

Computer Graphics

CH3 – Basic Transformation and Clipping

Outline

- **Transformations and Clipping**
 - **Basic Transformations**
 - Translation, Scaling, Rotation
 - **Composite Transformation**
 - **Clipping Algorithms**

Introduction

- Using output primitives and their attributes, we can create variety of pictures and graphs - also a need for altering or manipulating displays.
- Changes in ***orientation***, ***size***, and ***shape*** are accomplished with ***geometric transformations*** that alter the coordinate descriptions of objects.
- The basic geometric transformations are ***translation***, ***rotation***, and ***scaling***.
- Other transformations that are often applied to objects include ***reflection*** and ***shear***.

Basic Transformations

- Here, we first discuss general procedures for applying *Translation*, *Rotation*, and *Scaling* parameters to *reposition* and *resize* two-dimensional objects.
- Then, we consider how transformation equations can be expressed in a more convenient matrix formulation that allows efficient combination of object transformations.

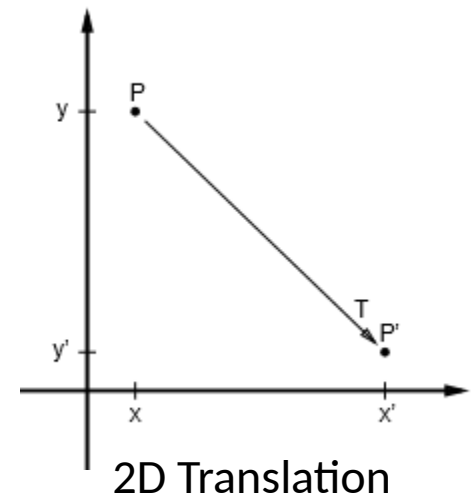
Basic Transformations

Translation

- A **translation** is applied to an object by repositioning it along a straight-line path from one coordinate location to another.
 - The point P is translated to the point P' by adding the translation distances T_x and T_y :
 - The above equations can be expressed in matrix form by:
 - So, the translation of the two dimensional vector $\begin{bmatrix} x \\ y \end{bmatrix}$ into $\begin{bmatrix} x' \\ y' \end{bmatrix}$ is given by:

where

T_x , and T_y



Basic Transformations

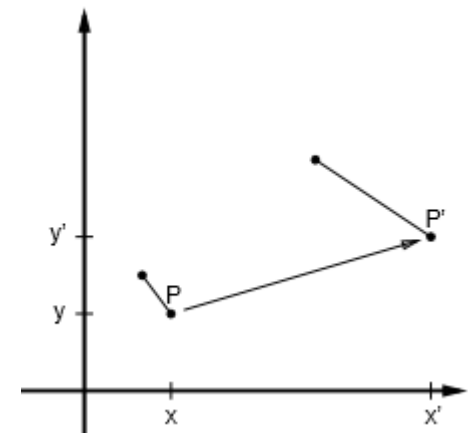
Scaling relative to origin

- A **scaling** transformation alters the size of an object by scaling factors s_x and s_y .
 - The vertex $P(x, y)$ is scaled into the vertex $P'(x', y')$ by multiplying it with the scaling factors s_x and s_y :
$$x' = s_x x$$
$$y' = s_y y$$
 - This can be expressed in matrix form by:

or

where

- If the transformation is called a **uniform scaling**.
- If it is called a **differential scaling**.



2D Scaling

Basic Transformations

Rotation relative to origin

- The point P , or in polar coordinates, is rotated anti-clockwise about the origin by θ into the point P' , or $(r, \theta + \phi)$.

– So

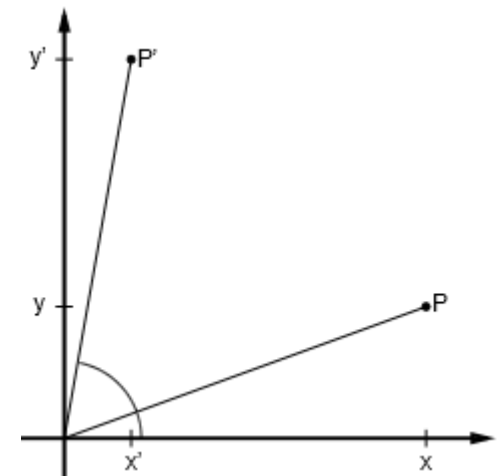
– Now,

and

– Therefore

– This can be expressed by:

where



2D Rotation

Basic Transformations

- All of the three basic transformations are ***rigid body transformations*** – that moves objects without deformation.
- That is, every point on the object is translated, scaled or rotated by the same amount.
 - A ***straight Line segment*** is *translated, scaled or rotated* by applying the transformation to each of the line endpoints and redrawing the line between the new endpoint positions.
 - ***Polygons*** are *translated, scaled or rotated* by adding the transformation vectors to the coordinate position of each vertex and regenerating the polygon using the new set of vertex coordinates and the current attribute settings.

