Parul® University

05201330 / 05202182 – Computer Graphics

Dr. Ghanshyam Rathod, Assistant Professor
Parul Institute of Computer Application - BCA

Parul®
University

DIGITAL LEARNING CONTENT

CHAPTER-3

Viewing and Clipping

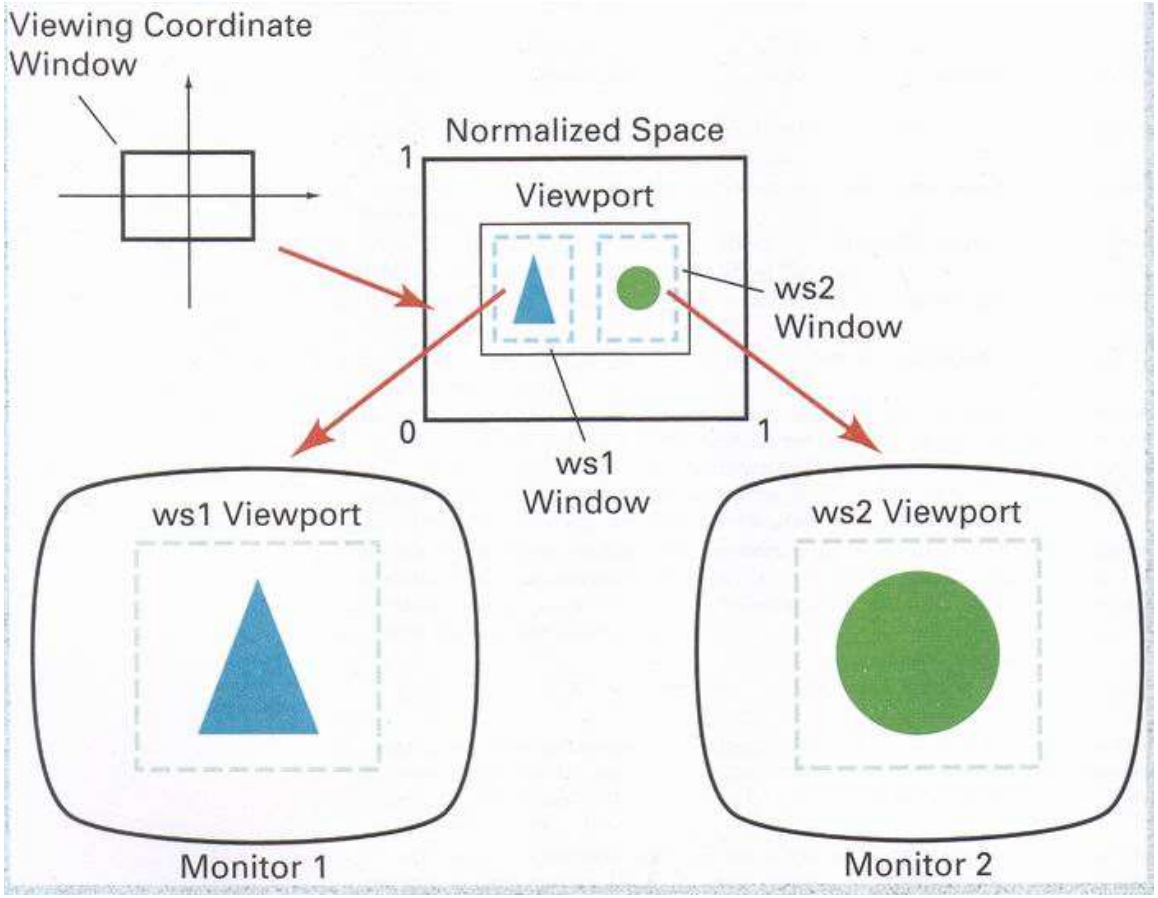
Concepts

Window: Area selected in world-coordinate for display is called window. It defines **what is to be viewed**.

Viewport: Area on a display device in which window image is display (mapped) is called viewport. It defines **where to display**.
In many cases window and viewport are rectangle, also other shape may be used as window and viewport such as polygon or circle.

Viewing Transformation: The process of converting coordinates from a "world window" (defined in world coordinates) to the corresponding coordinates on the display device.

Concepts


Viewing Coordinate
Window
Normalized Space
1
Viewport
ws2
Window
0
1
ws1
Window
ws1 Viewport
ws2 Viewport
Monitor 1
Monitor 2

Viewing in Computer Graphics

Viewing refers to the process of transforming a 3D scene into a 2D image that can be displayed on a screen. It involves defining how the objects in the scene are mapped to a viewable area. The process includes:

1. Setting up a Camera (Viewing Coordinate System):
   The viewing position is defined using a virtual camera or an eye position in the 3D space.

2. Viewing Transformation:
   Transforms the objects from the world coordinate system to the viewing coordinate system.

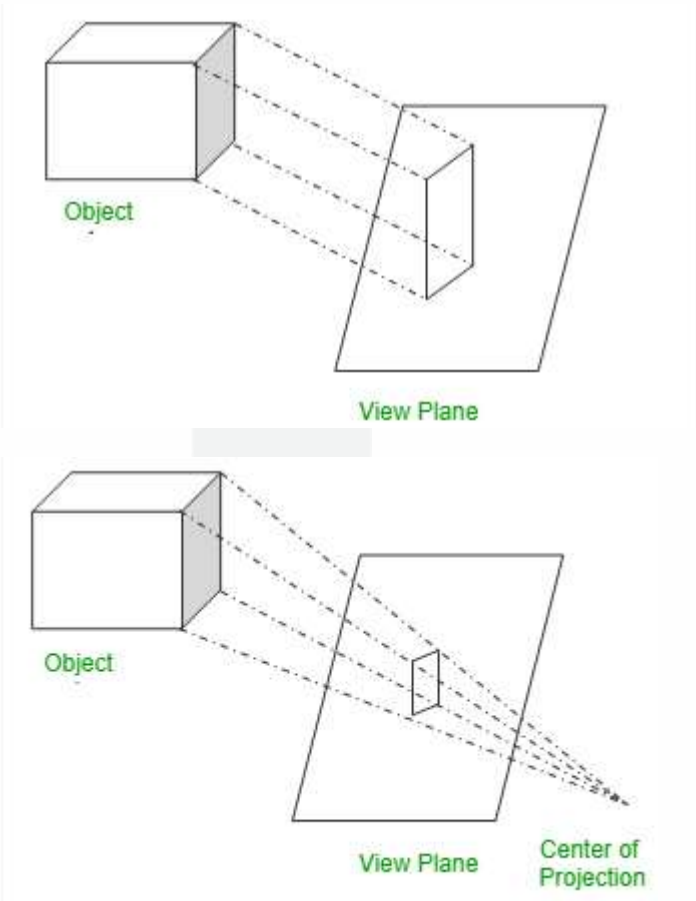
Viewing in Computer Graphics

3. Projection: Converts the 3D scene into a 2D representation.

   Projections can be of two types:

   - **Parallel Projection:** Maintains the relative dimensions of objects (used in engineering drawings).

   - **Perspective Projection:** Mimics how the human eye sees, with objects appearing smaller as they get farther away.

