SCS1302 COMPUTER GRAPHICS & MULTIMEDIA SYSTEMS

(Common to CSE & IT)

UNIT II 2D TRANSFORMATIONS AND VIEWING

Basic two dimensional transformations – Other transformations – 2D and 3D viewing – Line clipping – Polygon clipping – Logical classification – Input functions – Interactive picture construction techniques.

2-Dimensional Transformations

The Basic Transformations:

1. Translation
2. Scaling
3. Rotation

Other Transformations:

1. Reflection
2. Shearing

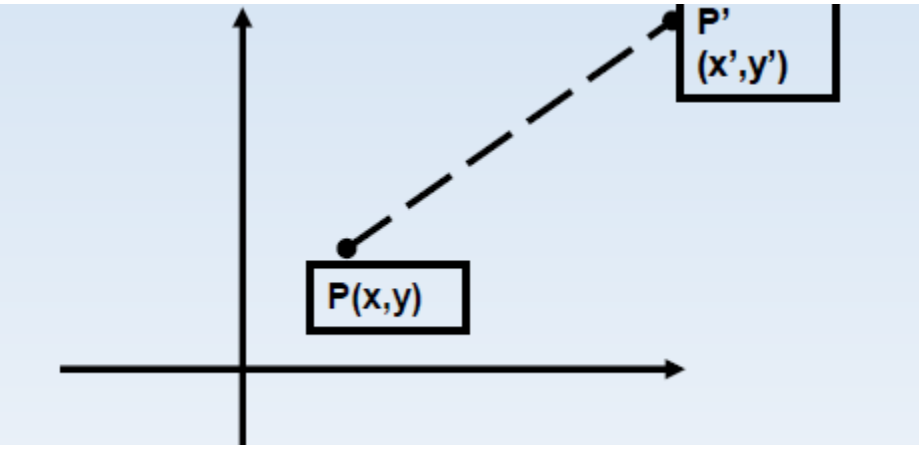
Translations

Displacement of an object in a given distance and direction from its original position.

- Rigid body transformation that moves object without deformation
- Initial Position point P (x, y)
- The new point P' (x', y')

  where

  x' = x + tx , y' = y + ty , tx and ty is the displacement in x and y respectively.

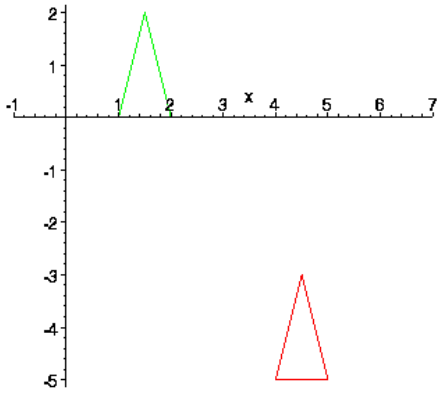


The translation pair (tx, ty) is called a translation vector or shift vector

Problem:

- Assume you are given a point at (x,y)=(2,1). Where will the point be if you move it 3 units to the right and 1 unit up? Ans: (x',y') = (5,2). How was this obtained? - (x',y') = (x+3,y+1). That is, to move a point by some amount *dx* to the right and *dy up*, you must add *dx* to the x-coordinate and add *dy* to the y-coordinate.
- What was the required transformation to move the green triangle to the red triangle? Here the green triangle is represented by 3 points

triangle = { p1=(1,0), p2=(2,0), p3=(1.5,2) }



Matrix/Vector Representation of Translations

A translation can also be represented by a pair of numbers, $t=(t_x,t_y)$ where t_x is the change in the x-coordinate and t_y is the change in y coordinate. To translate the point p by t, we simply add to obtain the new (translated) point

$$p' = p + t.$$

$$\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} tx \\ ty \end{bmatrix} = \begin{bmatrix} x + tx \\ y + ty \end{bmatrix}$$

