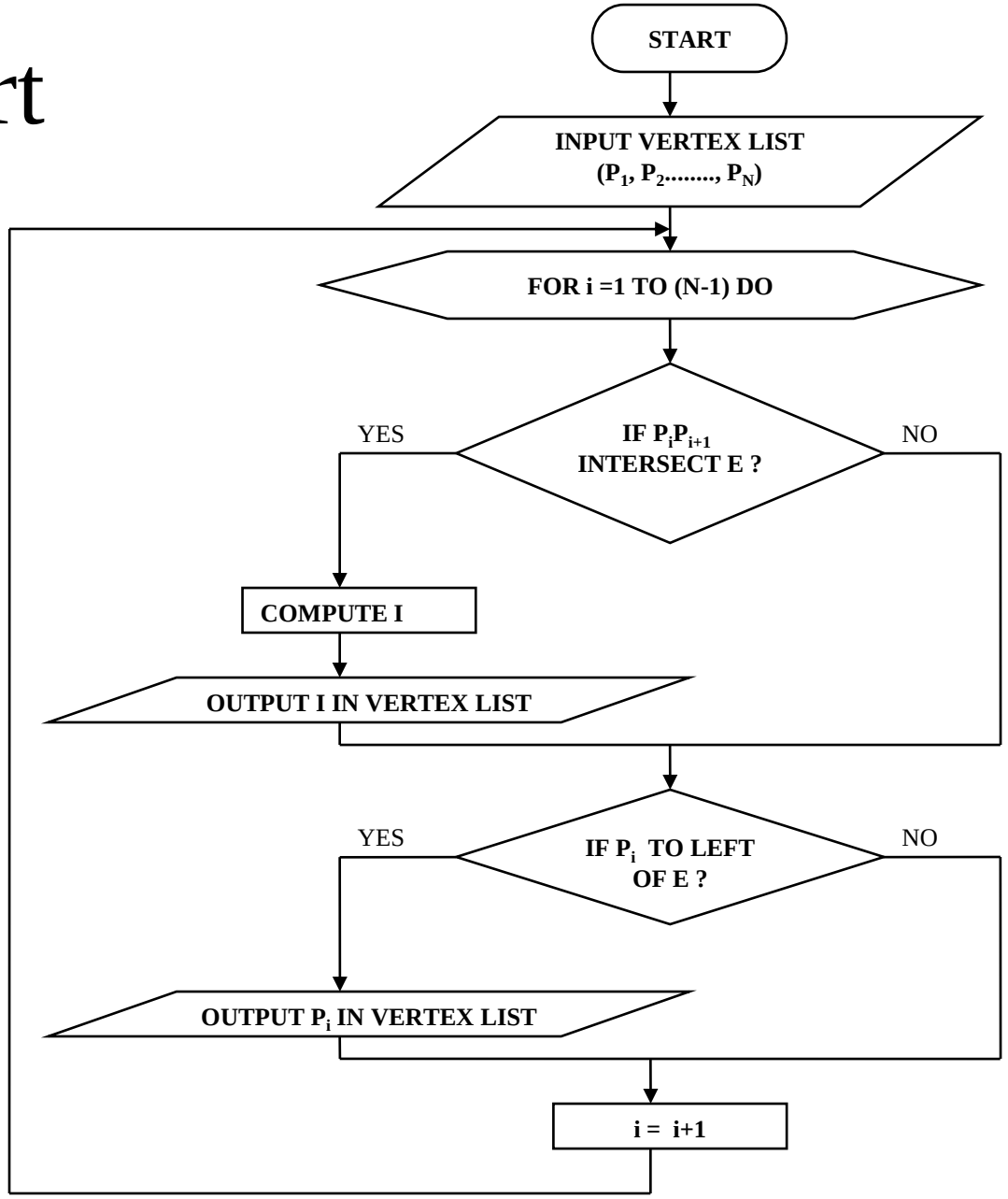
out → out
save nothing
Outside
(0 output)

