By: Mukunda Paudel
paudelmuku@gmail.com

COMPUTER GRAPHICS AND ANIMATION

Disclaimer

*This document is part of teaching materials for COMPUTER GRAPHICS AND ANIMATION under the Tribhuvan University syllabus for Bachelors of Arts in Computer Application (**BCA**).This document does not cover all aspect of learning COMPUTER GRAPHICS, nor are these be taken as primary source of information. As the core textbooks and reference books for learning the subject has already been specified and provided to the students, students are encouraged to learn from the original sources because this document cannot be used as a substitute for prescribed textbooks..*

*Various text books as well as freely available material from internet were consulted for preparing this document. Contents in This document are **copyrighted** to the instructor and authors of original texts where applicable.*

By: Mukunda Paudel
paudelmuku@gmail.com

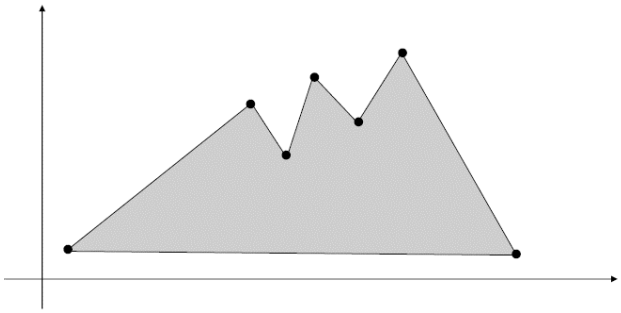
Unit 3: Clipping

2 Dimensional Viewing

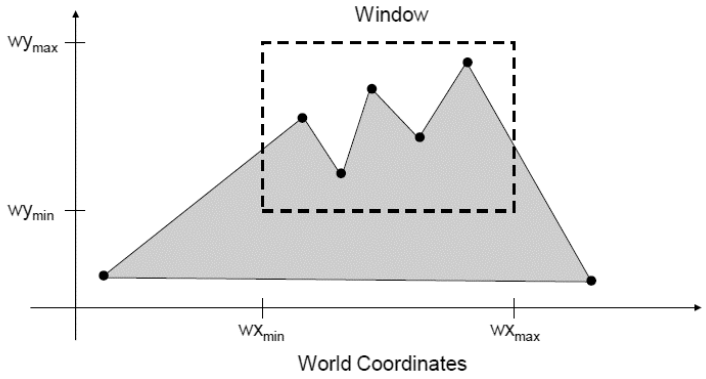
- Two Dimensional viewing is the formal mechanism for displaying views of a picture on an output device.
- Typically, a graphics package allows a user to specify which part of a defined picture is to be displayed and where that part is to be placed on the display device.
- Any convenient Cartesian coordinate system, referred to as the world-coordinate reference frame, can be used to define the picture.
- For a two-dimensional picture, a view is selected by specifying a subarea of the total picture area.
- The picture parts within the selected areas are then mapped onto specified areas of the device coordinates.
- Transformations from world to device coordinates involve translation, rotation, and scaling operations, as well as procedures for deleting those parts of the picture that are outside the limits of a selected display area.

Windowing Concept:

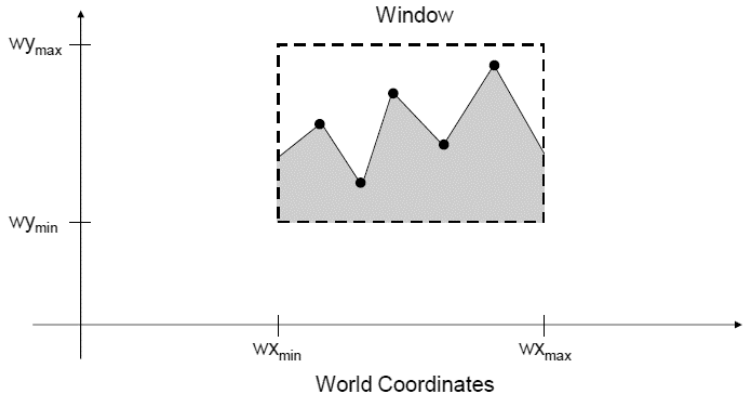
- A scene is made up of a collection of objects specified in world coordinates



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



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



Key Notes:

- A world-coordinate area selected for display is called a **window.**
- An area on a display device to which a window is mapped is called a **viewport**. In other words we can say that viewport is part of computer screen. It defines where to display.
- In many case window and viewport are rectangle, also other shape may be used as window and viewport.
- In general finding device coordinates of viewport from word coordinates of window is called as **viewing transformation.**
- The window defines what is to be viewed, the viewport defines where it is to be displayed.
- In computer graphics terminology, the term window originally referred to an area of a picture that is selected for viewing.
- Windows and viewport are rectangular in standard position, with the rectangular edge parallel to co-ordinate axis.

- The world window specifies which part of the world should be drawn. Whichever part lies inside the window should be drawn and whichever part lies outside should be clipped away and not drawn.
- OpenGL does the clipping automatically
- A mapping between world window and viewport is established by OpenGL
- OpenGL converts our coordinates to screen coordinates when we set up a screen window and viewport

Multiple Coordinates System

In a typical graphics program, we may need to deal with a number of different coordinate systems, and a good part of the work (and the cause of many headaches) is the conversion of coordinates from one system to another. Here is a list of some of the coordinate systems you may encounter.

1. Modelling coordinate/local/master coordinate (MC)

The shape of the individual object can be constructed in a scene within separate coordinate reference frame, is called Model Coordinate System.

It is used to define coordinates that are used to construct the shape of individual parts (objects) of a 2D scene.