Rotation

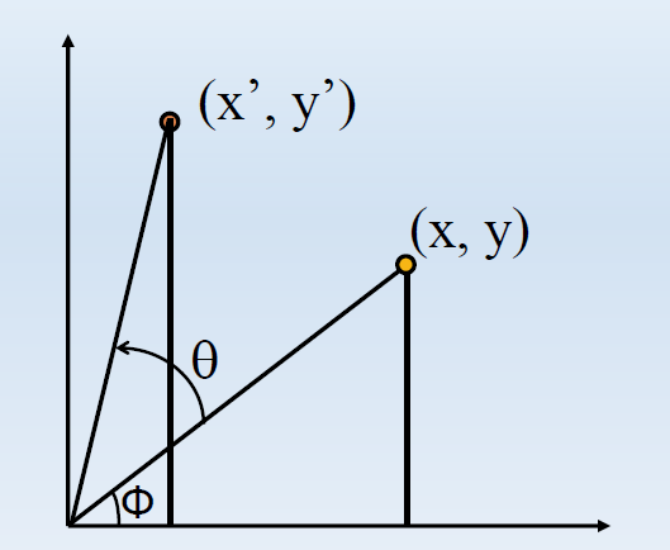
Rotation is applied to an object by repositioning it along a circular path in the xy plane.

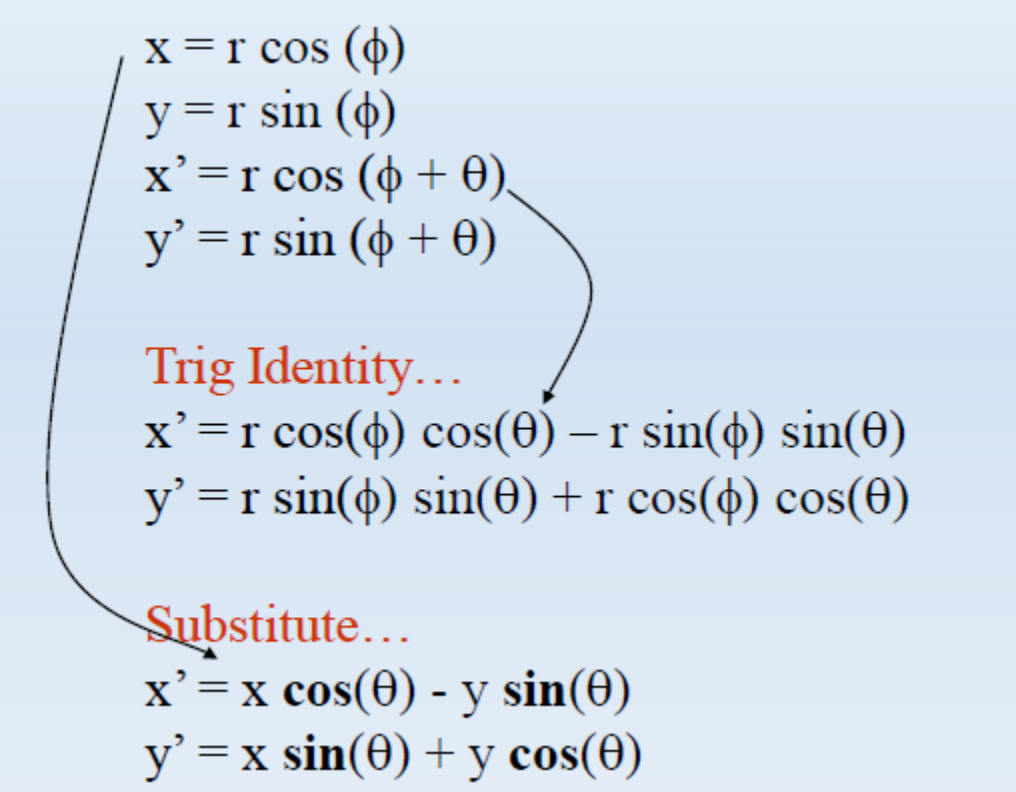
To generate a rotation, we specify

- Rotation angle θ
- Pivot point (xr , yr)

Positive values of θ for counterclockwise rotation

Negative values of θ for clockwise rotation.





Matrix Representation of

P'=R.P

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

Scaling

- Scaling alters the size of an object.
- Operation can be carried out by multiplying each of its components by a scalar
- Uniform scaling means this scalar is the same for all components

- Non-uniform scaling: different scalars per component

$$x' = x * sx$$

$$y' = y * sy$$

In matrix form:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} sx & 0 \\ 0 & sy \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