Sutherland-Hodgman Polygon Clipping

- Each example shows the point being processed (P) and the previous point (S)
- Saved points define area clipped to the boundary in question



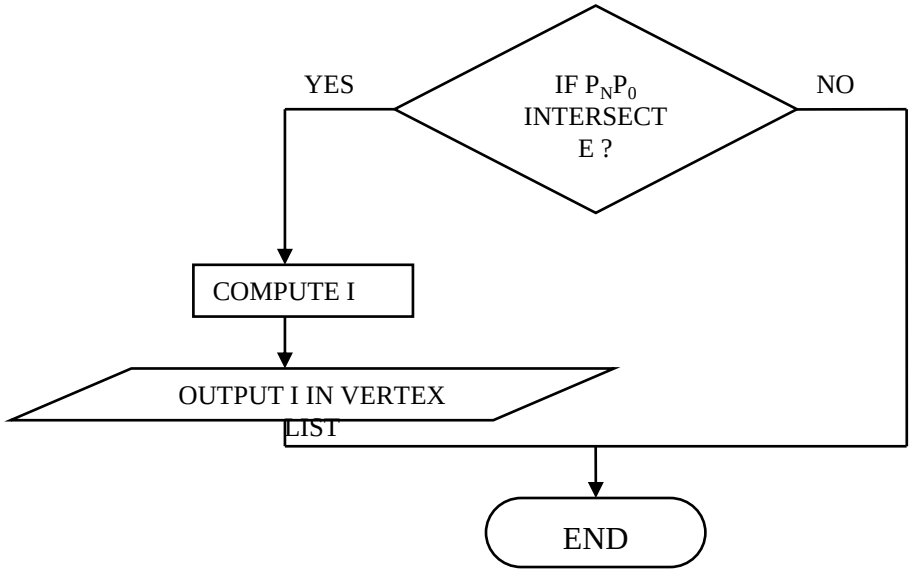
Flow Chart



Special case for first Vertex

Flow Chart

Special case for first Vertex



YOU CAN ALSO APPEND AN ADDITIONAL VERTEX $P_{N+1} = P_1$ AND AVOID SPECIAL CASE FOR FIRST VERTEX

Sutherland-Hodgeman Polygon Clipping

Inside/Outside Test:

Let *P(x,y)* be the polygon vertex which is to be tested against edge E defined form A(x_1, y_1) to B(x_2, y_2). Point *P* is to be said to the left (inside) of E or AB iff

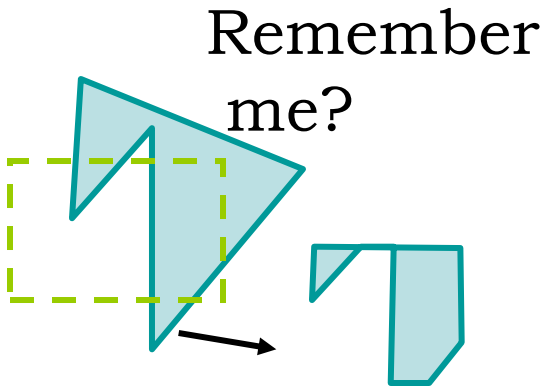
$$\frac{y - y_1}{y_2 - y_1} - \frac{x - x_1}{x_2 - x_1} > 0$$

