UNIT - III

Interactive Graphics

Introduction :→

+ Most display terminals provide the user with an alphanumeric keyboard. For some appⁿs, the keyboard is inconvenient or inadequate.

E.g - The user may wish to indicate one of a no. of symbols on the screen, in order to erase the symbol. ~~If each~~ he can do so by pointing at the symbol

- The user may similarly want to find the posⁿ of an item on the screen relative to other.

Thus pointing at item already on the screen & positioning new item are required for interactive systems.

The requirement of these two oprⁿ required & led to the development of a lot of input devices.

+ Ideally a graphical input device should lend itself both to pointing & to positioning. In reality such diversity is rare. Most devices are much better in positioning than at pointing. Fortunately, these devices by software & in this way hardware system can provide both of these capabilities.

+ Another important difference is b/w the devices that can be used on the screen & the devices that cannot be used on the screen. ~~The latter might appear to be less useful.~~ The later device visual feedback the [x] & [y] o/p of the device. to control the movement of a small cross known as [cursor] displayed on the screen.

+ The use of visual feedback has an additional advantage just as in any control system. It compensates for any lack of linearity in the device.

The linear device is one that that faithfully increases

+ The cursor however can be controlled quite easily even if the device behaves in fairly non-linear fashion.

e.g. the device may be much less sensitive near the left hand region of its travel:

- a 1-inch hand movement may change the x value by only 50 units
- whereas the same movement elsewhere may change x by 60 units.

This phenomenon has allowed simple, inexpensive devices like the mouse to be used very successfully for graphical Input.

UNIT - III

INTERACTIVE GRAPHICS

Pointing and positioning devices

- cursors
- light pens
- digitizing tablet
- the mouse
- track balls.

★ CURSORS :-

- The principle use of a cursor is to indicate loc^n on the screen i.e it is used as a locator.
- However, the shape of the cursor can also be used to indicate not only the location but the oper^n available, by clicking a 'mouse' button, e.g
  - → circle with a tail surrounding a plus sign – to indicate magnification
  - → hour glass or watch – used to indicate 'wait state' while the app^n is processing a request

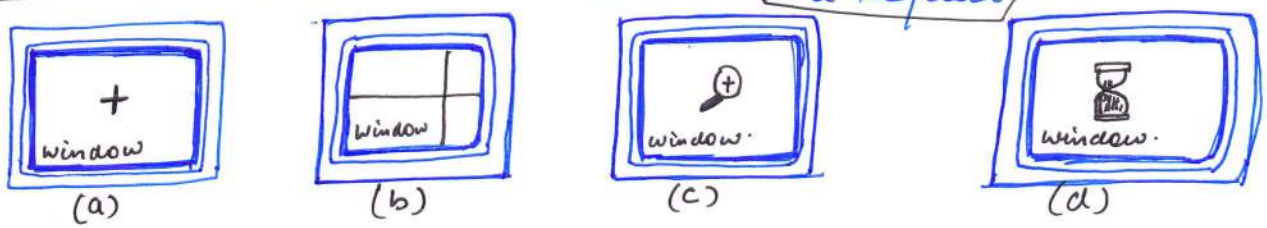


Fig 1 Cursor shapes. (a) Simple plus (b) Expanded plus for alignment (c) magnification icon (d) hour glass icon indicating wait

- Such cursors are iconic in nature i.e they are symbols used to convey complex information.
- Although a cursor is normally a small symbol, it can be expanded to full or nearly full screen / window size to aid in accomplishing interactive tasks.
- e.g if a simple upright cross or plus sign (+) – used to indicate pos^n expanding it to full screen/ windowsize is effective in

★ LIGHT PEN :→

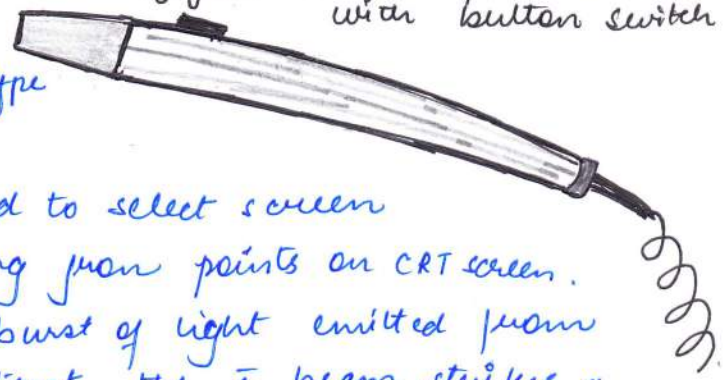


fig 2 A Light pen activated with button switch

- fig 2 shows the design of one type of light pen.
- Such pencil shaped devices are used to select screen pos^{n}s by detecting the light coming from points on CRT screen.
- They are sensitive to the short burst of light emitted from the phosphor coating at the instant the $\bar{e}$ beam strikes a particular point.
- An activated light pen, pointed at the spot on the screen as the $\bar{e}$ beam lights up that spot, generates an electrical pulse that causes the coordinate pos^n of the $\bar{e}$ beam to be recorded.
- As with cursor positioning devices, recorded light-pen coordinates can be used to position an object or to select a processing option.
- In concept the light pen is extremely simple.

2 Alternatives are shown in fig. 3 & fig 4

- In each case, the two main elements of the light pen are
  - → photocell &
  - → optical system
    - ↓ which focuses onto it any light in the pen's field of view.

A pen-shaped housing permits the light pen to be held in the hand and pointed at the display screen. On this housing is either a finger operated switch or a shutter that must be depressed to allow light to reach the photocell.

The output of the photocell is amplified and fed to a flip flop which is set whenever the pen is pointed at a sufficiently bright source of light. This flip flop can be read & cleared by the computer.

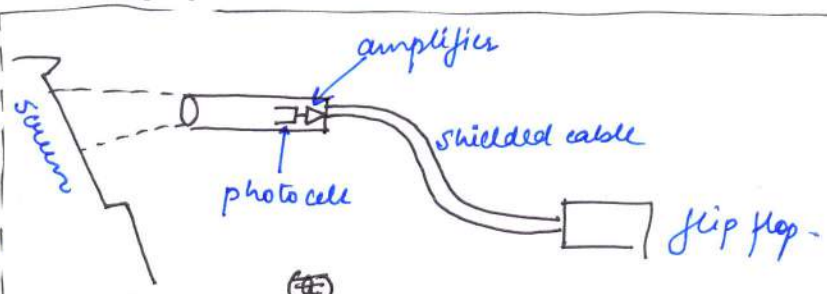


fig 3 The light pen using a hand-held photocell.