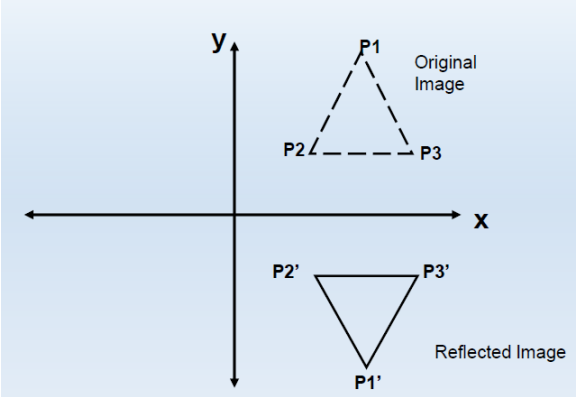
Reflection

A reflection is a transformation that produces a mirror image of an object

Generated relative to an axis of reflection

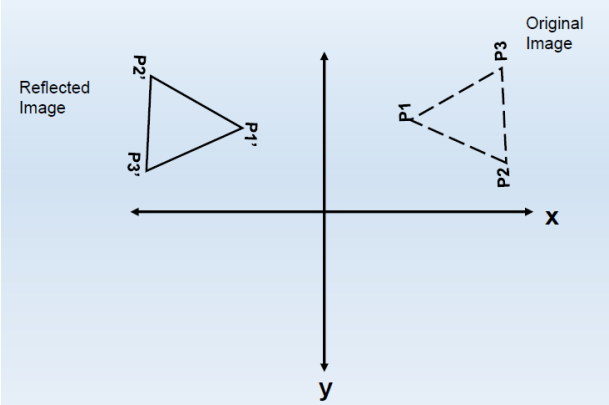
1. Reflection along x axis
2. Reflection along y axis
3. Reflection relative to an axis perpendicular to the xy plane and passing through the coordinate origin
4. Reflection of an object relative to an axis perpendicular to the xy plane and passing through point P
5. Reflection of an object with respect to the line y=x.

Reflection about x-axis:



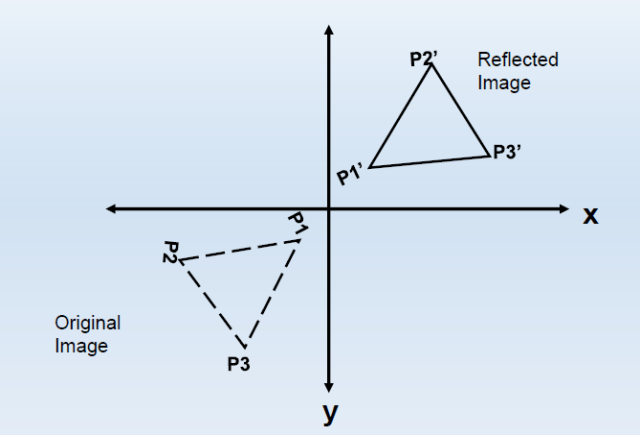
$$M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Reflection about y-axis:



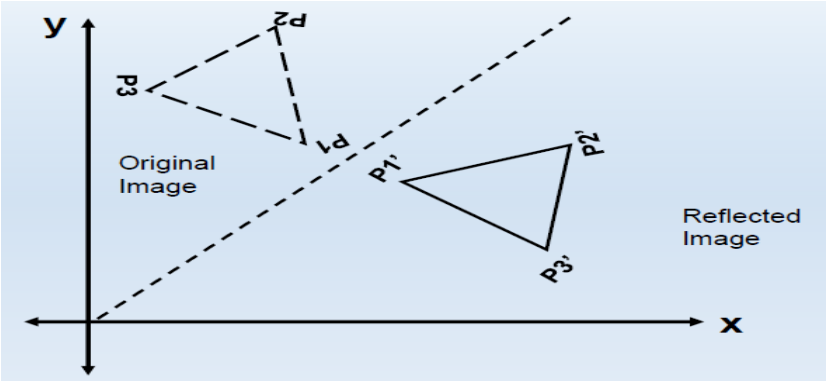
$$M = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Reflection relative to an axis perpendicular to the xy plane and passing through the coordinate origin:



$$M = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Reflection of an object with respect to the line y=x



$$M = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Shearing

A transformation that distorts the shape of an object such that the transformed object appears as if the object were composed of internal layers that had been caused to slide over each other.

