Lecture 6: 2D Viewing

Shobhanjana Kalita,

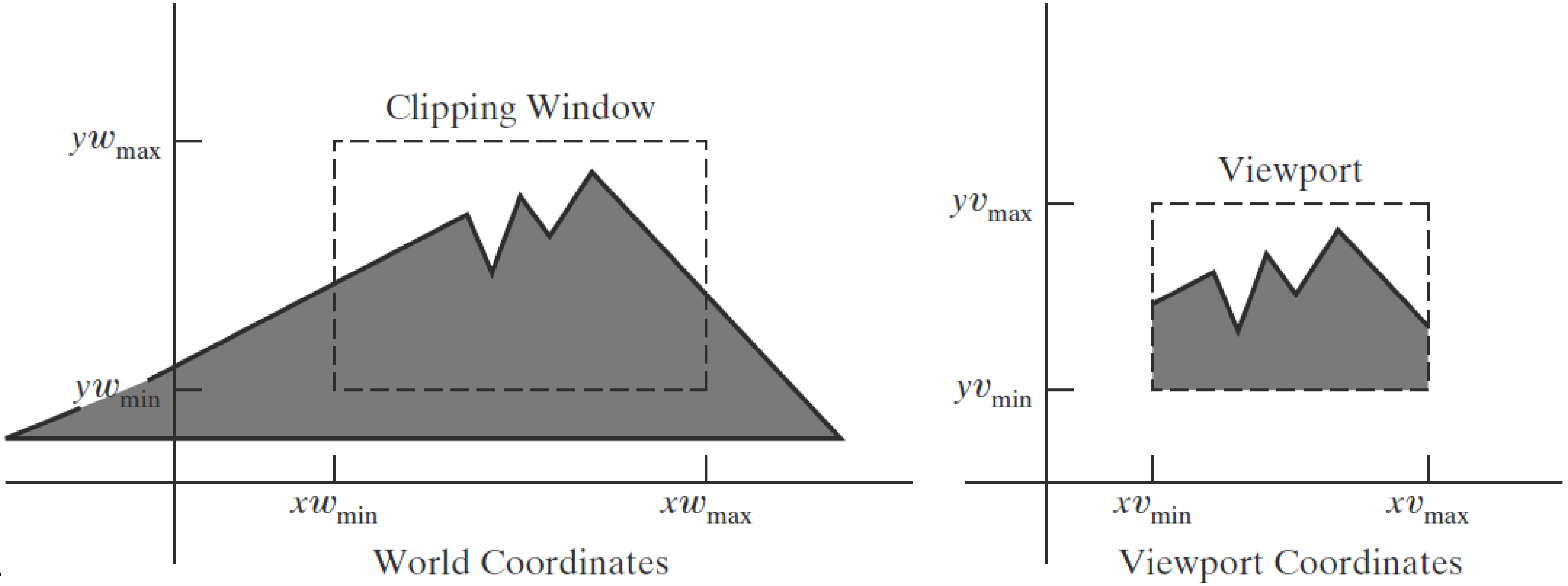
Dept. of CSE, Tezpur University

2D Viewing

- 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
- 2D viewing deals with the procedures for displaying views of a two-dimensional picture on an output device
  - A view for a 2D picture is selected by specifying a region of the xy plane that contains the total picture or any part of it
  - Picture parts within the selected areas are then mapped onto specified areas of the device coordinates
  - 2D viewing 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 scene area