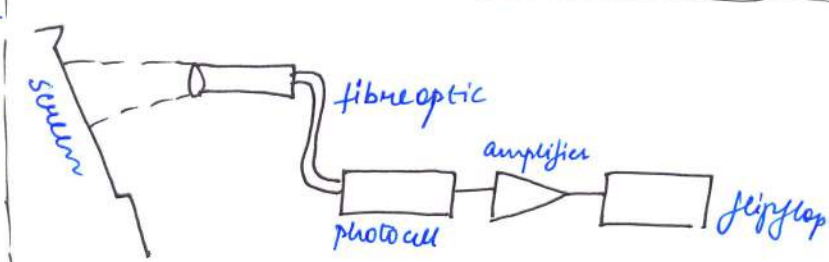


fig 4 The light pen using a fibre optic pipe

- To make use of light pen for positioning, some sort of Tracking program must be running in the computer.
- All light pen programs depend on a rapid response from the pen, when it is pointed at the screen.

A particularly fast response is required if the light pen is to be used with high speed displays.

Eg a display executes one instruction every 2 microseconds but the delay b/w displaying a pt or line & setting the light pen flip flop is 3 μs

By the time this happens, the display will be processing either the next instruction or the one after that.

The program may therefore incorrectly identify the seen item.

Disadvantages :→

(1) When a light pen is pointed at the screen, parts of the screen image is obscured by the hand & pen.

(2) Prolonged use of the light pen can cause arm fatigue.

(3) Light pens require special implementations for some appⁿs because they cannot detect posⁿs within black areas.

(4) Light pens sometimes give false readings due to background lighting in a room.

Books Referred
Bakers & Newman.

★ **DIGITIZING TABLET** :→ A common device for drawing, painting or interactively selecting coordinate posⁿs on an object is a Digitizer & The term tablet is used to describe a flat surface generally separate from the display, on which the user draws with a stylus.

- It is a rectangular plate, the top surface of which is sensitive to electromechanical interactions.
- It is an agent for digitization as instructed by the user through the use of stylus.
- The tablet may contain a set of regions to activate certain logical functions or commands.

Selections of those portions with mouse or stylus can give those functions or commands in the computer.

on the tablet is reproduced in the screen.

Graphic tablets :- are embedded with thin conducting wires. The press of stylus or mouse cross hair on the tablet changes resistance and generates **electromagnetic pulses**. Electric signal is induced in a wire coil in a wire coil in an activated stylus to record a tablet pos^n. Depending on the technology, either signal strength, coded pulses, or phase shifts can be used to determine the pos^n on the tablet.

Acoustic (or Sonic) Tablets :→ use sound waves to detect a stylus pos^n. Either strip microphones or point microphones can be used to detect the sound emitted by an electrical spark from a stylus tip. The pos^n of the stylus is calculated by timing the arrival of the generated sound at the different microphone pos^n.

Advantages → • Drawing can easily be changed • provides a capability of interactive graphics • sketch displayed is neater than on paper.

~~**3 dimensional digitizers**~~ :→ ~~use sonic~~

Disadvantages → • costly • suitable only for app^{ns} which required high resolution graphics.

★ THE MOUSE :→

- The mouse is a small hand held box used to position the screen cursor, resting on 2 metal wheels whose axes are at right angles.
- wheels or rollers on the bottom of the mouse can be used to record the amount & direction of movement.

  Another method for detecting mouse motion is with an optical sensor. → for this mouse is moved over a special mouse pad that has a grid of horizontal & vertical lines.

- Each wheel of the mouse is connected to a **shaft encoder** that delivers an electrical pulse for every incremental rotation of the wheel.
- As the mouse is rolled around on a flat surface, its movement in two orthogonal directions is translated into rotation of wheels. → These rotations can be measured by counting the pulses received from shaft encoders.
- The converted values may be held in registers accessible to the

- Push buttons may be mounted on the top of the mouse & the user can work them with his fingers as he moves the mouse.
- Ideally the computer should be able to read the posⁿ of these buttons whenever it reads the coordinates of the mouse.

Advantages :→

1) Simplicity & low cost
2) The user need not pick it up in order to use it – the mouse simply sits on the table surface until he needs it.
3) Moves the cursor faster than the arrow keys of keyboard

Disadvantages :-

1) It cannot be used for tracing data from paper, since a small rotation of the mouse or a slight loss of contact will cause a cumulative error in all readings.
2) It is very difficult to hand print characters for recognition by the computers.

Books Referred

[Cbr by Bakers & Newman]

★ THE TRACKBALL :-

- It is a ball that can be rotated with fingers or palm of the hand. (fig. on page 148 of Newman:) to produce screen cursor movement.
- Potentiometers, attached to the ball, measure the amount & direction of rotation.
- Trackballs are often mounted on keyboards or other devices such as Z mouse.
- In addition to feedback from the normal tracking symbol users obtain tactile feedback from the rotation rate or angular momentum of the ball.
- Trackballs are frequently equipped with buttons in order that they can be substituted for a mouse

Advantages :-

1) Trackerball is stationery so it doesnot require much space to use it.

2) They are more suitable for portable computers due to compact size.

3) It is sometimes considered better than mouse, because it requires little arm movement & less desktop space.

Books Referred.

Co by Bakers & Newmann & Dogus.

POSITIONING TECHNIQUES

- Positioning sometimes known as **locating** is one of the most basic graphical input techniques.
- The user indicates a posⁿ on the screen with an I/P device & this position is used to insert a symbol or to define the end point of a line.
- The **need for positioning** - occurs in geometric modeling appⁿs where the user frequently wishes to define a new element of the model or to reposition an existing one.

- Positioning involves the user in first moving the cursor or tracking cross to the desired spot on the screen & then notifying the computer by pressing a button or key.

- A single positioning operⁿ can be used to insert a symbol as shown in fig1

fig1 - Positioning a symbol

& two in succession can define the endpts of a line

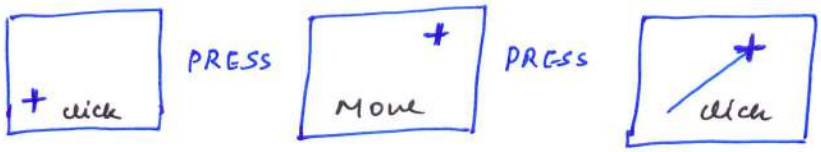


fig2 Line endpt positioning.

Positioning Constraints :→

A constraint is a rule that we wish certain info, in this case the input co-ordinate to obey. We enforce the constraint by applying a transformation to the input co-ordinate generating a new pair of co-ordinate that satisfy the constraint.

(1) **Modular constraint** :- It forces the input point to the nearest intersection on a grid. This can be applied to both symbols & to end point of line.