Homogeneous Coordinates

- Many graphics applications involve sequences of **geometric transformations**.
 - E.g., An animation might require an object to be translated and rotated at each increment of the motion.
- In design and picture construction applications, we perform translations, rotations, and scaling to fit the picture components into their proper positions.
- We consider how the matrix representations can be reformulated so that such transformation sequences can be efficiently processed.

Homogeneous Coordinates

- A combination of *translations*, *rotations* and *scaling* can be expressed as:
- Because scaling and rotation are expressed as matrix multiplication but translation is expressed as matrix addition,
 - it is not, in general, possible to combine a set of operations into a single matrix operation.
- ***Composition of transformations*** is often desirable if the same set of operations have to be applied to a list of position vectors.

Homogeneous Coordinates

- We can solve this problem by representing points in *homogenous coordinates*.
- In homogenous coordinates we add a third coordinate to a point, i.e., instead of representing a point by a pair (x, y) , we represent it as (x, y, w) .
- Two sets of homogenous coordinates represent the same point if one is a multiple of the other.
 - E.g. (x, y, w) and (kx, ky, kw) represent the same point.
- Given homogenous coordinates (x, y, w) , the Cartesian coordinates correspond to $(\frac{x}{w}, \frac{y}{w})$, i.e. the homogenous coordinates at which $w = 1$.
 - Points with $w = 0$ are called points at infinity.

Homogeneous Coordinates

- Representing points as 3-dimensional vectors, we can rewrite our transformation matrices as follows:

- **Translation**

- A translation by $\begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$ is represented by:

where

- **Scaling**

- Scaling by the factors s_x, s_y, s_z relative to the origin is given by:

where

Homogeneous Coordinates

- Representing points as 3-dimensional vectors, we can re-write our transformation matrices as follows:
- **Rotation**
 - Rotation about the origin by θ is given by:

where

Composition of 2D Transformations

- Using the associative property of matrix multiplication, we can combine several transformations into a single matrix.
- It can easily be shown that:

Composition of 2D Transformations

General fixed-point Scaling

- Scaling with factors s_x and s_y and with respect to the fixed point (x_f, y_f) can be achieved by the following list of operations:
 1. Translate by $(-x_f, -y_f)$ so that the fixed point is moved to the origin;
 2. Scale by s_x and s_y with respect to the origin;
 3. Translate by (x_f, y_f) to return the fixed point back to its original position.
- So, the general fixed-point scaling transformation matrix is:

Composition of 2D Transformations

General pivot-point rotation

- A rotation about the rotation-point or pivot-point by θ is achieved as follows:
 1. Translate by $(-x_p, -y_p)$;
 2. Rotate by θ ;
 3. Translate by (x_p, y_p) .
- The general pivot-point rotation matrix is therefore given by:

Composition of 2D Transformations

Inverse transformations

- The matrix of an inverse transformation is the inverse matrix of the transformation:
 - If then
 - It can easily be shown that:

Clipping Algorithms

- Usually only a specified region of a picture needs to be displayed.
 - This region is called the **clip window**.
- An algorithm which displays only those primitives (or part of the primitives) which lie inside a clip window is referred to as a ***clipping algorithm***.

Clipping Points

- A point lies inside the rectangular clip window if and only if the following inequalities hold:

Clipping Algorithms

Line Clipping

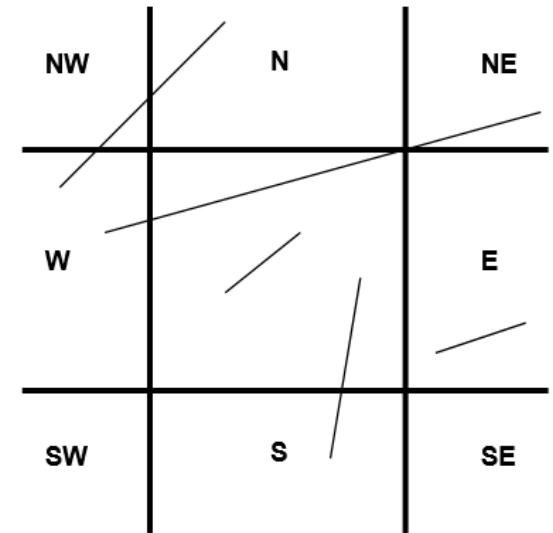
The Cohen-Sutherland algorithm for line clipping