2D Viewing

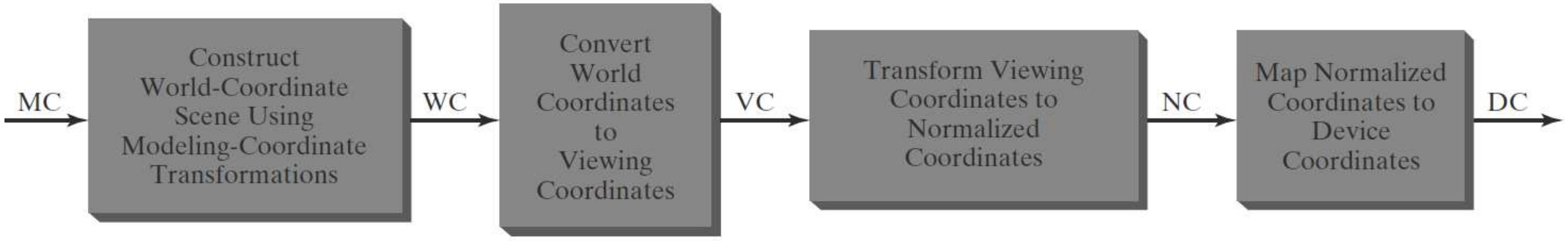
- A section of a two-dimensional scene that is selected for display is called a **clipping window** because all parts of the scene outside the selected section are "clipped" off
  - Sometimes the clipping window is referred to as the *world window or the viewing window*



2D viewing

- Graphics packages allow us also to control the placement within the display window using another "window" called the **viewport**
  - Objects inside the clipping window are mapped to the viewport; viewport is then positioned within display window
  - Changing the **position** of a viewport, we can view objects at different positions on the display area of an output device
  - Changing the **size** of viewports, we can change the size and proportions of displayed objects; zoom in and zoom out effects
- The clipping window selects **what** we want to see; the viewport indicates **where** it is to be viewed on the output device and **how**.

2D viewing pipeline


MC
Construct World-Coordinate Scene Using Modeling-Coordinate Transformations
WC
Convert World Coordinates to Viewing Coordinates
VC
Transform Viewing Coordinates to Normalized Coordinates
NC
Map Normalized Coordinates to Device Coordinates
DC

- The mapping of a two-dimensional, world-coordinate scene description to device coordinates is called a **2D viewing transformation**
  - ***window-to-viewport transformation*** or the ***windowing transformation***
- Once a world-coordinate scene has been constructed, we could set up a separate **2D viewing coordinate** reference frame for specifying the clipping window
  - Clipping window is defined in world coordinates, so viewing coordinates for 2D applications are same as world coordinates
  - To make viewing process ***independent*** output device requirements, graphics systems convert object descriptions to normalized coordinates
  - Contents of viewport are transferred to positions within the display window

Clipping Window

- Clipping windows can be any shape, size, and orientation
  - But a concave polygon or a window with nonlinear boundaries requires more processing
  - 2 ways to define the clipping window
- Set up a **viewing-coordinate system** within the world-coordinate frame
  - **Viewing frame** provides a reference for specifying a rectangular clipping window with any orientation and position
  - choose an origin P0=(x0,y0), and specify orientation
  - Based on the parameters transform the scene description to the viewing system