2. World coordinate system (WC)

A Model Coordinate System is the unique coordinate space of the model. Two distinct models, each with their own coordinate systems can't interact with each other. There needs to be a universal coordinate system that allows each model to interact with each other. This universal system is called World Coordinate System. For interaction to occur, the coordinate system of each model is transformed into the World Coordinate System

Also known as the "universe" or sometimes "model" coordinate system. This is the base reference system for the overall model, (generally in 3D), to which all other model coordinates relate.

OR,

Once individual objects have been identified or specified, those objects are placed into appropriate position within scene using reference frame is called world coordinate. It contains many objects with one dimension.

3. Viewing coordinate system (VC)

Used to define window in the world coordinate plane with any possible orientation, viewing some object or item at a time
Viewing co-ordinate system are used to define particular view of a 2D scene. Translation, scaling, and rotation of the window will generate a different view of the scene. For a 2-D picture, a view is selected by specifying a sub area (window) of the total picture area.

4. Device coordinate (DC)

Device co-ordinate are used to define coordinates in an output device. They are integers within the range (0, 0) to (x_{max}, y_{max}) for a particular output device.

OR,

When the world coordinate description of the scene is transformed to one or more output device reference frame for display, these coordinate system are referred to a device coordinate or screen coordinate in the case of video monitor.

5. Device coordinate/normalize device coordinate (NDC)

NVC's are used to make the viewing process independent of the output device (monitor, mobile, hard copy devices). Normally, the value of NVC is 0 to 1.

OR,

The coordinate are in range 0 to 1.

Generally a graphical system, first convert, world coordinate position to normalize device coordinate before final conversion to specified device coordinated. This makes the system independent of the various devices that might be used at a particular workstation.

Window to Viewport Transformation

Window to Viewport Transformation is the process of transforming a 2D world-coordinate objects to device coordinates. Objects inside the world or clipping window are mapped to the viewport which is the area on the screen where world coordinates are mapped to be displayed.

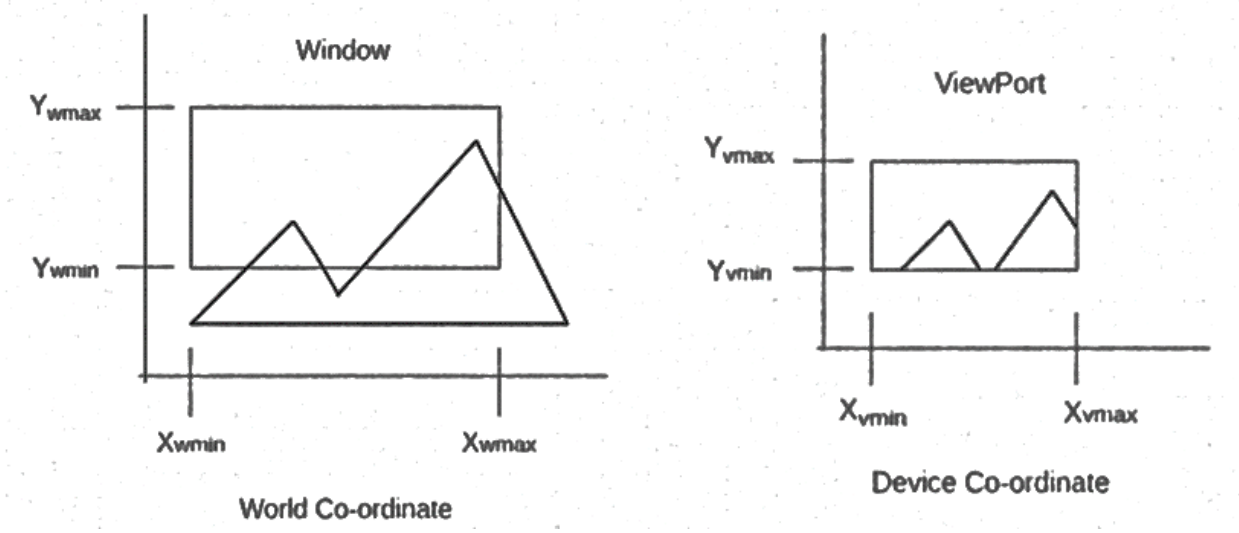


Figure: Mapping of picture section falling under rectangular Window onto a designated rectangular view port (i.e. viewing transformation)

It may be possible that the size of the Viewport is much smaller or greater than the Window. In these cases, we have to increase or decrease the size of the Window according to the Viewport.

Here,

World coordinate – It is the Cartesian coordinate w.r.t which we define the diagram, like X_{wmin}, X_{wmax}, Y_{wmin}, Y_{wmax}

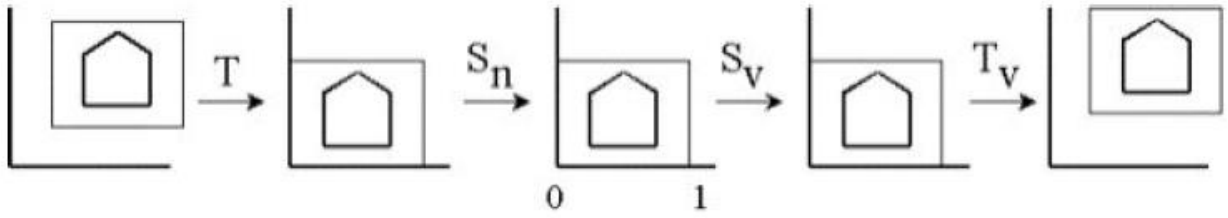
Device Coordinate –It is the screen coordinate where the objects is to be displayed, like X_{vmin}, X_{vmax}, Y_{vmin}, Y_{vmax}

Procedure for to transform a window to the view port we have to perform the following steps:

Step1: The object together with its window is translated until the lower left corner of the window is at the origin

Step2: The object and window are scaled until the window has the dimensions of the view port

Step3: Again translate to move the view port to its correct position on the screen (Setup Window,Translate window, Scale to normalize, Scale to view port, Translate to View port)



Application of windows to viewport transformation

i. By changing the position of the view port, we can view objects at different positions on the display area of an output device.
ii. ii. By varying the size of view ports, we can change size of displayed objects.
iii. iii. Zooming effects can be obtained by successively mapping different-sized windows on a fixed-sized view port
iv. iv. Panning effects (Horizontal Scrolling) are produced by moving a fixed-size window across the various objects in a scene.