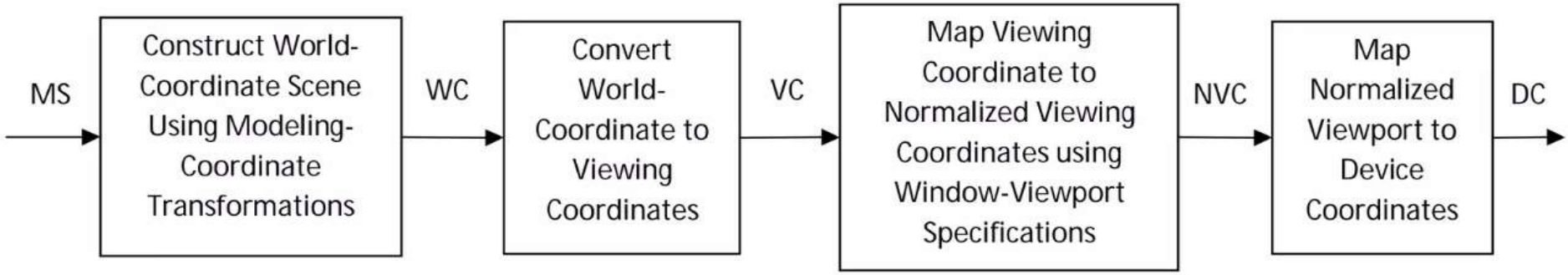


Viewing Transformation Pipeline

- We know that the picture is stored in the computer memory using any convenient Cartesian co-ordinate system, referred to as World Co-Ordinate System (WCS).
- However, when picture is displayed on the display device it is measured in Physical Device Co-Ordinate System (PDCS) corresponding to the display device.
- Therefore, displaying an image of a picture involves mapping the co-ordinates of the Points and lines that form the picture into the appropriate physical device co-ordinate where the image is to be displayed.
- This mapping of co-ordinates is achieved with the use of co-ordinate transformation known as viewing transformation.
- The viewing transformation which maps picture co-ordinates in the WCS to display co-ordinates in PDCS is performed by the following transformations.

Viewing Transformation Pipeline

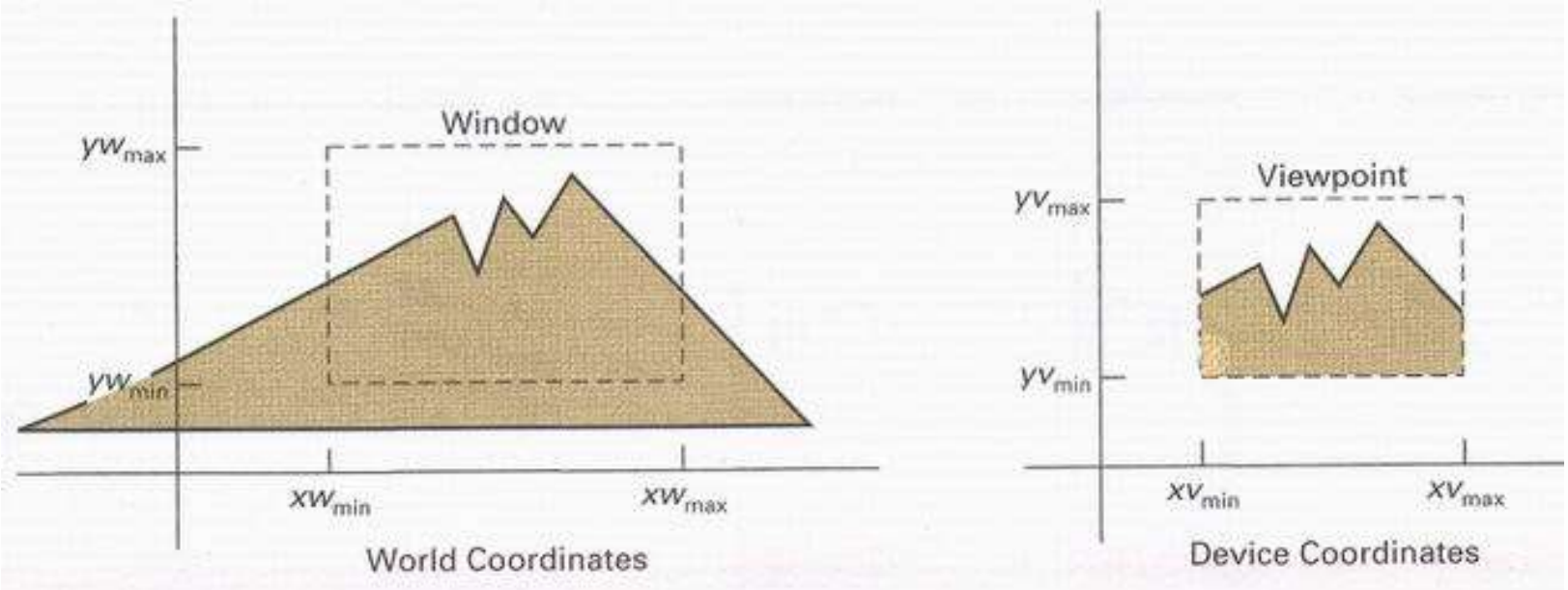
- Converting world co-ordinates to viewing co-ordinates.
- Normalizing viewing co-ordinates.
- Converting normalized viewing co-ordinates to device co-ordinates.



- **MS or MC (Modeling Coordinate):** Coordinates of 3D object in real world.
- **NVC:** It a coordinate system where all points are mapped to a standard range, typically between -1 and 1 on both the x and y axes, regardless of the actual screen resolution or viewport size, allowing for device-independent rendering and simplifying calculations during the graphics pipeline

Viewing Transformation Pipeline

- The mapping of a part of a world-coordinate scene to device coordinates is referred to as a viewing transformation.
- Figure shows the mapping of a picture section that falls within a rectangular window onto a designated & angular viewport.



Viewing Transformation Pipeline

- As shown in above figure, first of all we construct world coordinate scene modeling coordinate transformation.
- After this we convert viewing coordinates from world coordinates using window to viewport transformation.
- Then we map viewing normalized viewing coordinate in which we obtain values in between -1 to 1.
- At last, we convert 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.

Viewing Transformation Pipeline

- 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 defining with the “unit square” so that graphics package are more device independent which we call as normalized viewing coordinate.

Window and Viewport Transformation

- Mapping of window coordinate to viewport is called window to viewport transformation.
- It is done using transformation that maintains relative position of window coordinate into viewport.
- That means center coordinates of window must remains at center position in viewport.
- For, finding relative position by the equation below:

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

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

Window and Viewport Transformation

- By solving we can find position of viewport by below equation:

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

- Where, scaling factor are:

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

Window and Viewport Transformation

- This conversion is performed with the following sequence of transformations:

  1. Perform a scaling transformation using a fixed-point position of (xw_{min} , yw_{min},) that scales the window area to the size of the viewport.

  2. Translate the scaled window area to the position of the viewport.

- Relative proportions of objects are maintained if the scaling factors are the same (sx = sy). Otherwise, world objects will be stretched or contracted in either the x or y direction when displayed on the output device.

Window and Viewport Transformation

- Character strings can be handled in two ways when they are mapped to a viewport.
- The simplest mapping maintains a constant character size, even though the viewport area may be enlarged or reduced relative to the window.
- This method would be employed when text is formed with standard character fonts that cannot be changed.
- In systems that allow for changes in character size, string definitions can be windowed the same as other primitives.
- For characters formed with line segments, the mapping to the viewport can be carried out as a sequence of line transformations.

