2D Viewing

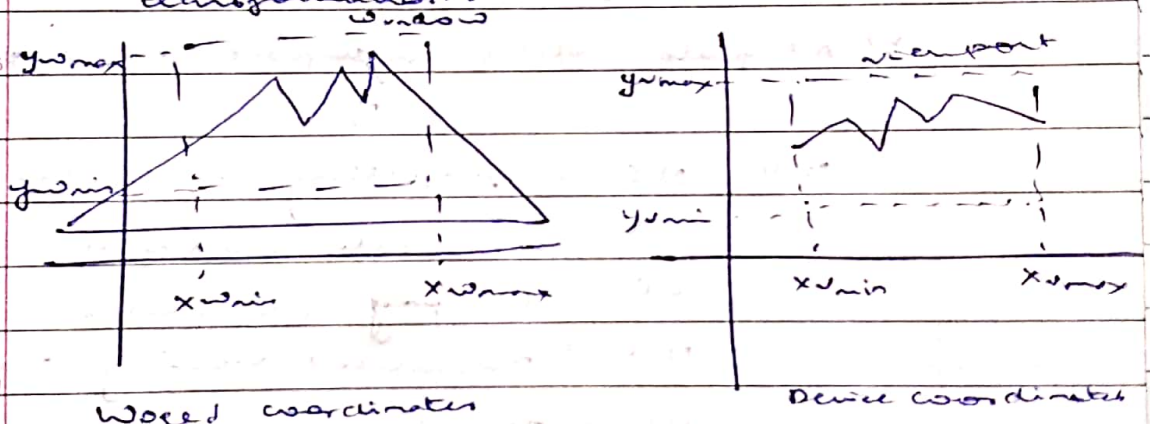
Viewing pipeline

→ Window: A world-coordinate area selected for display. Defines what is to be viewed.

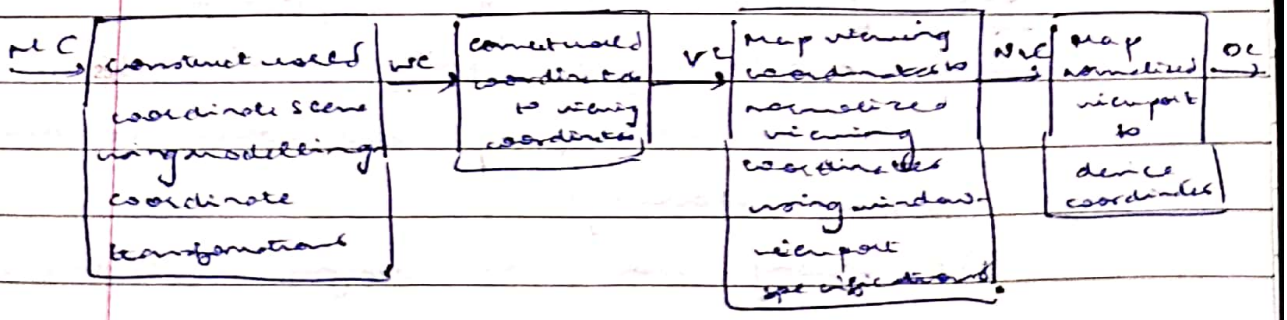
→ Viewport: Area of a display device to which a window is mapped. Defines where it is to be displayed.

→ Windows and viewports are rectangles in standard position.

→ Viewing transformation: Mapping of a part of a world-coordinate scene to device coordinates. Also called windowing transformation or window-to-viewport transformation.



2-D viewing transformation pipeline



Process of viewing transformation:

1) Construct scene in world coordinates using output primitives and attributes.
2) Obtain particular orientation for window.
   ↳ can create a 2D viewing coordinate system and define a window there.
   ↳ for arbitrary representation of rectangular window.
3) Once viewing reference frame is established, we can transform descriptions from world coordinates to viewing coordinates.
4) We then define viewport in normalized coordinates and map viewing coordinate description of the scene to normalized coordinates.
5) All parts outside viewport are clipped.

Note: MC → Modelling coordinates
WC → World coordinates
VC → Viewing coordinates
NVC → Normalized viewing coordinates
DC → Device coordinates

Viewing coordinate reference frame
↳ coordinate system provides the reference frame for specifying the world-coordinate window. Working:
1) A viewing-coordinate origin is selected at some world position: $P_0 = (x_0, y_0)$.
2) To establish the orientation, or rotation, of

this reference frame. Way to do this:

→ Specify a world vector V that defines the viewing y_v direction. vector V is called view up vector.

Now calculate components of unit vectors

$$v = (v_x, v_y) \text{ and}$$

$$u = (u_x, u_y)$$

for viewing y_v and x_v axes, respectively

→ These unit vectors are used to form first and second rows of rotation matrix R that aligns the viewing $x_v y_v$ axes with the world $x_w y_w$ axes.

→ To form composite transformation matrix,

i) translate view to world

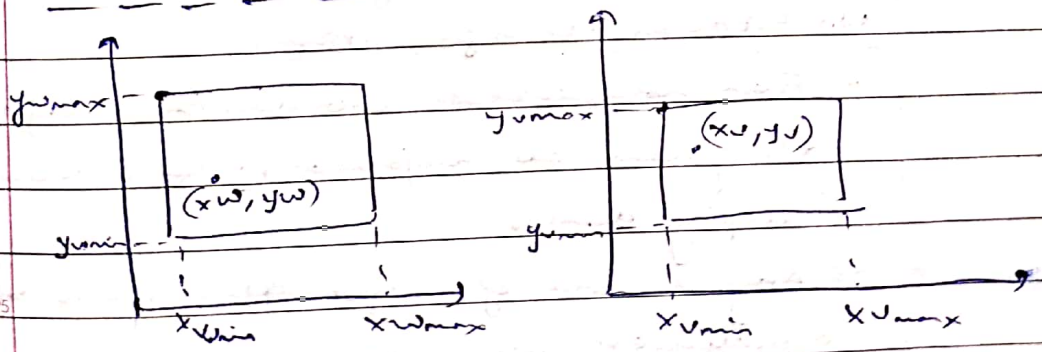
ii) rotate to align

$$M_{wc,vc} = R \cdot T$$

where T = Translation matrix

R = Rotation matrix

Window - To - Viewport coordinate Transformation



→ A point at position (x_w, y_w) in the window is mapped into position (x_v, y_v) in the associated viewport. To maintain the same relative placement in the viewport as the

window, we require that

$$\frac{x_v - x_{v_{min}}}{x_{v_{max}} - x_{v_{min}}} = \frac{x_w - x_{w_{min}}}{x_{w_{max}} - x_{w_{min}}} \quad -(1)$$

and $$\frac{y_v - y_{v_{min}}}{y_{v_{max}} - y_{v_{min}}} = \frac{y_w - y_{w_{min}}}{y_{w_{max}} - y_{w_{min}}}$$