Shear relative to the x-axis

$$\begin{bmatrix} 1 & sh_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

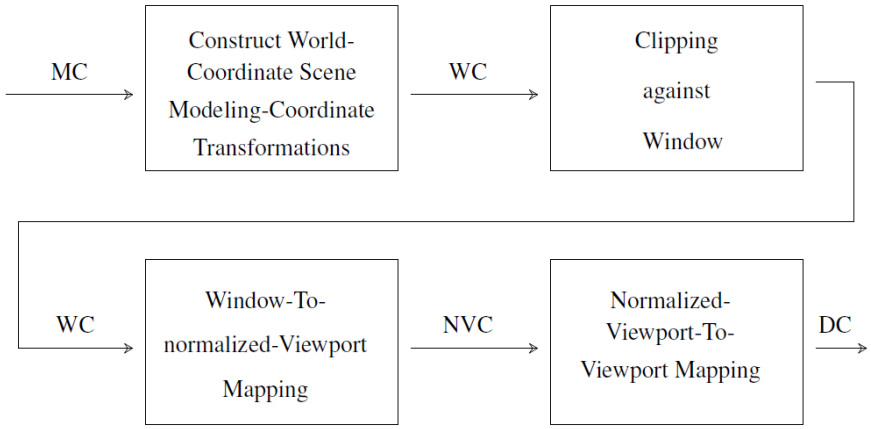
Shear relative to the y-axis

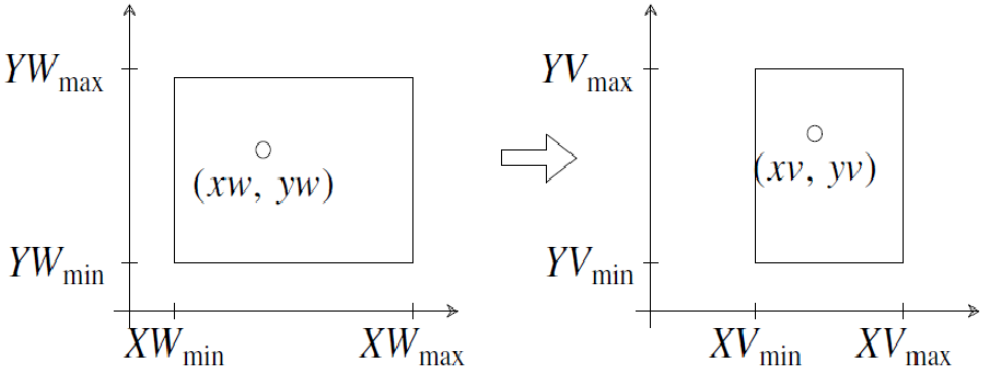
$$\begin{bmatrix} 1 & sh_y & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

2D VIEWING

The mapping of a 2D world coordinate system to device coordinates is called a two-dimensional viewing transformation.

The **clipping window** is the section of the 2D scene that is selected for viewing.
The **display window** is where the scene will be viewed.
The **viewport** controls the placement of the scene within the display window
A window-viewport transformation describes the mapping of a (rectangular) window in one coordinate system into another (rectangular) window in another coordinate system. This transformation is defined by the section of the original image that is transformed (clipping window), the location of the resulting window (viewport), and how the window is translated, scaled or rotated.





$$xv = xv_{\min} + (xw - xw_{\min})sx$$

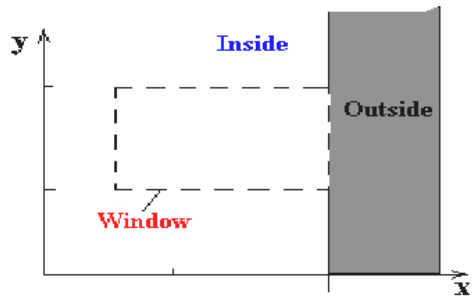
$$yv = yv_{\min} + (yw - yw_{\min})sy$$

sx and sy are scaling factors

$$sx = \frac{xv_{\max} - xv_{\min}}{xw_{\max} - xw_{\min}}, \qquad sy = \frac{yv_{\max} - yv_{\min}}{yw_{\max} - yw_{\min}}$$

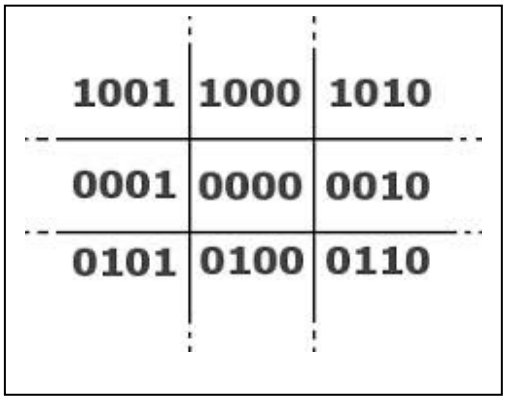
COHEN-SUTHERLAND LINE CLIPPING ALGORITHM

The Cohen-Sutherland line clipping algorithm quickly detects and dispenses with two common and trivial cases. To clip a line, we need to consider only its endpoints. If both endpoints of a line lie inside the window, the entire line lies inside the window. It is trivially accepted and needs no clipping. On the other hand, if both endpoints of a line lie entirely to one side of the window, the line must lie entirely outside of the window. It is trivially rejected and needs to be neither clipped nor displayed.