or $C = (x_2 - x_1)(y - y_1) - (y_2 - y_1)(x - x_1) > 0$

otherwise it is said to be the right/Outside of edge E

Weiler-Atherton Polygon Clipping

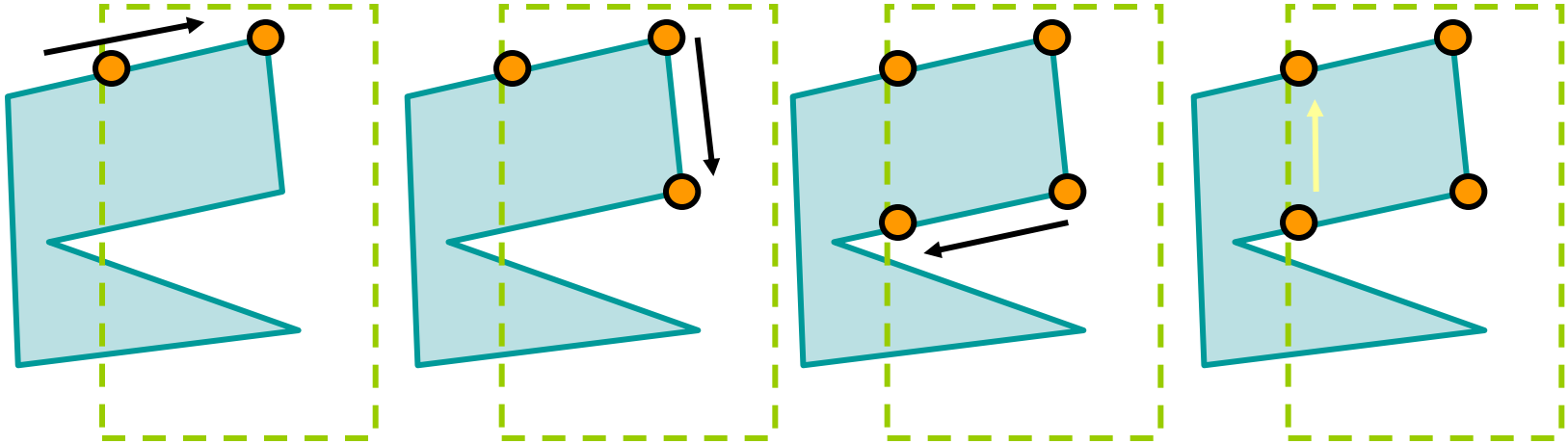
- Problem with Sutherland-Hodgeman:
  - Concavities can end up linked



- Weiler-Atherton creates separate polygons in such cases

Weiler-Atherton Polygon Clipping

- Example



add clip pt.
and end pt.

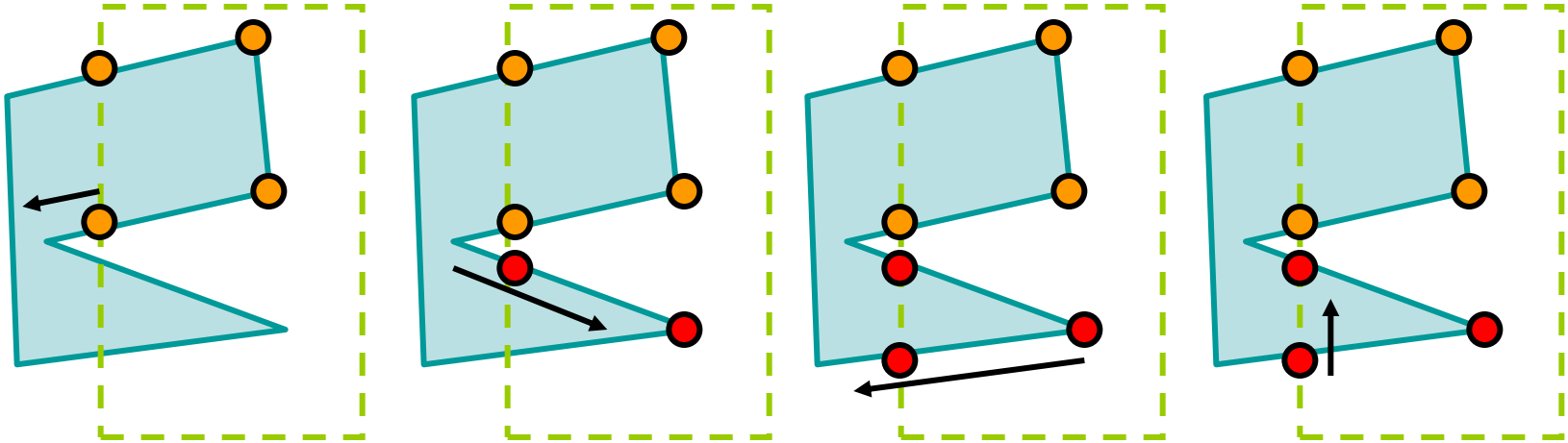
add end pt.