Solving these expressions for viewport position (x_v, y_v), we have

$$x_v = x_{v_{min}} + (x_w - x_{w_{min}}) s_x$$
$$y_v = y_{v_{min}} + (y_w - y_{w_{min}}) s_y \quad -(2)$$

where scaling factors are:

$$s_x = \frac{x_{v_{max}} - x_{v_{min}}}{x_{w_{max}} - x_{w_{min}}}$$

$$s_y = \frac{y_{v_{max}} - y_{v_{min}}}{y_{w_{max}} - y_{w_{min}}} \quad -(3)$$

(2) can be described as the sequence:

(a) Scaling (of $(x_{w_{min}}, y_{w_{min}})$ that scales window area to size of viewport.

(b) Translate the scaled window area to position of viewport.

Workstation Transformation

→ From normalized coordinate, object descriptions are mapped to the various display devices

→ Any number of output devices can be open in a particular application and another window-to-viewport transformation can be performed for each output device.

→ This mapping is accomplished by selecting a window area in normalized space and a

viewport area in the coordinates of the display device.

2D - viewing functions

1) evaluateViewOrientationMatrix (x0, y0, xV, yV, error, viewMatrix)

→ x0, y0 : coordinates of viewing origin

→ xV, yV : world-coordinate position for view up vector

→ error : generated if input parameters are in error. otherwise the

→ viewMatrix for world-to-viewing transformation is calculated.

* Defines a viewing reference system in

2) To set up elements of window-to-viewport mapping matrix:

evaluateViewMappingMatrix (xwmin, xwmax, ywmin, ywmax, xVmin, xVmax, yVmin, yVmax, error, viewMappingMatrix)

→ xwmin, xwmax, ywmin, ywmax: window limits

→ xVmin, xVmax, yVmin, yVmax: viewport limits

3) Store combination of viewing and window-viewport mappings for various workstations in a viewing table

SetViewRepresentation (ws, viewIndex, viewMatrix, viewMappingMatrix, xclipmin, xclipmax, yclipmin, yclipmax, clipxy)

→ ws: output workstation

→ viewIndex: integer identifier for this particular window-viewport pair.

→ wwMatrix and viewMappingMatrix can be concatenated and referenced by viewIndex

→ clipxy: clip or noclip

4) Selects a particular set of options from view table

setViewIndex (viewIndex)

5) Apply workstation transformation by selecting workstation window-viewport pair:

setWorkstationWindow (ws, xwsWindmin, xwsWindmax, ywsWindmin, ywsWindmax)

setWorkstationViewport (ws, xwsVportmin, xwsVportmax, ywsVportmin, ywsVportmax)

Clipping operations

→ Any procedure that identifies those portions of a picture that are either inside or outside of a specified region of space is referred to as a clipping algorithm or simply clipping.

→ Region against which an object is to clipped is called a clip window.

→ Applications of clipping:

(i) extracting part of a defined scene for viewing

(ii): identifying visible surfaces in 3D views

(iii) anticalising line segments or object boundaries

(iv) creating objects using solid-modelling procedures

(v) displaying a multiwindow environment

(vi) drawing and painting operations that allow parts of a picture to be selected for copying, moving, erasing or duplicating.

→ Types:

(1) point clipping (2) line clipping

(3) Area clipping (4) curve clipping

(5) Text clipping

Point Clipping

→ point P(x,y) is clipped from a clip window (xwmin, xwmax, ywmin, ywmax) if it does not satisfy any one of the following:

$xw_{min} \leq x \leq xw_{max}$

$yw_{min} \leq y \leq yw_{max}$.

Line clipping

→ Test a given line segment to detect if it lies completely inside the clipping window or completely outside.

→ If above condition is not satisfied, perform intersection calculations with one or more clipping boundaries and calculation of multiple intersection points.

→ To avoid calculations, devise below algorithms:

$$x = x_1 + u(x_2 - x_1)$$

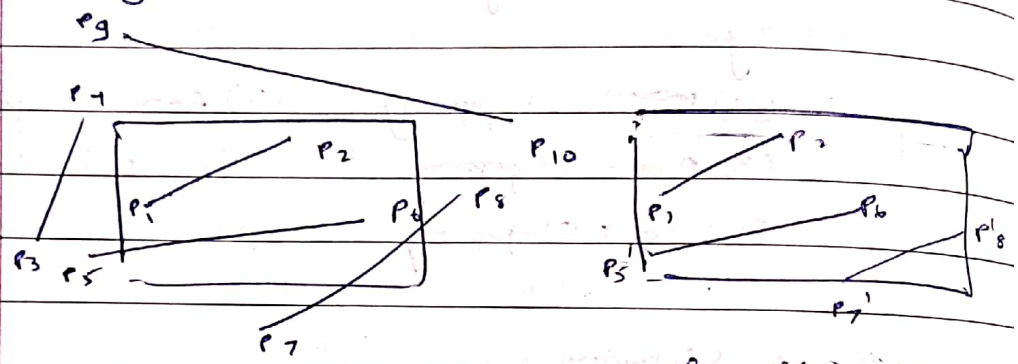
$$y = y_1 + u(y_2 - y_1) \qquad 0 \le u \le 1$$

At (x,y), values for u are calculated to check for intersection.

If u ≠ 0 to 1 line does not enter interior of window at boundary.

If u = 0 to 1, line does cross into clipping area.

Line segments that are parallel to window edges can be handled as special cases.



Before clipping

After clipping

Cohen Sutherland line clipping

→ oldest and most popular.

→ speeds up processing of line segments by performing initial tests that reduce the number of intersections that must be calculated.