Clipping Operations

- Clipping refers to the process of removing portions of graphics primitives (like lines, polygons, or text) that lie outside the viewing area or clipping window. It ensures that only the visible parts of objects are displayed on the screen.
- Generally, 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. The region against which an object is to be clipped is called a clip window.
- The clipping operation can be performed on different objects. Thus, it can be divided as:
    1. Point Clipping
    2. Line Clipping (straight-line segments)
    3. Area Clipping (polygons)
    4. Curve Clipping
    5. Text Clipping

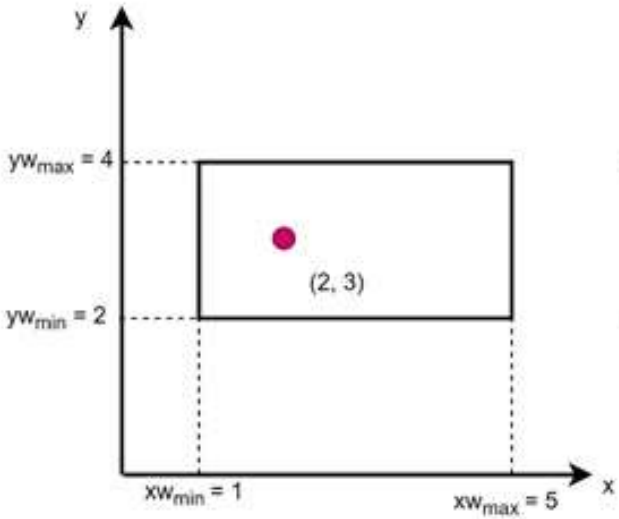
Point Clipping

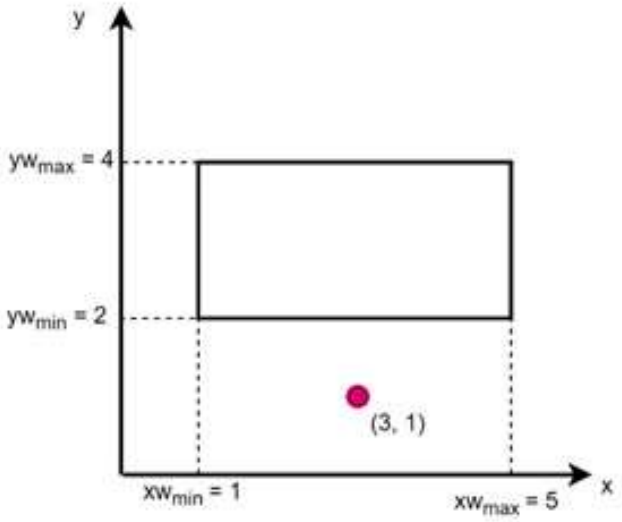
- Assuming that the clip window is a rectangle in standard position, we save a point P = (x , y) for display if the following inequalities are satisfied:

$$xw_{min} <= x <= xw_{max}$$
$$yw_{min} <= y <= yw_{max}$$

- Where the edges of the clip window (xw_{min} , xw_{max} , yw_{min} , yw_{max}) can be either the world-coordinate window boundaries or viewport boundaries. If any one of these four inequalities is not satisfied, the point is clipped (not saved for display).
- Although point clipping is applied less often than line or polygon clipping, some applications may require a point-clipping procedure. For example, point clipping can be applied to scenes involving explosions or sea foam that are modeled with particles (points) distributed in some region of the scene.

Point Clipping





Here the (xw_{min}, yw_{min}) = (1, 2) and (xw_{max}, yw_{max}) = (5, 4), the point coordinate is (2, 3)

For the first case,
It satisfies the conditions:

$1 < 2 < 5$

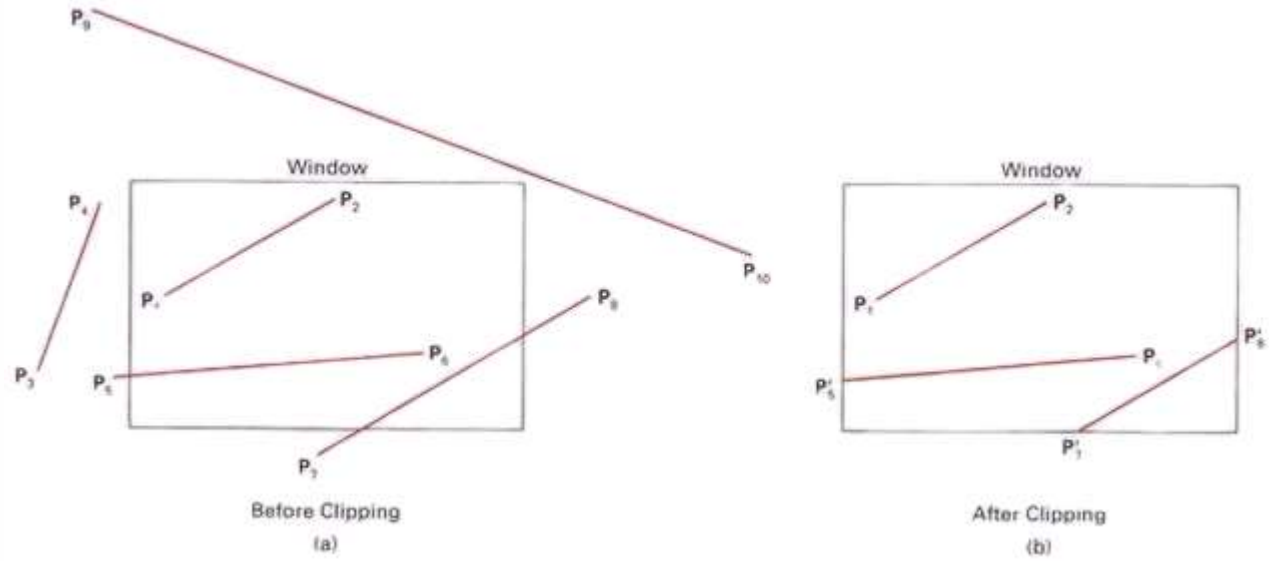
$2 < 3 < 4$

For the second case, the point is at (3, 1) so it does not satisfy, the condition for y, and it is clipped.

$1 < 3 < 5$

$2 < 1 < 4$ (invalid condition)

Line Clipping



- Figure shows possible relationships between line positions and a standard rectangular clipping region. A line clipping procedure involves several parts.
- First, we can test a given line segment to determine whether it lies completely inside the clipping window.

Line Clipping

- If it does not, we try to determine whether it lies completely outside the window.
- Finally, if we cannot identify a line as completely inside or completely outside, we must perform intersection calculations with one or more clipping boundaries.
- We process lines through the “inside-outside” tests by checking the line endpoints.
- A line with both endpoints inside all clipping boundaries, such as the line from P1 to P2 is saved.
- A line with both endpoints outside any one of the clip boundaries (line P3-P4 in Fig) is outside the window.
- All other lines cross one or more clipping boundaries, and may require calculation of multiple intersection points.
- To minimize calculations, we try to devise clipping algorithms that can efficiently identify outside lines and reduce intersection calculations.

Line Clipping

- For a line segment with endpoints (x_1 , y_1) and (x_2 , y_2) and one or both endpoints outside the clipping rectangle,

- The clipping operation can be decided from the values

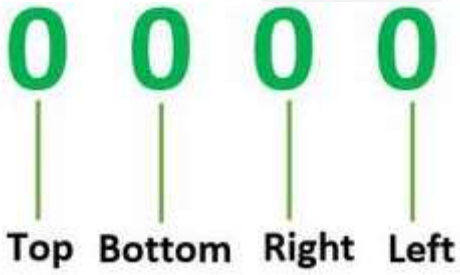
$$x = x_1 + u (x_2 - x_1)$$
$$y = y_1 + u (y_2 - y_1)$$

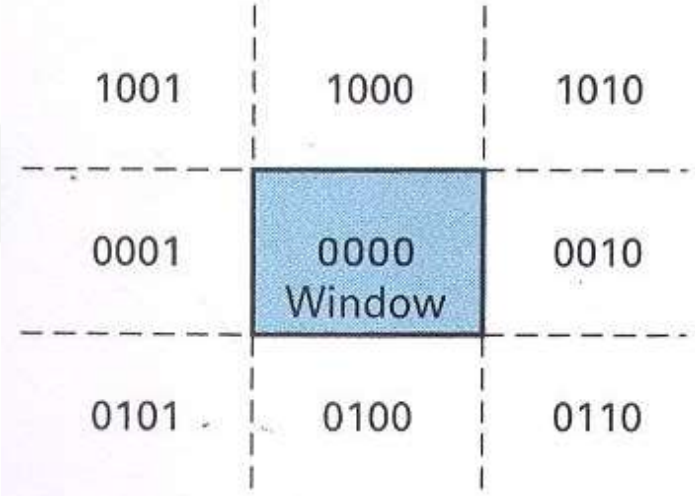
$$0 <= u <= 1$$

- If the value of u is outside the range 0 to 1 then, line is outside.
- If the value of u is within the range of 0 to 1 then, it crosses the clipping area.

