BCA -

UNIT - III

Page No.
Date

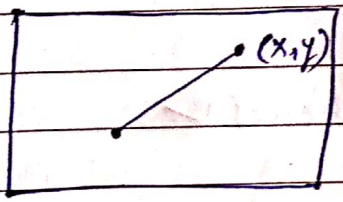
PRIMITIVES OPERATIONS

Display devices are different in nature but most graphics system offer a similar set of graphics primitive commands.

(1) Absolute line command

LINE - ABS - 2(X, Y)

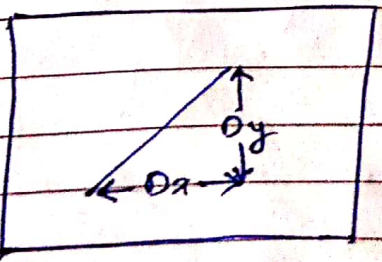
It will draw a line from the current position to the specified point (x, y).



(2) Relative line command

LINE - REL - 2(DX, DY)

It will indicate how far we have to move from the current position.



(3) MOVE - ABS - 2 (X, Y)

It will move pen position to the specified co-ordinate position (X, Y).

(4) MOVE - REL - 2 (DX, DY)

It will move pen position relative to specified co-ordinate position. (Dx, Dy).

DISPLAY FILE INTERPRETER

Q what is display file?

Ans It is a file which is used to keep information about the image in form of instructions.

By using these instructions we can construct the image whenever required.

Saving instructions occupies less space than saving an image itself.

Display file interpreter is used to convert these instructions into actual image.

Q what is Display file interpreter?

Ans We can think of our display file Interpreter as a m/c which executes

instructions one by one written in display file, the result of execution is a "visual image".

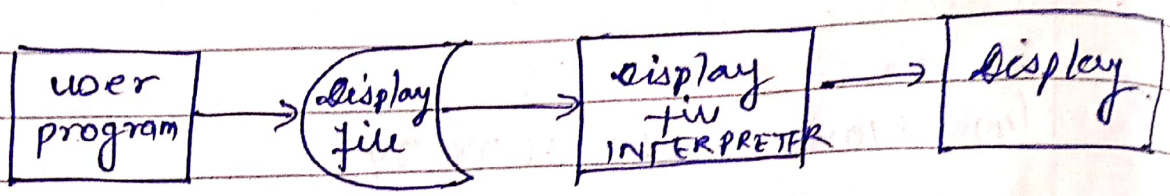


fig (1) → Display file and Interpreter

Display file interpreter serves as an interface between our graphics program and the display device.

When we write a graphics program for a specific display device then it is not portable but when we use display file to store code of our program then our prog. is portable.

Here we need display file interpreter to convert our instructions to an actual image.

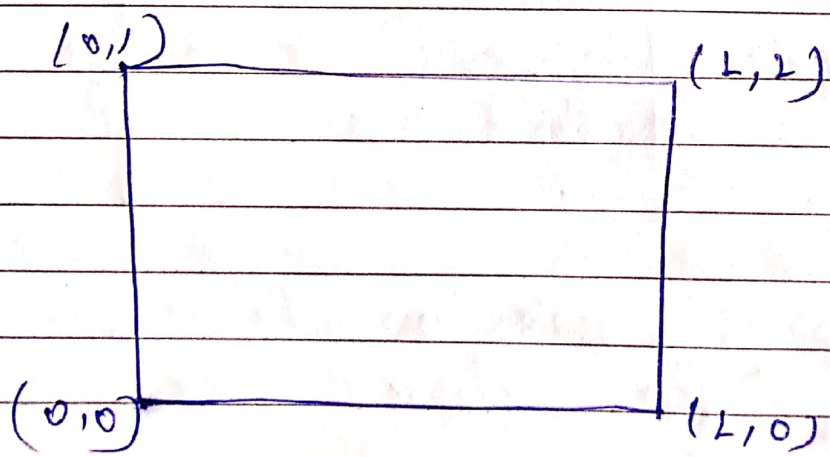
Display files are some times also called files.

Page No.
Date

Q. What is Normalized device co-ordinates? explain their significance.

Ans. Normalized device co-ordinates are device independent units. By using normalized device co-ordinates we can create an image which is device independent.

Actually different display devices have different sizes, (screen) which is measured in pixels. If we want our program to be device independent then we must specify the co-ordinates using normalized device - co-ordinates.



formula for converting NDC to actual DC

x_s = width * x_n + width - start-

y_s = height * y_n + height - start-

Q Explain structure of display file. write display file algorithms.

Ans DISPLAY - FILE STRUCTURE :-

Instruction of display file contains two parts -

i) opcode :

i.e operation code It indicates what type of command it is and what operation we have to perform on the given data. (Line or More)

ii) Operand :

which specifies co-ordinates (x, y)

Display file is made up of series of instructions of these types.