Viewing Transformation pipeline

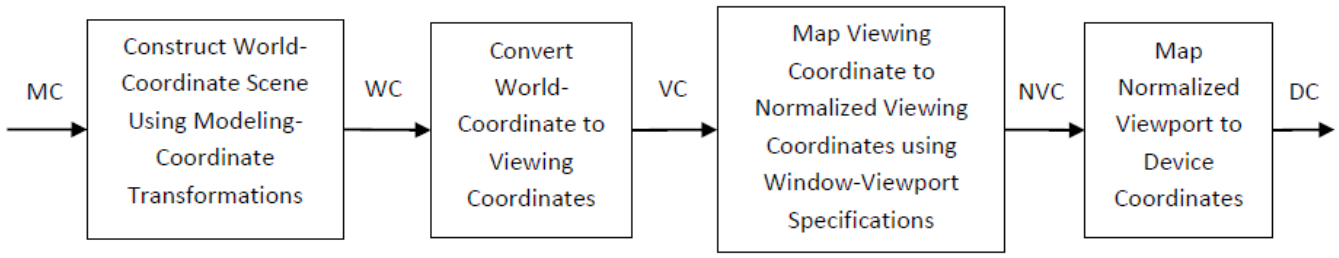


Figure: 2D viewing Pipeline

As shown in figure above first of all construct world coordinate scene using modeling coordinate transformation.

- ❖ After this, convert viewing coordinates from world coordinates using window to viewport transformation.
- ❖ Then map the viewing coordinate to normalized viewing coordinate in which we obtain values in between 0 to 1.

- At last, convert the normalized viewing coordinate to device coordinate using device driver software which provide device specification.
- Finally device coordinate is used to display image on display screen.
- By changing the viewport position on screen we can see image at different place on the screen.
- By changing the size of the window and viewport we can obtain zoom in and zoom out effect as per requirement.
- Fixed size viewport and small size window gives zoom in effect, and fixed size viewport and larger window gives zoom out effect.
- View ports are generally defines with the unit square so that graphics package are more device independent which we call as normalized viewing coordinate.

Window-To-Viewport Coordinate Transformation (mapping)

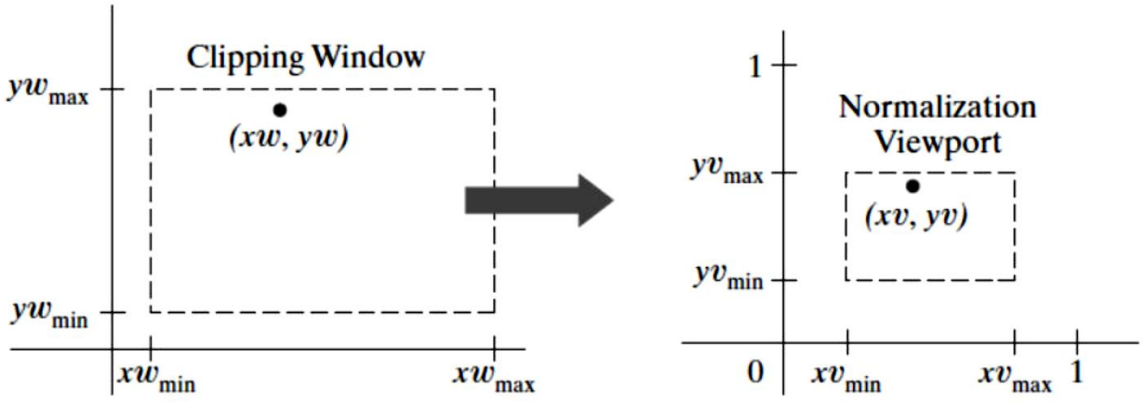


Figure: Windows to viewport mapping

- **A** point at position (x_w, y_w) in the designated window is mapped to viewport coordinate position (x_v, y_v) to maintain the same relative placement in the viewport as in the window
- A window can be specified by four world coordinates: x_{wmin}, x_{wmax}, Y_{wmin} and Y_{wmax}.
- Similarly a view port can be described by four device coordinates: x_{vmin}, x_{vmax}, Y_{vmin} and Y_{vmax}

To maintain the same relative placement in the viewport (or to calculate the point (x_V, y_V)) as in window the following proportional ratios must be equal.

Now, the relative position of the object in window and viewport are same

Then, for X-coordinate,

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

And, for Y- coordinate

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

Solving these or after calculating X and Y coordinate, we get.

$$x_v = x_{vmin} + (x_w - x_{wmin})s_x$$
$$y_v = y_{vmin} + (yw - y_{wmin})s_y$$

Where, s_x is scaling factor of x coordinate and s_y is scaling factor of y coordinate and determined by,

$$s_x = \frac{x_{vmax} - x_{vmin}}{x_{wmax} - x_{wmin}}$$
$$s_y = \frac{y_{vmax} - y_{vmin}}{y_{wmax} - y_{wmin}}$$

Workstation transformation.

- Number of display device can be used in application and for each we can use different window to viewport transformation. This mapping is called the **workstation transformation**
- As shown in following figure two different displays devices are used and we map different window-to-viewport on each one.