Inside-Outside Window Codes

To determine whether endpoints are inside or outside a window, the algorithm sets up a half-space code for each endpoint. Each edge of the window defines an infinite line that divides the whole space into two half-spaces, the inside half-space and the outside half-space, as shown below.



For any endpoint (x , y) of a line, the code can be determined that identifies which region the endpoint lies. The code's bits are set according to the following conditions:

- First bit set **1** : Point lies to **left** of window $\mathbf{x} < \mathbf{x_{min}}$
- Second bit set **1** : Point lies to **right** of window $\mathbf{x} > \mathbf{x_{max}}$
- Third bit set **1** : Point lies below(**bottom**) window $\mathbf{y} < \mathbf{y_{min}}$
- fourth bit set **1** : Point lies above(**top**) window $\mathbf{y} > \mathbf{y_{max}}$

The sequence for reading the codes' bits is LRBT (Left, Right, Bottom, Top).

Once the codes for each endpoint of a line are determined, the logical AND operation of the codes determines if the line is completely outside of the window. If the logical AND of the endpoint codes is not zero, the line can be trivally rejected. For example, if an endpoint had a code of 1001 while the other endpoint had a code of 1010, the logical AND would be 1000 which indicates the line segment lies outside of the window. On the other hand, if the endpoints had codes of 1001 and 0110, the logical AND would be 0000, and the line could not be trivally rejected.

The logical OR of the endpoint codes determines if the line is completely inside the window. If the logical OR is zero, the line can be trivally accepted. For example, if the endpoint codes are 0000 and 0000, the logical OR is 0000 - the line can be trivally accepted. If the endpoint codes are 0000 and 0110, the logical OR is 0110 and the line can not be trivally accepted.

Algorithm

The Cohen-Sutherland algorithm uses a divide-and-conquer strategy. The line segment's endpoints are tested to see if the line can be trivally accepted or rejected. If the line cannot be trivally accepted or rejected, an intersection of the line with a window edge is determined and the trivial reject/accept test is repeated. This process is continued until the line is accepted.

To perform the trivial acceptance and rejection tests, we extend the edges of the window to divide the plane of the window into the nine regions. Each end point of the line segment is then assigned the code of the region in which it lies.

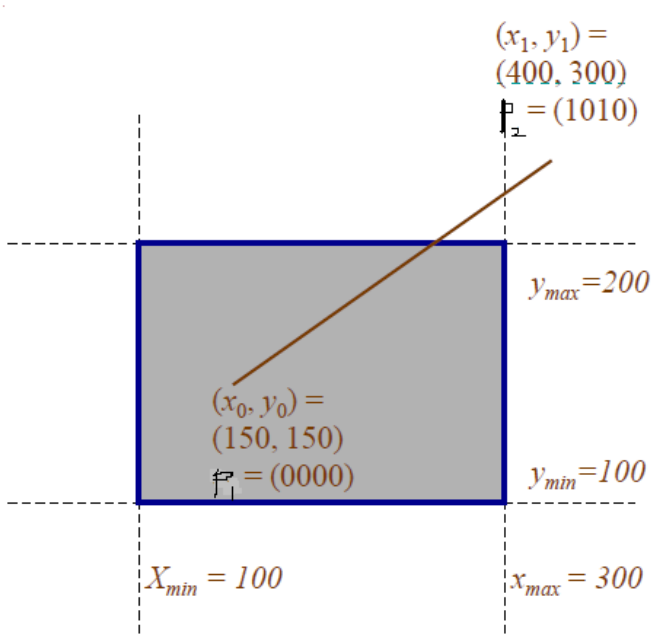
1. Given a line segment with endpoint p1=(x0,y0) and p2=(x1,y1)
2. Compute the 4-bit codes for each endpoint.

   If both codes are 0000,(bitwise OR of the codes yields 0000) line lies completely inside the window: pass the endpoints to the draw routine.