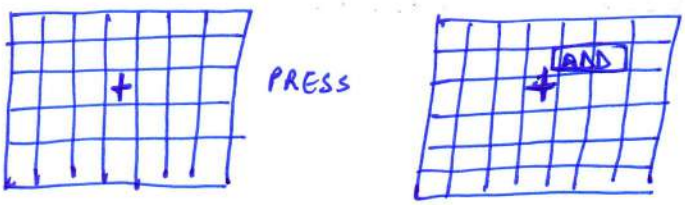


fig3 Modular constraint applied to positioning a symbol.

(2) DIRECTIONAL CONSTRAINT :-

- It applied to straight line. Many appⁿs use only horizontal & vertical lines & are easier to operate if all input lines are constrained to be either horizontal or vertical
- The user specifies two end pts, the program determines whether the line they specify is more nearly horizontal or vertical ~~and~~ draws a line parallel to corresponding axis. (fig 4. & fig5)

(3) GRAVITY FIELD EFFECT :- (fig6 & fig7)

- we may wish to attach a line or symbol to a point in the existing picture that dosnot lie on a grid intersection.
- We cannot use a modular constraint & must instead use a gravity field effect.
- Around each of the lines we define an invisible region, shaped approximately like either a dumbbell or a sausage.
- If input point lie in this region we must replace it with the nearest point on the line itself.

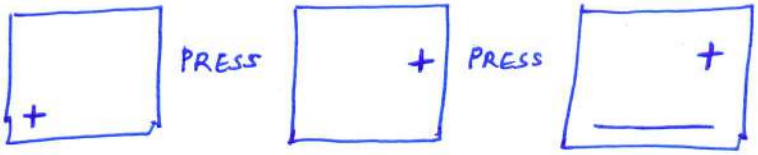


fig4 Horizontal / vertical line constraint

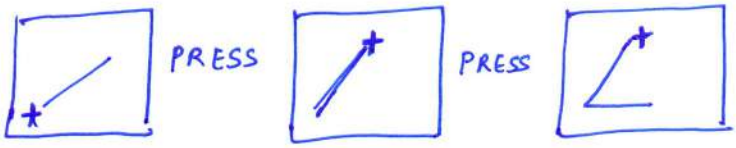


fig5 use of a horizontal line constraint to align endpts vertically

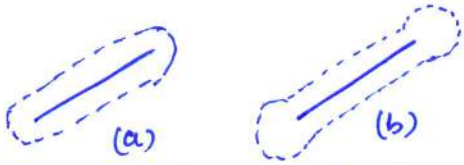


fig6 gravity fields (a) uniform (b) with increased field strength at line endpts

ELASTIC OR RUBBER BAND TECHNIQUES

- Rubber banding is a popular technique of drawing geometric primitives such as line, polylines, rectangle, circle & ellipse on the computer screen.

- It becomes an integral part & de facto standard with the Graphical User Interface (GUI) for drawing & is almost universally accepted by all windows based applications.

- The user specifies the line in the normal way by positioning its two end pts.

  As he moves from the first endpoint to the second, the program displays a line from the first endpoint to the cursor position, thus he can see the lie of the line before he finishes positioning it

- The effect is of an elastic line stretched between the first end point and the cursor; hence the name for this technique.

- Consider the different linear structures in fig 1 (a) - (d). depending on the position of the cross-hair cursor. The user may move the cursor to generate more possibilites. & select the one which suits him for specific appⁿ.

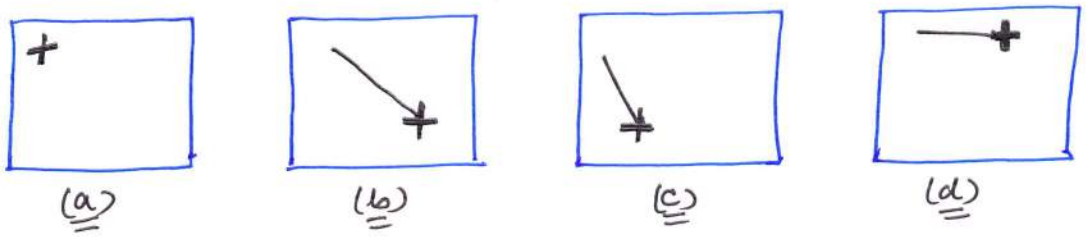


Fig 1 Demonstration of rubber banding: Look at the cross-hair cursor: (a) The start point of the line to be drawn is selected

(b) - (d) 3 diff lines are shown depending on the posⁿ of the cursor representing the end-point - the user selects the desired line.

SELECTION OF TERMINAL POINTS OF THE LINE :→

- The user moves the cursor to the appropiate posⁿ & selects

- Then, as the cursor is moved, the line changes taking the latest posⁿ of the cursor as the end-point.
- As long as the button is held down, the state of the rubber band is active.

The process is explained with the state transition diagram of Rubber banding in fig 2