→ every line endpoint in a picture is assigned a 4-digit binary code, called region code that identifies the location of point relative to the boundaries of the clipping rectangle.

bit	4	3	2	1
pos	above	below	right	left

→ . . ⇒ point is in that relative position

Camlin Page
Date / /

→ 0000 : window

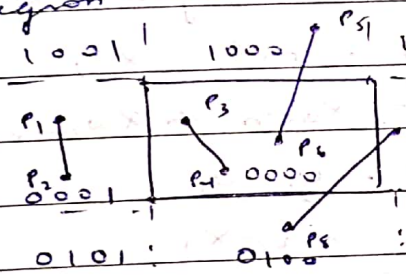
1001	1000	1010
0001	0000	0010
0101	0100	0110

→ Bit 1 : $x < x_{wmin}$ $(x - x_{wmin})$ Bit 3 : $y < y_{min}$ $(y - y_{min})$

Bit 2 : $x > x_{wmax}$ $(x_{wmax} - x)$ Bit 4 : $y > y_{max}$ $(y_{max} - y)$

1) Calculate differences b/w endpoint coordinates and clipping boundaries

2) Use the constant sign bit of each difference calculation to set corresponding value in the region code.



case (1) P_1 and P_2

0001 → N2

& 0001 → N2

0001 → N2

completely outside ⇒ reject

case (2) : P_3 and P_4

0000 → Z

& 0000 → Z

0000 → Z

Accept the line (no clipping)

case (3) : P_5 and P_6

1010 N2 clipping

& 0000 Z

0000 Z

(partially inside)

Camlin Page
Date / /

Case (4) : P7 and P8

0010 → N2
0100 → N2
0000 → ?

} clipping

possibly inside

Intersection points with a clipping boundary can be calculated using the slope-intercept form of the line equation. For a line with endpoint coordinates (x_1, y_1) and (x_2, y_2), the y coordinates of the intersection point with a vertical boundary can be obtained as:

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

where $x = x_{wmin}$ or x_{wmax}

$$m = \frac{y_2 - y_1}{x_2 - x_1}$$

Similarly x coordinate with horizontal boundary

$$x = x_1 + \frac{y - y_1}{m}$$

where $y = y_{wmin}$ or y_{wmax}

In case (3): P_5' needs to be calculated.

$P_5' = 0000$ z
0000 z
0000 → Accept

Clip P_5' and P_5.

Algorithm:

Step 1: For each end point, region code is assigned

Step 2: The line is accepted if both endpoints

Camlin Page
Date / /

raw region code 0000

Step 3: Else, logical AND operation is performed for both region code.

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

Step 3.2: Else clipping is required.

Step 3.2.1: An end point of line is selected that is outside the window

Step 3.2.2: Find the intersection point of the window boundary

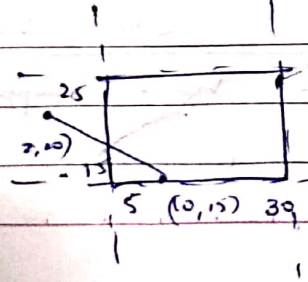
Step 3.2.3: The end point is replaced with the intersection point and the region code is updated.

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

Step 4: Repeat steps for other lines.

Q1) Clip the line with the coordinates $(x_1, y_1) = (2, 20)$ and $(x_2, y_2) = (10, 15)$ against a window having lower left corner at (5, 15) and upper right corner at (30, 25) using cohen sutherland line clipping algorithm.

Ans



Camlin Page
Date / /

1) Region code for $(x_1, y_1) = (2, 20)$

Bit 1: $x - x_{wmin} = 2 - 5 = -3$ (-ve) 1

Bit 2: $x_{wmax} - x = 30 - 2 = 28$ (+ve) 0

Bit 3: $y - y_{wmin} = 20 - 15 = 5$ (+ve) 0

Bit 4: $y_{wmax} - y = 25 - 20 = 5$ (+ve) 0

Region code = 0001

2) Region code for $(x_2, y_2) = (10, 15)$

Bit 1: $x - x_{wmin} = 10 - 5 = 5$ (+ve) = 0

Bit 2: $x_{wmax} - x = 30 - 10 = 20$ (+ve) 0

Bit 3: $y - y_{wmin} = 15 - 15 = 0$ (+ve) 0

Bit 4: $y_{wmax} - y = 25 - 15 = 10$ (+ve) 0

Region code = 0000

0001 → N Z

0000 → Z

0000 → Z

Clipping at point 1. vertical intersection

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

$= 20 - \frac{5}{8}(5 - 2)$

$= 18.125$

$(5, 18.125)$

$\therefore m = \frac{y_2 - y_1}{x_2 - x_1}$

$= \frac{-15.2}{18.2}$

$= \frac{-5}{5}$

$= -1$

$x = x_{min}$

$\therefore$ intersection at x_{min}

3) Region code for $(x_1', y_1') = (5, 18.125)$

Bit 1: $x - x_{wmin} = 5 - 5 = 0$ 0

Bit 2: $x_{wmax} - x = 30 - 5 = 25$ 0

Bit 3: $y - y_{wmin} = 18.125 - 15 = 3.125$ 0

Bit 4: $y_{wmax} - y = 25 - 18.125$ 0

Region code = 0000

Hence, we have clipped the line segment

Camlin Page
Date / /

Q2) window : LL (5, 15) UR (30, 25)

$(x_1, y_1) = (2, 16)$ $(x_2, y_2) = (10, 15)$

Ans- At (2, 16)

Bit 1: 2 − 5 = −3 < 0 → 1

2: 30 − 16 = 14 > 0 → 0

3: 16 − 16 = 1 → 0

4: 25 − 16 = 9 → 0

Region code = 1000

At (10, 15)

Bit 1: 10 − 5 = 5 → 0

2: 30 − 10 = 20 → 0

3: 15 − 15 = 0 → 0

4: 25 − 15 = 10 : 0 → 0

Region code: 0000

0001 → NZ

0000 → Z

0000 → Z

Point 1 has to be clipped.