   If both codes have a 1 in the same bit position (bitwise AND of the codes is not 0000), the line lies outside the window. It can be trivially rejected.

3. If a line cannot be trivially accepted or rejected, at least one of the two endpoints must lie outside the window and the line segment crosses a window edge. This line must be clipped at the window edge before being passed to the drawing routine.
4. Examine one of the endpoints, say p1=(x0,y0) . Read P_1's 4-bit code in order: Left-to-Right, Bottom-to-Top.
5. When a set bit (1) is found, compute the intersection I of the corresponding window edge with the line from P_1 to P_2. Replace P_1 with I and repeat the algorithm.

Example illustration



For p1(150,150),

(From Right to left)

First bit: x – xmin = 150 – 100 = 50 = set the bit to 0

Second bit: xmax – x = 300 – 150 = 150 =set the bit to 0

Third bit: y – ymin = 150 – 100 =50 = set the bit to 0

Fourth bit: ymax – y = 200 – 150 = 50 = set the bit to 0

So the region code for P1 is 0000

For p2(400,300),

(From Right to left)

First bit: x – xmin = 400 -100 = 300 = set the bit to 0

Second bit: xmax – x = 300 – 400 = - 100 = set the it to 1

Third bit: y – ymin = 300 – 300=0= set the bit to 0

Fourth bit: ymax – y = 200 – 300 = -100 = set the bit to 1.

So the region code for P2 is 1010

P1 is inside and p2 is in the top and above region code

To find the intersection point:

$$y = y_1 + m\,(x_{boundary} - x_1)$$

$$x = x_1 + (y_{boundary} - y_1) / m$$

Where $x_{boundary}$ is set either to xmin or xmax

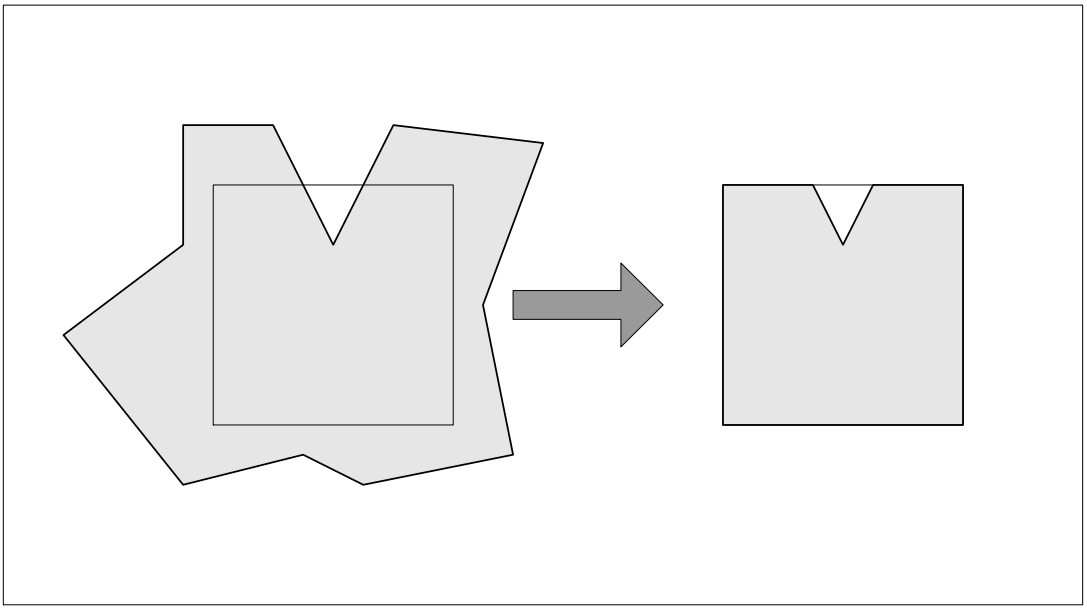
Where $y_{boundary}$ is set either to ymin or ymax