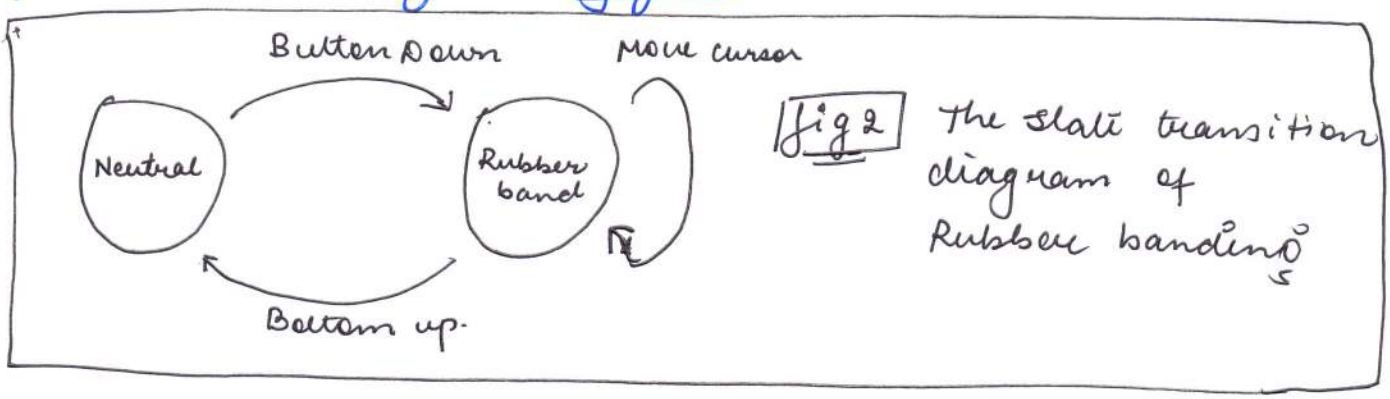


fig 2 The state transition diagram of Rubber banding

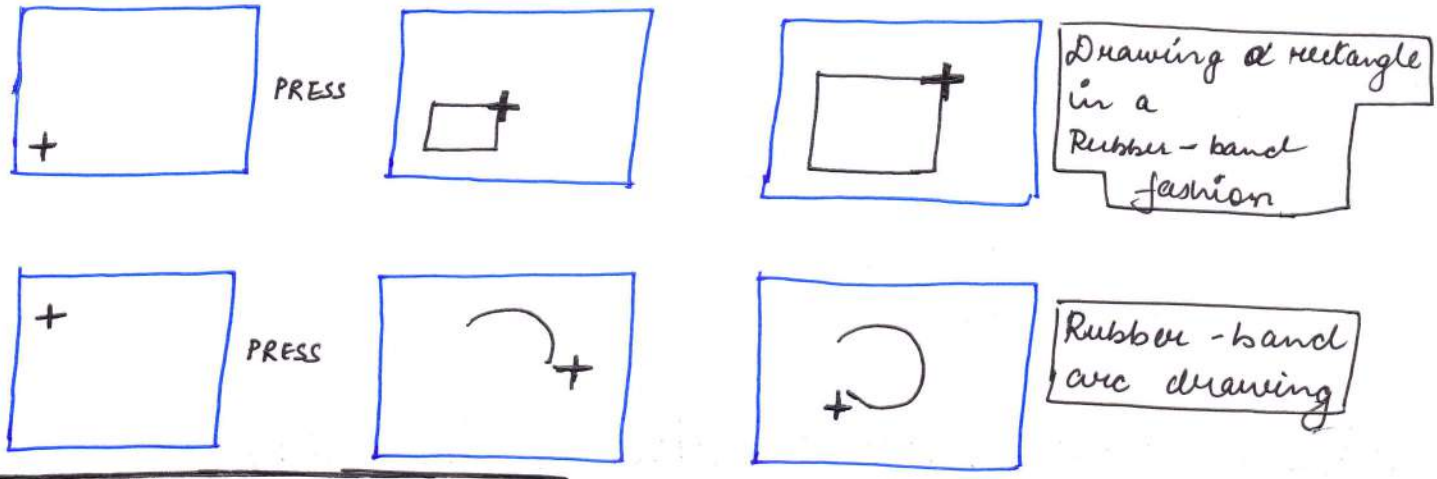
- When the user is happy with the final posⁿ, the pressed button is released and the line is drawn b/w the start and the last posⁿ of the cursor.

e.g. This is widely followed in MS-Windows based appⁿs like in the case of paint brush drawing package.

- Other geometric entities can be drawn in a rubber-band fashion: examples include
  - horizontally or vertically constrained lines
  - rectangles
  - arcs of circles

  fig 3

The technique is very helpful in drawing relatively complex entities such as rectangles ~~or~~ and arcs.



PROBLEM WITH THIS TECHNIQUE :- A rubber band line must be redrawn

ZOOMING & PANNING

- **Zooming** is a transformation often provided with an imaging software.

  This transformation effectively scales down or blows up a pixel map or a portion of it with the instructions from the user.

  Such scaling is commonly implemented at pixel level rather than at the coordinate level.

  - A video display or an image is essentially a pixel map i.e a collection of pixels which are the smallest addressable elements of a picture.

- The process of **Zooming** replicates pixels along successive scan lines

- **e.g** for a zoom factor of **two**,

  ⇒ each **pixel value** is used **four times**
  twice on each of the two successive scan lines

- **fig 1** shows the effect of zooming by a factor of 2.

fig 1 Effect of zooming by a factor of two

- Such integration of pixels sometimes involves replication using a set of ordered patterns, commonly known as **Dithering**

- The two most common dither types are
  (a) ordered dither &
  (b) random dither

- These are widely used, especially when the grey levels (shades of brightness) are synthetically generated

- fig 2 → where ~~and~~ a portion of the image is zoomed

- **Panning** :→ The process of panning acts as a qualifier to the zooming transformation.
  - This step moves the scaled up portion of the image to the centre of the screen & depending on the scale factor, fills up the entire screen.

Advantage :→

Effective increase in the zoom area in all four directions even if the selected image portion (for zooming) is close to the screen boundary.

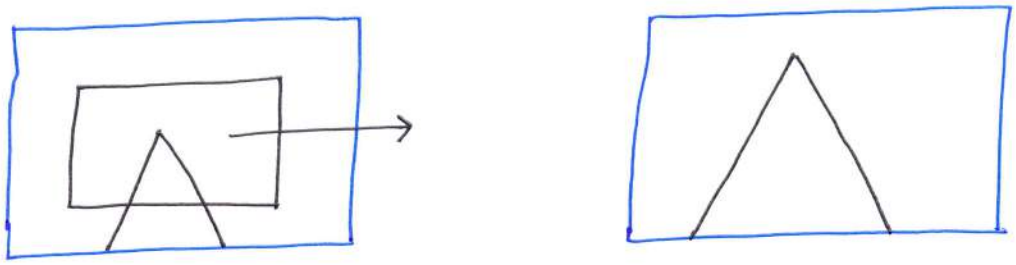


fig 2 The zooming of the selected portion of an image (In the case of panning, the blown up part is shifted to the centre of the screen)

Books Referred

— Fundamentals of computer graphics & Multimedia by D. P. Mukherjee

WINDOWING :→

window to viewport mapping

(Topic covered in 2nd unit)

INKING :→

- If we sample the position of a graphical input device at regular intervals and display a dot at each sampled position, a trail will be displayed of the movement of the device.

  This technique which closely simulates the effect of drawing on paper is called Inking

- For many years the main use of inking has been in conjuction with on-line character - recognition programs

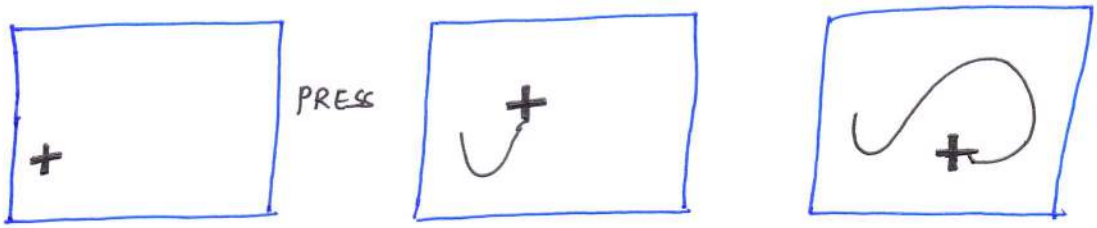


fig 1 Freehand inking

SCISSORING :→

In computer graphics, the deleting of any parts of an image which falls outside of a window that has been sized & laid over the original image.
Also called clipping

CLIPPING

- 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 **clipping algorithm** or **clipping**
- The region against which an object is to clipped is called **clip window**. Everything outside the window is discarded.