Cohen-Sutherland Line Clipping

- This is the oldest and the most popular line clipping procedure.
- In this procedure both the endpoints of line contain a four-digit binary code, known as Region Code: This code identifies the location of point with respect to the clip window.





In this,

- bit 1 represents: Left
- bit 2 represents: Right
- bit 3 represents: Below
- bit 4 represents: Top

Cohen Sutherland line clipping algorithm divides a two-dimensional space into **9 regions** and then efficiently determines the lines and portions of lines that are inside the given rectangular area

Cohen-Sutherland Line Clipping

- Nine regions are created, eight "outside" regions and one "inside" region.
- For a given line extreme point (x, y), we can quickly find its region's four-bit code.
- Four-bit code can be computed by comparing x and y with four values (x_min, x_max, y_min and y_max).
  1. If x is less than x_min then bit number 1 is set.
  2. If x is greater than x_max then bit number 2 is set.
  3. If y is less than y_min then bit number 3 is set.
  4. If y is greater than y_max then bit number 4 is set

Cohen-Sutherland Line Clipping

- If the value of bit is 1, it indicates the true condition e.g. 0101 means the point is Bottom and Left side of the Rectangle.
- By using such representation, we can make following conclusions:
  1. If the region code of both endpoints is 0000, then the line is completely inside.
  2. If both the endpoints contain 1 in same bit position, then the line is completely outside. This can also be checked by logical AND operation. If the result is not 0000 then, line is completely outside.
  3. If the line is not completely inside or outside, then we have to find out the intersection point of line with window.

Cohen-Sutherland Line Clipping

- 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' coordinate of the intersection point with a vertical boundary can be obtained with the calculation

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

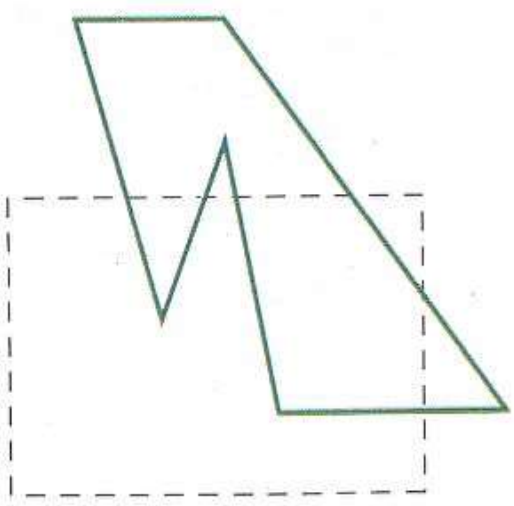
- where the x value is set either to xw_{min}, or to xw_{max}, and the slope of the line is calculated as $m = (y_2 - y_1)/(x_2 - x_1)$.
- Similarly, if we are looking for the intersection with a horizontal boundary, the x coordinate can be calculated as

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

with y set either yw_{min} or to yw_{max}

Polygon Clipping

Polygon clipping operation is carried out by modifying line clipping algorithms. We can consider the polygon as a set of lines and thus clip the selected region. The only problem in this is that the closed area of the polygon cannot be decided.

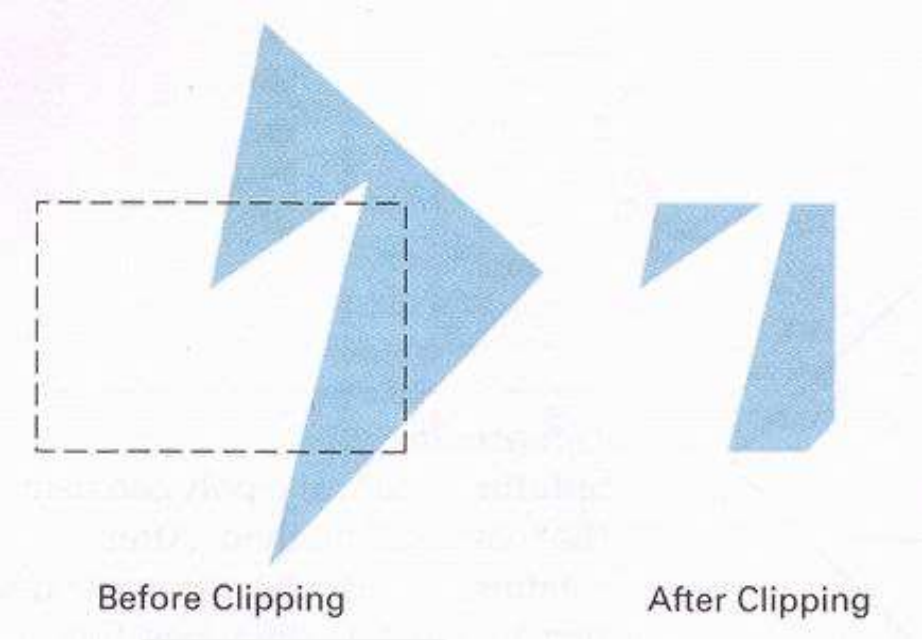


A polygon boundary processed with a **line clipper** may be displayed as a series of unconnected line segments as shown in figure, depending on the orientation of the polygon to the clipping window.

Display of a polygon processed by a line-clipping algorithm.

Polygon Clipping

- What we really want to display is a bounded area after clipping, as in figure.
- For polygon clipping, we require an algorithm that will generate one or more closed areas that are then scan converted for the appropriate area fill.
- The output of a **polygon clipper** should be a sequence of vertices that defines the clipped polygon boundaries.



Display of a correctly clipped polygon

Polygon Clipping (Sutherland Hodgman Algorithm)

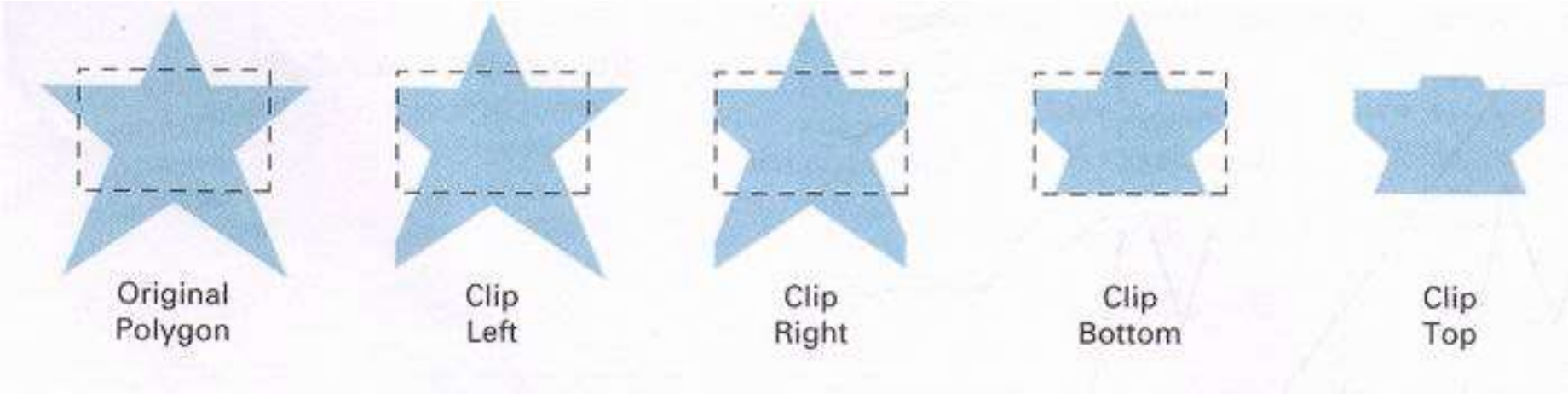
We can correctly clip a polygon by processing the polygon boundary as a whole against each window edge.

This could be accomplished by processing all polygon vertices against each clip rectangle boundary in turn.