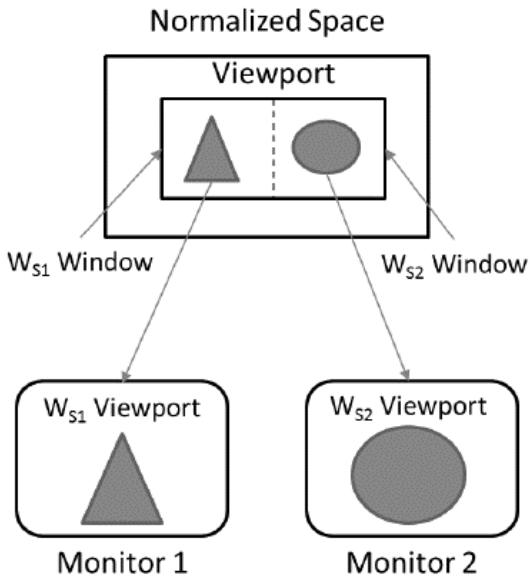


Figure: Workstation Transformation

Numerical

Problem-01: Window port is given by (100,100,300,300) and view port is given by (50, 50,150,150).Convert the window port coordinate (200,200) to the view port coordinate.

Solution:

$(x_{wmin}, y_{wmin}) = (100, 100)$

$(x_{wmax}, y_{wmax}) = (300, 300)$

$(x_{vmin}, y_{vmin}) = (50, 50)$

$(x_{vmax}, y_{vmax}) = (150, 150)$

$(x_w, y_w) = (200,200)$

$(x_v, y_v) = ?$

We know,

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

= (150- 50) / (300 - 100) = 0.5

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

= (150- 50) / (300 - 100) = 0.5

The equations for mapping window co-ordinate to view port coordinate is given by
$x_v = x_{vmin} + (x_w - x_{wmin})Sx$
$yv = y_{vmin} + (y_w - y_{wmin})Sy$

Hence,

$x_v = 50 + (200 - 100) * 0.5 = 100$
$y_v = 50 + (200 - 100) * 0.5 = 100$

The transformed view port coordinate is (100,100).

Problem-02

Given Coordinate for world coordinate representation are $\mathbf{X_{wmin} = 20}$ $\mathbf{X_{wmax} = 80,}$ $\mathbf{Y_{wmin} = 40,}$ $\mathbf{Y_{wmax} = 80}$ and coordinate of viewport are $\mathbf{X_{vmin} = 30,}$ $\mathbf{X_{vmax} = 60,}$ $\mathbf{Y_{vmin} = 40,}$ $\mathbf{Y_{vmax} = 60}$. A point (X_w, Y_w) be (30, 80) on the window. Calculate same point on viewport.

Solution:

Calculate scaling factor of x coordinate S_x and scaling factor of y coordinate S_y as:

S_x = (60 - 30) / (80 - 20) = 30 / 60
S_y = (60 - 40) / (80 - 40) = 20 / 40

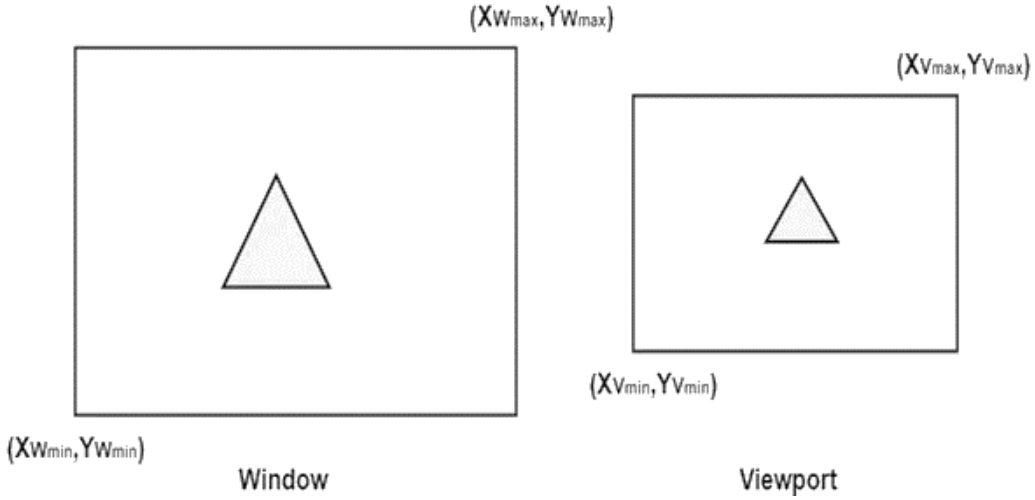
So, now calculate the point on viewport (X_v, Y_v).

X_v = 30 + (30 - 20) * (30 / 60) = 35

$Y_v = 40 + (80 - 40) * (20 / 40) = 60$

So, the point on window (X_w, Y_w) = (30, 80) will be **(X_v, Y_v) = (35, 60)** on viewport

Matrix Representation viewing Transformation



Step1: Translate window to origin

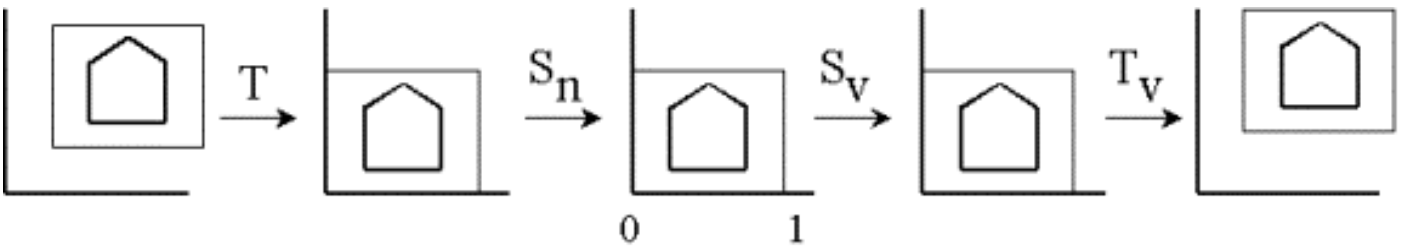
Step2: Scaling of the window to match its size to the viewport

