2D Viewing & Clipping

By
Poonam Saini
Dept. of Computer Science & Engineering
Sir Padampat Singhania University
Udaipur

2D Viewing & Clipping

Coordinate Systems

- **World Coordinate System (WCS):** Identifies locations of objects in the world in the application.
- **Model Coordinate System(MCS):** Identifies the shapes of object and it is attached to the object. Therefore the MCS moves with the object in the WCS.
- **Viewing Coordinate System (VCS):** Defined by the viewpoint and view site.

2D Viewing & Clipping

- 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 coordinate system is referred to as the world coordinate reference frame, can be used to define picture.
- For a 2D picture, a view is selected by specifying a subarea of the total picture area.
- A user can select a single area for display or several areas could be selected for simultaneous display or for an animation.

2D Viewing & Clipping

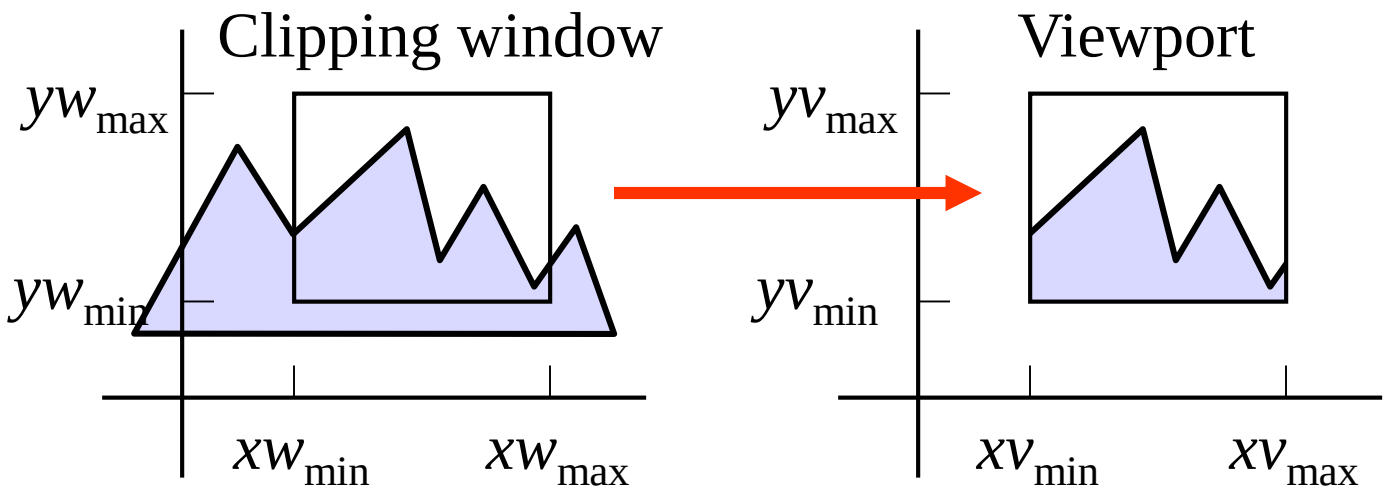
- Transformation from world to device coordinates involve translation, rotation and scaling operations as well as procedures for deleting those parts of the picture that are outside the limits of a selected display area.
- **WINDOW:** A world coordinate area selected for display is called a window.
- **VIEWPORT:** An area on a display device to which a window is mapped is called viewport.
- The window defines what is to be viewed and the viewport defines where is to be displayed.

Viewing Transformation

- The mapping of a part of a world-coordinate scene to a device coordinate is referred to as a viewing transformation or window-to-viewport transformation or the windowing transformation.

2D Viewing pipeline

World Coordinates: Device Coordinates:



Clipping window:

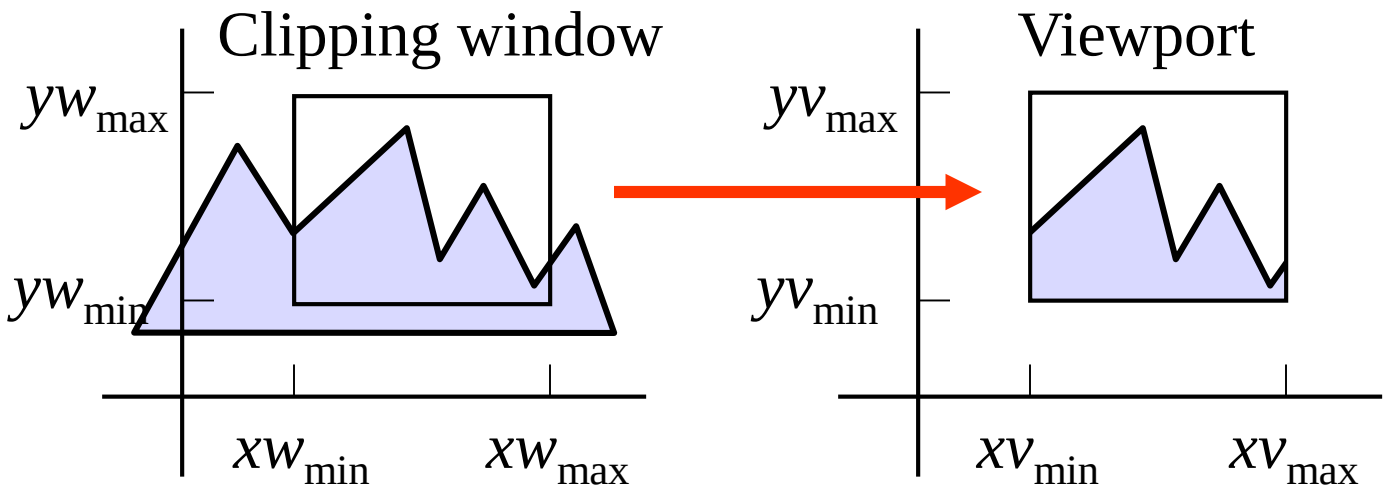
What do we want to see?

Viewport:

Where do we want to see it?

2D Viewing pipeline 2

World Coordinates: Device Coordinates:

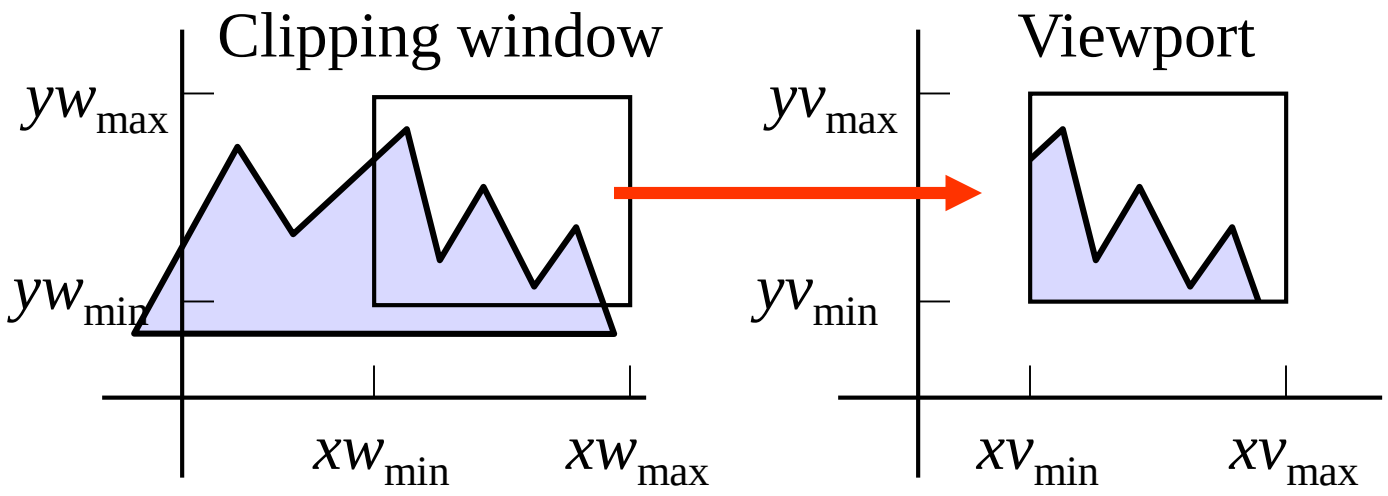


Clipping window:

Planning…

2D Viewing pipeline 2

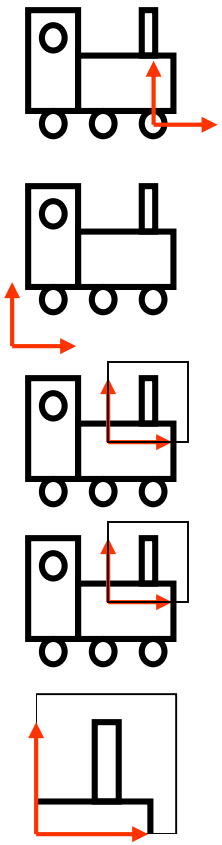
World Coordinates: Device Coordinates:



Clipping window:

Planning…

2D Viewing pipeline 4



MC: Modeling Coordinates

Apply model transformations

WC: World Coordinates

Determine visible parts

VC: Viewing Coordinates

To standard coordinates

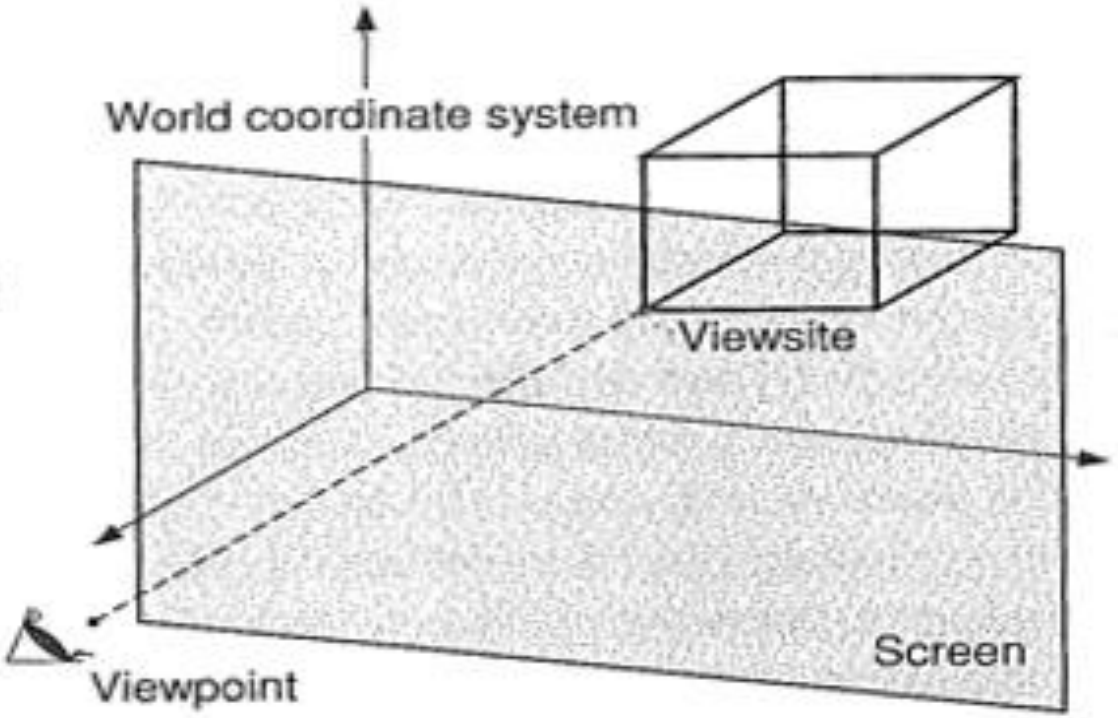