- Clipping a straight line against a rectangular clip window results in either:
 1. A line segment whose endpoints may be different from the original ones, or
 2. Not displaying any part of the line. This occurs if the line lies completely outside the clip window.
- The Cohen-Sutherland line clipping algorithm considers each endpoint at a time and truncates the line with the clip window's edges until the line can be trivially accepted or trivially rejected.
- A line is trivially rejected if both endpoints lie on the same outside half-plane of some clipping edge

Clipping Algorithms

The Cohen-Sutherland algorithm for line clipping

- The plane is partitioned into nine segments by extending the clip window's edges.
- Each segment is assigned a 4-bit code according to where it lies with respect to the clip window:



- For example bit 2 is set if and only if the region lies to the south of (below) the clip window.

Clipping Algorithms

The Cohen-Sutherland algorithm for line clipping

- The Cohen-Sutherland algorithm starts by assigning an **outcode** to the two line endpoints (say and).
 - If both *outcodes* are () the line lies entirely in the clip window and is thus trivially accepted.
 - If the two *outcodes* have at least one bit in common (), then they lie on the same side of the window and the line is trivially rejected.
 - If a line cannot be accepted or rejected, it is then truncated by an intersecting clip edge and the previous steps are repeated.

Clipping Algorithms

The Cohen-Sutherland algorithm for line clipping

Algorithm:

Step 1: Assign a region code for each end points.

Step2: If both endpoints have a region code 0000 then accept the line. Else apply Step 3.

Step3: Perform the logical AND operation for both region codes.

– If the result is not 0000, then reject the line.

– Else clip the line by:

- Choose an endpoint of the line that is outside the window.
- Find the intersection point at the window boundary (based on region code).
- Replace endpoint with the intersection point and update the region code.

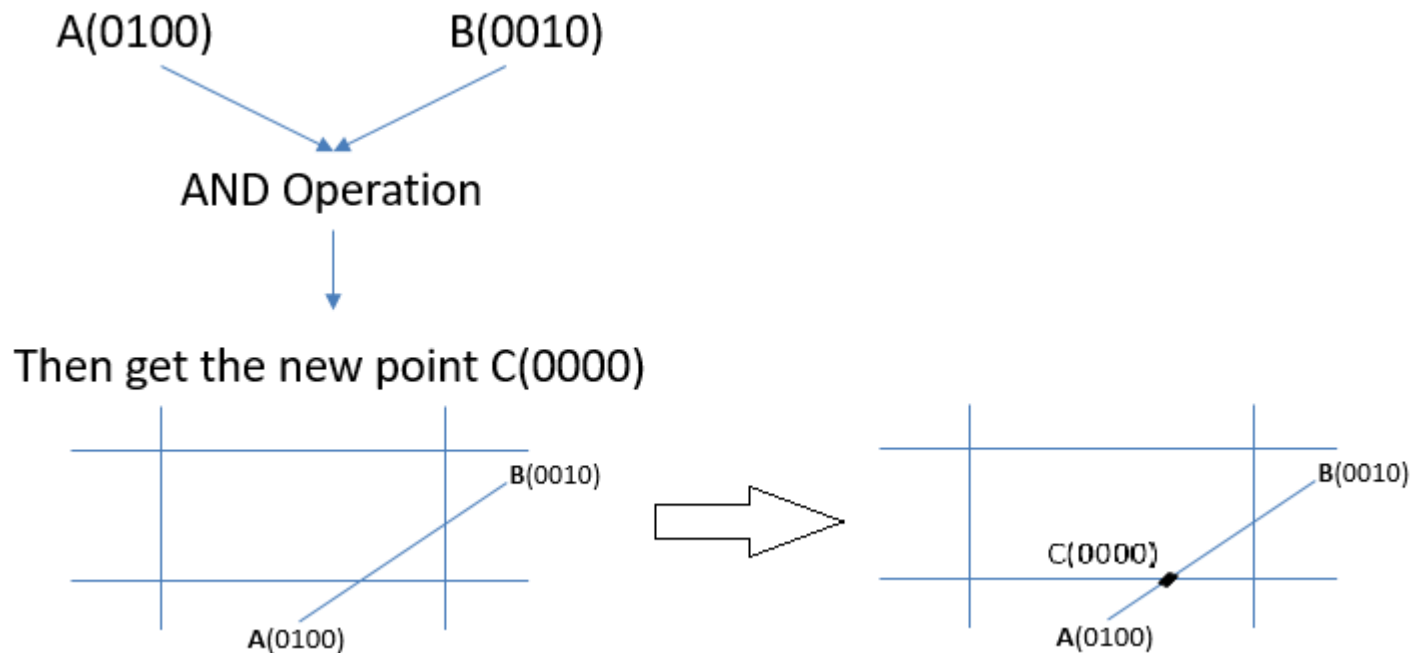
Step 4: Repeat Step 2 until we find a clipped line either trivially accepted or rejected.