Beginning with the initial set of polygon vertices, we could 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**, as shown in figure

Polygon Clipping (Sutherland Hodgman Algorithm)



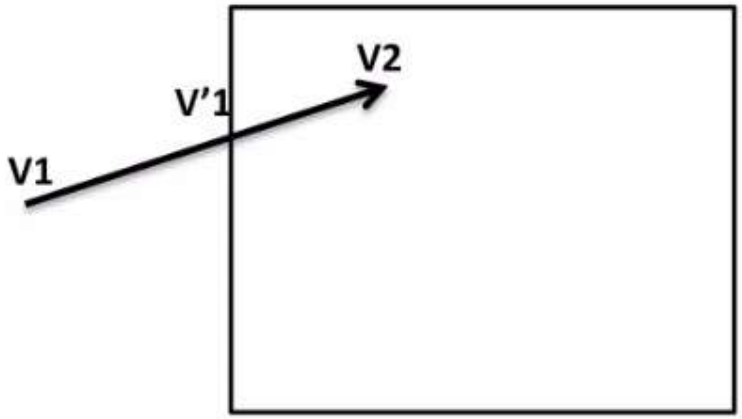
At each step, a new sequence of output vertices is generated and passed to the next window boundary clipper.

Polygon Clipping (Sutherland Hodgman Algorithm)

It consist of Four Cases:

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

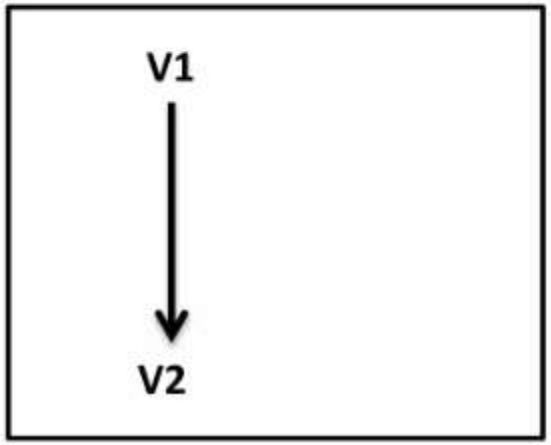
Save {v'1 and v2}



Polygon Clipping (Sutherland Hodgman Algorithm)

2. If the both vertices of the edge are inside the window. Then only second vertex is added to the output vertex list.

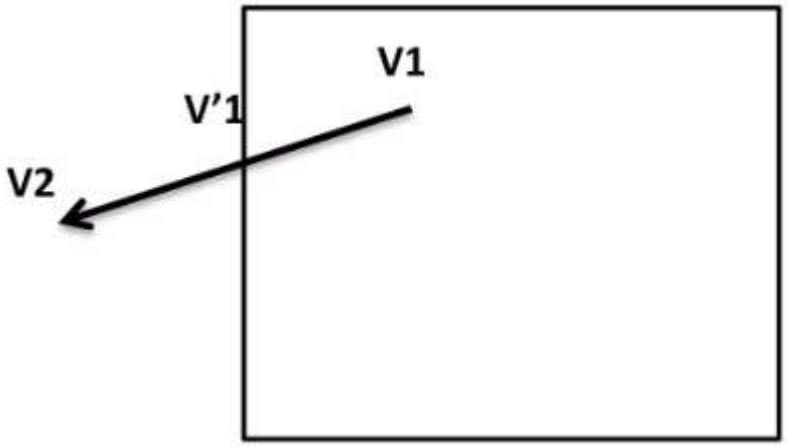
Save {v2}



Polygon Clipping (Sutherland Hodgman Algorithm)

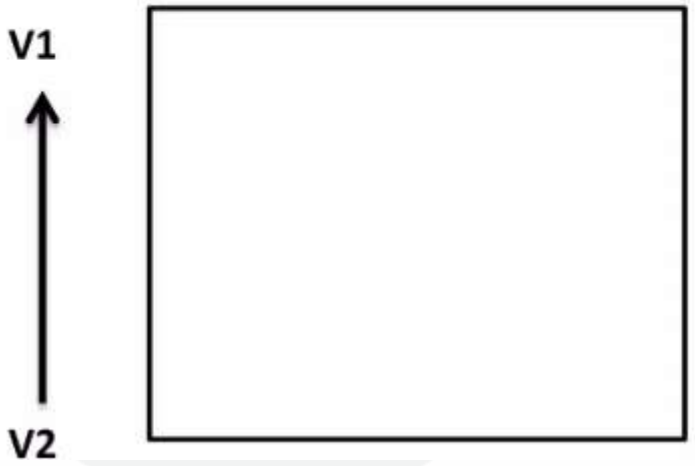
3. If the first vertex of the edge is inside the window and second vertex is outside then only the intersection point of the polygon edge with the window boundary is added to the output vertex list.

Save {v'1}



Polygon Clipping (Sutherland Hodgman Algorithm)

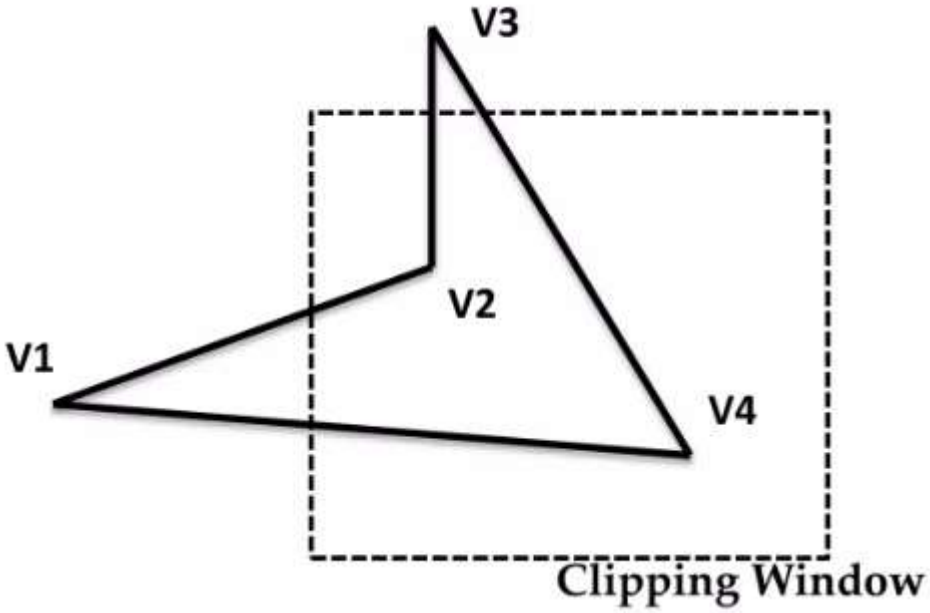
4. If the both vertices of the edge are outside the window. Then nothing is added to the output vertex list.



Save {Ø}

Polygon Clipping (Sutherland Hodgman Algorithm)

Example: For a polygon and clipping window as shown in figure is the original image. We are now applying, Left, Right, Bottom, and Top clipping

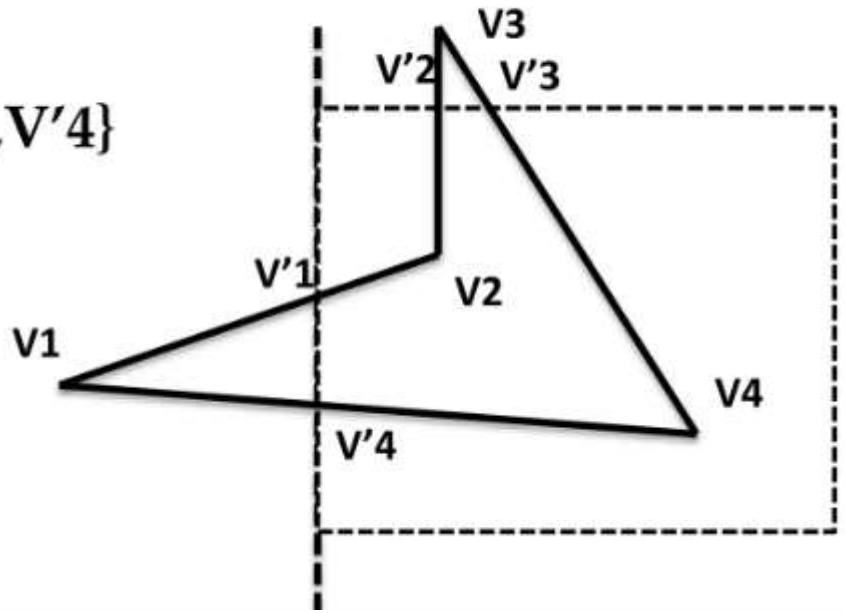


Polygon Clipping (Sutherland Hodgman Algorithm)