NC: Normalized Coordinates

Clip and determine pixels

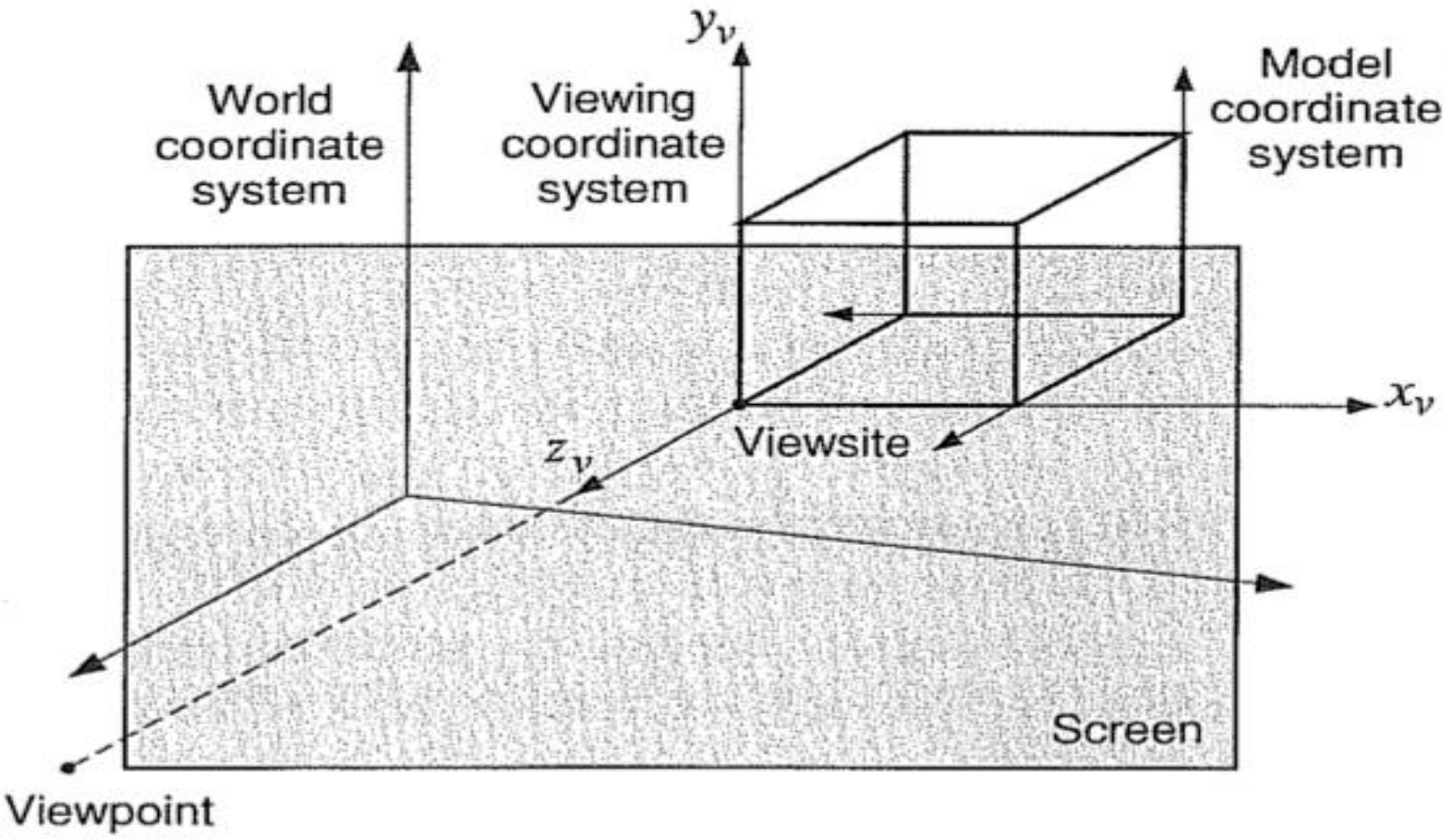
DC: Device Coordinates

Coordinate Systems, viewpoint and view site

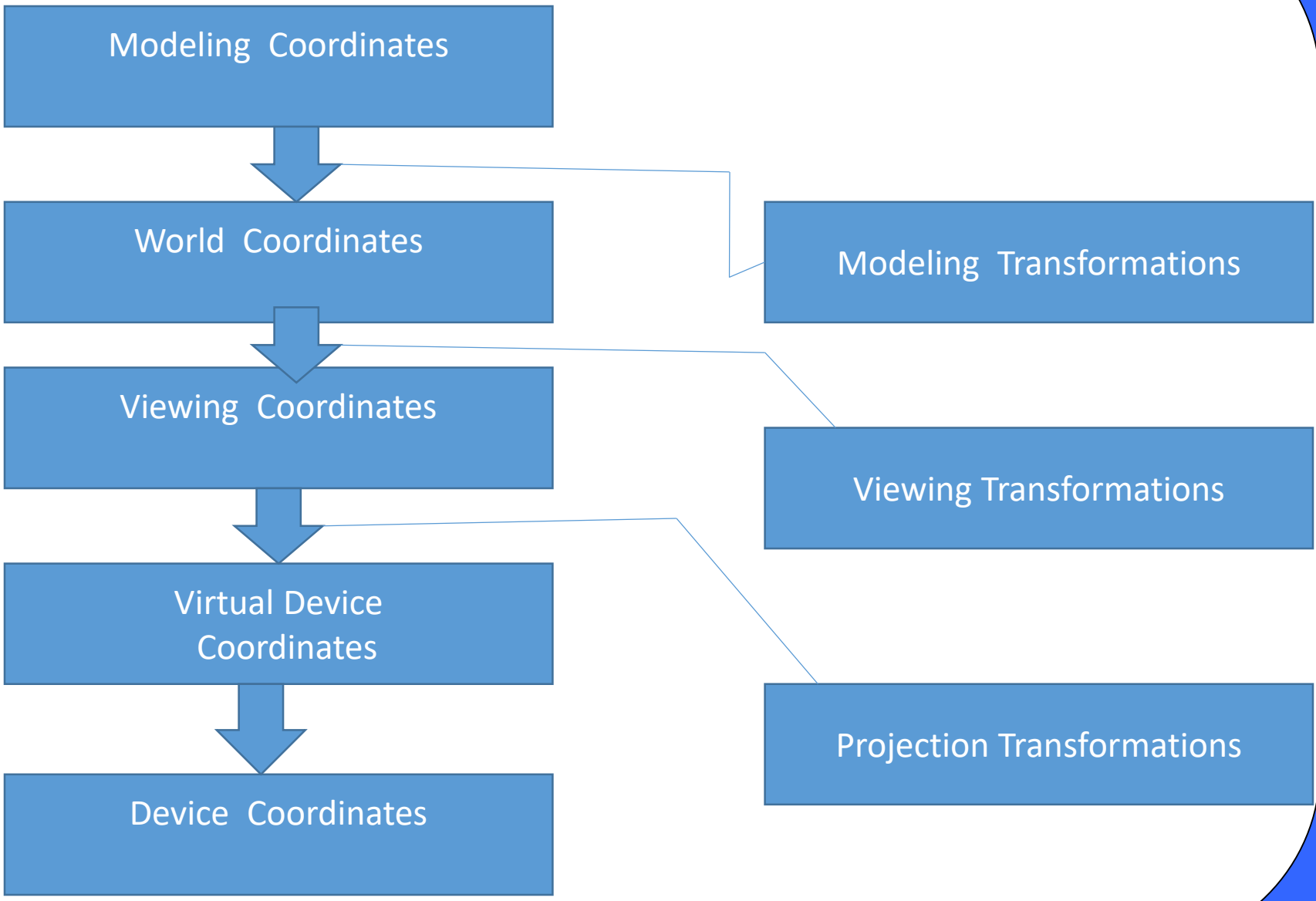
Viewpoint and viewsite



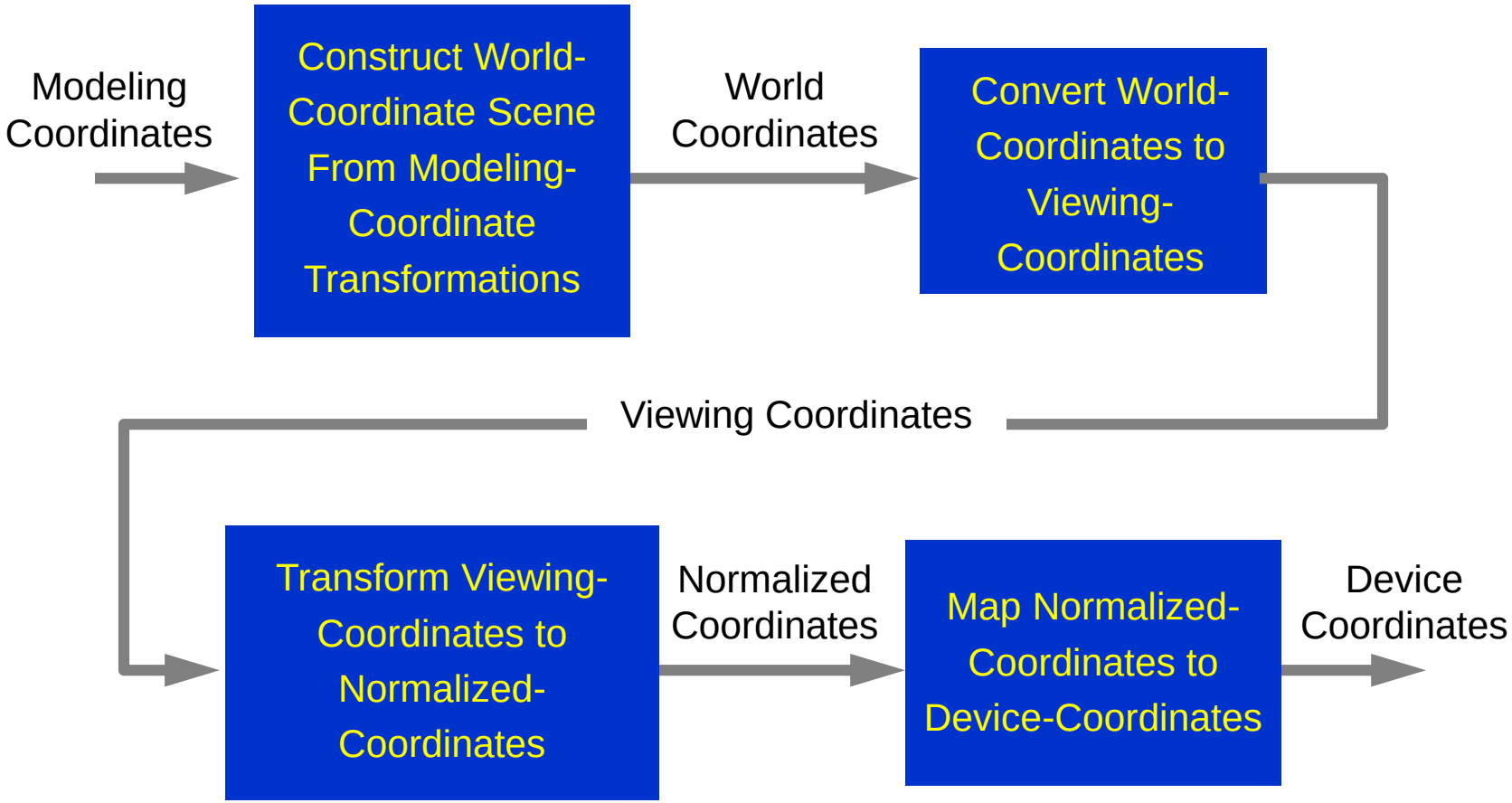
Coordinate Systems



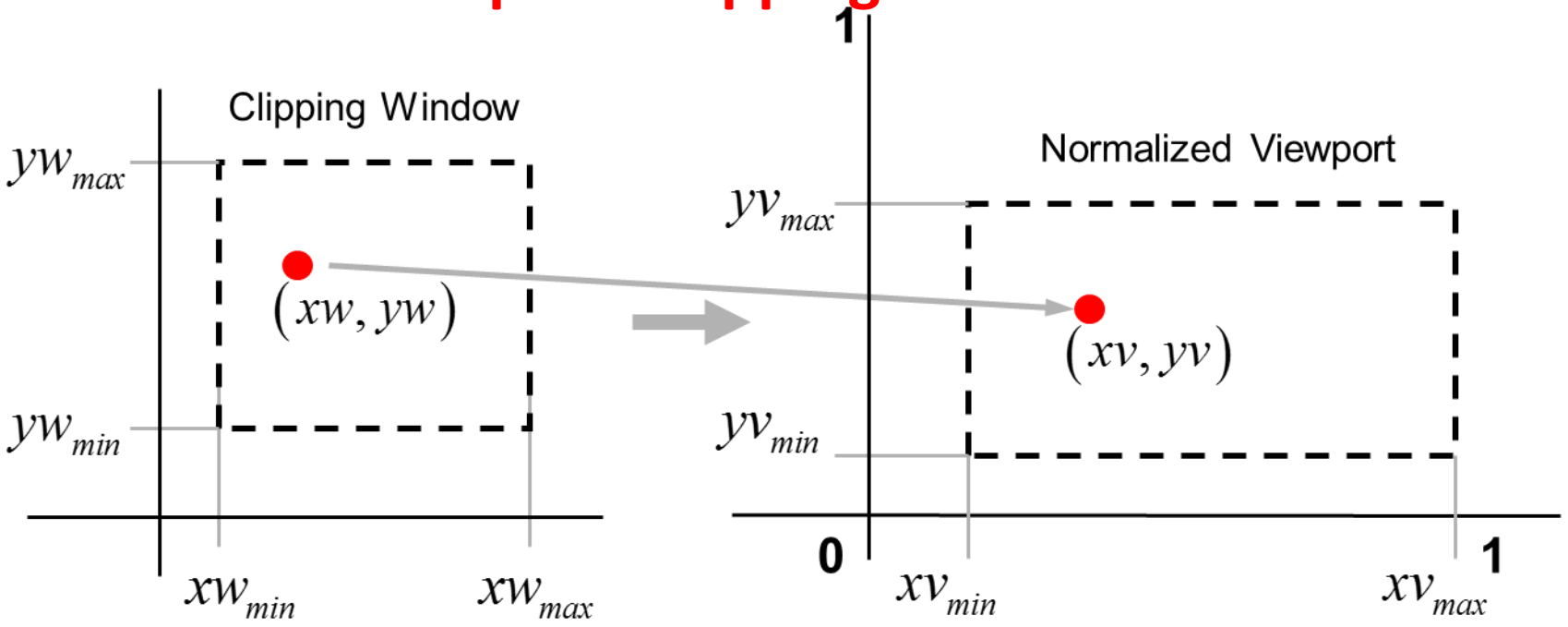
Transformations between Coordinate Systems