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

$= 16 + \frac{15-16}{10-2}(5 - 2)$

$= 15.625$

(5, 15.625)

At (5, 15.625)

Bit 1: 5 − 5 = 0 → 0

Bit 2: 30 − 5 = 0 → 0

Bit 3: 15.625 − 15 = 0.625 → 0

Bit 4: 25 − 15.625 → 0

Region code = 0000

⇒ successfully clipped.

Camlin Page
Date / /

03) (10,12) and (34,27) for window

UR = (x_{max}, y_{max}) = (30, 25) LL = (x_{min}, y_{min}) = (5, 15)

Ans- At (10,12)

Bit 1: 10 − 5 = 5 → 0

2: 30 − 10 = 20 → 0

3: 12 − 15 = −3 → 1

4: 15 − 12 = 3 → 0

Region code = 0100

At (34,27)

Bit 1: 34 − 5 → 0

2: 30 − 34 → 1

3: 27 − 15 → 0

4: 25 − 27 → 1

Region code = 1010

0100 → NZ

1010 → NZ

0000 → Z

Perform clipping at both points.

(a) Point 1 (10,12) Cut horizontally.

$y = y_{min} = 15$ [∵ below]

$x = x_1 + \frac{y_{min} - y_1}{m}$

$m = \frac{27-12}{34-10} = \frac{15}{24} = \frac{5}{8}$

$x = 10 + \frac{(15-12) \times 8}{5}$

$= 14.8$

Point = (14.8, 15) UBRL 1010

(b) Point 2 (34,27)

$x = x_{wmax} = 30$

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

$= 27 + \frac{5}{8}(30 - 34)$

$= 24.5$

$(30, 24.5)$

Check for x for each Region code in that case is 0010

Region code at $(14.8, 15)$

Bit 1: $14.8 - 5$ 0

Bit 2: $30 - 14.8$ 0

3: $15 - 15$ 0

4: $25 - 15$ 0

Region code = 0000

⇒ clipping successful.

At $(30, 24.5)$

Bit 1: $30 - 5$ 0

2: $30 - 30$ 0

3: $24.5 - 15$ 0

4: $25 - 24.5$ 0

Region code = 0000

⇒ clipping successful.

Q4) Clip the line P1P2 with coordinates $(x_1, y_1) = (-13, 5)$ and $(x_2, y_2) = (17, 11)$ against a window having lower left corner at $(-8, -4)$ and upper right corner at $(12, 8)$ using cohen sutherland line clipping Algorithm.

Ans - $(x_1, y_1) = (-13, 5)$ $(x_2, y_2) = (17, 11)$

$LL = (\overset{min}{-8}, -4)$ $UR = (\overset{max}{12}, 8)$

At $(-13, 5)$

Bit 1 = $-13 + 8 = -5$ 1

Bit 2 = $12 + 13 = 25$ 0

Bit 3 = $5 + 4 = 9$ 0

Bit 4 = $8 - 5 = 3$ 0

Camlin Page
Date / /

U B R L

Region code = 0 0 0 1

(vertical $y = y_t$ ... left

At (17, 11)

Bit 1 : 17 + 8 — 0

2 : 12 − 17 — 1

3 : 11 + 4 — 0

4 : 8 − 11 — 1

Region code : 1 0 1 0

0 0 0 1 → N Z

1 0 1 0 → N Z

0 0 0 0 → Z

Point 1 and 2 to be clipped.

(a) For point 1 : (−13, 5)

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

$$= 5 + \frac{11-5}{17+13}(-8+13)$$

$$= 5$$

Point = (−8, 5)

Region code will be as:

Bit 1 : −8 + 8 — 0

2 : 12 + 8 — 0

3 : 5 + 4 — 0

4 : 8 − 5 — 0

Region code : 0 0 0 0

⇒ successfully clipped.

(b) For point 2 : (17, 11)

Assume vertical intersection

$$y = y_1 + m_{mx}(x - x_1)$$

$$= 11 + \frac{11-5}{17+13}(12-17)$$

$$= 10$$

(12, 10)

Bit 1: 12 + 8 — 0

2: 12 − 12 — 0

3: 10 + 4 — 0

4: 8 − 10 — 1

Region code = 1000

⇒ not accepted.

Assume horizontal intersection:

$$x = x_1 + \frac{y - y_1}{m} \quad [y = y_{max}]$$

$$= 17 + \frac{(8 - 11)\,30}{6}$$

$$= 2$$

(2, 8)

Bit 1: 2 + 8 — 0

2: 12 − 2 — 0

3: 8 + 4 — 0

4: 8 − 8 — 0

Region code = 0000

⇒ (2, 8) is accepted.

Points are (−8, 5) and (2, 8)

Q5) Find the region codes for a given line segment PQ with the endpoints given as P(−1, 5) and Q(−3, 8) and then clip the line wrt a rectangular window with lower left corner at (−3, 1) and upper right corner at (2, 6) using cohen sutherland line clipping algorithm. 2,

Ans- $(x_1, y_1) = (-1, 5)$ $(x_2, y_2) = (3, 8)$

LL = (−3, 1) UR = (2, 6)

At (−1, 5)

Camlin Page
Date / /

Bit 1: $-1 + 3$ → 1
2: $2 + 1$ → 0
3: $5 - 1$ → 0
4: $6 - 5$ → 0

Region code = 0001

At (3, 8)

Bit 1: $3 + 3$ → 0
Bit 2: $2 - 3$ → 1
3: $8 - 1$ → 0
4: $6 - 8$ → 1

Region code = 1010

0001
1010
0000 → 2

Both points P and Q need to be clipped at left vertical.