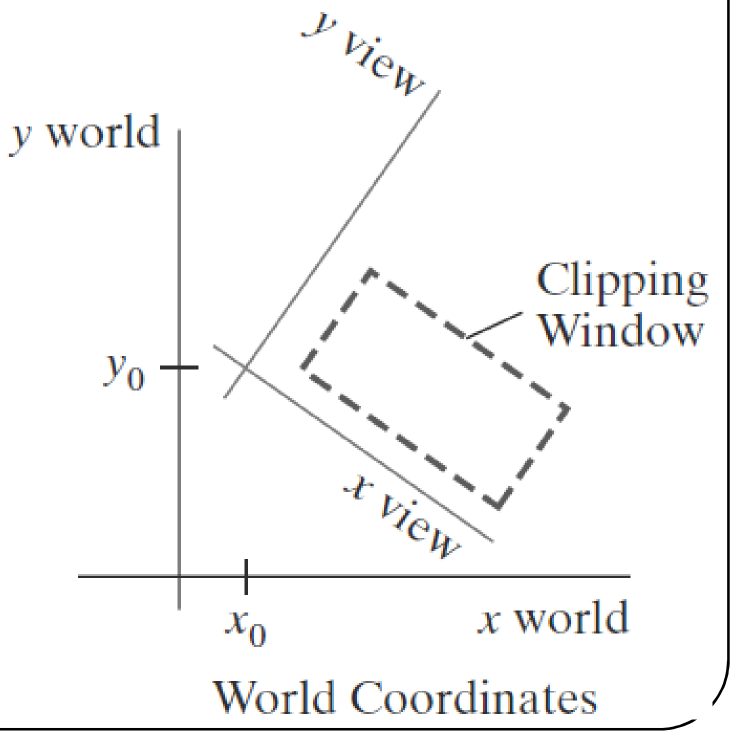
Clipping window - in VC system

- Transforming **viewing-coordinate (VC) system**
  - Translate the viewing origin to the world origin
  - Rotate the viewing system to align it with the world frame
  - Object positions WC are then converted to VC with

  the composite two-dimensional

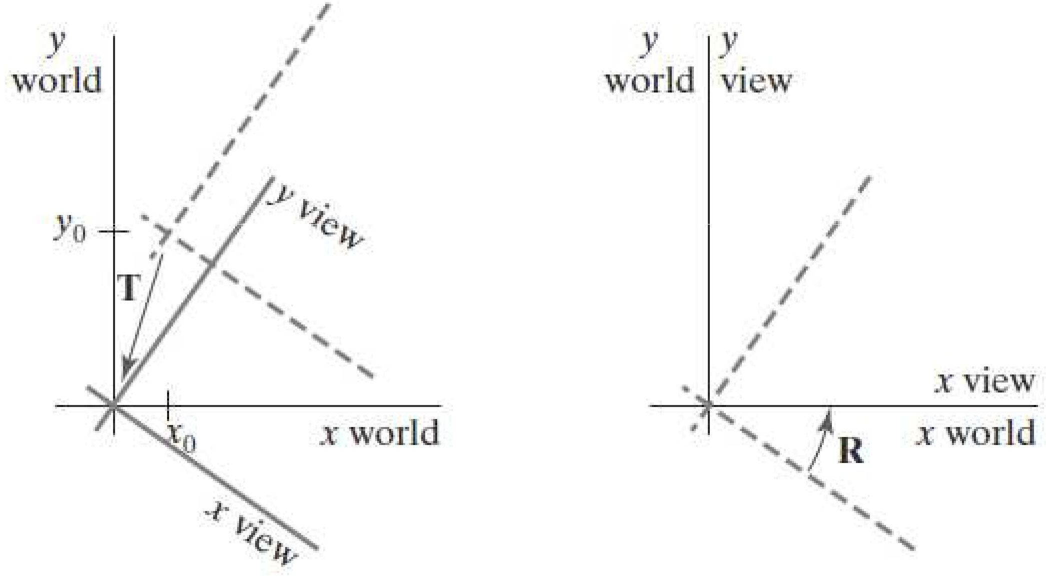
  transformation matrix

$$\mathbf{M}_{WC,VC} = \mathbf{R} \cdot \mathbf{T}$$



Clipping window – WC system

- Rectangular clipping window in world coordinates (WC) is typically provided in a graphics-programming library
- Define opposing ends of the clipping window rectangle in WC
- Perform sequence of transformations as required for objects in the clipping window but without a requiring a frame of reference