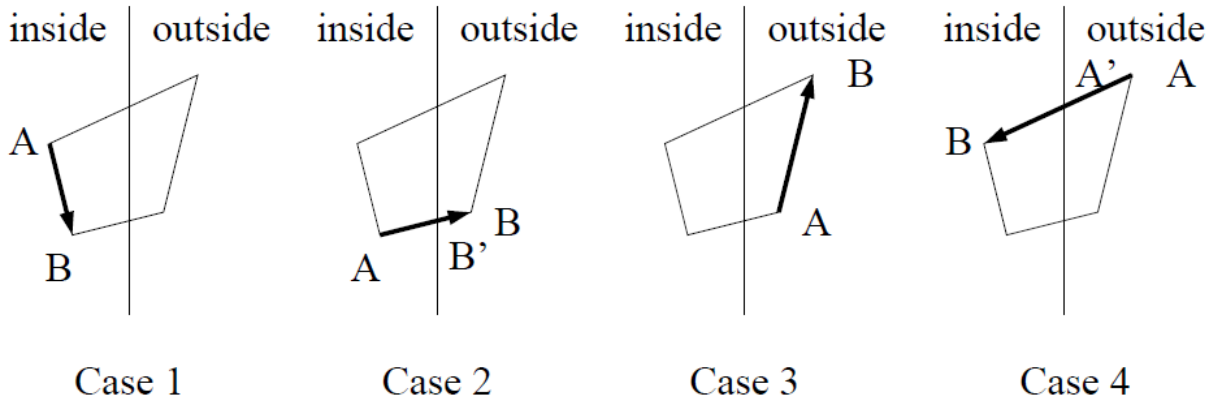
POLYGON CLIPPING

An algorithm that clips a polygon is rather complex. Each edge of the polygon must be tested against each edge of the clipping window, usually a rectangle. As a result, new edges may be added, and existing edges may be discarded, retained, or divided. Multiple polygons may result from clipping a single polygon. We need an organized way to deal with all of these cases.

SUTHERLAND HODGEMAN POLYGON CLIPPING

The Sutherland and Hodgman's polygon clipping algorithm is used in this tool. This algorithm is based on a divide-and-conquer strategy that solves a series of simple and identical problems that, when combined, solve the overall problem. The simple problem is to clip a polygon against a single infinite clipping edge. This process outputs the series of vertices that define the clipped polygon. Four clipping edges, each defining one boundary of the clipping window, are used to successively to fully clip the polygon.



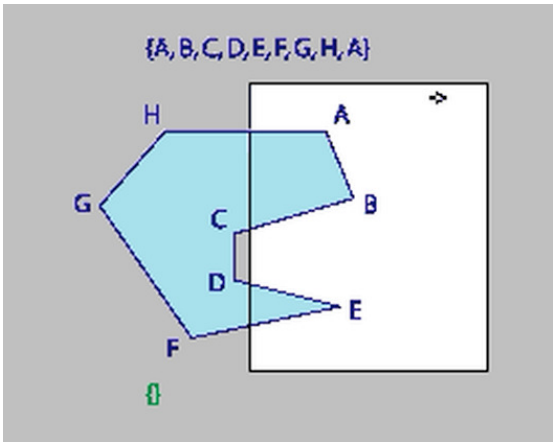


Assuming vertex A has already been processed,

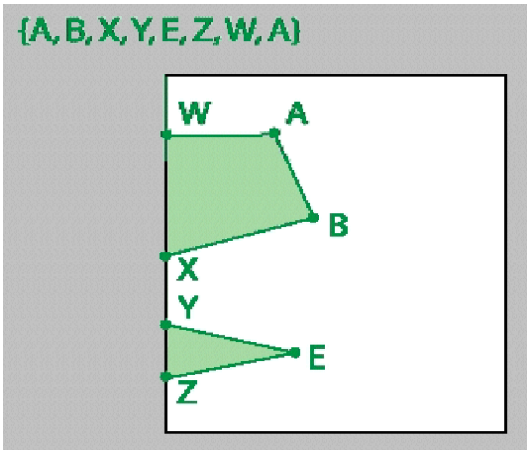
Case 1 — vertex B is added to the output list
Case 2 — vertex B' is added to the output (edge AB is clipped to AB')
Case 3 — no vertex added (segment AB clipped out)

Case 4 — vertices A' and B are added to the output

Sample Polygon

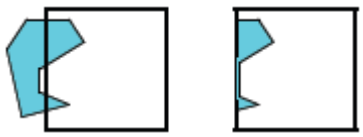


After Clipping



Drawback of Sutherland Hodgeman Algorithm:

Clipping of the concave polygon → Can produce two CONNECTED areas



LOGICAL CLASSIFICATION OF INPUT DEVICES

To make the graphical package independent of the hardware devices.

Input functions are designed based on the type of input data.