(a) Clipping P(−1, 5)

$$y = y_1 + m(x_{min} - x_1)$$
$$= 5 + \frac{8-5}{3+1}(-3+1)$$
$$= 3.5$$

$P' = (-3, 3.5)$

Bit 1: $-3 + 3$ → 0
2: $2 + 3$ → 0
3: $3.5 - 1$ → 0
4: $6 - 3.5$ → 0

Region code = 0000

∴ P' (−3, 3.5) and successfully clipped

(b) clipping (3, 8)

Let us take horizontal clipping

$$x = x_1 + \frac{y - y_1}{m}$$

$$= 3 + \frac{(6 - 8)(4)}{3}$$

$$= 0.333$$

$$(0.333, 6)$$

Bit 1: 0.333 + 3 → 0

2: 2 - 0.333 → 0

3: 6 - 1 → 0

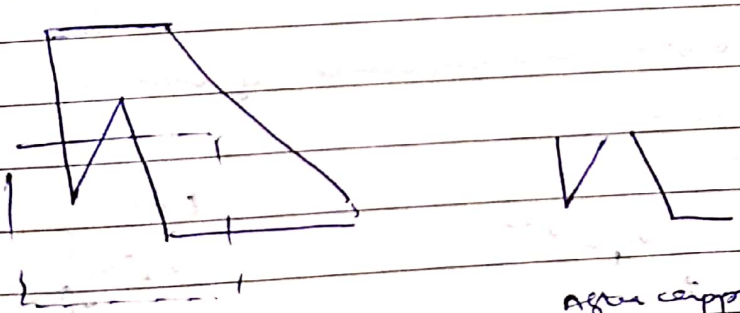
4: 6 - 6 → 0

Region code = 0000

∴ Q' = (0.333, 6)

and successfully clipped.

Polygon clipping



Before clipping

After clipping

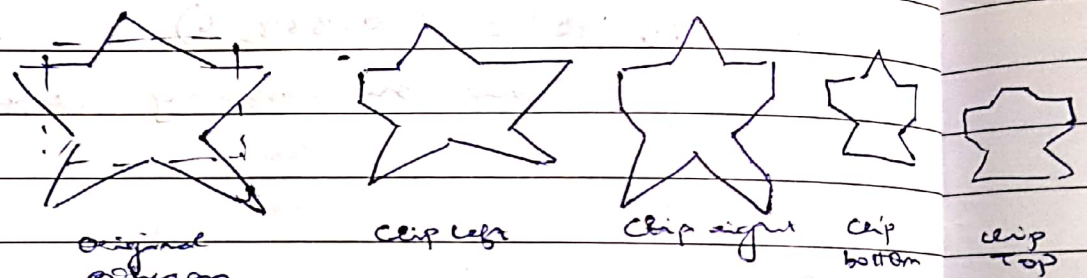
→ an algorithm that will generate one or more closed areas that are then scan converted for the appropriate area fill.

→ O/P of polygon clipper should be a sequence of vertices that defines the clipped polygon boundaries.

Sutherland-Hodgeman Polygon clipping

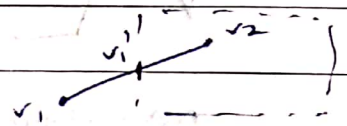
→ processing the polygon vertices against each clip rectangle boundary.

ie we first clip the polygon against the left rectangle boundary to produce a new sequence of vertices. The new set of vertices could then be successively passed to a right boundary clipper, a bottom boundary clipper, and a top boundary clipper. At each step, a new sequence of output vertices is generated and passed to the next window boundary clipper.



To find new sequence of vertices we have 4 cases.

(i)

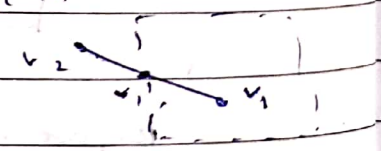


outside → inside

O/P : $v_1' v_2$

(intersect + dest)

(ii)

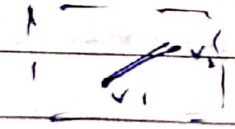


inside - outside

O/P : intersect

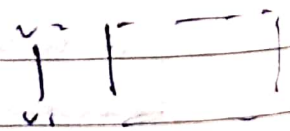
$= v_1'$

(iii)



in → in

O/P dest v_2

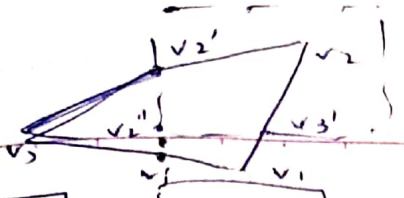


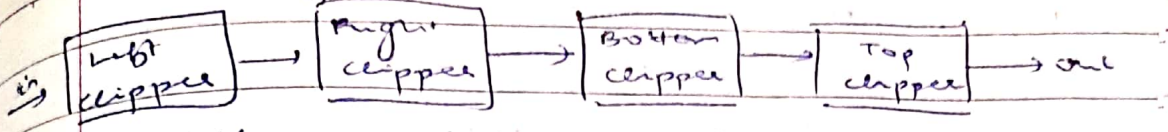
out - out

O/P : null

Camlin | Page
Date / /

eg(1)





$v_1 \longrightarrow v_1 \longrightarrow v_1 \longrightarrow$ NULL

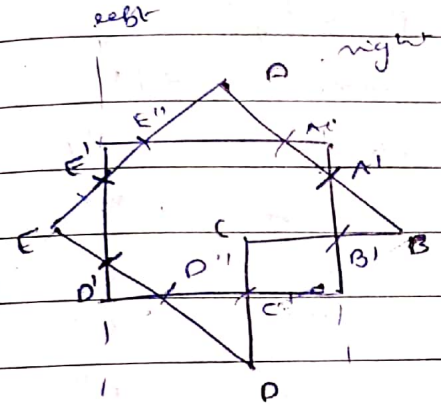
$v_2 \longrightarrow v_1' \longrightarrow v_1' \longrightarrow$ NULL

$v_3 \longrightarrow \{v_2', v_3\} \longrightarrow \{v_2', v_3\} \longrightarrow \{v_2'', v_2'\} \longrightarrow v_2'', v_2'$

$v_3 \longrightarrow v_3$

$v_3' \longrightarrow v_3'$

eg(2):



(i) against left edge

vertex	case	output
AB	in-in	B
BC	in-in	C
CD	in-in	D
DE	in-out	D'
EA	out-in	E'A

(ii) against right edge.

vertex	case	output
AB	in-out	A'
BC	out-in	B'C
CD	in-in	D
DD'	in-in	D'
D'E'	in-in	E'
E'A	in-in	A

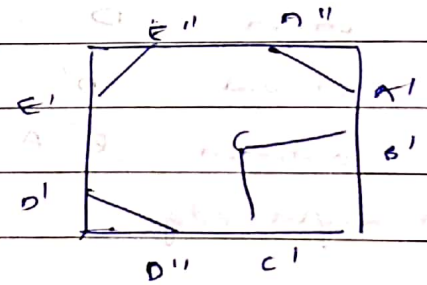
(iii) against bottom edge

vertex	case	output
AA'	in-in	A'

A'B'	in-in	B'
B'C	in-in	C
CD	in-out	C'
DD'	out → in	D''D
D'E'	in-in	E'
E'A	in-in	A

(iv) against top edge

vertex	case	O/P
AA'	out-in	A''A'
A'B'	in-in	B'
B'C	"	C
CC'	"	C'
C'D''	"	D''
D''D'	"	D'
D'E'	"	E'
E'A	in-out	E''



Cases are explained as :

(1) If the first vertex is outside the window boundary and the second vertex is inside, both the intersection point of the polygon edge with the window boundary and the second vertex are added to the output vertex list.