add clip pt.
cache old dir.

follow clip edge until
a) new crossing found
b) reach pt. already added

Weiler-Atherton Polygon Clipping

- Example (cont)



continue from cached location

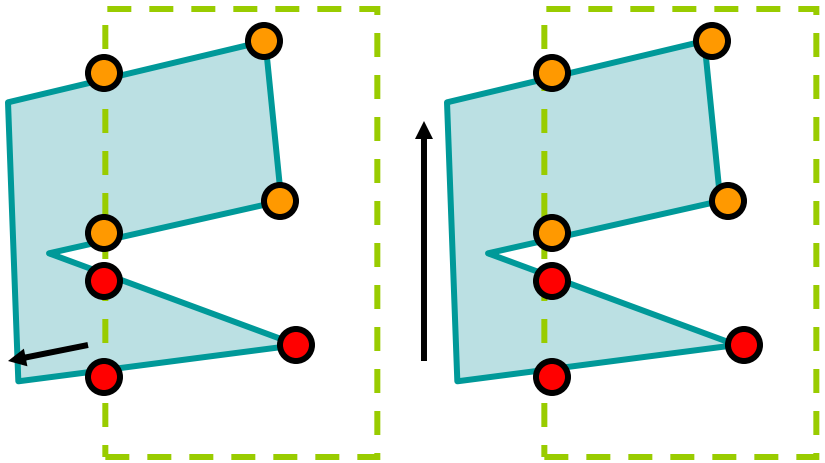
add clip pt. and end pt.

add clip pt. cache dir.

follow clip edge until
a) new crossing found
b) reach pt. already added

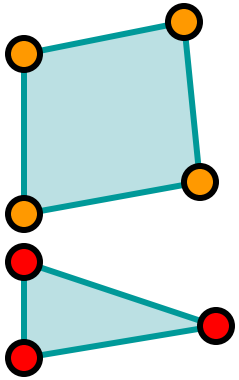
Weiler-Atherton Polygon Clipping

- Example (concluded)



continue from cached location

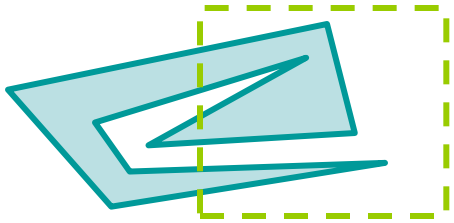
nothing added finished



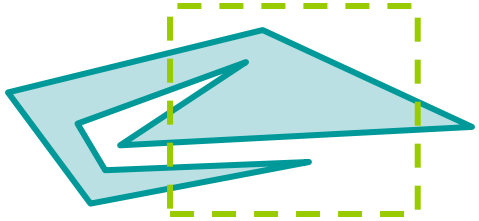
Final result:
Two *unconnected* polygons

Weiler-Atherton Polygon Clipping

- Difficulties:
  - What if the polygon re-crosses edge?