2D viewing transformation pipeline



Window to Viewport Mapping



Maintain relative size and position between clipping window and viewport.

$$\frac{xv - xv_{\min}}{xv_{\max} - xv_{\min}} = \frac{xw - xw_{\min}}{xw_{\max} - xw_{\min}} \qquad \frac{yv - yv_{\min}}{yv_{\max} - yv_{\min}} = \frac{yw - yw_{\min}}{yw_{\max} - yw_{\min}}$$

Solving for (xv, yv) obtains:

$$xv = s_x xw + t_x, \quad yv = s_y yw + t_y, \text{ where}$$

Scaling factors: $$s_x = \frac{xv_{\max} - xv_{\min}}{xw_{\max} - xw_{\min}}, \quad s_y = \frac{yv_{\max} - yv_{\min}}{yw_{\max} - yw_{\min}}$$

Translation factors:

$$t_x = \frac{xw_{\max} xv_{\min} - xw_{\min} xv_{\max}}{xw_{\max} - xw_{\min}}, \quad t_y = \frac{yw_{\max} yv_{\min} - yw_{\min} yv_{\max}}{yw_{\max} - yw_{\min}}$$

This can also be obtained by composing transformations:

$$\mathbf{M}_{\substack{\text{window,} \\ \text{norm\_viewport}}} =$$

$$\mathbf{T}(xv_{\min}, yv_{\min}) \cdot \mathbf{S}(s_x, s_y) \cdot \mathbf{T}(-xw_{\min}, -yw_{\min}) = \begin{bmatrix} s_x & 0 & t_x \\ 0 & s_y & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

World clipping window can first be mapped to normalized square between -1 and +1, where clipping algorithm takes place, and then transform the scene into viewport given in display coordinates.

$$\mathbf{M}_{\substack{\text{window,}\\ \text{norm\_square}}} = \begin{bmatrix} \dfrac{2}{xw_{\max} - xw_{\min}} & 0 & -\dfrac{xw_{\max} + xw_{\min}}{xw_{\max} - xw_{\min}} \\ 0 & \dfrac{2}{yw_{\max} - yw_{\min}} & -\dfrac{yw_{\max} + yw_{\min}}{yw_{\max} - yw_{\min}} \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{M}_{\substack{\text{norm\_square,}\\ \text{viewport}}} = \begin{bmatrix} (xv_{\max} - xv_{\min})/2 & 0 & (xv_{\max} + xv_{\min})/2 \\ 0 & (yv_{\max} - yv_{\min})/2 & (yv_{\max} + yv_{\min})/2 \\ 0 & 0 & 1 \end{bmatrix}$$

Clipping Operations

- The procedure that identifies those portions of the picture that are either inside or outside of a specified region of space is referred to as a clipping algorithm or clipping.
- CLIP WINDOW: The region against which an object is to be clipped is called a clip window.

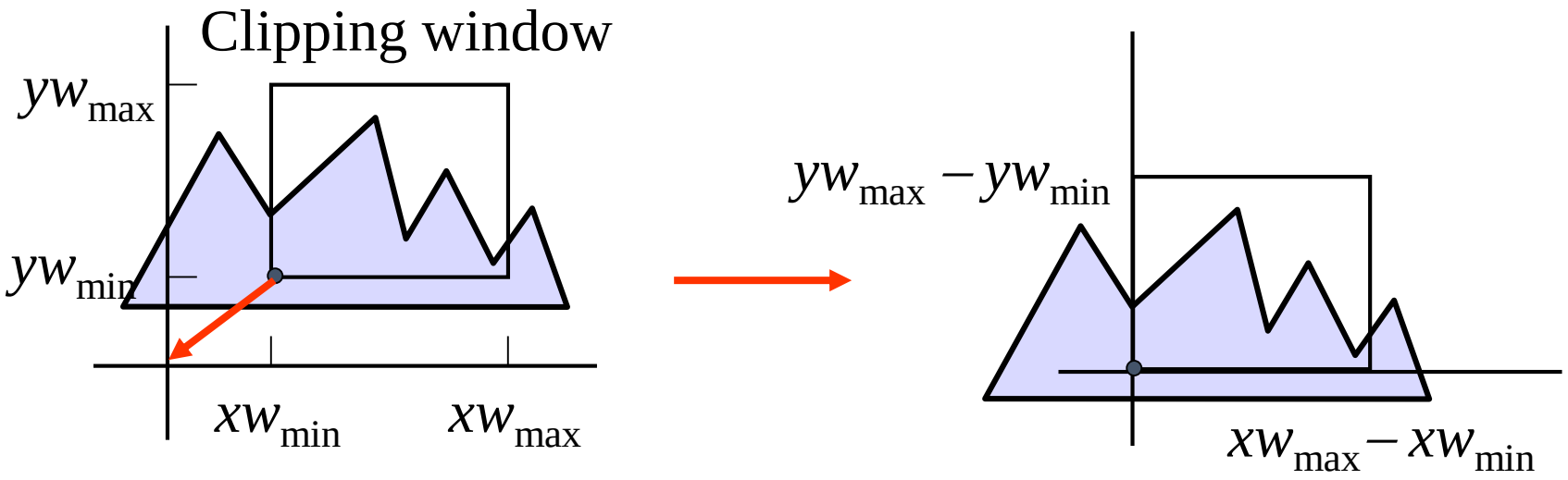
Applications of Clipping

- Extracting parts of a defined scene for viewing.
- Identifying visible surfaces in 3D views.
- Antialiasing line segments or object boundaries.
- Creating objects using solid modelling procedures.
- Displaying multiwindow environment.
- Drawing and painting operations that allows parts of a picture to be selected for copying, moving, erasing and duplicating.

Different Approaches for Clipping

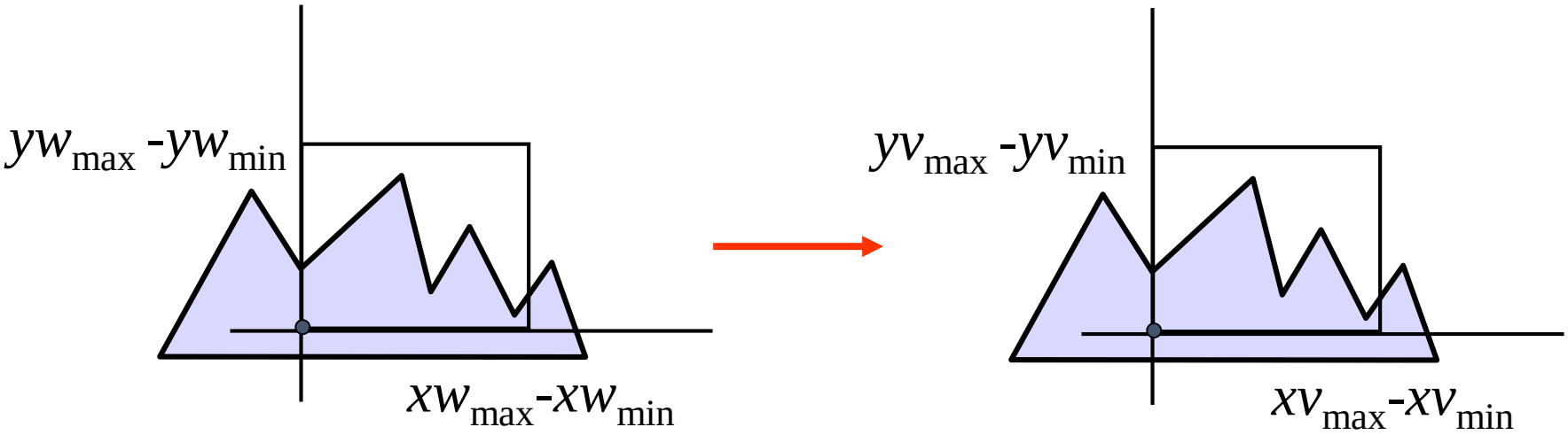
- Method 1: Clipping algorithms can be applied in World Coordinates(WC), so that only the content of the window interior are mapped to device coordinates.
- Method 2: Complete WC are mapped first to device coordinates or normalized device coordinates then clipped against the viewport boundaries.