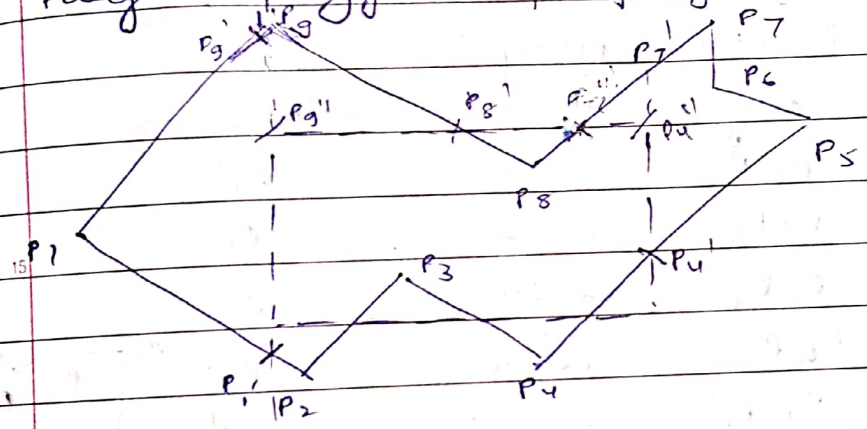
(2) If both input vertices are inside the window boundary, only the second vertex is

added to the output vertex list.

(3) If the first vertex is inside the window boundary and the second vertex is outside, only the edge intersection with the window boundary is added to the output vertex list.

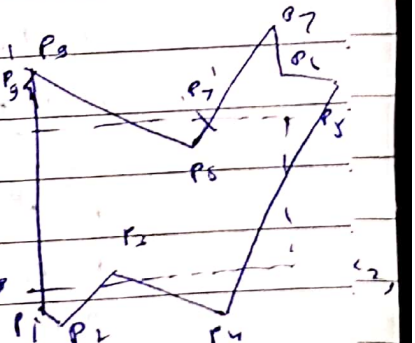
(4) If both input vertices are outside the window boundary, nothing is added to the output list.

Q) Clip the following polygon using sutherland Hodgeman polygon clipping algorithm



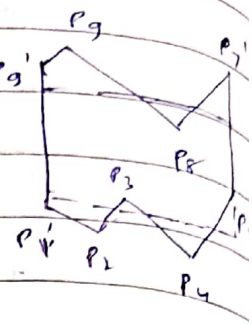
Ans (i) left clipping

edge	case	o/p vertex
P_1P_2	out → in	$P_1'P_2$
P_2P_3	in → in	P_3
P_3P_4	"	P_4
P_4P_5	"	P_5
P_5P_6	"	P_6
P_6P_7	"	P_7
P_7P_8	"	P_8
P_8P_9	"	P_9
P_9P_1	in → out	P_9'



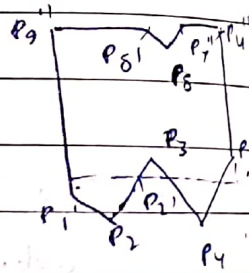
(ii) right clipping

edge	case	O/P vertex
$P_1'P_2$	in-in	P_1'
P_2P_3	"	P_2
P_3P_4	"	P_3 P_g'
P_4P_5	in→out	P_4'
P_5P_6	out-out	—
P_6P_7	out-out	—
P_7P_8	out-in	$P_7'P_8$
P_8P_g	in-in	P_8
P_gP_g'	in-in	P_g
$P_g'P_1'$	in-in	P_g'



(iii) top clipping

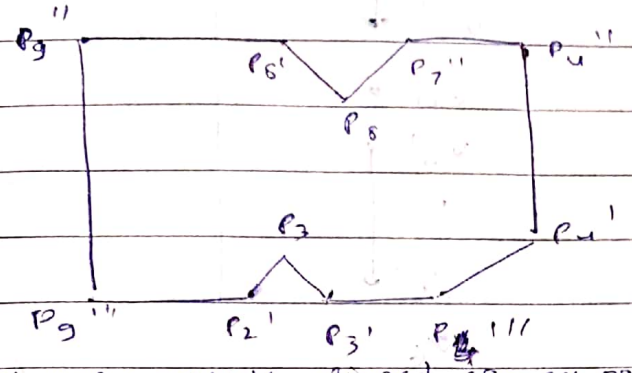
edge	case	O/P vertex
$P_1'P_2$	in-in	P_1'
P_2P_3	in-in	P_2
P_3P_4	in-in	P_3
P_4P_4'	in-in	P_4
$P_4'P_7'$	in-out	P_4''
$P_7'P_8$	out-in	$P_7''P_8$
P_8P_g	in-out	P_8'
P_gP_g'	out-out	—
$P_g'P_1'$	out-in	$P_g''P_1'$



(iv) bottom clipping

edge	case	O/P vertex
$P_1'P_2$	out-out	—
P_2P_3	out-in	$P_2'P_3$
P_3P_4'	in-out	P_3'
P_4P_4'	out-in	$P_4''P_4'$
$P_4'P_4''$	in-in	P_4'
$P_4''P_7''$	in-in	P_4''

$P_7'' P_8$	in-in	P_7'
$P_8 P_5'$	"	P_8
$P_8' P_9''$	"	P_5'
$P_9'' P_1'$	in-out	P_9'''