1. Left Clipped:

Output vertex=

{ V'1, V2, V'2,V3,V'3,V4,V'4}

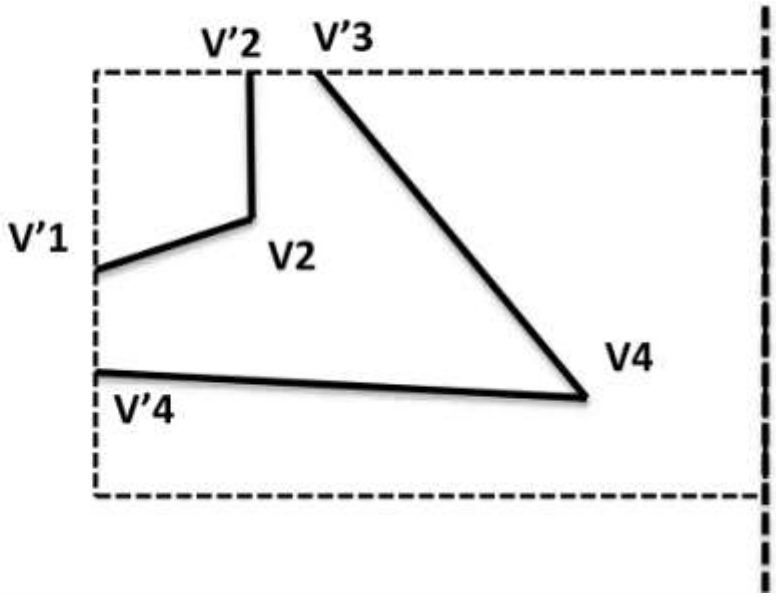


Polygon Clipping (Sutherland Hodgman Algorithm)

2. Right Clipped:

Output vertex=

{ V'1, V2, V'2, V'3,V4,V'4}

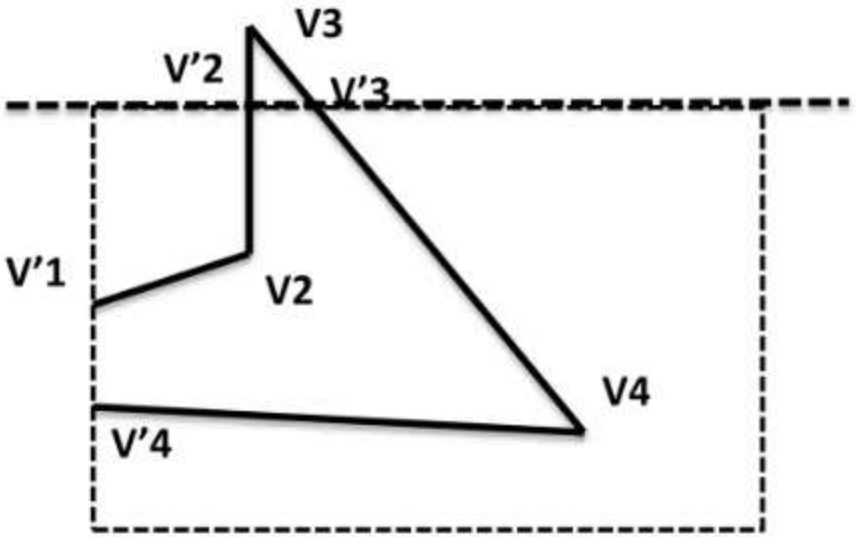


Polygon Clipping (Sutherland Hodgman Algorithm)

3. Top Clipped: :

Output vertex=

{ V'1, V2, V'2, V'3,V4,V'4}

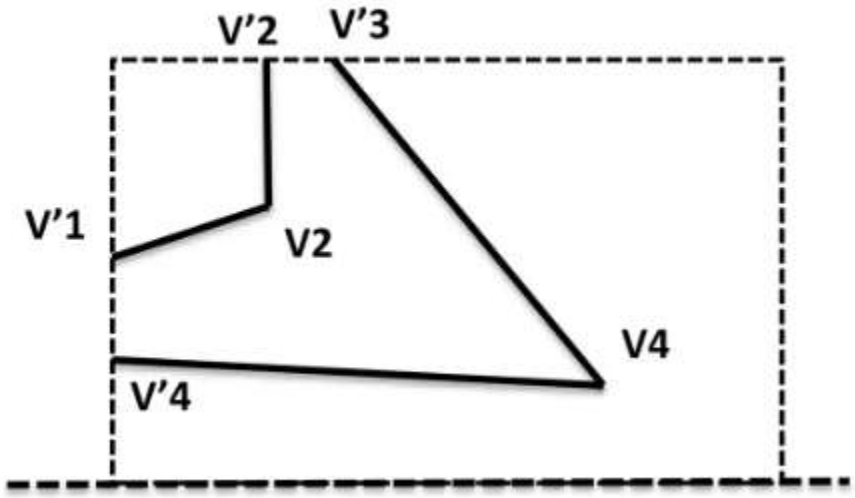


Polygon Clipping (Sutherland Hodgman Algorithm)

4. Bottom Clipped:

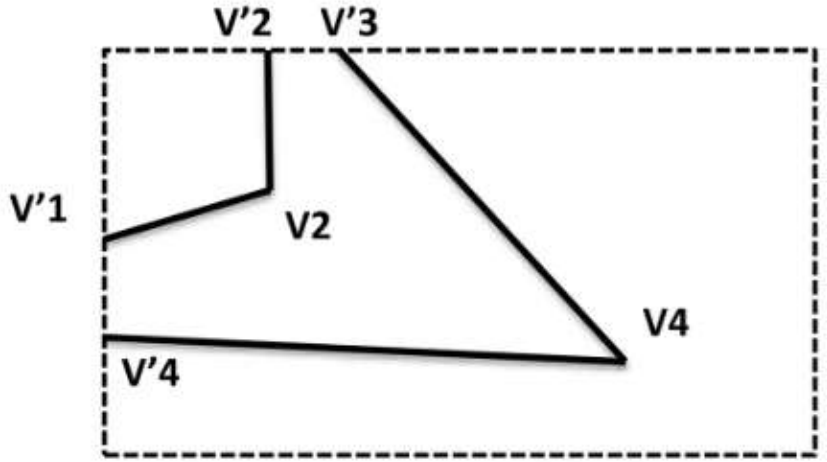
Output vertex=

{ V'1, V2, V'2, V'3,V4,V'4}



Final Clipped polygon:

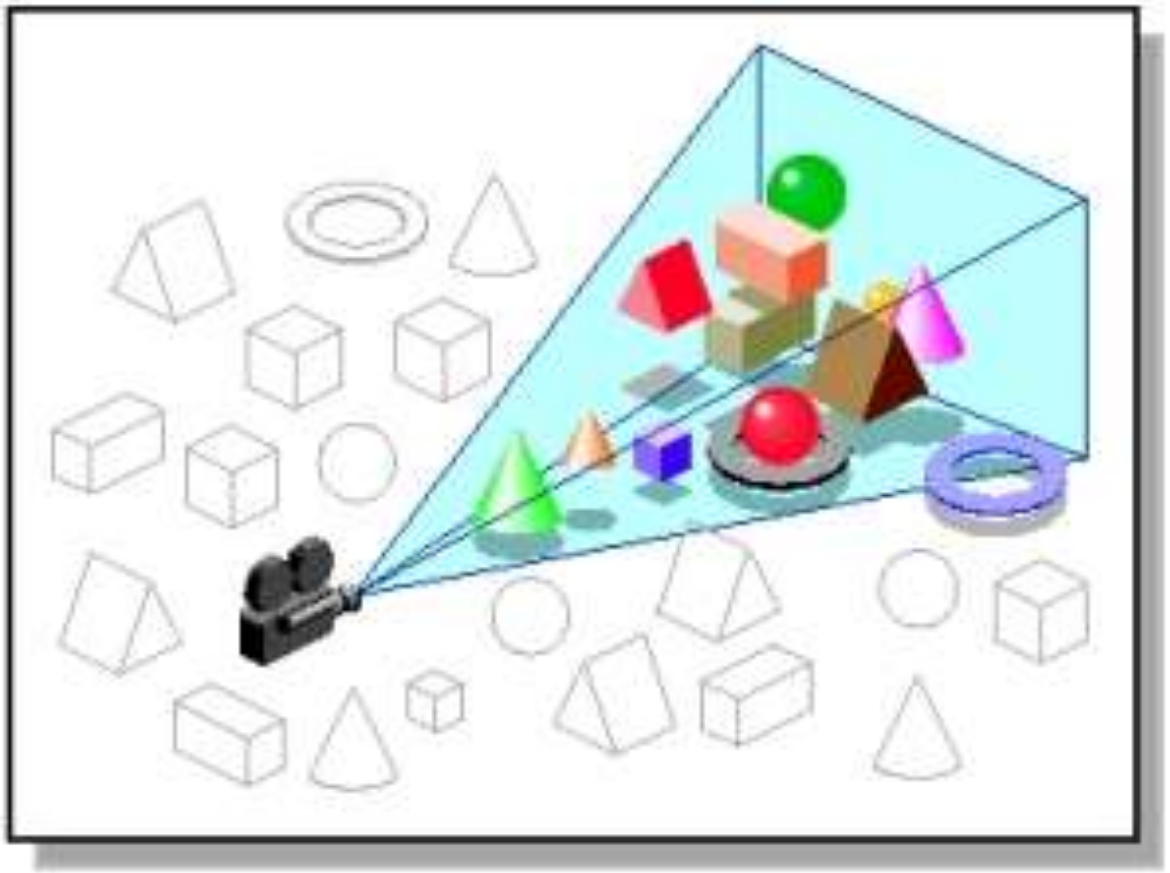
{ V'1, V2, V'2, V'3,V4,V'4}



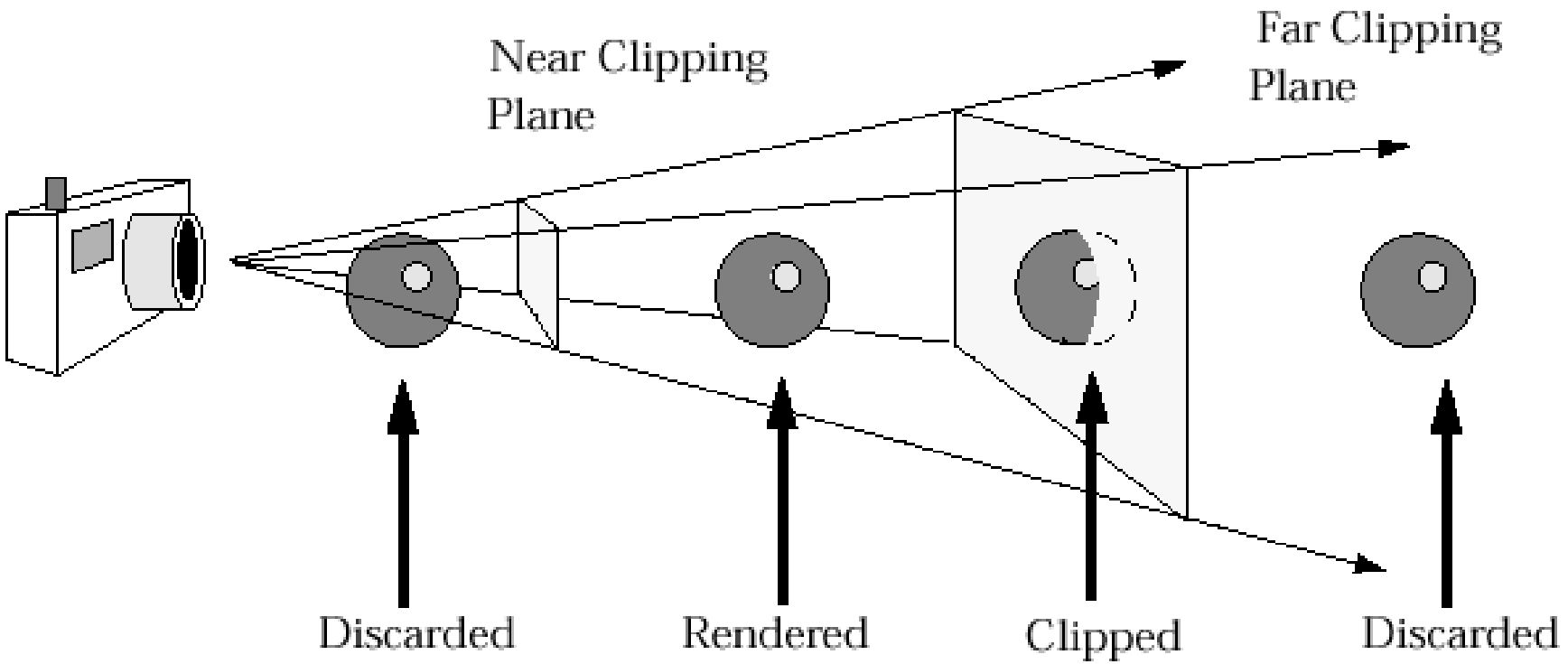
3D Clipping

- Just like the case in two dimensions, clipping removes objects that will not be visible from the scene.
- The point of this is to remove computational effort.
- 3-D clipping is achieved in two basic steps
  - ✓ Discard objects that can't be viewed
    - i.e. objects that are behind the camera, outside the field of view, or too far away
  - ✓ Clip objects that intersect with any clipping plane
- Discarding objects that cannot possibly be seen involves comparing an objects bounding box/sphere against the dimensions of the view volume
  - ✓ Can be done before or after projection