Procedure :-

* Clipping algo can be applied in world coordinates so that only the contents of window interior are mapped to device coordinates.
* Complete world coordinates picture can be first mapped to device coordinates or normalized device coordinates, then clipped against the viewport boundaries.
* world coordinate clipping removes those primitives outside the window from further consideration, thus eliminating the processing necessary to transform these primitives to device space.
* viewport clipping, on the other hand can reduce calculations by allowing concentration of viewing & geometric transformation matrices.

Clipping is of following types :-

1) Point clipping
2) Line clipping
3) Area clipping (Polygon)
4) Curve clipping
5) Text clipping

(1) **POINT CLIPPING** :- Assuming that the clip window is a rectangle in standard position, world coordinate save a point $P = (x, y)$ for display, if following inequalities are satisfied.

- where the edges of the clip window $\begin{pmatrix} xw_{min}, & xw_{max} \\ yw_{min}, & yw_{max} \end{pmatrix}$

can either the world coordinate window boundaries or viewport boundaries.

Note :- If any one of these power inequalities is not satisfied, the point is clipped.

(2) LINE CLIPPING :- A line clipping procedure involves several parts

→ first, we test a given line segment to determine whether it lies

(a) completely inside the clipping window

(b) completely outside the clipping window

→ finally, if not identify both of the cases, then we must perform intersection calculation with one or more clipping boundary.

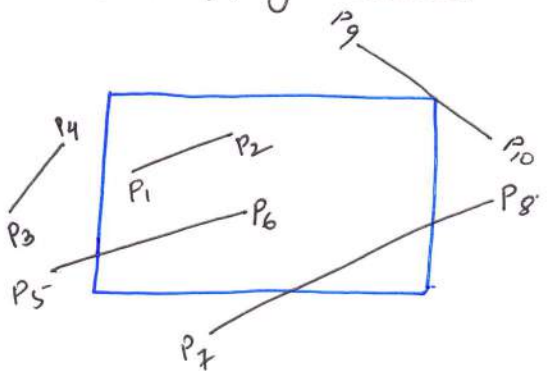
Clipping categories :→

(i) Visible :- both endpts of the line within the window

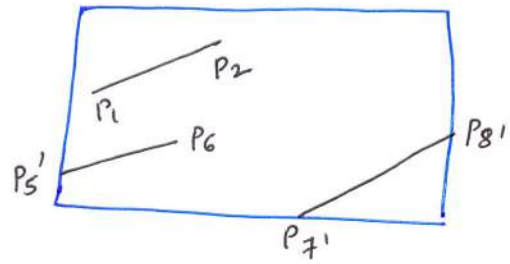
(ii) Non-visible :- the line definitely lies outside the window. This will occur if the line from (x_1, y_1) to (x_2, y_2) satisfies one of the following four inequalities :-

$x_1, x_2 > x_{max}$	$y_1, y_2 > y_{max}$
$x_1, x_2 < x_{min}$	$y_1, y_2 < y_{min}$

(iii) Clipping candidate :- the line is in neither category (i) nor (ii)



Before clipping



After clipping

(a) COHEN - SUTHERLAND LINE CLIPPING :-

This is one of the oldest and most popular line clipping procedure.

In this, every line end pt in a picture is assigned a four digit binary code, called Region code that identifies the location of the pt. relative to the boundaries of the clipping rectangle.

~~The Algorithm employs an efficient procedure for finding the category of a line. It proceeds in 2 steps :-~~

~~(i) Assign a 4-bit region code to each end pt. of the line :→~~

- By numbering the bit posⁿ in the region code as 1 through 4 from right to left, the coordinate region can be correlated with bit partitions as:-

bit 1 :	left	A value of 1 in any bit posⁿ → indicate that the pt is in the relative posⁿ.
bit 2 :	Right	
bit 3 :	below	Otherwise, bit set to 0.
bit 4 :	above	

- Bit values in the region code are determined by comparing end pt coordinates value (x, y) to the clip boundaries.
  Bit 1 is set to 1 if $x < x_{wmin}$
  Other 3 bits values can be determined using similar comparisons.

The Algorithm is carried out in 2 steps :→.

Step 1 :→ Assign a 4-bit region code to each endpt. of the line

y_{max}	1001	1000	1010
y_{min}	0001	0000 window	0010
	0101	0100	0110
	x_{min}	x_{max}	

The code is determined with the following two steps :-

(i) Calculate differences b/w endpt coordinates & clipping boundaries.

(ii) Use the resultant sign bit of each difference calculation to set the corresponding value in the region code.

Step 2 :- The line is visible if both region codes are 0000
not visible if bitwise logical AND of the codes is not 0000
a candidate for clipping - if bitwise logical AND of the region codes is 0000

For a line in category 3,

→ The intersection pts. with a clipping boundary can be calculated using the slope intercept form of the line equation.

→ For a line with endpt coordinate (x_1, y_1) (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 x value is set either to xw_{min} or xw_{max} & the slope of the line is calculated with a horizontal boundry.

→ the x coordinate can be calculated as :-

$$x = x_1 + \frac{(y - y_1)}{m}$$

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

Herein $m = \frac{y_2 - y_1}{x_2 - x_1}$ (slope).

(b) MID - POINT SUB DIVISION LINE CLIPPING :-

→ In this, we subdivide the line at its midpt and in this way avoid direct computation of the line's point of intersection with the edges of the screen.

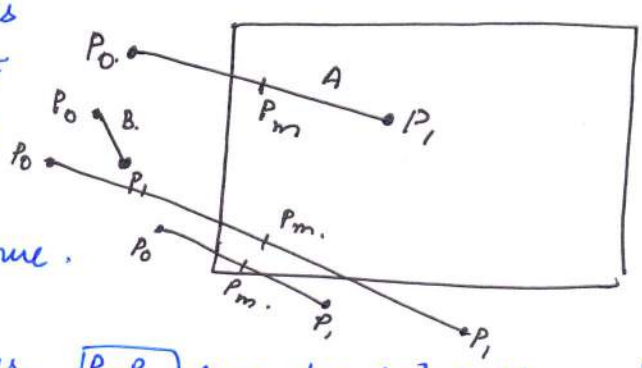
→ This modification makes the algo more suitable for machines w/o hardware for multiplication and division.

→ The midpt variation of the algo has been implemented in H/W in clipping divider

→ The algo clips the line by finding endpts. of its visible segment.

The steps carried out in this process :→.

Step 1 :- we test whether P_1 is visible., if so, it is farthest visible pt. from P_0 & the process is complete.
If it is invisible, we continue.