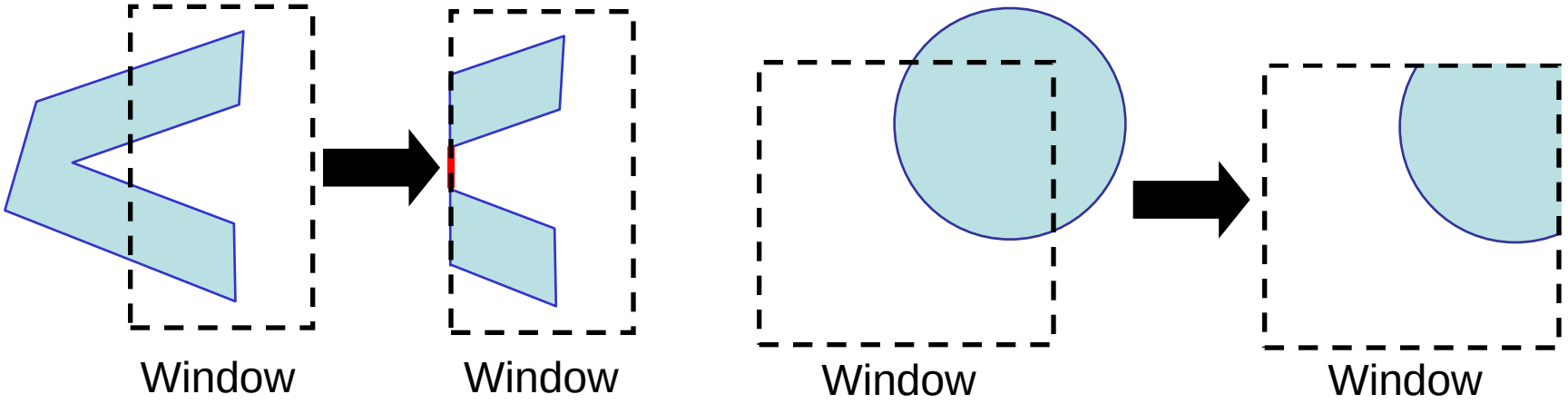
  - How many “cached” crosses?



  - Your geometry step must be able to *create* new polygons
    - Instead of 1-in-1-out

Other Area Clipping Concerns

- Clipping concave areas can be a little more tricky as often superfluous lines must be removed



- Clipping curves requires more work
  - For circles we must find the two intersection points on the window boundary