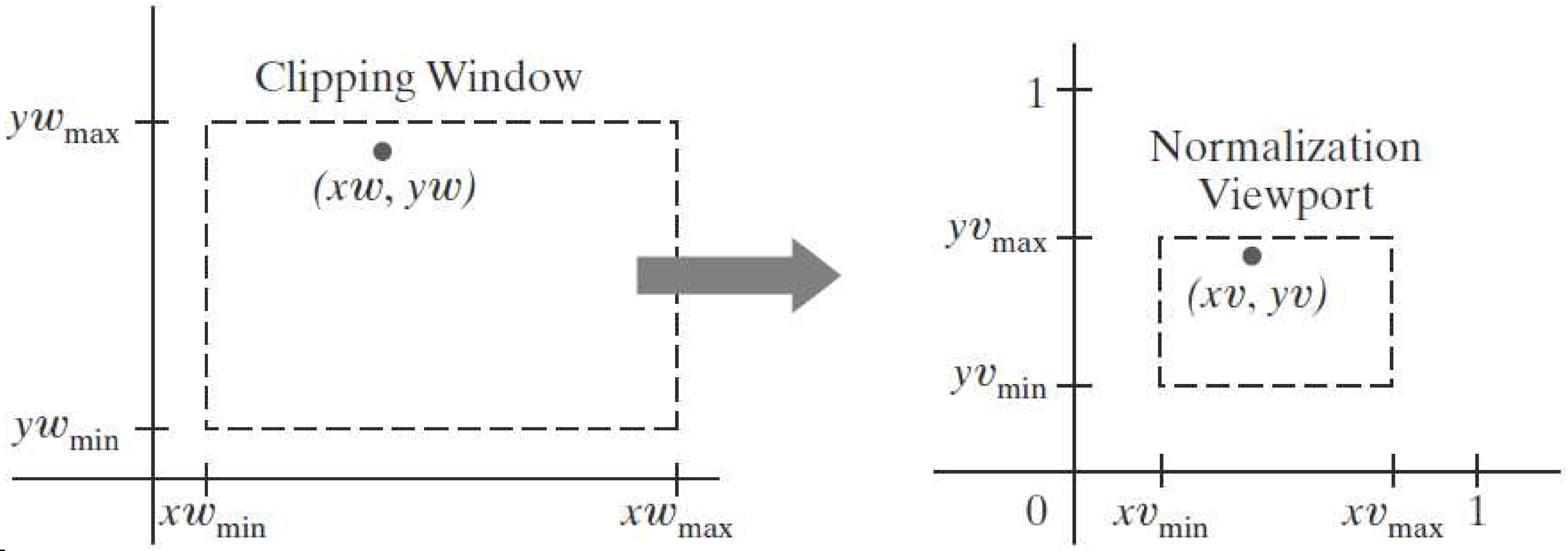


Normalization

- Normalization: Specify coordinate values between 0 and 1
- normalization and window-to-viewport transformations can be combined into one operation.
  - In this case, the viewport coordinates in the range from 0 to 1 so that the viewport is positioned within a unit square
  - Unit square containing the viewport is mapped to the output display device
- normalization and clipping routines are applied before the viewport transformation
  - The viewport boundaries are specified in screen coordinates relative to the display window position

Window to Viewport

- Object descriptions are transferred to this normalized space using a transformation that maintains the same relative placement of a point in the viewport as it had in the clipping window
- Position *(xw, yw)* in the clipping window is mapped to position *(xv, yv)* in the associated viewport



Window to Viewport

- To transform the world-coordinate point into the same relative position within the viewport, we require,

$$\frac{xv - xv_{\min}}{xv_{\max} - xv_{\min}} = \frac{xw - xw_{\min}}{xw_{\max} - xw_{\min}}$$

$$\frac{yv - yv_{\min}}{yv_{\max} - yv_{\min}} = \frac{yw - yw_{\min}}{yw_{\max} - yw_{\min}}$$

- Solving these equations for *(xv, yv)*

$$xv = s_x xw + t_x$$

$$yv = s_y yw + t_y$$

such that $s_x = \dfrac{xv_{\max} - xv_{\min}}{xw_{\max} - xw_{\min}}$ $\quad t_x = \dfrac{xw_{\max} xv_{\min} - xw_{\min} xv_{\max}}{xw_{\max} - xw_{\min}}$