Step 2 :- we check whether P_0P_1 can be trivially rejected, in which case, the process is complete & no O/P is generated. otherwise we continue.

Step 3 :- we divide P_0P_1 at its midpoint P_m. This is a guess at the farthest visible point. If the segment P_mP_1 can be trivially rejected.
we have overestimated & we repeat step 2 using the segment P_0P_m.
otherwise we repeat the step 2 with segment P_mP_1

Note :- The no. of subdivisions that must be performed during each search is at most equal to = no. of bits of precision in the representation of x & y

If we have line with end pts. (x_1, y_1) & (x_2, y_2), then the mid-point coordinates (x_m, y_m) can be written as.

$$x_m = \frac{x_2 + x_1}{2} \qquad y_m = \frac{y_2 + y_1}{2}$$

(C) LIANG - BARSKY ALGORITHM :-

→ The following parametric eq^n represent a line from (x_1, y_1) to (x_2, y_2) along with its infinite extension

$$x = x_1 + \Delta x \cdot u$$
$$y = y_1 + \Delta y \cdot u$$

where $\Delta x = x_2 - x_1$, $\Delta y = y_2 - y_1$

→ For point (x, y) inside the window

→ The line itself corresponds to $\boxed{0 \le u \le 1}$, we traverse along the extended line with $\boxed{u}$, increasing from $\boxed{-\infty}$ to $\boxed{\infty}$

We first move from $\boxed{\text{outside}}$ to the $\boxed{\text{inside}}$ of the clipping window's two boundary lines (bottom & left), & then move from $\boxed{\text{inside}}$ to the $\boxed{\text{outside}}$ of the other two boundary lines (top & right)

→ If we use $\boxed{u_1}$ & $\boxed{u_2}$ where $\boxed{u_1 \le u_2}$ to represent the $\boxed{\text{beginning}}$ & $\boxed{\text{end}}$ of the visible portion of the line we have,

$$\boxed{\begin{aligned} u_1 &= \text{maximum}(0, u_l, u_b) \\ u_2 &= \text{minimum}(1, u_t, u_r) \end{aligned}}$$

where u_l, u_b, u_t, u_r → corresponds to the intersection pt. of the extended line with window's left, bottom, top, right boundary respectively.

→ Rewrite the four inequalities

$$\boxed{p_k u \le q_k} \qquad k = 1, 2, 3, 4 \text{ ----}$$

where

$p_1 = -\Delta x$ $\quad q_1 = x_1 - x_{min}$ (left)

$p_2 = \Delta x$ $\quad q_2 = x_{max} - x_1$ (right)

$p_3 = -\Delta y$ $\quad q_3 = y_1 - y_{min}$ (bottom)

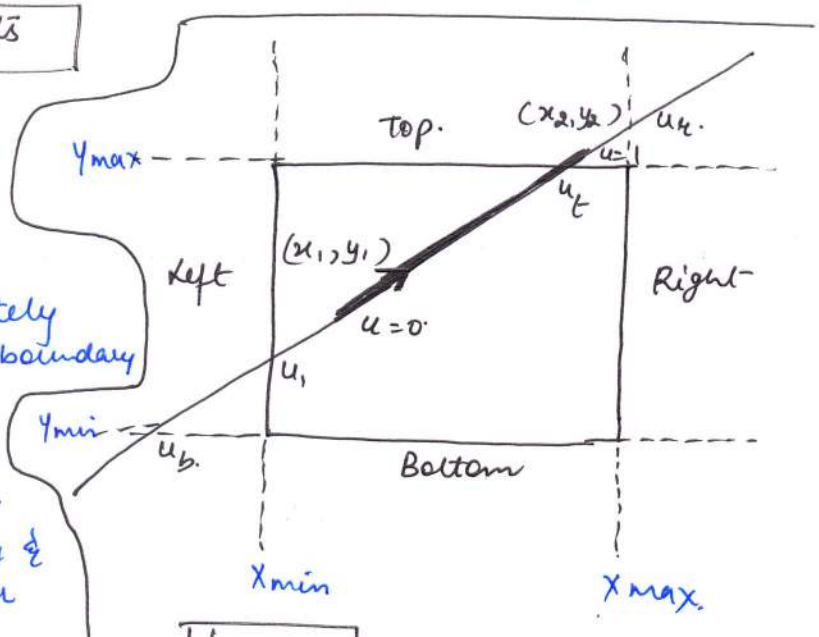
$p_4 = \Delta y$ $\quad q_4 = y_{max} - y$ (top)

Observe the following facts

→ If $\boxed{p_k = 0}$, the line is parallel to corresponding boundary &

★ if $\boxed{q_k < 0}$, line completely outside the boundary & eliminated

★ if $\boxed{q_k \ge 0}$, line is inside the boundary & needs further consideration



→ If $\boxed{p_k < 0}$, the extended line proceeds from $\boxed{\text{outside}}$ to $\boxed{\text{inside}}$ of the corresponding boundary line

→ If $\boxed{p_k > 0}$, the extended line proceeds from $\boxed{\text{inside}}$ to outside

→ If $\boxed{p_k \neq 0}$, the value of $\boxed{u}$ that corresponds to the intersection point is $\boxed{q_k / p_k}$

There are four steps in this algo

Step 1 :- If $\boxed{p_k = 0}$ & $\boxed{q_k < 0}$ for any k, eliminate the line & stop otherwise proceed.

Step 2 :- for all k, such that $\boxed{p_k < 0}$

Calculate $\boxed{r_k = \frac{q_k}{p_k}}$

Let $\boxed{u_1}$ be the $\boxed{\max}$ of the set containing $\boxed{0}$ & the calculated $\boxed{r}$ values.

Step 3 :- For all k, such that $\boxed{p_k > 0}$

calculate $\boxed{r_k = \frac{q_k}{p_k}}$

Let $\boxed{u_2}$ be the $\boxed{\min}$ of the set containing $\boxed{1}$ & the calculate $\boxed{r}$ values

Step 4 :- If $\boxed{u_1 > u_2}$

Eliminate the line since it is completely $\boxed{\text{outside}}$ the clipping window

Otherwise,
use $\boxed{u_1}$ & $\boxed{u_2}$ to calculate the endpts. of the clipped line.

Advantages :-

(1) It is more efficient than the Cohen Sutherland algo. since intersection calculation are reduced.

(3) POLYGON CLIPPING :→ In this we consider the case of using a polygonal clipping window to clip a polygon.