2D Clipping

1. Introduction
2. Point Clipping
3. Line Clipping
4. Polygon/Area Clipping
5. **Text Clipping**
6. Curve Clipping

Text Clipping

Text clipping relies on the concept of bounding rectangle

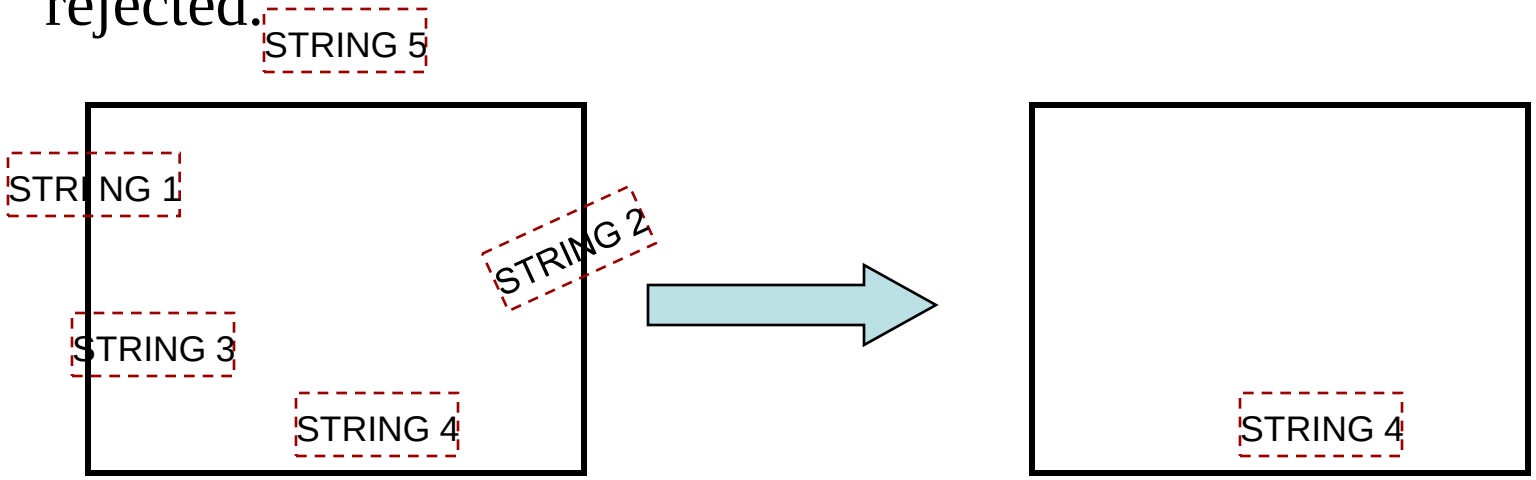
TYPES

1. All or None String Clipping
2. All or None Character Clipping
3. Component Character Clipping

Text Clipping

1. All or None String Clipping

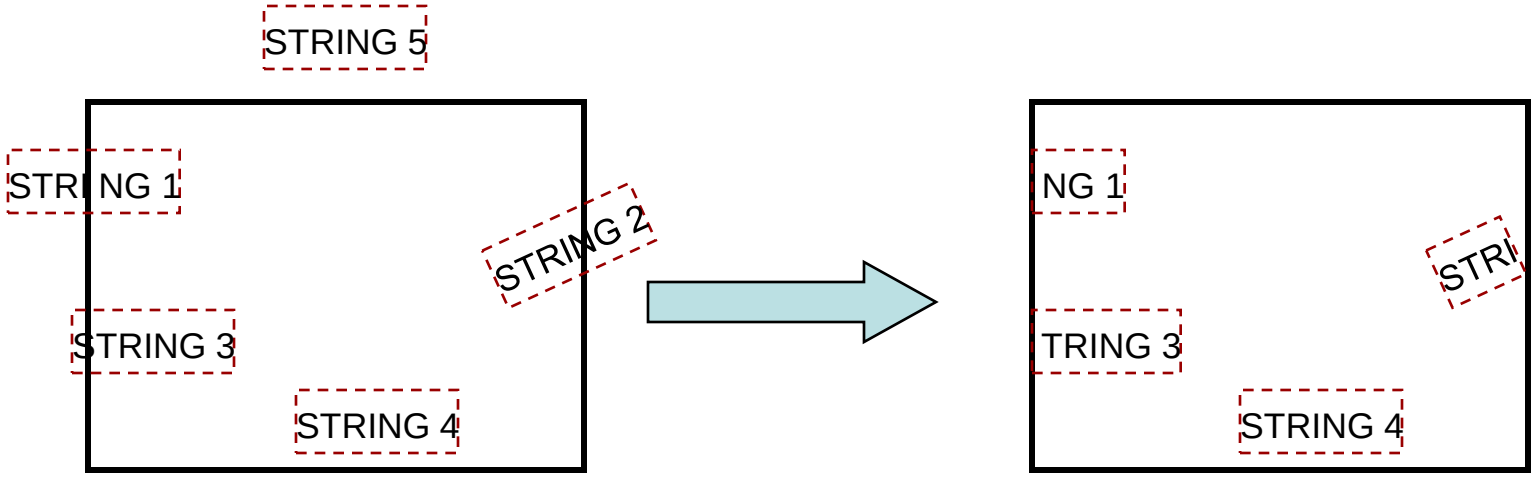
- In this scheme, if all of the string is inside window, we clip it, otherwise the string is discarded. This is the fastest method.
- The procedure is implemented by consider a ***bounding rectangle*** around the text pattern. The boundary positions are compared to the window boundaries. In case of overlapping the string is rejected.