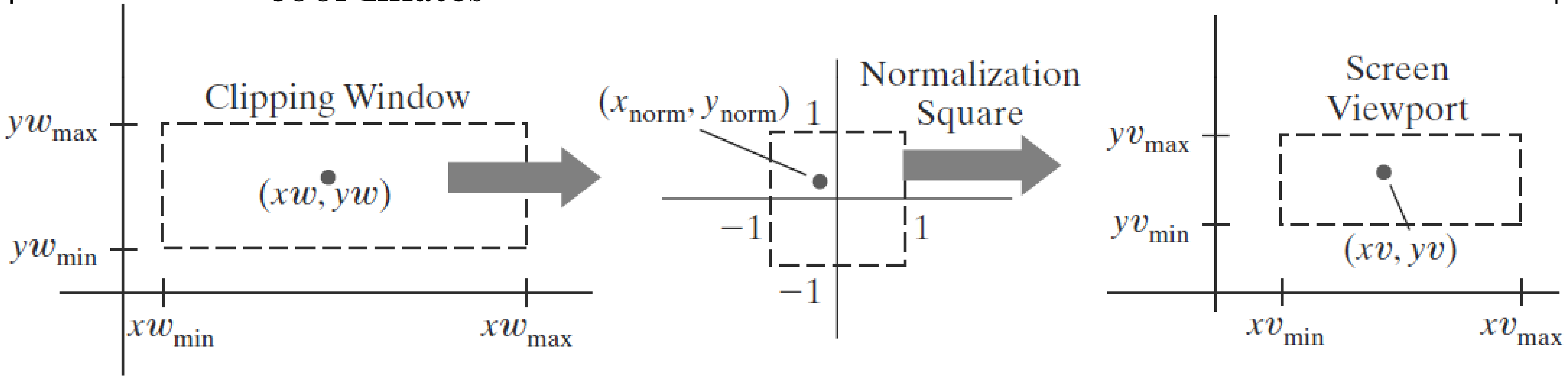
$$s_y = \frac{yv_{\max} - yv_{\min}}{yw_{\max} - yw_{\min}} \qquad t_y = \frac{yw_{\max} yv_{\min} - yw_{\min} yv_{\max}}{yw_{\max} - yw_{\min}}$$

Window to viewport

- Window-to-viewport transformation
  - Relative placement of object descriptions is maintained
  - Relative proportions of objects, on the other hand, are maintained only if the aspect ratio of viewport is same as that of clipping window i.e. $sx = sy$
  - Else, world objects will be stretched or contracted in either the x or y directions (or both) when displayed on the output device
- Clipping window to normalized viewport
  - clipping routines can be applied using either the clipping-window boundaries or the viewport boundaries
  - normalized coordinates are transformed into device coordinates
  - the unit square can be mapped onto the output device by translation and scaling

Window to Viewport

- Clipping window into a normalized square
  - clipping routines applied on normalized coordinates
  - transfer the scene description to a viewport specified in screen coordinates



  - transfer the contents of the clipping window into the normalization square - similar to window-to-viewport transformation

Window to Viewport

- Window to normalized square transformation matrix
  - Obtained by *xvmin* , *yvmin* = –1 and *xvmax, yvmax*= +1

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

- Normalized square to viewport transformation matrix
  - Obtained by *xwmin* , *ywmin* = –1 and *xwmax, ywmax*= +1

$$\mathbf{M}_{\text{normsquare, viewport}} = \begin{bmatrix} \frac{xv_{\text{max}} - xv_{\text{min}}}{2} & 0 & \frac{xv_{\text{max}} + xv_{\text{min}}}{2} \\ 0 & \frac{yv_{\text{max}} - yv_{\text{min}}}{2} & \frac{yv_{\text{max}} + yv_{\text{min}}}{2} \\ 0 & 0 & 1 \end{bmatrix}$$

Clipping Algorithms

- A procedure that eliminates those portions of a picture that are either inside or outside a specified region of space is referred to as a **clipping algorithm** or simply **clipping**
- Usually a clipping region is a rectangle in standard position, although any shape could be used
- Applications
  - Viewing pipeline
  - antialias object boundaries
  - construct objects using solid-modeling methods
  - manage a multiwindow environment, and to
  - allow parts of a picture to be moved, copied, or erased in drawing and painting programs

Point Clipping

- Given a clipping rectangle in standard position, we save a two-dimensional point **P** = *(x, y)* for display if