One possible method to store this instruction is to use three seperate arrays - one to store opcode DF-OP, one to store x co-ordinate DF-X and one to store y co-ordinate DF-Y.

suppose we have to read s^{th} instruction from the display file then we have to

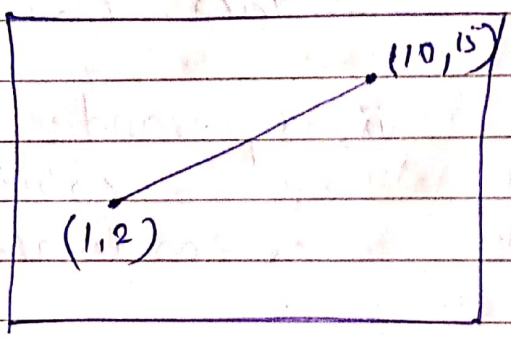
Page No.
Date

Need DF-OP[5], DF-X[5], DF-Y[5] for one single instruction.

The display-file must be large enough to store all commands needed to create our image.

Let's define opcode first before writing commands in display file-

OP = 1 for MOVE
= 2 for LINE



Suppose we to store instructions to draw above line then -

DF-OP	DF-X	DF-Y
1	1	2
2	10	15

→ Let's define algorithm to write instruction in display file.

Page No.
Date

Algorithm PUT-POINT(OP, X, Y)

starts

If FREE > DFsize then
Return
error "display file full"
stop.
ELse
DF-OP[FREE] ← OP
PF-X[FREE] ← X
DF-Y[FREE] ← Y
FREE = FREE + 1
Return
END.

STOP

The Algorithm put-point is used to store op, x, y coordinates in display file.

FREE → shows free location in display file

DF-size → shows size of display file

After storing the instruction increment free by 1 so that it points next free location

Page No.
Date

* Algorithm get_point (n, OP, x, y)

begin

op ← DF-OP[n]

x ← DF-X[n]

y ← DF-Y[n]

Return;

END

→ Algorithm get_point Retrieves instr. from the display file from the given location.

n → the number of the desired instruction

op — opcode

x — x co-ordinate

y — y co-ordinate

* Algorithm display_file_enter (OP)

begin

put_point(OP, DF-pen-x, DF-pen-y)

Return.

END

Here DF-pen-x and DF-pen-y gives current pen position

Page No.
Date

* Algorithm move-abs-2(x,y)
begin
DF-pen-x ← x
DF-pen-y ← y
display-file-enter(1);
Return
END.

* Algorithm line-abs-2(x,y)
begin
DF-pen-x ← x
DF-pen-y ← y
display-file-enter(2);
Return;
end;

* Algorithm move-rel-2(Dx, Dy)
begin
DF-pen-x = DF-pen-x + Dx
DF-pen-y = DF-pen-y + Dy
display-file-enter(1);
Return.
end

* Algorithm Line-rel-2(Dx, Dy)
begin
DF-pen-x = DF-pen-x + Dx
DF-pen-y = DF-pen-y + Dy
display-file-enter(2);
end.

DISPLAY - FILE Algorithms

They are also known as algorithms for display file interp...

The interpreter will read the instr... from a portion of the display file and carry out the operation like line and move command.

Algorithm MOVE (X, Y)

used to move pen on location (x, y)

(x, y) → Normalized co-ordinates

FRAME - PEN - X
FRAME - PEN - Y
} → Actual screen co-ordinates

WIDTH
HEIGHT
} → screen dimensions.

width - start
Height - start
} → co-ordinates of lower left co-ordinates

WIDTH - END
HEIGHT - END
} → co-ordinates of upper right corner.

BEGIN

FRAME - PEN - X ← max (width start, min (width - end, x * width + width start))

FRAME-PEN-Y = MAX (Height-start, min (Height-end, y * Height + Height-start))

Return;

END

* In Bresenham we are using formula for converting normalized device co-ordinates to actual screen co-ordinates.

* Max and MIN functions are used as a safeguard, they prevent it from generating a value outside the bounds of actual display.

* If (X,Y) were to corresponds to a point outside the screen area, the MAX and MIN functions would clamp (~~clamp~~) the corresponding screen co-ordinates position to the display boundary.

* Algorithm DDLINE (X,Y)
used to draw a line
(X,Y) → normalized co-ordinates
FRAME-PEN-X, FRAME-PEN-Y] - pen position i.e actual screen co-ordinates
WIDTH, HEIGHT] - screen dimentions.

Page No.
Date

Width - start
Height - start
] coordinates of lower left corner

Width - End
Height - End
] co-ordinates of upper right corner.

x1, y1 → old end points of the line segment.