Clipping window



- Clipping window usually an *axis-aligned* rectangle
- Sometimes rotation
- From world to view coordinates: $\mathbf{T}(-xw_{min}, -yw_{min})$ possibly followed by rotation
- More complex in 3D

To normalized coordinates 1



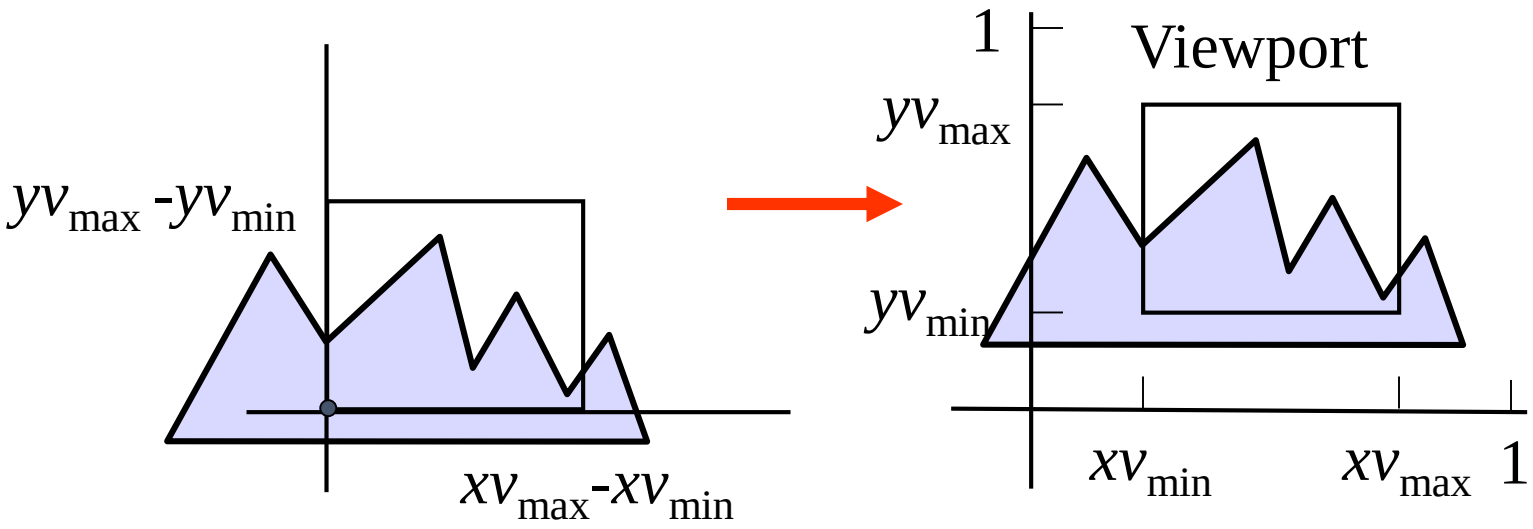
Scale with :

$$\mathbf{S}\left(\frac{xv_{max}-xv_{min}}{xw_{max}-xw_{min}},\frac{yv_{max}-yv_{min}}{yw_{max}-yw_{min}}\right)$$

If the two scale factors are unequal,

then the aspect-ratio changes : distortion!

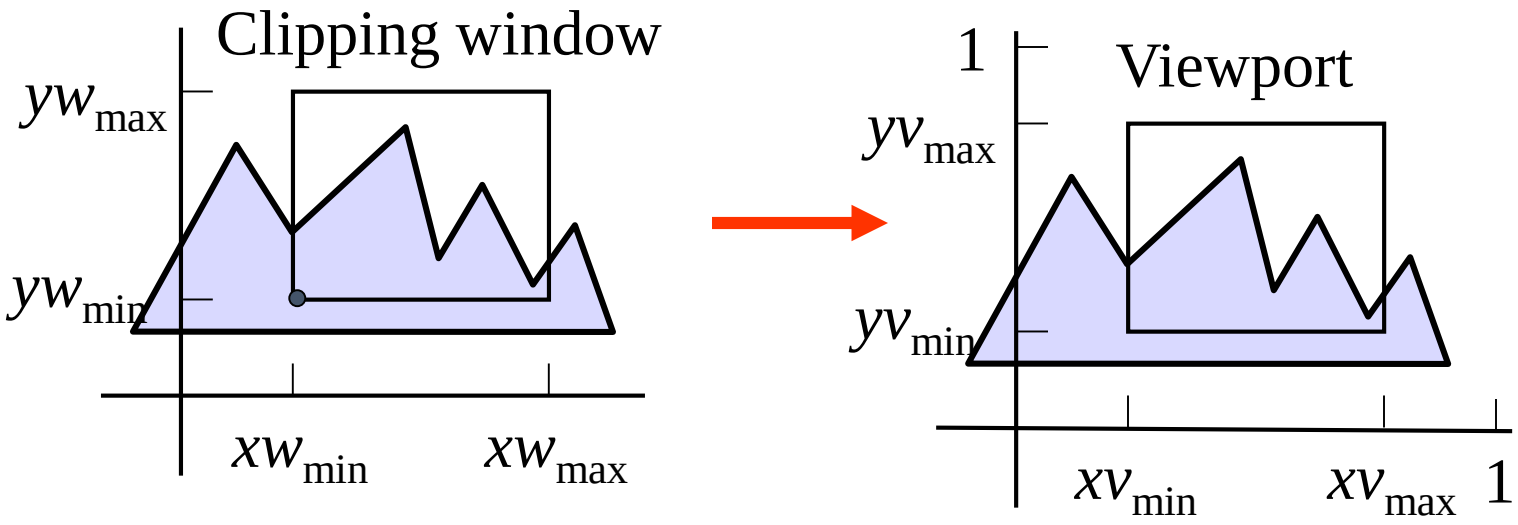
To normalized coordinates 2



Translate with :