$S_x = (Xv_{max} - Xv_{min}) / (Xw_{max} - Xw_{min})$

$S_y = (Yv_{max} - Yv_{min}) / (Yw_{max} - Yw_{min})$

Step3: Again translate viewport to its correct position on screen. (i.e. inverse of step 1)

Above three steps can be represented in matrix form:



View CM=T * S * T^{-1}

I.e. Use Transformation as:

1. Set up window
2. Translate window
3. Scale to normalize
4. Scale to viewport
5. Translate to Viewport

Clipping:

- Any Procedure that identifies those portions of a picture that are either inside or outside of a specified region of a space is referred to as a **Clipping algorithm** or simply **clipping**.
- The region against which an object is to clip is called a **Clip Window.**
- Clipping is necessary to remove those portions of the objects which are not necessary for further operations.
- Clipping excludes unwanted graphics from the screen.
- Depending on the application, the clip window can be a general polygon or it can even have curved boundaries. But Rectangular clip regions are preferred and standard.
- For the viewing transformation, we want to display only those picture parts that are within the window area. Everything outside the window is discarded.
- Clipping algorithm can be applied in World Coordinates, so that only the contents of the window interior are mapped to Device coordinates.
- So there are three cases
    1. The object may be completely outside the viewing area defined by the window port.
    2. The object may be seen partially in the window port.
    3. The object may be seen completely in the window port
- For case i and ii, clipping operation is necessary but not for case iii.

Application of Clipping

- Extracting part of a defined scene for viewing.
- Identifying visible surfaces in three-dimensional views.
- Creating objects using solid-modeling procedures.
- Displaying a multi window environment.
- Anti-aliasing line segments or object boundaries.

- Drawing and painting operations that allow parts of a picture to be selected for copying, moving erasing, or duplicating.

Types of Clipping (based on different output primitives)

- Point Clipping
- Line Clipping (straight-line segments)
- Area Clipping (polygons)
- Curve Clipping
- Text Clipping

Point Clipping:

- In point clipping we eliminate those points which are outside the clipping window and draw points which are inside the clipping window.
- Let's consider clipping window is rectangular boundary with edge (x_{wmin},x_{wmax},y_{wmin},y_{wmax}).
- So for finding whether given point is inside or outside the clipping window we use following inequality:

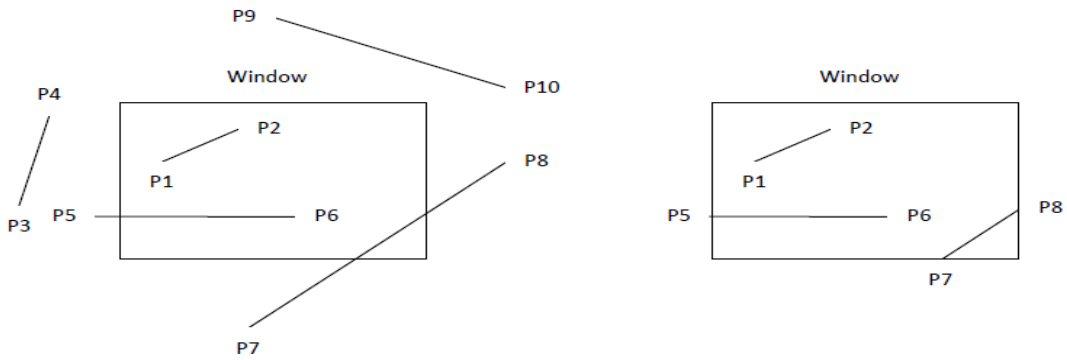
$$xwmin \leq x \leq xwamx$$
$$ywmin \leq y \leq ywamx$$

- If above both inequality is satisfied then the point is inside otherwise the point is outside the clipping window.

Line Clipping

Line clipping involves several possible cases.

1. Completely inside the clipping window.
2. Completely outside the clipping window.
3. partially inside and partially outside the clipping window.

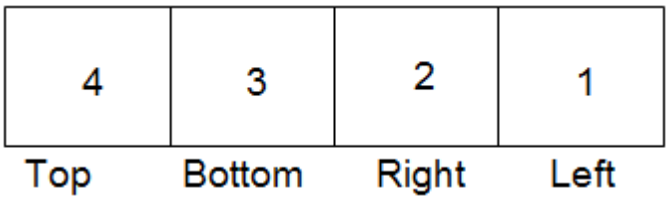


- Line which is completely inside is display completely. Line which is completely outside is eliminated from display. And for partially inside line we need to calculate intersection with window boundary and find which part is inside the clipping boundary and which part is eliminated.

Some of clipping procedure are discuss below.

Cohen-Sutherland Line Clipping

- One of the oldest and most popular line-clipping procedure.
- Danny Cohen and Ivan Sutherland
- Can be used only on a rectangular window
- World space is divide 9 different regions based on the windows boundaries.
- Middle region is the clipping window.
- Each region is assigned a unique 4-bit code, called **region code.**
- Region codes indicate the position of the regions with respect to the window