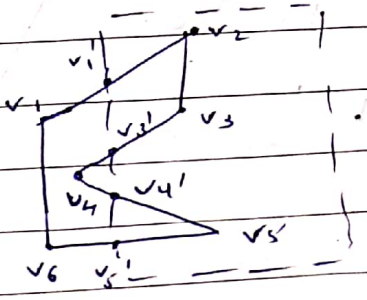
limitation: only correctly clipped convex polygon ($0° < \angle < 180°$)

Weiler Atherton Polygon clipping

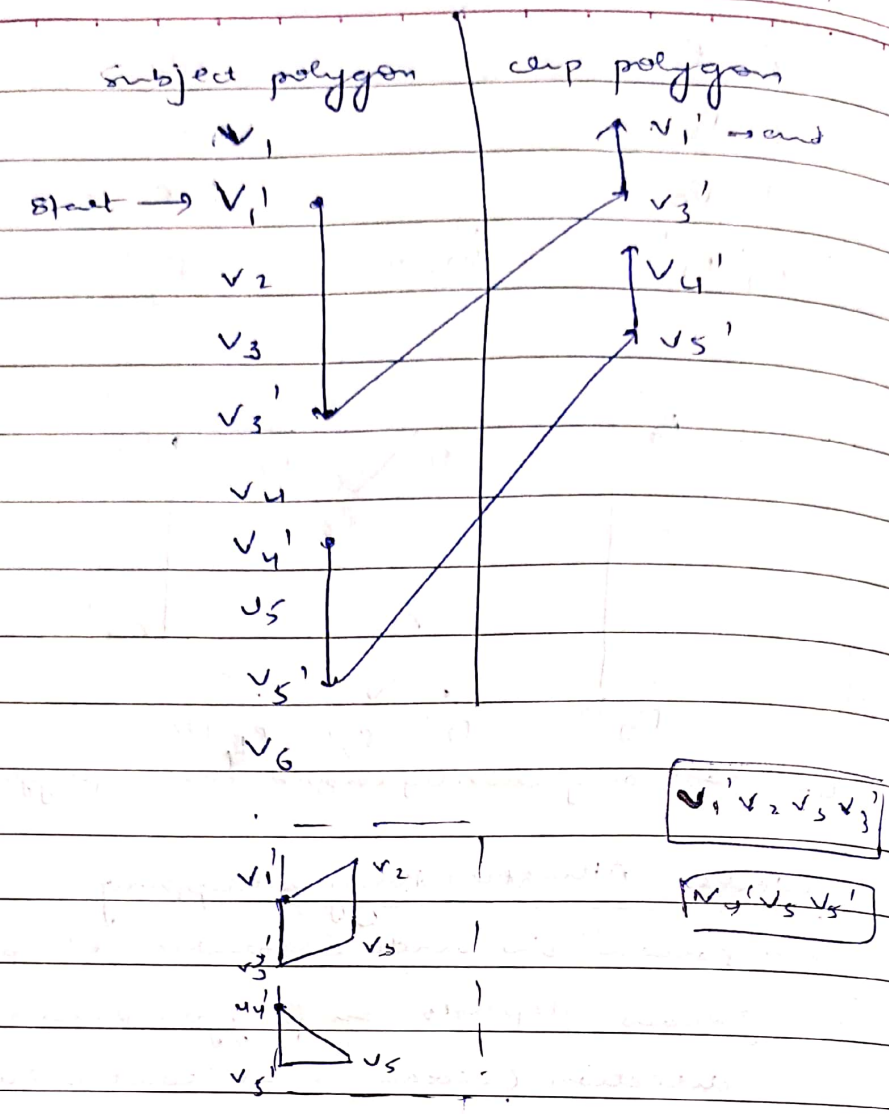
→ follow the window boundaries: pathline follow depends on polygon processing direction (clockwise or counterclockwise) and whether the pair of polygon vertices currently being processed represents an out-in pair or in-out pair. For clockwise processing, use following rules:

1) out-in : follow polygon boundary.

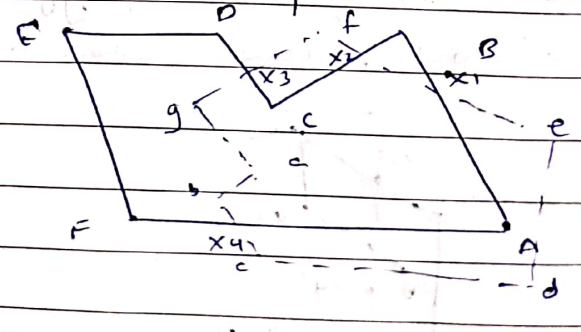
2) in-out, follow window boundary in clockwise



→ create 2 lists: one for object polygon and the other for clipping polygon.

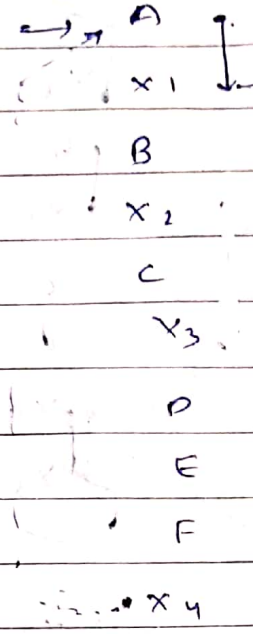


Q2) Explain how polygon in Figure 1 will be clipped using the Weiler Atherton Polygon clipping algorithm (solid line represents polygon and dashed line represents window)

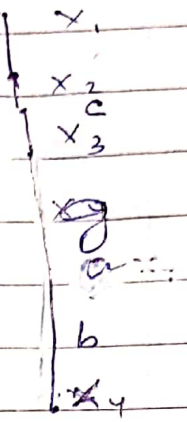


Ans- Mark intersections with window as x_1, x_2, x_3 and x_4 as marked in the figure. Create 2 lists:

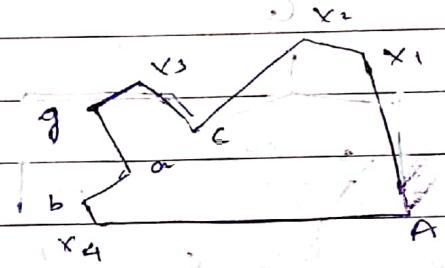
Subject Polygon



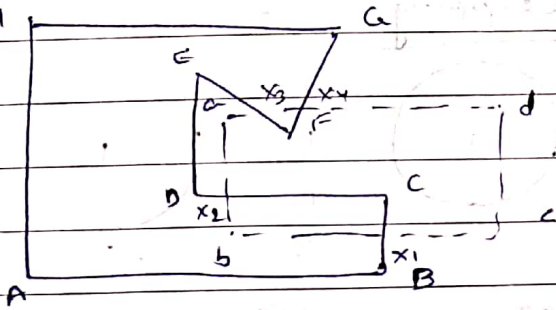
Clip Polygon



d
e
f



Q3) Clip the following polygon using Weiler Atherton polygon clipping Algorithm.