Step 5: Repeat Step 1 for other lines.

Clipping Algorithms

The Cohen-Sutherland algorithm for line clipping

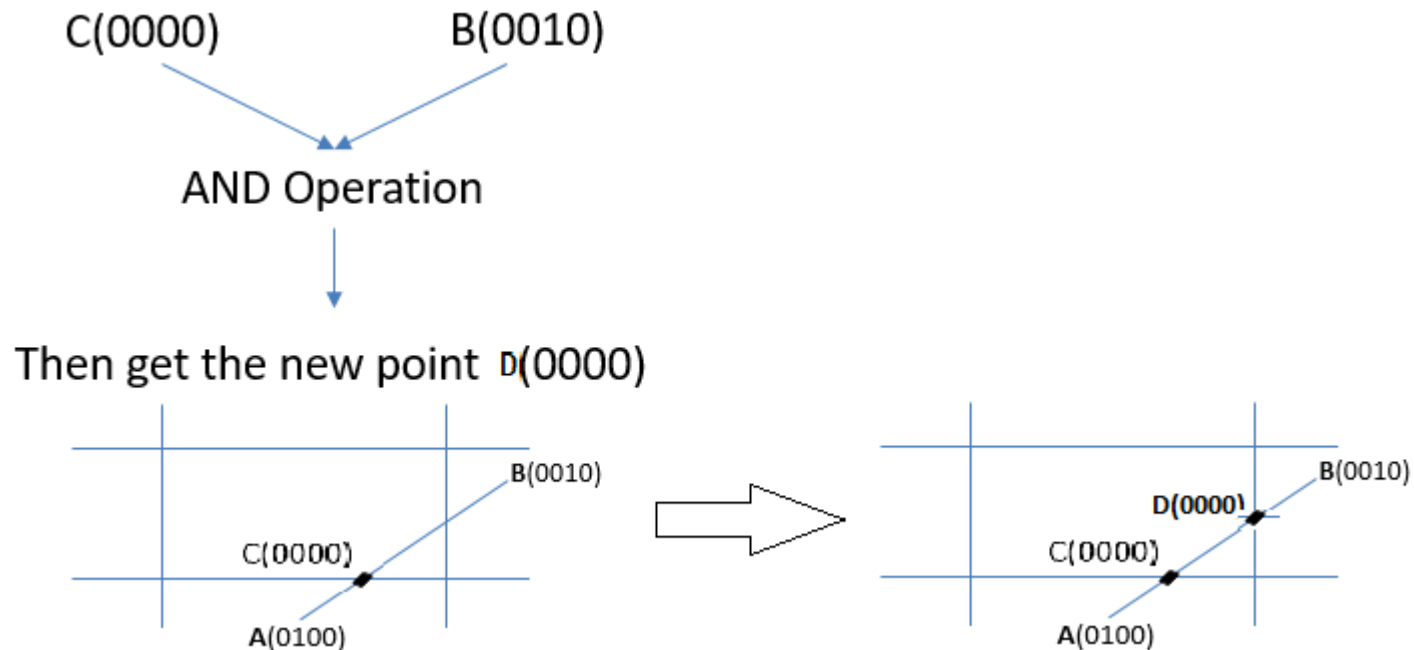
Example:



Clipping Algorithms

The Cohen-Sutherland algorithm for line clipping

Example:



Then we have the final line after clipping CD .

Clipping Algorithms

Polygon Clipping

Sutherland-Hodgman polygon clipping algorithm

- A polygon can be clipped against a rectangular clip window by considering each clip edge at a time.