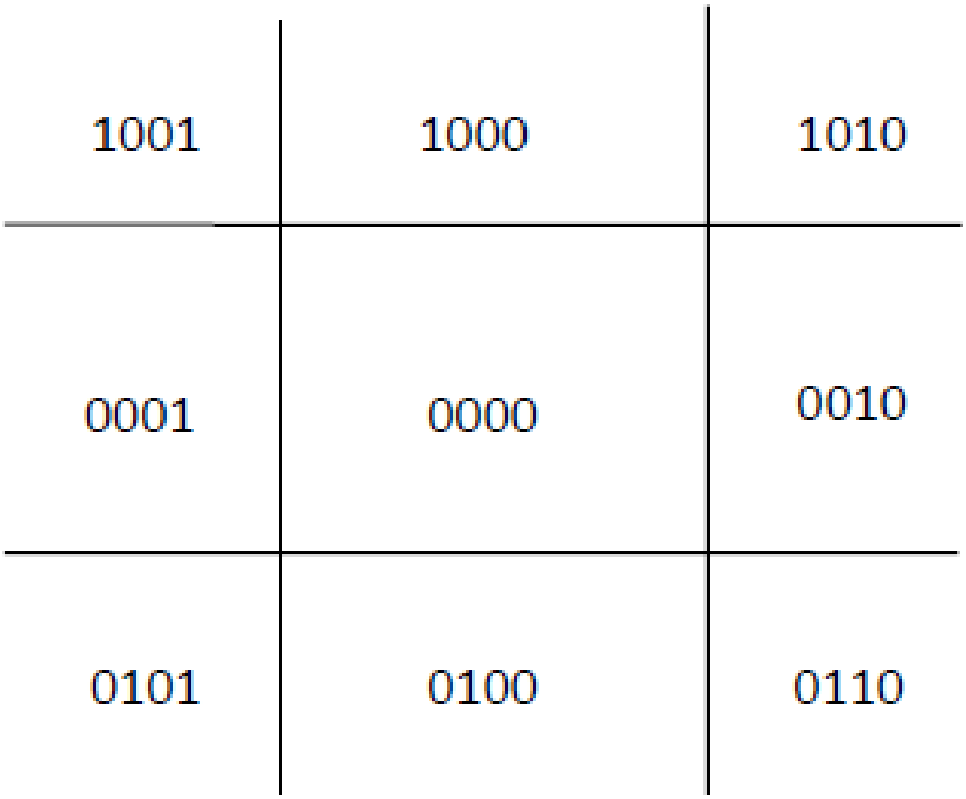
TBRL

- Any point inside the clipping window has a region code 0000.

The rightmost bit is the first bit. The bits are set to 1 based on the following scheme.

1) First bit Set- If the end point is to the left of the window.
2) Second bit Set- If the end point is to the right of the window.
3) Third bit Set- If the end point is to the below the window.
4) Fourth bit Set- If the end point is to the above the window.

Otherwise the bit is set to Zero.

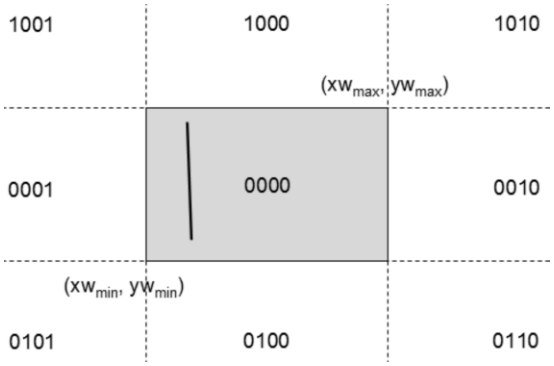


- It is obvious that, if both end point codes are zero, then both ends of the line lie inside the window & the line is visible.

Algorithm:

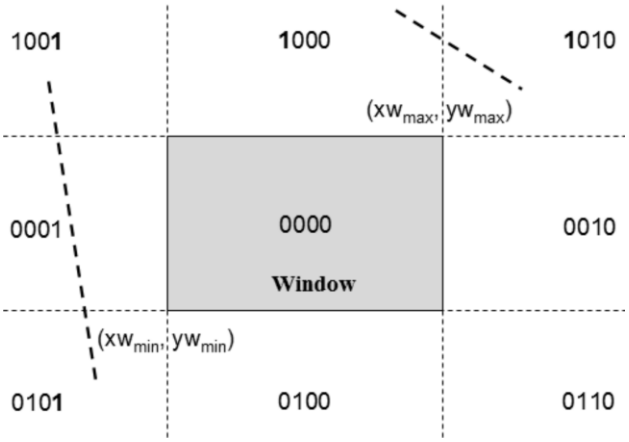
1. Given a line segment with end points $P_1=(x_1,y_1)$ and $P_2(x_2,y_2)$, compute 4 bit region code for each end point.
2. To clip a line, first we have to find out which regions its two endpoints lie in.

Case I:
If both end point code is 0000, then the line segment is completely inside the window. I.e. if bitwise (logical) OR of the codes yields 0000, then the line segment is accepted for display.



Case II:

If the two end point region code both have a 1 in the same bit position, then the line segment lies completely outside the window, so discarded. I.e. if bitwise (logical) AND of the codes not equal to 0000, then the line segment is rejected.



Case III:

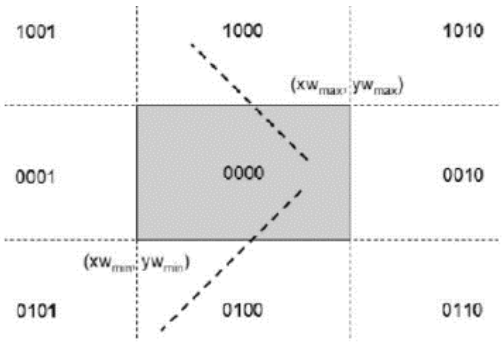
If lines cannot be identified as completely inside or outside we have to do some more calculations.