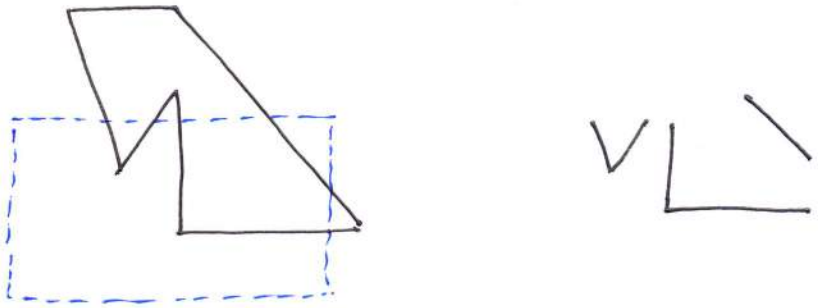
→ A polygon boundry processed with a line clipper may be displayed as a series of unconnected line segments depending upon the orientation of the polygon of the clipping window. (fig)

There are 3 types of polygon :→

★ **Convex** :- A polygon is called **convex**, if the line joining any two interior points of the polygon lies completely inside the polygon.

★ **Concave** :- A non convex polygon is said to be **concave**.

★ **+vely oriented** :- A polygon with vertices $P_1 \ldots P_N$ (& edges P_{i-1}, P_i & $P_N P_1$) is said to be +vely oriented if a tour of the vertices in the given order produces a counterclockwise circuit.



Before clipping After clipping.

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

For **polygon clipping**, we require an algorithm that will generate one or more closed areas that are then scan converted for the appropiate area fill.

The output of a polygon clipper should be sequence of vertices that defines the clipped polygon boundaries.

fig Display of a

(a) THE SUTHERLAND - HODGEMAN ALGO (For Concave Polygons)

- 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 boundry 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 & a top boundary clipper

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

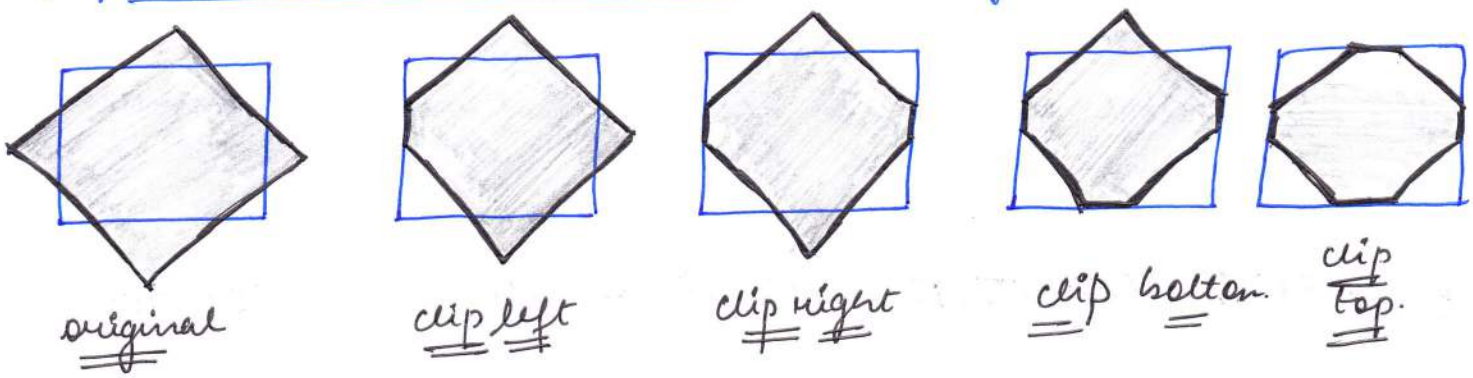


fig. Clipping a polygon against successive window boundaries.

Let $P_1 \ldots P_N$ be the vertex list of the polygon to be clipped.
Let edge E, determined by end pts. A & B be any edge of the +vely oriented, convex clipping polygon.

we clip each edge of the polygon in turn against the edge E of the clipping polygon forming a new polygon whose

① If both P_{i-1} & P_i are to the left of the edge,
⇒ vertex P_i is placed on the vertex output list of the polygon.

② If both P_{i-1} & P_i are to the right of the edge.
⇒ Nothing is placed on the vertex o/p list

③ If P_{i-1} is to the left & P_i is to right of the edge E,
⇒ the intersection point I of the line segment P_{i-1}, P_i with the extended edge E is calculated & placed on the vertex o/p list

④ If P_{i-1} is to the right & P_i is to the left of the edge E,
⇒ the intersection point I of the line segment P_{i-1}, P_i with extended edge E is calculated.
Both P_i & I placed on vertex o/p list.

Note :- Special attention is necessary in using the Sutherland-Hodgeman algo in order to avoid unwanted effects.

(b) WEILER ATHERTON POLYGON CLIPPING : (for Concave polygon)

- when the clipped polygon have two or more separate sections, then it is the concave polygon handled by this algo.
- The vertex - processing procedure for window boundaries are modified. so that concave polygon are displayed.

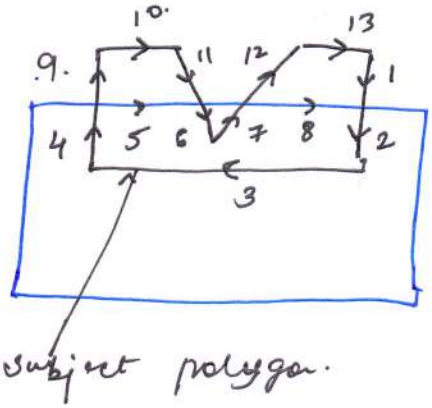
Let the clipping window be initially called clip polygon & polygon to be clipped the subject polygon.

→ we start with an arbitrary vertex of the subject polygon & trace around its border in the clockwise direction until an intersection with the clip polygon is encountered

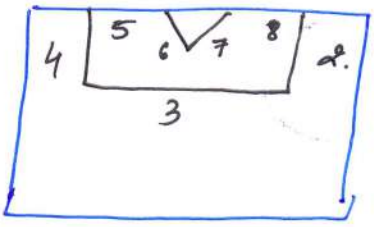
(i) If the edge enters the clip, polygon record the intersection point & continues to trace the subject polygon.

to follow the clip polygon in the same manner (i.e treat the clip polygon as subject polygon & the subject polygon as clip polygon & proceed as before).

→ Whenever our path of traversal forms a sub-polygon, we output the sub-polygon as part of the overall result.



(a) Clip polygon.



(b)

→ We then continue to trace the rest of the original subject polygon from a recorded intersection point that makes the beginning of a non-yet-traced edge or portion of an edge. The algo terminates when the entire border of the original subject polygon has been traced exactly.

(4) CURVE CLIPPING :-

- Curve clipping procedures will involve non-linear eq^ns &
- requires more processing than for objects with linear boundaries
- The bounding rectangle for a circle or other curved obj, can be used first to test for overlap with a rectangular clip window

★ If bounding rectangle is completely inside window

⇒ we save the ~~project~~ object.