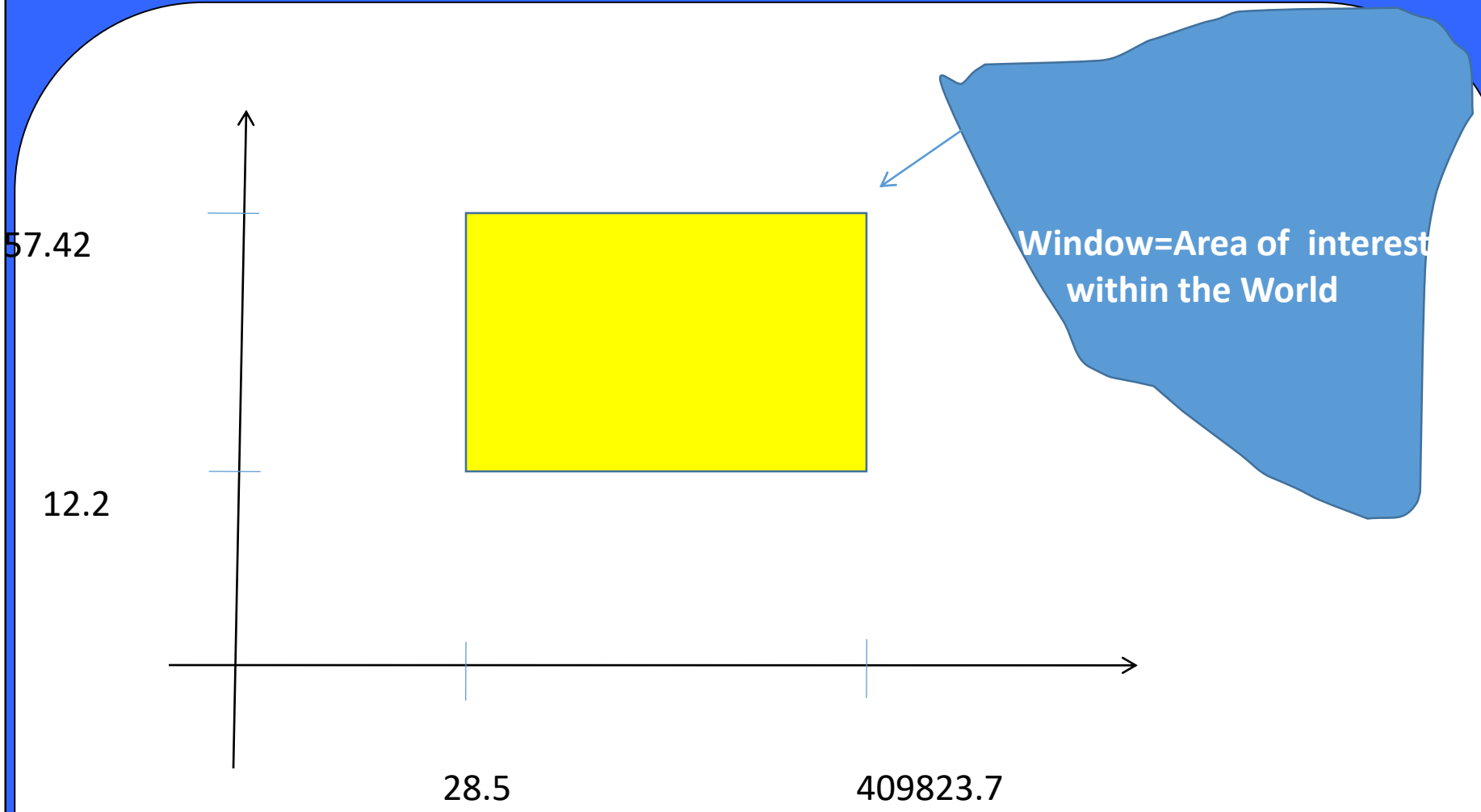
$$\mathbf{T}(xv_{min}, yv_{min})$$

To normalized coordinates 3



All together :

$$\mathbf{T}(xv_{min}, yv_{min})\mathbf{S}\left(\frac{xv_{max}-xv_{min}}{xw_{max}-xw_{min}}, \frac{yv_{max}-yv_{min}}{yw_{max}-yw_{min}}\right)\mathbf{T}(-xw_{min}, -yw_{min})$$



Assume window is rectangular

World coordinates are chosen at the convenience of the application or user

Example:

World coordinates

Window

(1,1)

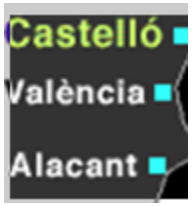
(0,0)

Normalized Device Coordinates

(0,1023)

(1,1)

.83,.9



.45,.32

Viewport

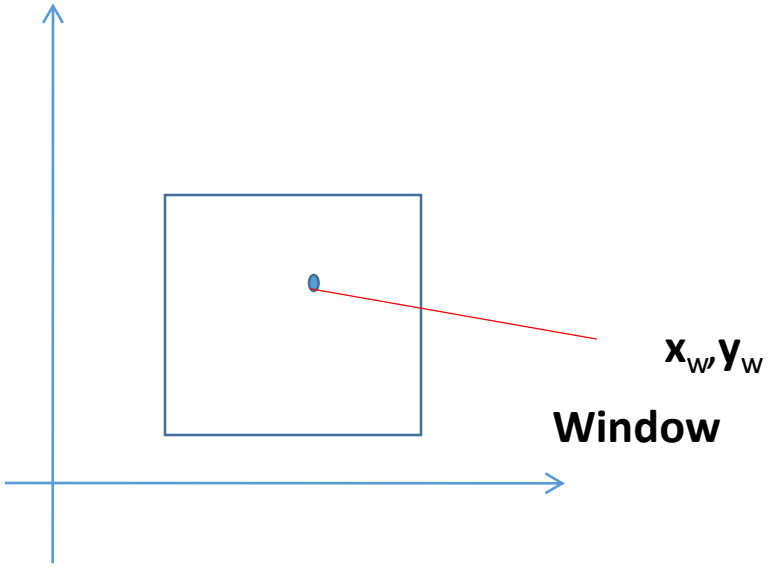
(0,0)

(0,0)

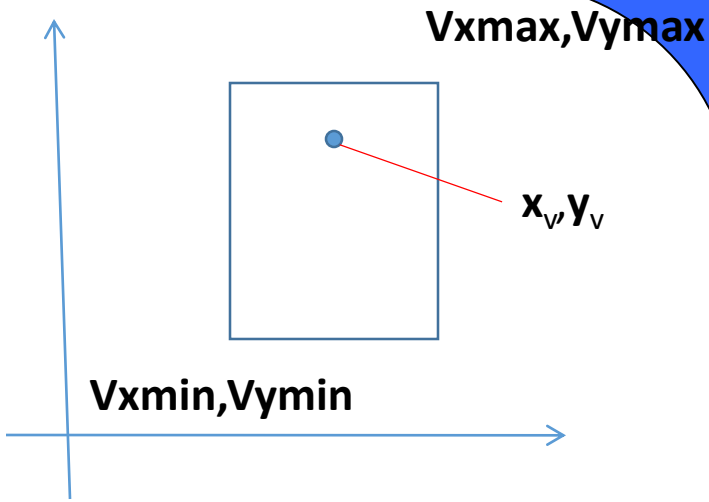
(1279,0)

Normalized Device Coordinates

Screen Coordinates



WORLD COORDINATES



NORMALIZED DEVICE COORDINATES

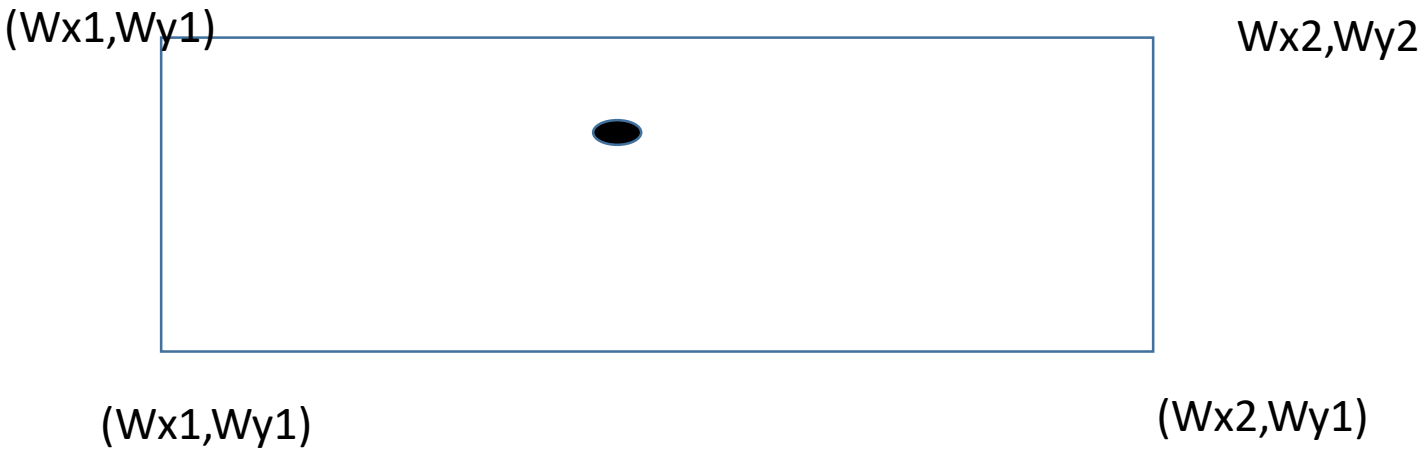
$$Xv = \frac{Vxmax - Vxmin}{Wxmax - Wxmin}(Xw - Wxmin) + Vxmin$$

$$Yv = \frac{Vymax - Vymin}{Wymax - Wymin}(Yw - Wymin) + Vymin$$

Types of clipping

- Point clipping
- Line clipping
- Area clipping
- Curve clipping
- Text clipping