- Locator
- Stroke
- Valuator
- choice
- pick

Locator Devices:

- To select a co-ordinate on the screen.
- When the cursor is at the desired position on the screen, a button is clicked to select that co-ordinate point.
- Mouse, joystick, trackballs, space balls, digitizers.. Can be used as locator devices.
- Keyboard- along with the four , additional four buttons are provided to move horizontally.
- Keyboard along with the above can also be used

Stroke Device:

- A sequence of co-ordinate points can be selected.
- Locator devices in continuous mode.
- Graphical tablet or digitizer can also be used.

Valuator Devices:

- To give scalar input values like temperature and voltage levels.
- Floating point numbers within any range can be given as input
- E.g.: Set of Control dials.
- Rotation in one direction increases the values..
- Slide-potentiometers, keyboards with a set of numeric keys can also be used.
- Display sliders, buttons, rotating scales and menus on screen.

Choice Devices:

- To construct a picture.
- Selection from a list or menu of alternatives.
- Either a keyboard or stand-alone “button box” can be used for selecting item from the menu.
- The selected screen position(x,y) is compared with the menu option.

- Touch panels are also used.

Pick Devices:

- To select parts of the screen that are to be edited.
- Choice devices can be used here.
- If the selected point lies in the bounding rectangle of a particular object, then that'll be selected.
- If it lies in the area of 2 objects, then the squared distance from the point to the line segment both the objects are compared.

(Diagrams needed)

When triggered, input devices return information (their measure) to the system
Mouse - returns position information
Keyboard - returns ASCII code

The different modes are:

Request mode:

- The measure of the device is not returned to the program until the device is triggered

Sample mode

- Measure is returned immediately after the function is called in the user program (device sample).
- No trigger needed
- Useful in apps where the program guides the user

Event mode

- Can handle multiple inputs
- When device triggered an event is generated
- Identifier for device placed in the event queue
- Event queue process is independent of the application, asynchronous

The input class function can be given by:

set…Mode(ws,deviceCode,inputMode,echoFlag)

ASSIGNMENT OF INPUT-DEVICE CODES

Device Code	Physical Device Type
1	Keyboard
2	Graphics Tablet
3	Mouse
4	Joystick
5	Trackball
6	Button

setLocatorMode(1,2,sample,noecho)

setTextMode(2,1,request,echo)

setPickMode(4,3,event,echo)

The Request mode can be given by:

request…(ws,deviceCode,status, …)

Locator and Stroke Input in Request Mode:

The request functions for these two logical input classes are:

requestLocator (ws, devCode, status, viewIndex, pt)

requestLocator (ws, devCode, nMax, status, viewIndex, n, pts)

Where nmax is the maximum number of points that can go in the input list

viewIndex is assigned a two dimensional view index number. Lower this value higher is the priority.

Interactive Picture Construction Techniques

Interactive picture- construction methods are commonly used in variety of applications, including design and painting packages. These methods provide user with the capability to position objects, to constrain fig. to predefined orientations or alignments, to sketch fig., and to drag objects around the screen. Grids, gravity fields, and rubber band methods are used to aid in positioning and other picture construction operations. The several techniques used for interactive picture construction that are incorporated into graphics packages are:

(1) Basic positioning methods:- coordinate values supplied by locator input are often used with positioning methods to specify a location for displaying an object or a character string. Coordinate positions are selected interactively with a pointing device, usually by positioning the screen cursor.

(2) constraints:-A constraint is a rule for altering input coordinates values to produce a specified orientation or alignment of the displayed coordinates. the most common constraint is a horizontal or vertical alignment of straight lines.

(3) Grids:- Another kind of constraint is a grid of rectangular lines displayed in some part of the screen area. When a grid is used, any input coordinate position is rounded to the nearest intersection of two grid lines.

(4) Gravity field:- When it is needed to connect lines at positions between endpoints, the graphics packages convert any input position near a line to a position on the line. The conversion is accomplished by creating a gravity area around the line. Any related position within the gravity field of line is moved to the nearest position on the line. It illustrated with a shaded boundary around the line.

(5) Rubber Band Methods:- Straight lines can be constructed and positioned using rubber band methods which stretch out a line from a starting position as the screen cursor.

(6) Dragging:- This methods move object into position by dragging them with the screen cursor.

(7) Painting and Drawing:- Cursor drawing options can be provided using standard curve shapes such as circular arcs and splices, or with freehand sketching procedures. Line widths, line styles and other attribute options are also commonly found in painting and drawing packages.

(DIAGRAMS-REFER BOOK)