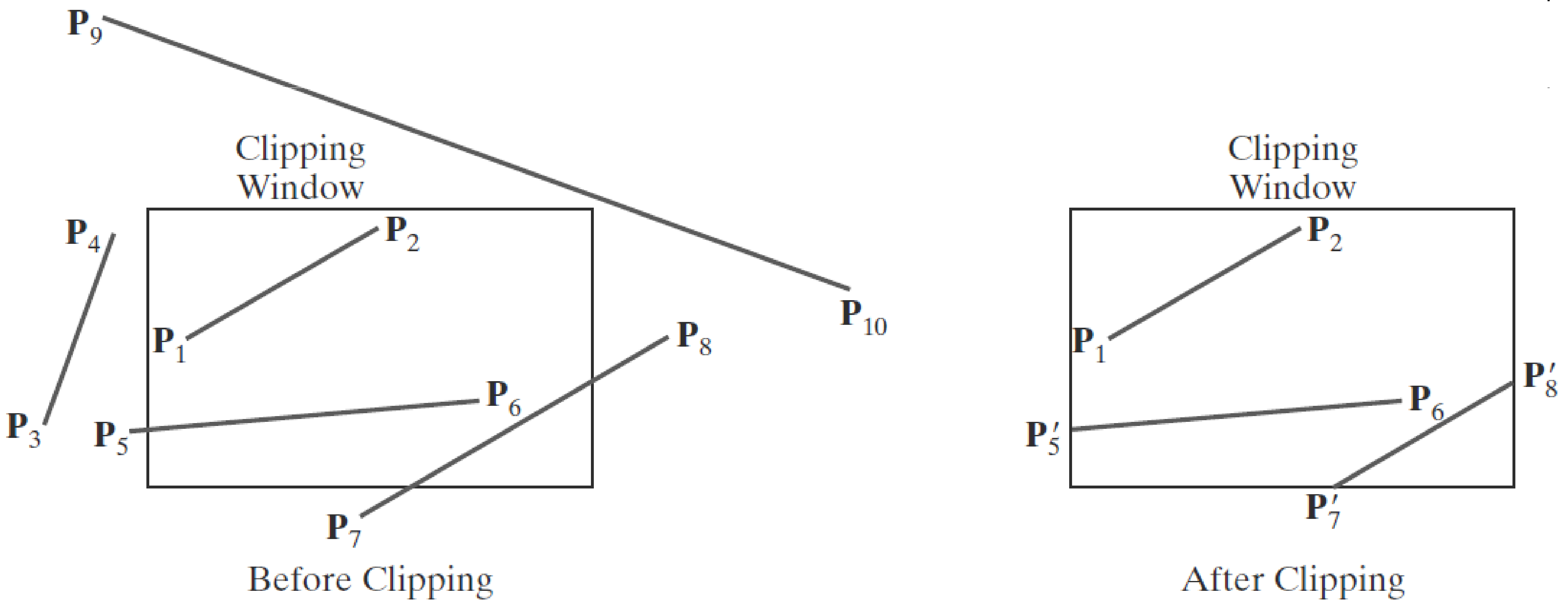
$$xw_{\min} \le x \le xw_{\max}$$

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

- If any of these inequalities is not satisfied, the point is clipped
  - not saved for display

Line Clipping

- A line-clipping algorithm processes each line in a scene to determine whether the entire line or any part of it is to be saved for display



Line Clipping

- Need to compute intersection points of lines with clipping window
  - minimize intersection calculation for better algorithm
- Test whether a line segment is completely inside the clipping window or completely outside
  - Completely inside – easy; both points are inside – P1P2
  - Completely outside – difficult; both points outside?? – P3P4
- If not possible – Test whether any part of a line crosses the window interior

Line Clipping

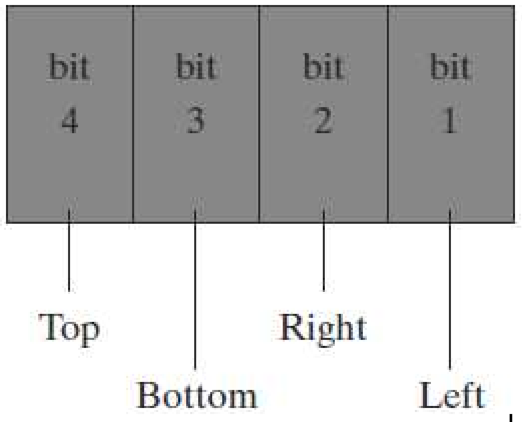
- Use the parametric equation of line

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

  to determine intersections by assigning the coordinate value for that edge to either *x* or *y* and solving for parameter *u*.
- If the value of *u* is outside the range from 0 to 1
  - line segment does not intersect that window border line
  - Else part of the line is inside the border
- Eg: boundary is *xwmin,* so substitute this value for *x*, solve for *u*, and calculate corresponding *y*-intersection value
- Process inside portion of the line segment against the other clipping boundaries
  - until the section inside the window is found