Point Clipping



Assume, clip window is a rectangle in standard position, save appoint P(x,y) for display if the following inequalities are satisfied:

$$XWmin \leq x \leq XWmax$$

$$YWmin \leq y \leq YWmax$$

Point Clipping

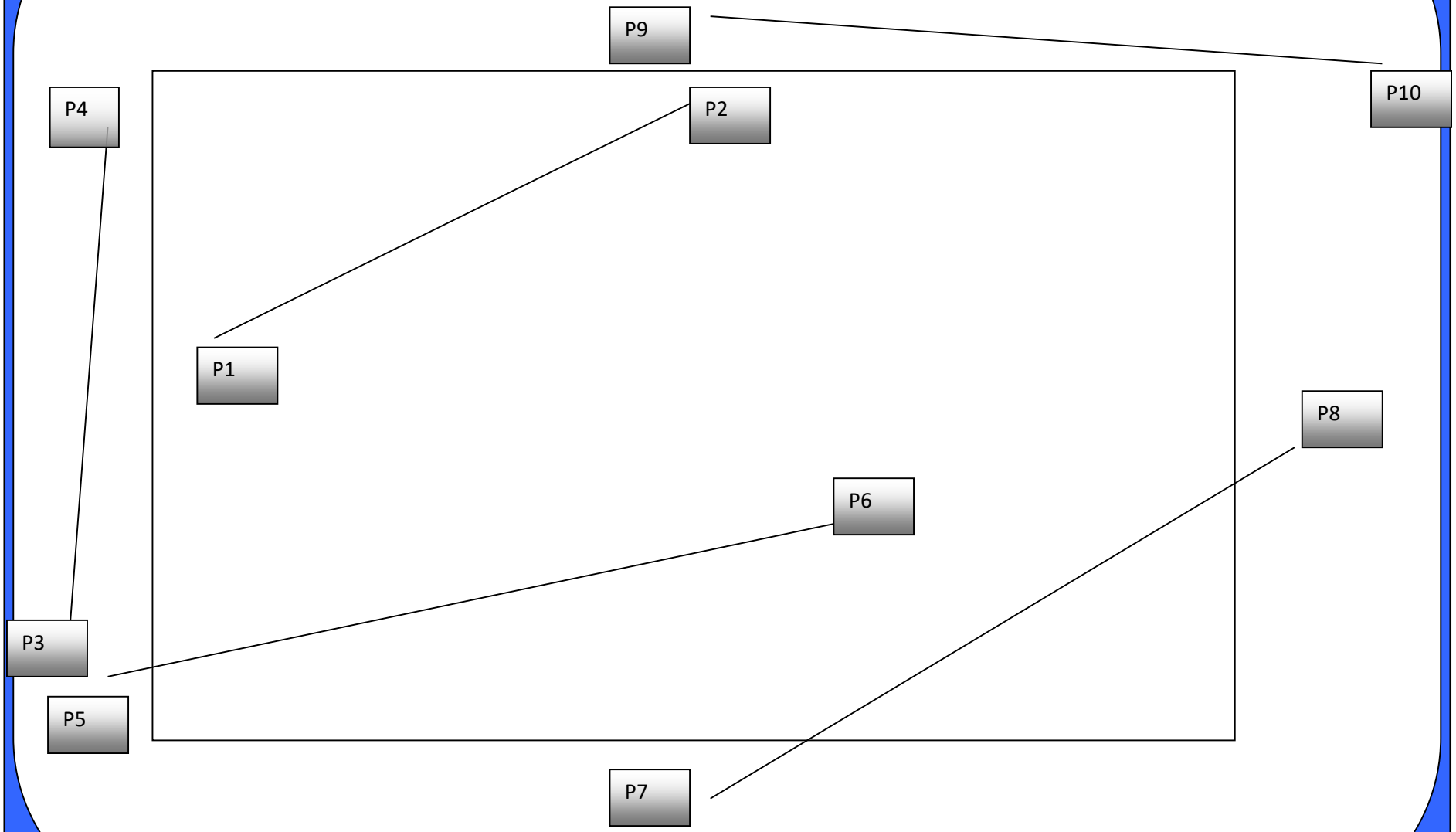
Where the edges of the clip window are:

$$(XWmin, XWmax, YWmin, YWmax)$$

And they can be either the world coordinate window boundaries or viewport boundaries.

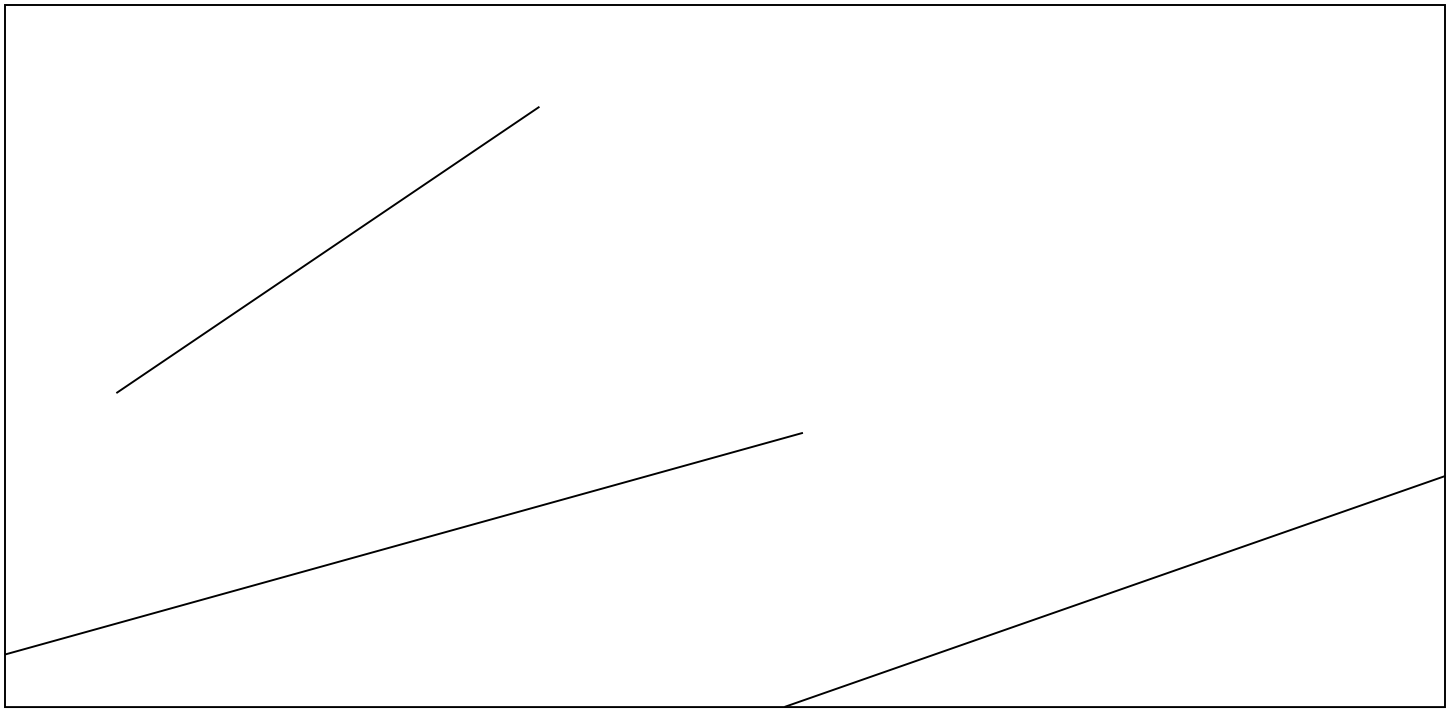
If one of the four inequality is not satisfied then the point is clipped or not saved for display