Text Clipping

2. All or None Character Clipping

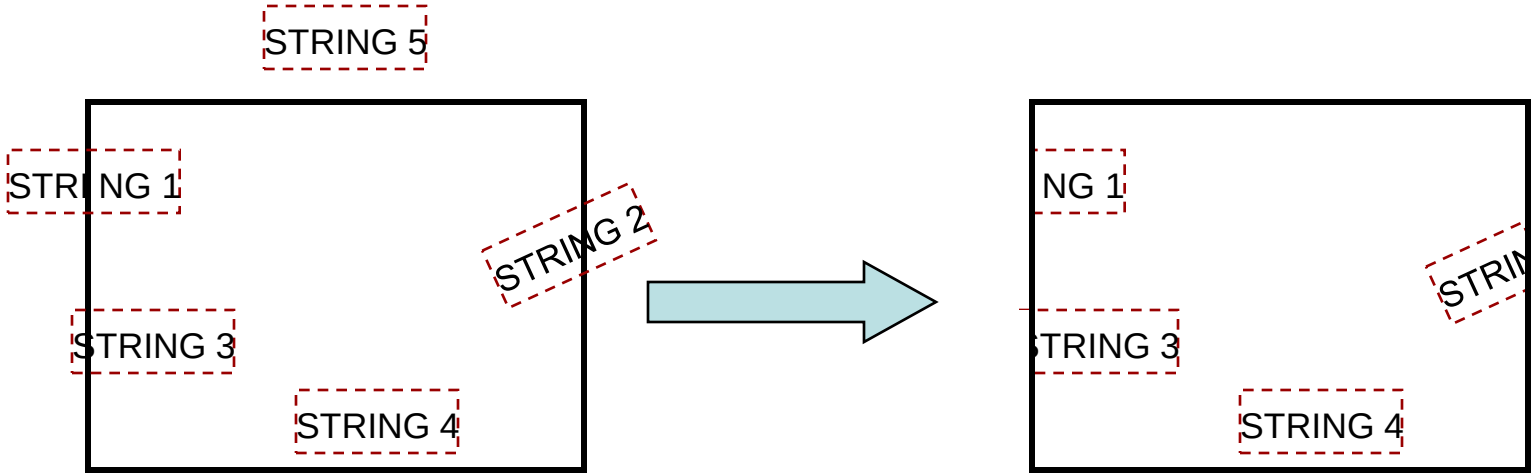
- In this scheme, we discard only those characters that are not completely inside window.
- Boundary limits of individual characters are compared against window. In case of overlapping the character is rejected.



Text Clipping

3. Component Character Clipping

- Characters are treated like graphic objects.
  - Bit Mapped Fonts : Point Clipping
  - Outlined Fonts : Line/Curve Clipping
- In case of overlapping the part of the character inside is displayed and the outside portion of the character is rejected.