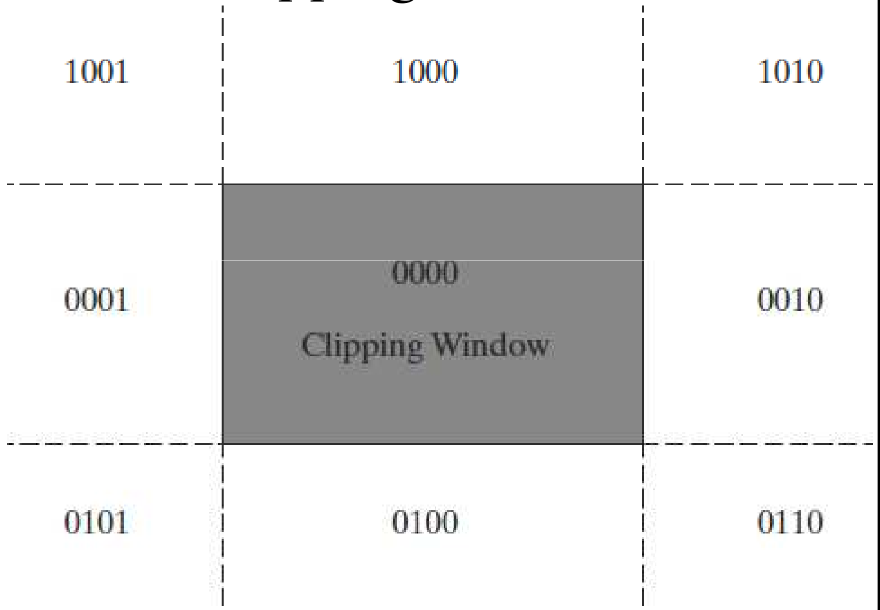
Line Clipping – Cohen-Sutherland

- One of the earliest algorithms for fast line clipping
- Processing time is reduced by performing more tests before proceeding to the intersection calculations
- Every line endpoint in a picture is assigned a four-bit binary value – **region code**
  - each bit position is used to indicate whether the point is inside or outside one of the clipping-window boundaries
  - 1 (or *true*)=> endpoint is outside that window
  - 0 (or *false*) => endpoint is not outside (it is inside or on) the corresponding window edge



Line Clipping – Cohen-Sutherland

- Bit values in a region code determined by comparing the coordinate values *(x, y)* endpoints to the clipping boundaries
  - Bit 1 is set to 1 if $x < xwmin$
  - Bit 2 is set to 1 if $x > xwmax$
  - Bit 3 is set to 1 if $y < ywmin$
  - Bit 4 is set to 1 if $y > ywmax$



- Instead of using inequality testing, determine the values for a region-code:
  - Calculate differences between endpoint coordinates and clipping boundaries.
  - Use the resultant sign bit of each difference calculation to set the corresponding value in the region code.

Line Clipping – Cohen-Sutherland

- For any given line,
  - If both endpoints have region code 0000 – completely inside
  - If both endpoints has a region-code value of 1 in at least one same bit position – completely outside the clipping rectangle, and we eliminate that line segment
- Lines that cannot be identified as completely inside or completely outside are checked for intersections
- As each clipping-window edge is processed, a section of the line is clipped, and remaining part of the line is checked against the other window borders
  - Eliminate sections until either the line is totally clipped or the remaining part of the line is inside the clipping window