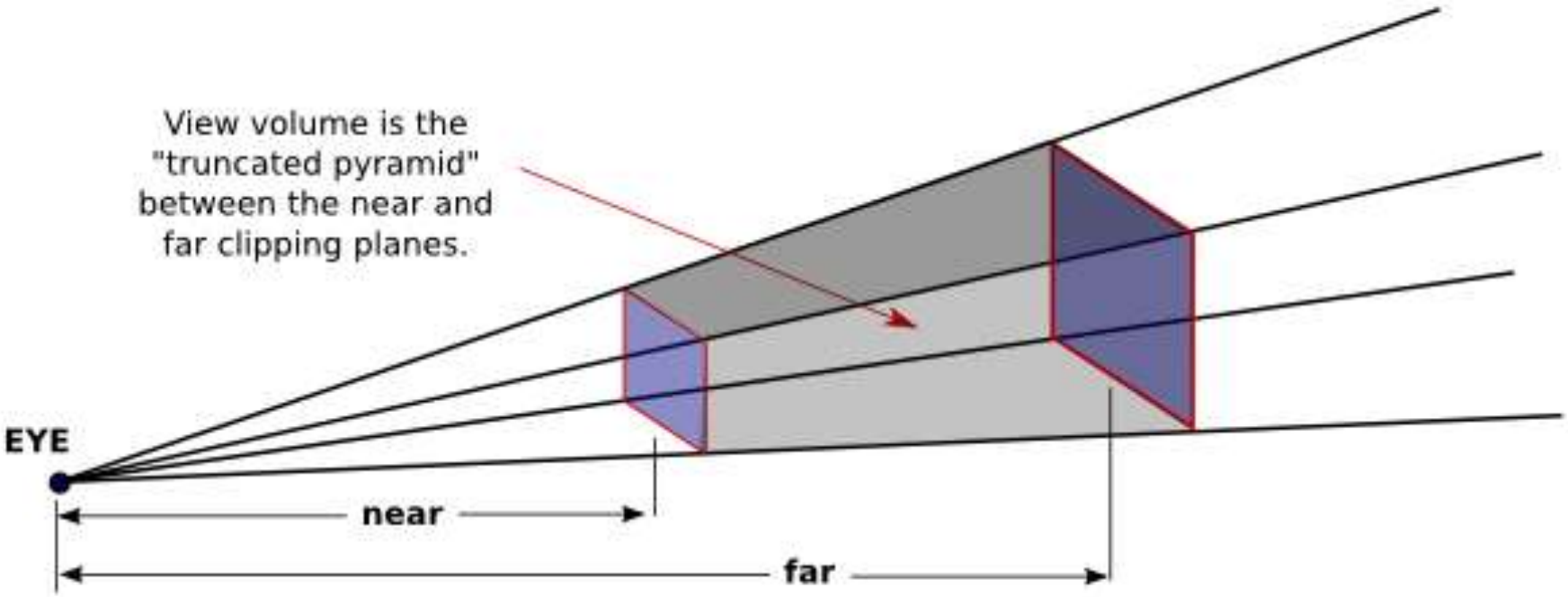
3D Clipping



3D Clipping


Near Clipping Plane
Far Clipping Plane
Discarded
Rendered
Clipped
Discarded

3D Clipping


View volume is the
"truncated pyramid"
between the near and
far clipping planes.
EYE
near
far

View Volume

- In 3D computer graphics, the view volume defines the region of space that is visible from the camera's perspective and will be projected onto the display screen.
- It acts as a spatial filter, determining which objects or parts of objects are rendered in the final image.
- Types of View Volumes:
- The shape of the view volume depends on the type of projection used:

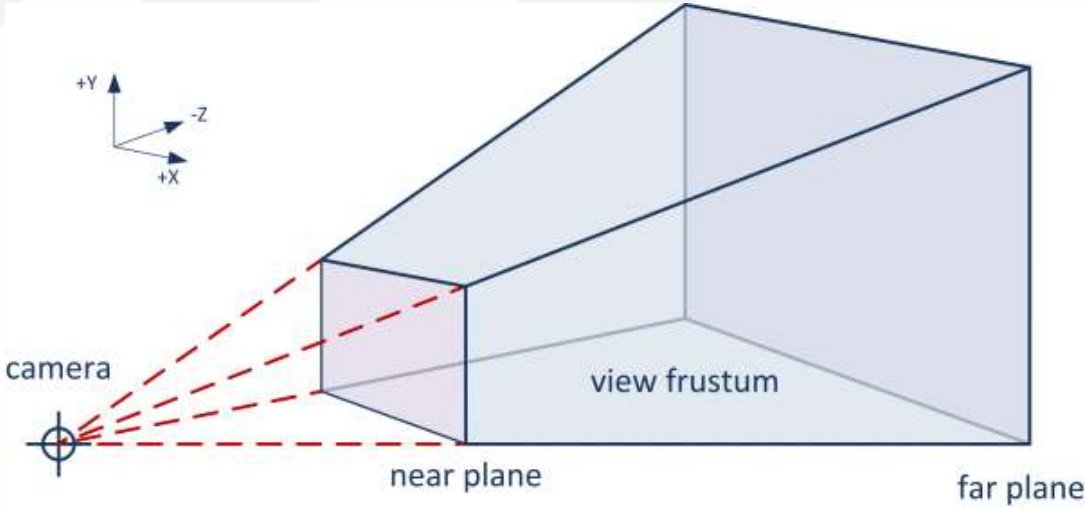
1. **Perspective Projection:**
   - *View Volume Shape:* A frustum, which resembles a truncated pyramid.
   - *Characteristics:* Objects farther from the camera appear smaller, mimicking human vision.
   - *Usage:* Commonly used in applications requiring realistic depth perception, such as simulations and video games.

View Volume

- Types of View Volumes:
- The shape of the view volume depends on the type of projection used:
  2. **Orthographic (Parallel) Projection:**
     - *View Volume Shape:* A rectangular parallelepiped (box-like structure).
     - *Characteristics:* Objects maintain their size regardless of depth, preserving true dimensions.
     - *Usage:* Ideal for technical drawings, engineering designs, and architectural plans where accurate measurements are crucial.

Components of the View Volume

- **Near Clipping Plane:** The closest boundary to the camera; objects closer than this plane are not rendered.
- **Far Clipping Plane:** The farthest boundary from the camera; objects beyond this plane are excluded from rendering.
- **Field of View (FOV):** The extent of the observable world seen at any given moment, determining the width and height of the view volume.



Normalized View Volume

- The normalized view volume (also known as the canonical view volume) is a standardized coordinate space into which the original view volume is transformed.
- This transformation simplifies subsequent rendering processes, such as clipping and projection. In this normalized space:
- Coordinates are mapped to a consistent range, typically [−1, 1] for all axes in many graphics systems.
- The complex, camera-specific view volume is converted into a uniform cube (or parallelepiped), making mathematical operations more straightforward.

View Port Clipping

- In 3D rendering, the viewport represents the rectangular area of the render target where the 3D scene is projected.
- Clipping is the process of excluding parts of objects or scenes that fall outside this viewport, ensuring that only the visible elements are processed and displayed.

Clipping Planes

- Clipping in 3D involves two primary planes:
- **Near Clipping Plane:** This plane is closest to the camera. Objects closer than this plane are not rendered.

View Port Clipping

- **Far Clipping Plane:** This plane is farther from the camera. Objects beyond this plane are also not rendered.
- Adjusting these planes helps in managing which parts of the scene are visible and can prevent rendering artifacts.
- For instance, setting the near clipping plane too close can cause nearby objects to be clipped unexpectedly, while setting the far clipping plane too far can lead to depth buffer precision issues.

View Port Clipping

Practical Applications

- 3D Modeling Software: In applications like Blender, users can adjust the viewport clipping parameters to work efficiently with scenes of varying scales. This adjustment ensures that objects are not inadvertently clipped during modeling. blender.stackexchange.com
- Game Development: Developers utilize clipping planes to optimize rendering performance. By culling objects outside the player's view or beyond a certain distance, they can reduce the computational load.
- Medical Imaging: Clipping planes allow practitioners to focus on specific sections of 3D scans, providing clearer insights into particular areas of interest.

Clipping in Homogeneous Coordinates

- Clipping in homogeneous coordinates is a method used in computer graphics to remove any part of a 3D object that is outside of the viewing frustum (the visible area of the 3D scene). This is done by transforming the 3D object into homogeneous coordinates and then applying a series of clipping planes to remove any parts of the object that are outside of the viewing frustum.
- The steps of clipping in homogeneous coordinates are as follows:
  1. Convert the original 3D coordinates of the object into homogeneous coordinates by adding a fourth dimension to the coordinates. This fourth dimension represents the weight of the vertex, and it is set to 1 for all visible vertices.

Clipping in Homogeneous Coordinates

- The steps of clipping in homogeneous coordinates are as follows:
  2. Multiply the homogeneous coordinates of the object by the transformation matrix.
     This matrix is used to project the object onto the clipping planes.
  2. Test the transformed coordinates against the clipping planes using the equation: Ax' + By' + Cz' + Dw' = 0.
     If the result is greater than zero, it means that the vertex is inside the clipping plane, and it should be kept.
     If the result is less than zero, it means that the vertex is outside the clipping plane, and it should be discarded.

DIGITAL LEARNING CONTENT

Parul® University

www.paruluniversity.ac.in