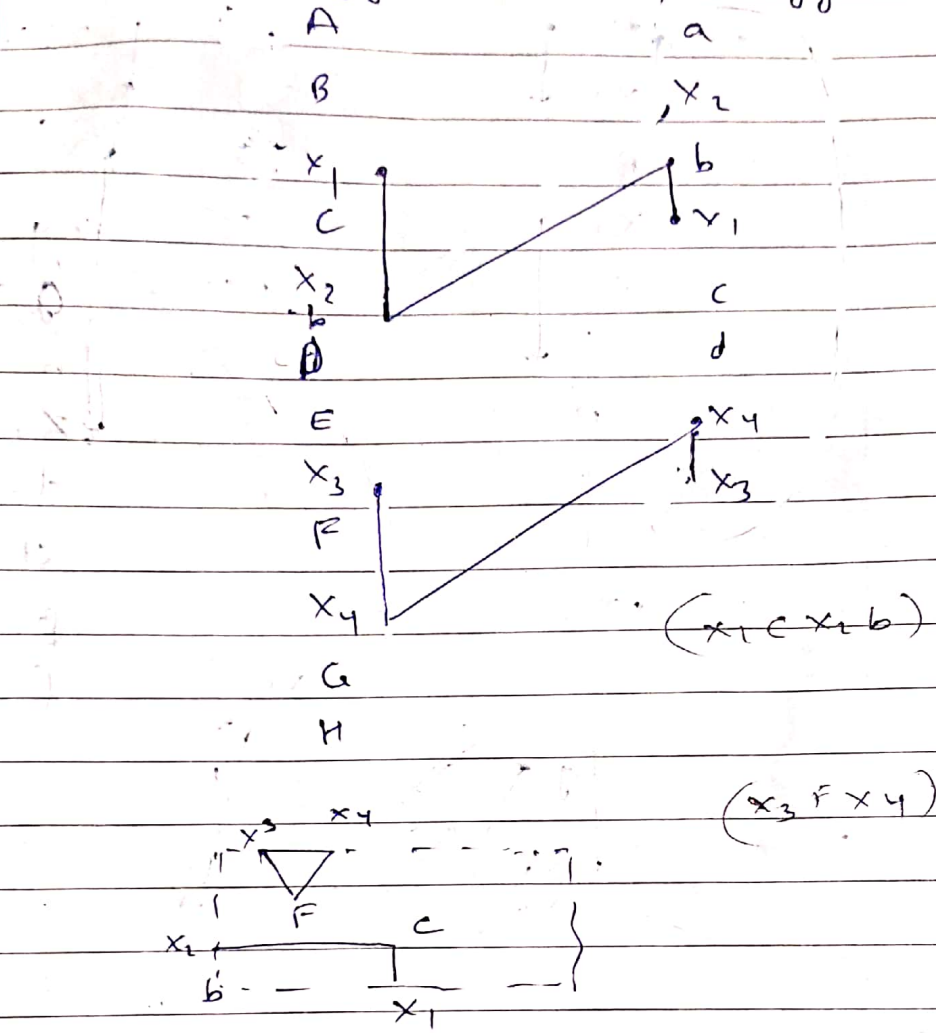


Ans - Mark intersections with window as x_1, x_2, x_3, x_4 as shown.

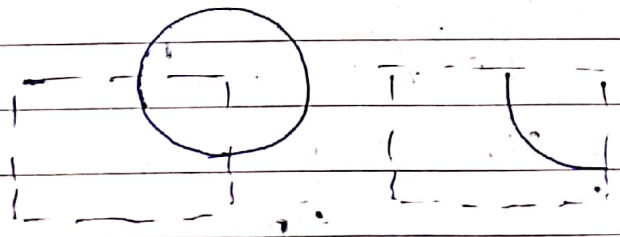
Draw 2 lists as below:

Camlin | Page
Date / /

Subject Polygon Clip Polygon



Curve Clipping



Before clipping After clipping

→ clipping areas with curved boundaries

→ requires more processing than for objects with linear boundaries.

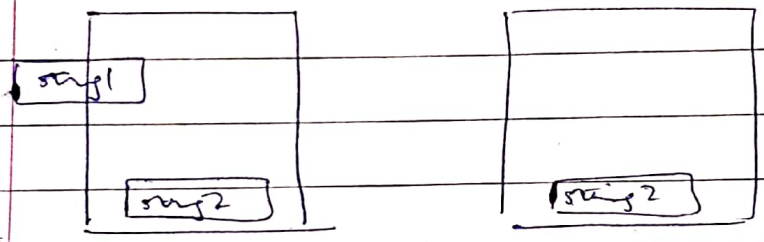
→ bounding rectangle for curved object can be used first to test for overlap with a rectangular clip window.

→ If bounding rectangle for the object is completely inside the window, we save the object.

→ If rectangle is determined to be completely outside the window, we discard the object.

→ For other cases, we can use coordinate extents of individual quadrants for circles and ellipses.

Text Clipping



Before Clipping After Clipping

→ Technique used depends on methods used to generate characters and the requirements of a particular application.

→ all-or-none string-clipping strategy: If all of the string is inside a clip window, we keep it. Otherwise, the string is discarded. The procedure is implemented by considering a bounding rectangle around the text pattern. produces fastest text clipping.

→ all-or-none character-clipping strategy: Here, we discard only those characters that are not completely inside the window. In this case, the boundary limits of individual characters are compared to the window.

→ clip components of individual characters:
each character is treated in a way like lines.
If an individual character overlaps a clip window boundary, we clip off the parts of the character that are outside the window.
Characters defined with bit maps would be clipped by comparing the relative position of the individual pixels in the character grid patterns to the clipping boundaries.