Line Clipping – Cohen-Sutherland

- To determine a boundary intersections, use the slope intercept form of the line equation
- For a line with endpoint coordinates (*x0, y0*) and (*xend, yend*),
  - For intersection point with a vertical clipping border line, *y*-coordinate is obtained with the calculation-

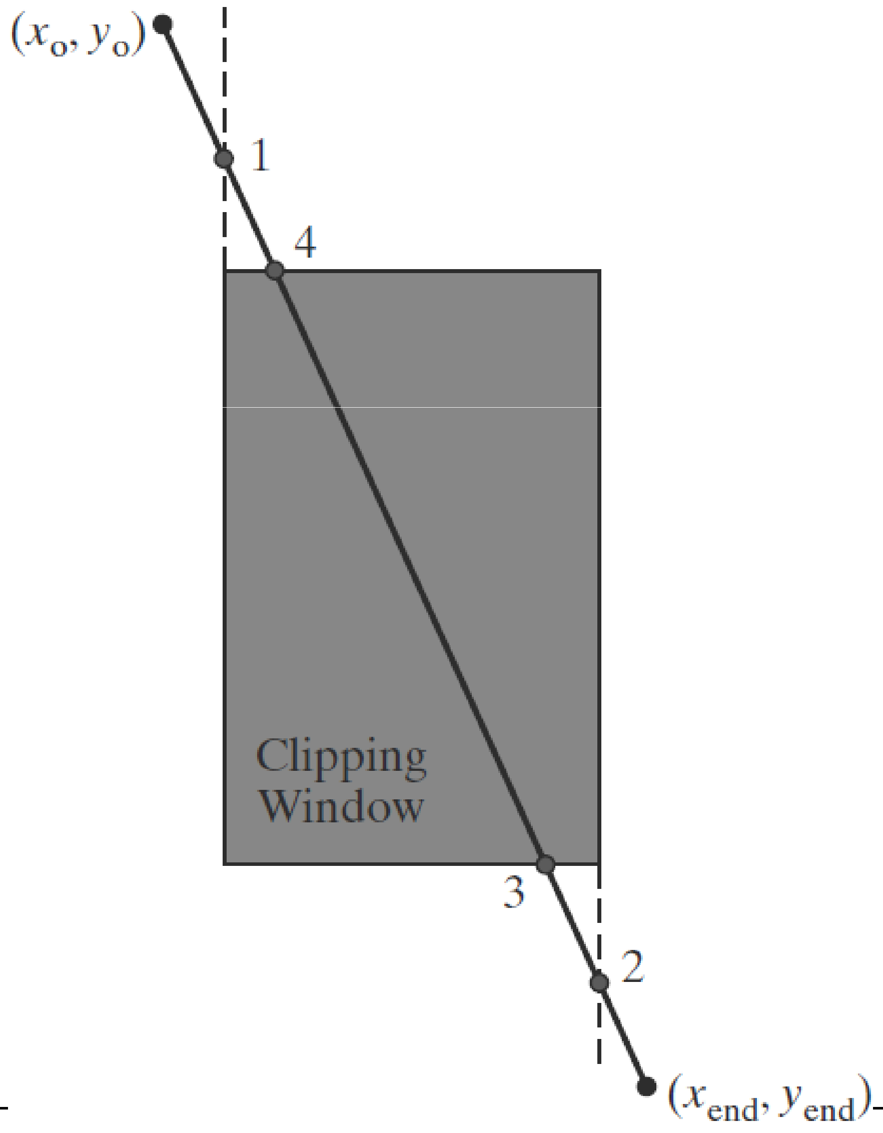
    $y = y_0 + m(x - x_0)$ $\qquad$ $m = (y_{\text{end}} - y_0)/(x_{\text{end}} - x_0)$

    where the *x* value is set to either *xwmin* or *xwmax*
  - Similarly, for intersection with a horizontal border, the *x*-coordinate is calculated as

    $$x = x_0 + \frac{y - y_0}{m}$$

    with *y* set either to *ywmin* or to *ywmax*

Line Clipping – Cohen-Sutherland



Four intersection positions (labeled from 1 to 4) for a line segment that is clipped against the window boundaries in the order left, right, bottom, top