Line Clipping



Before Clipping

Line Clipping



After Clipping

Line Clipping Procedure

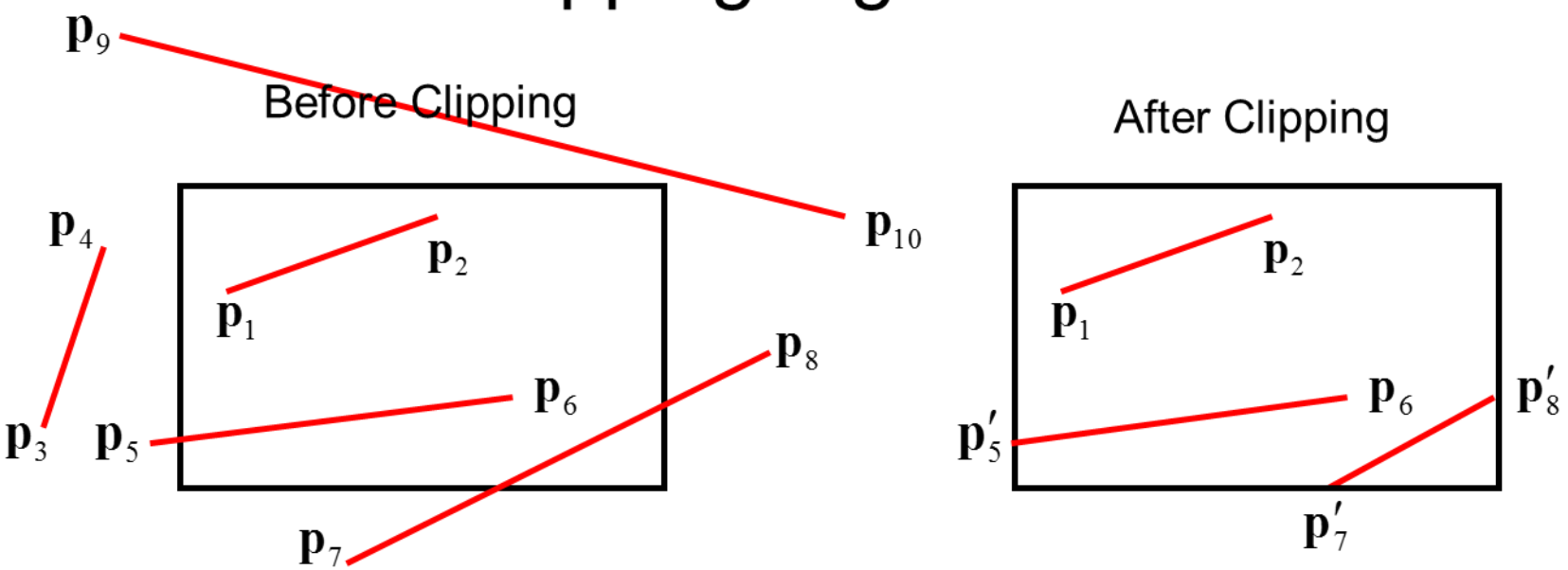
Lines are processed using inside-outside tests by checking the line endpoints.

Case 1: A line with both endpoints inside the clipping boundaries is saved i.e. visible.

Case 2: A line with both end points outside the clipping boundaries is outside the window i.e. not visible.

Case 3: The lines cross one or more clipping boundaries. Therefore, multiple intersection points are to be calculated.

Clipping Algorithms



Parametric equations of line segment from (x_0, y_0) to $(x_{\text{end}}, y_{\text{end}})$

$$x = x_0 + u(x_{\text{end}} - x_0), \quad y = y_0 + u(y_{\text{end}} - y_0), \quad 0 \le u \le 1.$$

Used to determine the parts contained in clipping window.

Cohen-Sutherland Line Clipping Algorithm

Divide the line clipping process into two phases:

PHASE 1: Identify those lines which intersect the clipping window and so need to be clipped.

PHASE 2: Perform the clipping.

Cohen-Sutherland Line Clipping Algorithm

PHASE 1: Procedure for finding the category of the line.

1. Assign a 4-bit region code to each end point of the line.

(a) Each bit position in the region code is used to indicate one of the four relative coordinate positions w.r.t. the clip window.

bit 1: Left
bit 2: right
bit 3: below
bit 4: above

Cohen-Sutherland Line Clipping Algorithm

Procedure for finding the category of the line.

(b) A value of one for any bit position indicates that the point is in the relative position, otherwise the bit is set to 0.

(c) If a point is inside the clipping window then the region code will be 0000.

Cohen-Sutherland Line Clipping Algorithm

PHASE 1: Procedure for finding the category of the line.

(d) Calculating bit values in the region code:

Compare the endpoint coordinate values(x,y) to the clip boundaries.

Bit 1=1 if $x<xw_{min}$

Bit 2=1 if $x>xw_{max}$

Bit 3=1 if $y<yw_{min}$

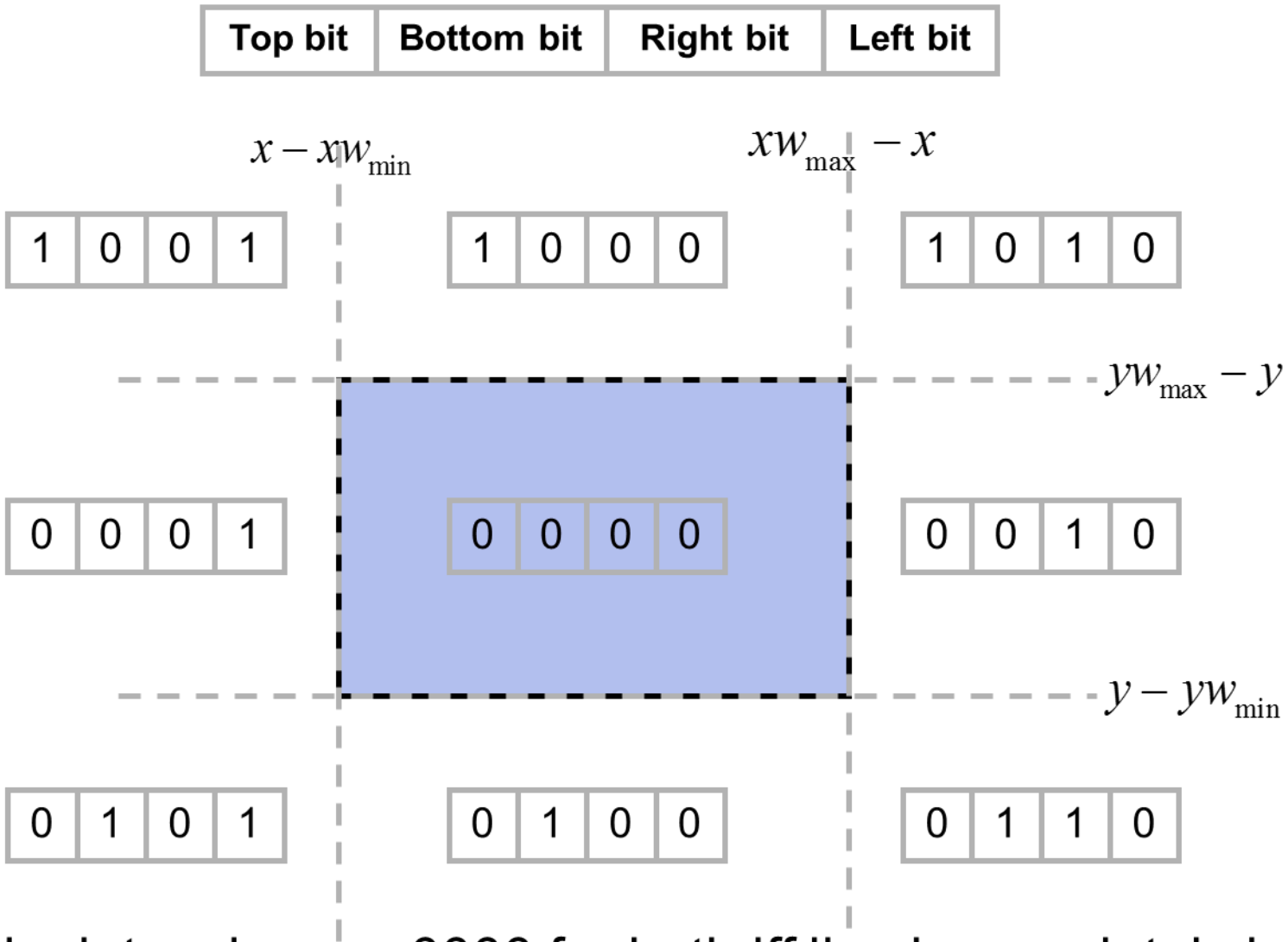
Bit 4=1 if $y>yw_{max}$

Cohen-Sutherland Line Clipping Algorithm

- Intersection calculations are expensive. Find first lines completely inside or certainly outside clipping window. Apply intersection only to undecided lines.
- Perform cheaper tests before proceeding to expensive intersection calculations.

Cohen-Sutherland Line Clipping Algorithm

- Assign code to every endpoint of line segment.
  - Borderlines of clipping window divide the plane into two halves.
  - A point can be characterized by a 4-bit code according to its location in half planes.
  - Location bit is 0 if the point is in the positive half plane, 1 otherwise.
  - Code assignment involves comparisons or subtractions.
- Completely inside / certainly outside tests involve only logic operations of bits.



Endpoint codes are 0000 for both iff line is completely inside.

If endpoint codes has 1 in same bit, line is certainly outside.

Lines that cannot be decided are intersected with window border lines.

Each test clips the line and the remaining is tested again for full inclusion or certain exclusion, until remaining is either empty or fully contained.

Endpoints of lines are examined against left, right, bottom and top borders (can be any order).

For a line in Category 3, find the intersection point of the line with one of the boundaries of the clipping window or with the infinite extension of one of the boundaries.