Here we find the intersection points with a clipping boundary using the slope intercept form of the line equation

Here, to find the visible surface, the intersection points on the boundary of window can be determined as:

$y - y_1 = m\,(x - x_1)$

Where, $m = (y_2 - y_1) / (x_2 - x_1)$



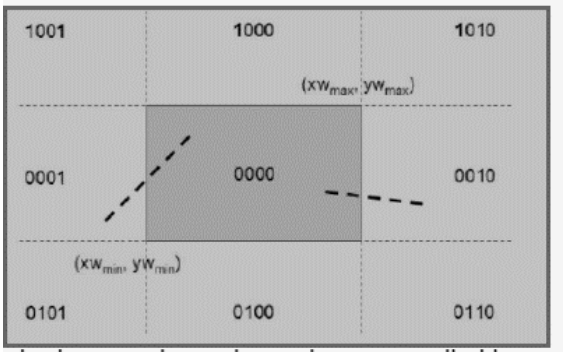
i. If the intersection is with horizontal boundary: $x = x_1 + \frac{y - y_1}{m}$

Where y is set to either y_{wmin} or y_{wmax}.

ii. If the intersection is with vertical boundary :

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

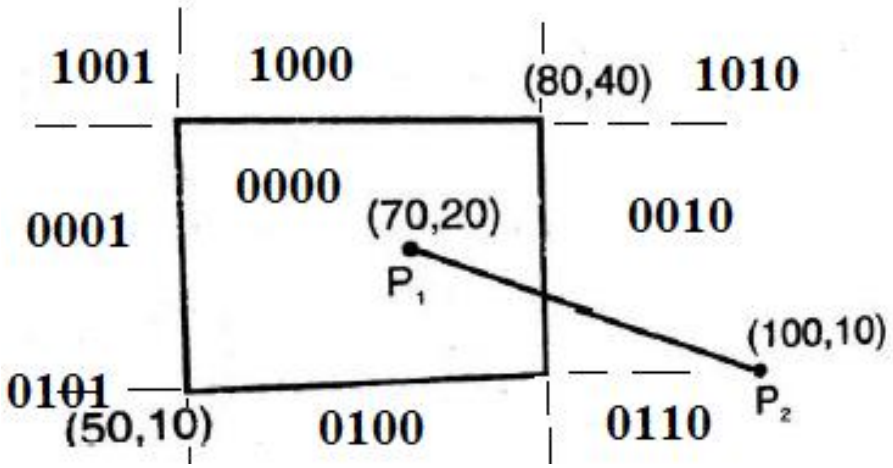
Where x is set to either x_{wmin} or x_{wmax}



3. Assign a new four-bit code to the intersection point and repeat until either cas1 or case2 are satisfied.

Example: Use the Cohen –Sutherland algorithm to clip the line P_1 (70, 20) and P_2 (100, 10) against a window lower left hand corner (50, 10) and upper right hand corner (80, 40)**.**

Solution:



Step-1: Assign 4 bit binary code to the two end point

$P_1 = 0000$ $P_2 = 0010$

Step-2: Calculate bitwise **OR:**

P_1 **|** $P_2 = 0000$ **|** $0010 = 0010$

Since P_1 **|** P_2! = 0000, hence the two point doesn't lies completely inside the window.

Step-3: Finding bitwise **AND**:

P_1 & $P_2 = 0000$ & $0010 = 0000$

Since P_1 & $P_2 = 0000$, hence line is partially visible.

Step- 3.1: Now, to find the intersection of P_1 and P_2 with the boundary of Window,

We have,

$P_1(x_1, y_1) = (70, 20)$
$P_2(x_2, y_2) = (100, 10)$

Slope m = (10-20) / (100-70) = -1/3

We have to find the intersection with right edge of window,

Here, x=80
y=?

We have

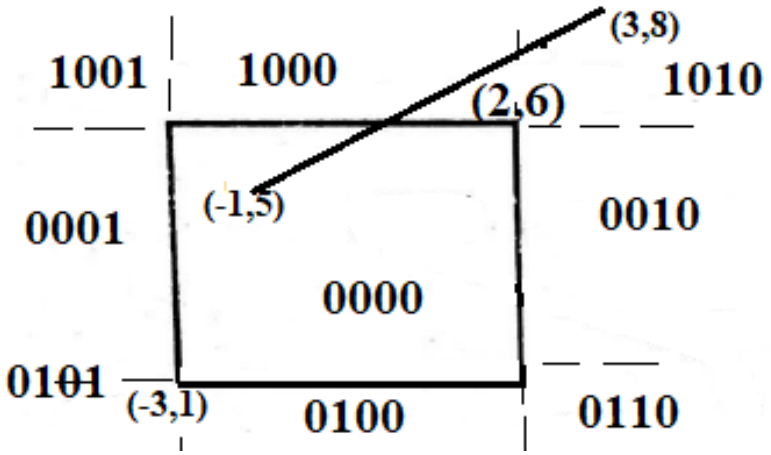
$y = y_2 + m(x-x_2)$
$= 10 + (-1/3)(80-100)$
$= 10 + 6.67$
$= 16.67$

Thus, the intersection point P3 = (80, 16.66), so discarding the line segment that lie outside the boundary i.e. P_3P_2, we get new line P_1P_3 with co-ordinate P_1 (70, 20) and P_3 (80, 16.67)

Problem: clip a line M (-1, 5) and P (3, 8) using Cohen Sutherland line clipping algorithm having window coordinates (-3, 1) and (2, 6). (Classwork)

Answer: point of intersection (1/3, 6) hence (-1, 5) to (1/3, 6) visible

Solution:



From figure,

Region code for two endpoint are $p_1 = 0000$ and $p_2 = 1010$

Step-1: Calculate logical OR

$P_1 \mid p_2 = 0000 \mid 1010 = 1010$
$P_1 \mid p_2 ! = 0000$ therefore, calculate logical AND