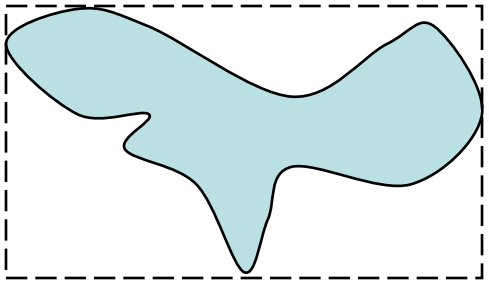


2D Clipping

1. Introduction
2. Point Clipping
3. Line Clipping
4. Polygon/Area Clipping
5. Text Clipping
6. **Curve Clipping**

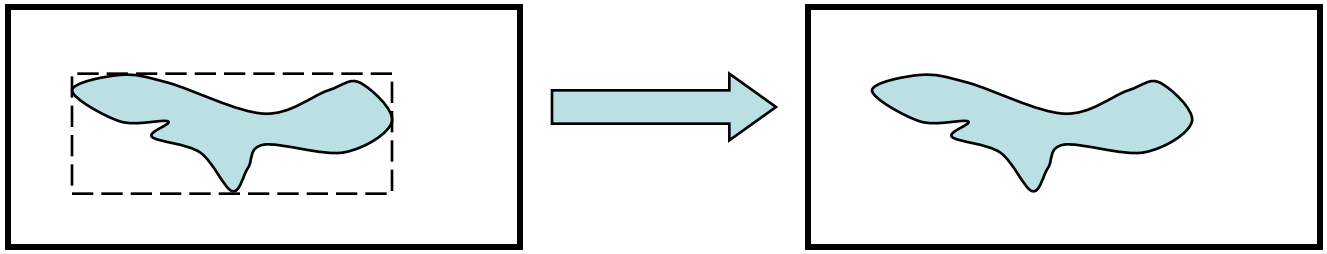
Curve Clipping

- Areas with curved boundaries can be clipped with methods similar to line and polygon clipping.
- Curve clipping requires more processing as it involve non linear equations.
- ***Bounding Rectangles*** are used to test for overlap with rectangular clip window.

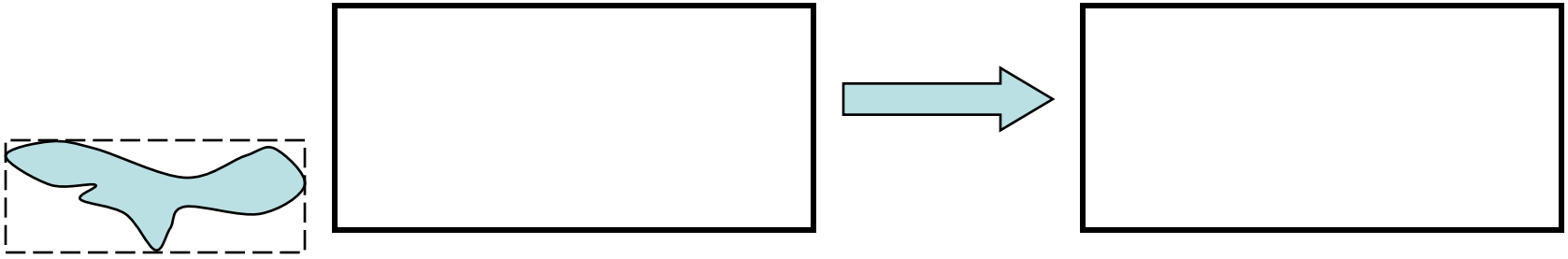


Curve Clipping

- If bounding rectangle is completely inside the object/curve is saved.

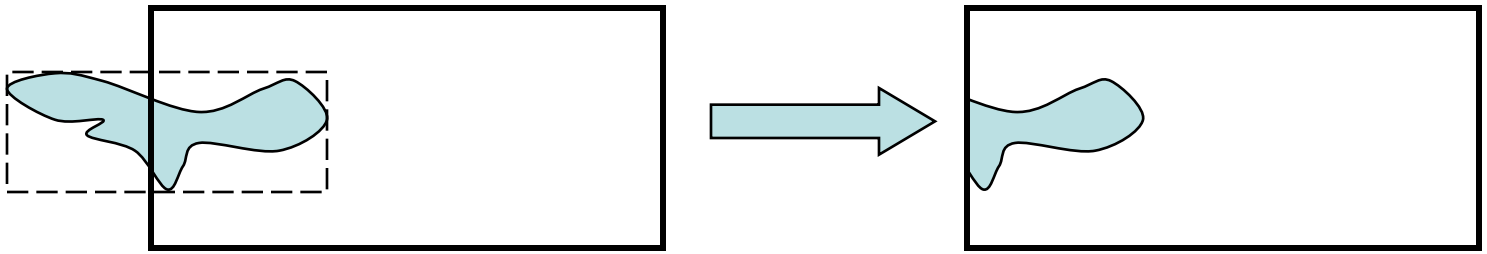


- If bounding rectangle is completely outside the object/curve is discarded.



Curve Clipping

- If both the above tests fails we use other computation saving approaches depending upon type of object
  - **Circle:** Use coordinate extent of individual quadrant, then octant if required.
  - **Ellipse:** Use coordinate extent of individual quadrant.
  - **Point:** Use point clipping



Any Question !