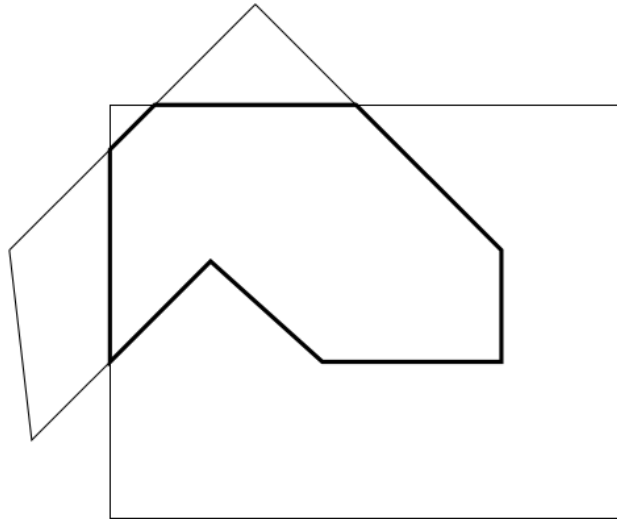


Clipping Algorithms

Polygon Clipping

Sutherland-Hodgman polygon clipping algorithm

- For filled polygons, a new polygon boundary must be computed.



- At each clipping stage, a new polygon is created by removing the outside vertices and inserting the intersections with the clip boundary.

Clipping Algorithms

Polygon Clipping

Sutherland-Hodgman polygon clipping algorithm

Clipping against an edge

- Given the polygon , we can create a new polygon by
 - considering the vertices in sequence and
 - deciding which ones (and any intersections with the edge) have to be inserted into the clipped polygon vertex list.
- Suppose that has been processed in the previous step.
- There are four possible cases which need to be considered while processing the next vertex .

Clipping Algorithms

Polygon Clipping

Sutherland-Hodgman polygon clipping algorithm

Clipping against an edge

1. If P_1 is outside the window and P_2 is inside, then the intersection of the line P_1P_2 with the clipping edge, and the vertex P_2 have to be inserted into the new polygon list.
2. If P_1 and P_2 are both outside the clip window then no point need be inserted.
3. If both P_1 and P_2 are inside the window, P_2 is inserted into the list, and finally.
4. If P_1 is inside the window and P_2 is outside, only the intersection of the line P_1P_2 with the clip boundary is inserted.

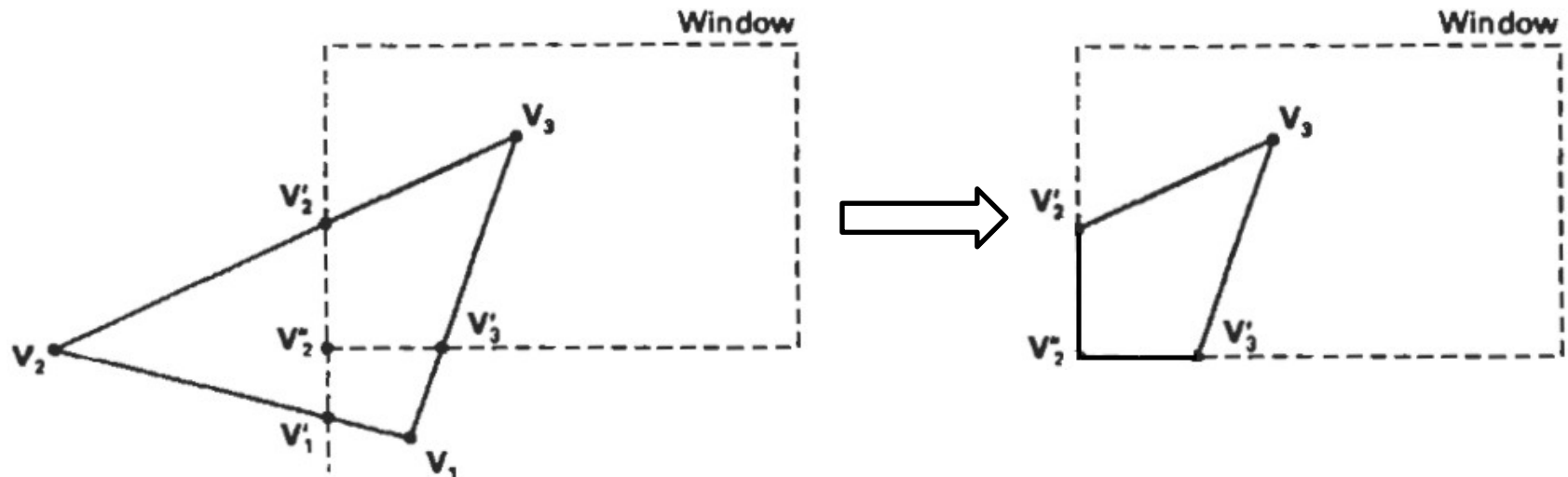
Clipping Algorithms

Polygon Clipping

Sutherland-Hodgman polygon clipping algorithm

Clipping against an edge

- **Example:** Processing the vertices of the polygon below through a boundary-clipping pipeline.