★ If rectangle is completely outside the window,

⇒ we discard the object

- But if the bounding rectangle test fails, we can look for other computation saving approaches.

- For circle, we can use the coordinate extents of individual quadrants & then octants for preliminary test the coordinate extents of individual quadrants.

fig clipping a filled circle.

(5) **TEXT CLIPPING** :- There are several techniques that can be used to provide text clipping.

★ The simplest method for processing character strings relative to a window boundary is to use the

All-or-none string clipping strategy :-

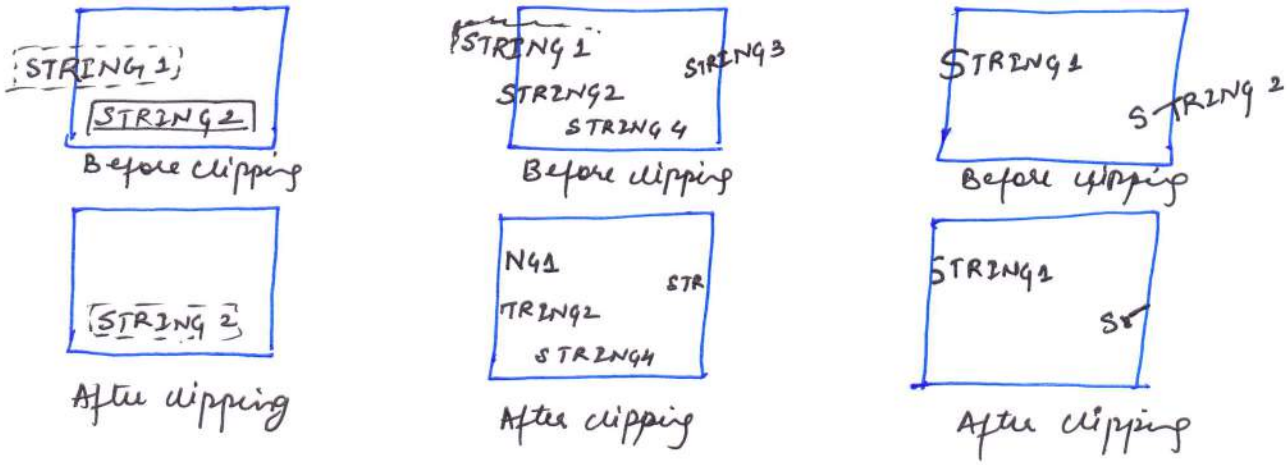
If all of the string is inside a clip window, we keep it. Otherwise it is discarded → by considering a bounding rectangle around the text pattern. The boundary pos^ns of the rectangle are then compared to the window boundaries & the string is rejected, if there is any overlap.

Advantage This method produces the fastest text clipping.

★ All-or-none character clipping strategy :→

Here we discard only those characters that are not completely inside the window.

★ Compute



All or None string clipping strategy

All or None character clipping strategy.

Components of individual character clipping strategy.

fig Text clipping using bounding rectangle

X

Books Referred

- Computer Graphics by Donald Hearn & Baker
- Computer Graphics by Schaum's Series
- Principles of ICG by Newman

MOUSE PROGRAMMING

- GUI & mouse go hand in hand.
- The mouse has more or less become standard I/P device with any GUI.
- The use of a mouse requires a program to sense its presence. We have to load a device driver program that attaching of mouse.
- A device driver is program while sense the signal coming from the port to which the mouse is attached.
- On sensing the signals, the device translates these into the related actions in the screen.
- The mouse programming in C language load [Ax] register & issues interrupt number [33H] following is the list of functions used in C language for mouse programing.

init mouse :- Initialize the mouse. This fun checks whether mouse driver have been loaded or not by issuing interrupt [33H] & service no.

```
init mouse()
{   i.x.ax = 0;      // Stores Ax = 0
    init 86|0X33 , di & 0);
    return (0.x, ax);
}
```

This fun returns the value FFFFH if mouse support is available use returns 0.

show mouse ptr() :- This fun displays the mouse pointer on screen this fun uses service no. under interrupt no. 33H.

Hidemouse ptr() :- Hides the mouse ptr. This fun uses service nos under interrupt no. 33H.

```
hide mouse ptr()
```

getmouse () :- This fun returns the mouse pos^n & also button status. Uses service no. 3 under interrupts. 33H.

```
getmouse ( int * button, int *x, int *y)
  {  i.x.ax = 3;
     int 86 ( 0x33, &i, &o);
     * button = o.x.bx
       *x = o.x        /* x coordinate */
       *y = o.x.dx     /* y coordinate */
```

The status of mouse button is stored in Bx register

If $\boxed{Bx = 1}$ → left button is clicked

$\boxed{Bx = 2}$ → right " " "

= 4 → center " " "

return the location of cursor in cartesian sys (x, y) in Cx & Dx register.

restricted mouse ptr () :- This fun restricted the movement of mouse in a sp. region. Similar to viewport, the movement of mouse is limited to a particular area & mouse cant go beyond that area.

```
restrict mouse ptr ( int x, int y, int x2, int y2)
  {  i.x.ax = 7;   // sets horizontal limit of ptr
     i.x.cx = x;   // min x value
     i.x.dx = x2;  // max x value
     int 86 ( 0x33, &i, &o);
     i.x.ax = 8    // sets vertical limit for ptr
     i.x.cx = y1
     i.x.dx = y2
     int 86 ( 0x33, &i, &o) }
```

This fun restricts the movement of mouse in special rectangle with indpts coordinates (x_1, y_1) (x_2, y_2) & (x_2, y_2)

F also uses two services of interrupt 33H. service no. 7 restricts the horizontal movement of a mouse pointer whereas service 8 restricts the vertex movement of mouse.

[illegible] lib fun $\boxed{int86()}$

Various appⁿs of mouse programming :→

(1) Free hand drawing
(2) can generate Rubber band
(3) Diff shapes can be generated quickly
(4) Menu driver.

Mouse can be used in text mode or graphics mode.
initgraph() is the fxn responsible for switching mode from graphics to text.

Detect is the ~~mouse~~ macro defined in 'graphics.h'.
It request initgraph to automatically ~~det~~ which graphics driver to load in order to s/w to highest resolution graphics mode.

Init graph fⁿ takes 3 parameters

1) Graphic driver (gd) 2) graphic mode (gm)
3) path to driver file.

The fxn sets up the numeric values to gd & gm. One in graphic mode $\boxed{\text{get max()}}$ & $\boxed{\text{get many ()}}$ to obtain max $\boxed{x}$ & $\boxed{y}$ coordinate in current graphic mode.

Rectangle fxn can be used to set the viewport area which restricts any drawing activity within the view.

X