Step-2: calculate P_1 & p_2

$P_1 \& p_2 = 0000 \& 1010 = 0000$ i.e. line is partially visible

Step-3: given line intersects in horizontal boundary therefore,

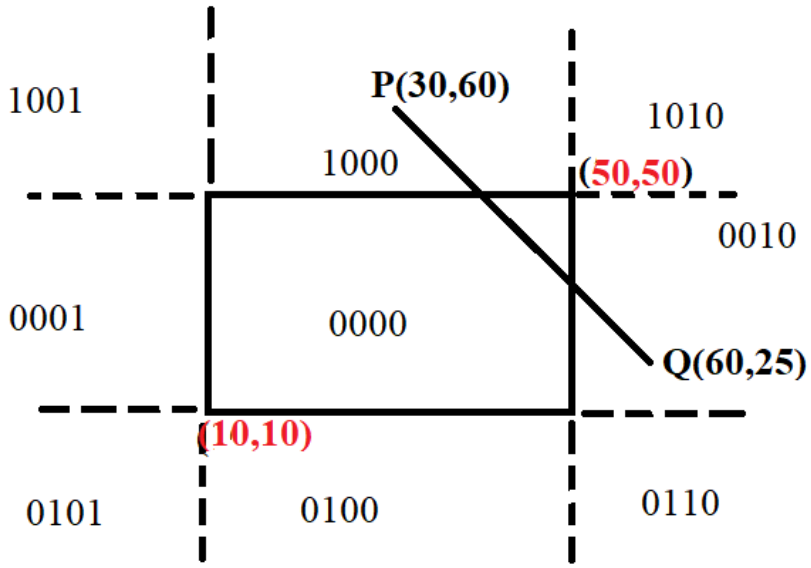
$X = x_1 + (y-y_1) / m$ where, $m = (8-5) (/ 3 +1) = ¾$

Now, $x = (-1) + (6-5) / (3/4) = 1 / 3$

Thus, the intersection point P3 = (1/3, 6) so discarding the line segment that lie outside the boundary i.e. P_3P_2, we get new line P_1P_3 with co-ordinate P_1 (-1, 5) and P_3 (1/3, 6)

Problem: Size of windows port is defined by (10, 10) and (50, 50). Apply the Cohen-Sutherland line clipping algorithm to clip a line (30, 60) to (60, 25). (*Homework)*

Solution:



From figure,

Region code for two endpoint are P = 1000 and Q = 0010

Step-1: Calculate logical OR

P | Q = 1000 | 0010 = 1010
P | Q ! = 0000 therefore, calculate logical AND

Step-2: calculate P & Q

P & Q= 1000 & 0010 = 0000 i.e. line is partially visible

Here, Line intersects on both boundary (i.e. Horizontal and Vertical). Therefore, calculate the value of both x and y.

Step-3: Given line intersects in horizontal boundary therefore,

$X= x_1 + (y-y_1) / m$ where, m= (25-60) / (60 -30) = - (7/6)

Now, $x = (-60) + (50-25) / (-7/6) = 38.57$

Also, given line intersects on vertical boundary too. Then,

$y = y_2 + m(x-x_2)$
$= 25 + (-7/6) (50-60)$
$= 36.67$

Thus, the intersection point P1 = (38.57, 50) and P2 = (50, 36.67) so discarding the line segment that lie outside the boundary i.e. P_1P_2, we get new line P_1P_2 with co-ordinate P_1 (38.57, 50) and P_2 (50, 36.67)

Polygon Clipping

- For polygon clipping we need to modify the line clipping procedure because in line clipping we need to consider about only line segment while in polygon clipping we need to consider the area and the new boundary of the polygon after clipping.

Sutherland-Hodgeman Polygon Clipping Algorithm

- Sutherland–Hodgeman algorithm is used for clipping polygons.
- A single polygon can actually be split into multiple polygons.
- The algorithm clips a polygon against all edges of the clipping region in turn.

Process:

- You can take any order but LEFT, RIGHT, BOTTOM, TOP is General.

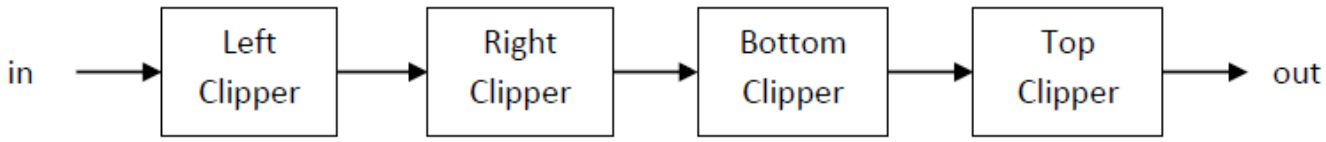


Figure: Processing the vertices of the polygon through boundary clipper

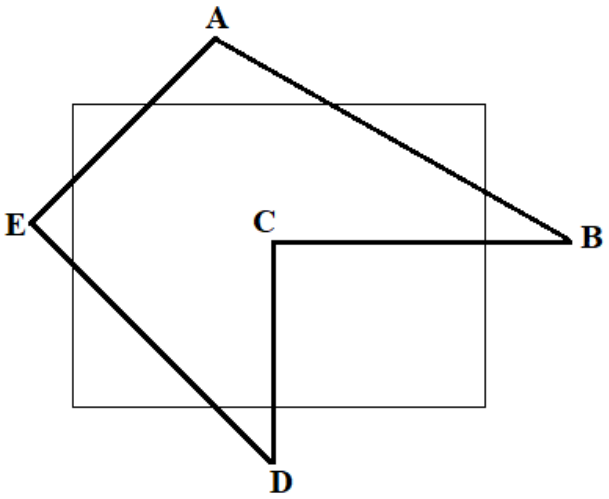
There are four possible cases for any given edge of given polygon against clipping edge.

Cases	Action	Output
1. **Both** Vertices are **inside**	Only the second vertex is added to the output list	Second Vertex
2. **First** vertex is **outside** while **second** vertex is **inside**	Both the point of intersection of the edge with the clip boundary and the second vertex are added to the output list	Second Vertex and Intersection vertex

3. **First** vertex is **inside** while **second** vertex is **outside**	Only the point of intersection of the edge with the clip boundary is added to the output list	Intersection point / Vertex
4. **Both** vertices are **outside**	No vertices are added to the output list	None

Example:

Clip the given polygon using Sutherland-Hodgeman polygon clipping algorithm



Let's proceed algorithm

Solution: on board