- After all vertices are processed through the pipeline, the vertex list for the clipped polygon is $[, , ,]$.

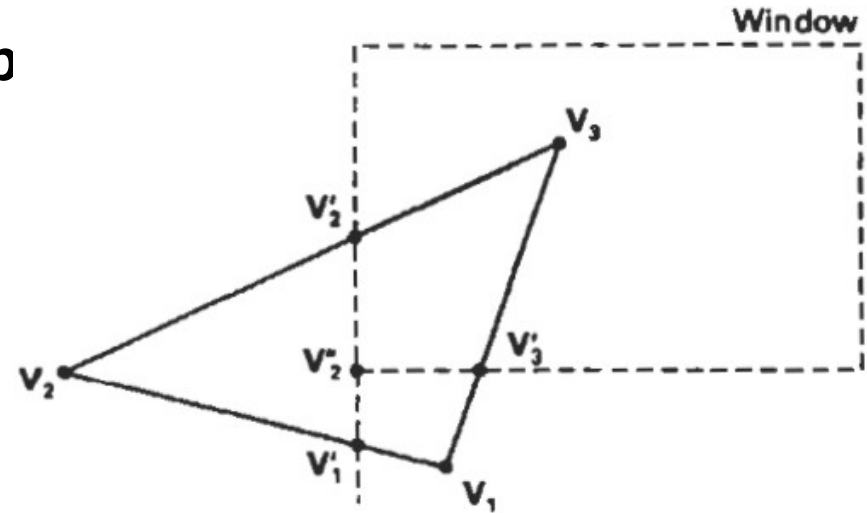
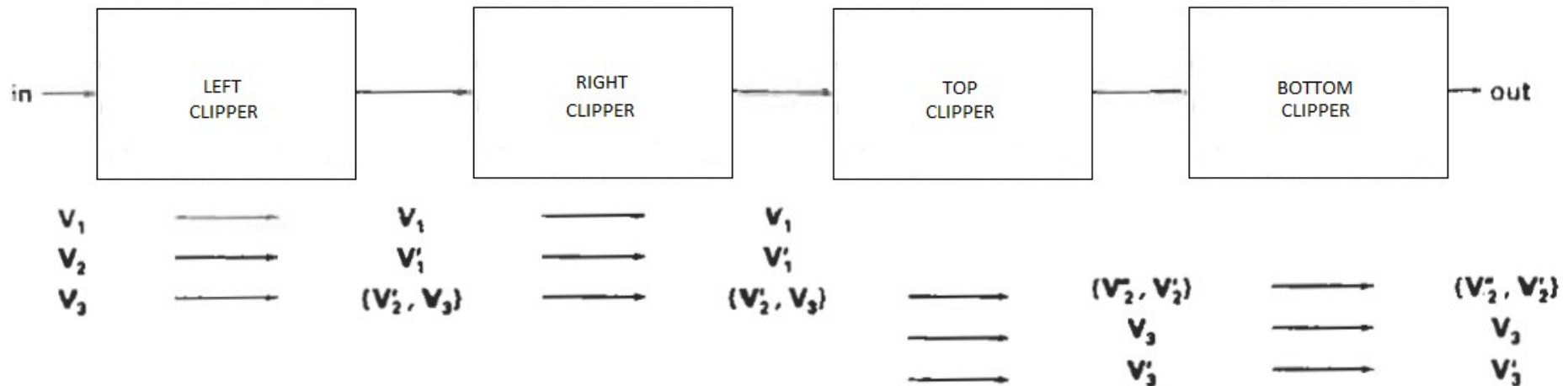
Clipping Algorithms

Polygon Clipping

Sutherland-Hodgman polygon clip

Clipping against an edge

- Example: cont....



Thank You!!!

End of CH3!

Computer Graphics

CH4 – Windows and Viewports

Outline

- **Windows and Viewports**
 - **Co-ordinates Representation**
 - **Viewing Pipeline**
 - **Window to Viewport Transformation**
 - **Zooming and Panning effects with windows and viewports**