Begin

x1 ← frame-pen-x
y1 ← frame-pen-y

frame-pen-x ← MAX(width-start, min(width-end, x * width + width-start));

frame-pen-y ← max(Height-start, min(Height-end, y * Height + Height-start));

call bresenham's () to draw line.

END

Algorithm INTERPRET (START, COUNT)

This will read and execute instructions from display file
START — starting location from display file.

count — total no. of instructions to be interpreted.

for example if we want to execute 10 instructions from display file starting from instruction 3 then start will have value 3 and count will have value 10.

```
Begin
    for nth = start to start + count - 1
    do
    begin
        get_point(nth, op, x, y);
        If (op = 1) then
            call domore(x, y);
        Else
            If (op = 2) then
                call Dolin(x, y);
            Else
                Return error;
    End
    Return;
END
```

DISPLAY CONTROL

→ It is the main component of any device, generating a video signal.

→ It is an integrated circuit that produces a TV video signal in a video display system.

→ Video display controller generate the timing of the video signals and the blanking interval signals.

→ A video controller chip is integrated in the main computer system logic but can also manipulate video RAM contents in an independent fashion.

→ The display controller reads the memory address of the frame buffer and generates the appropriate interface signal to encode the pixel color directed to the display.

→ The display controller copies the contents from the memory every frame and o/p to the display across one of the many physical interface standard.

Page No.
Date

Attributes of o/p primitives
ch-4

* LINE ATTRIBUTE *

Basic attributes of a line are -

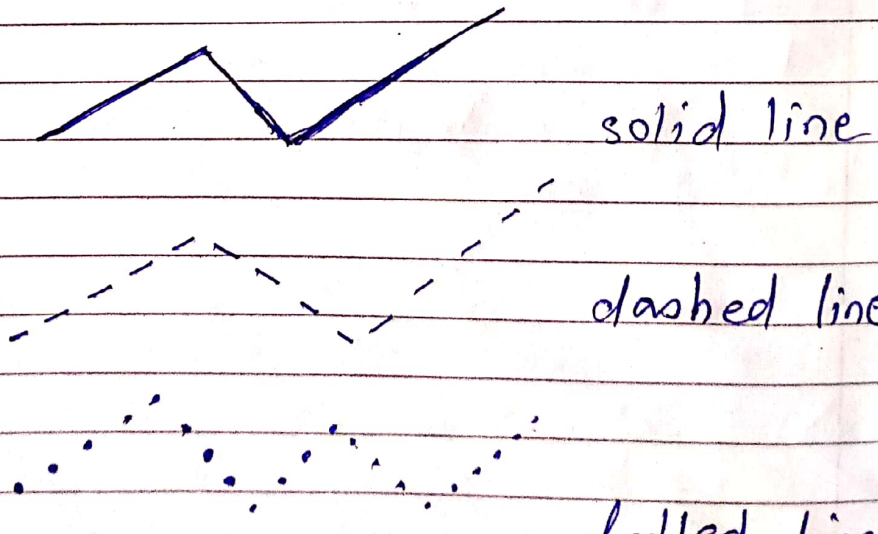
i) type
ii) width
iii) color

(i) LINE TYPE

The line type attribute can have following values like :-

a) solid lines
b) dashed lines
c) dotted lines

Example -



To set a line type attribute in PHIGS a uses uses following functions —

setlinetype (lt)

where lt = 1, 2, 3 or 4

1 – solid lines
2 – dashed lines
3 – dotted lines
4 – dashed-dotted lines

(ii) LINE WIDTH

→ This option depends on the capabilities of the o/p device.

→ for example – If we are using video monitor then heavy lines can be drawn via adjacent parallel lines.

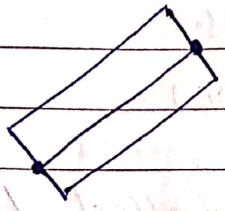
If we are using pen-plotter then we need to change pen to draw a heavy line.

In PHIGS attribute, a line command is used to set the line-width.

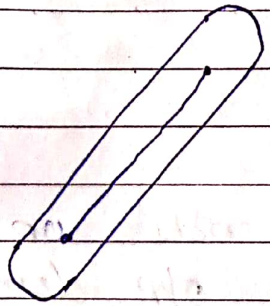
we set- line width by using command.

setlinewidthscalfactor(lw)

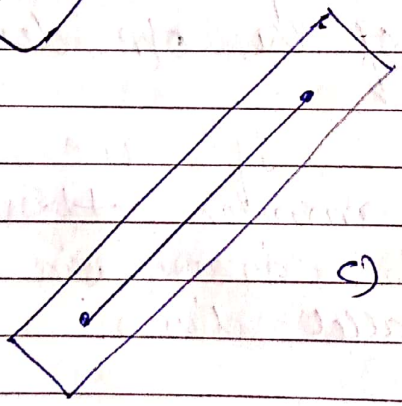
lw → +ve no.



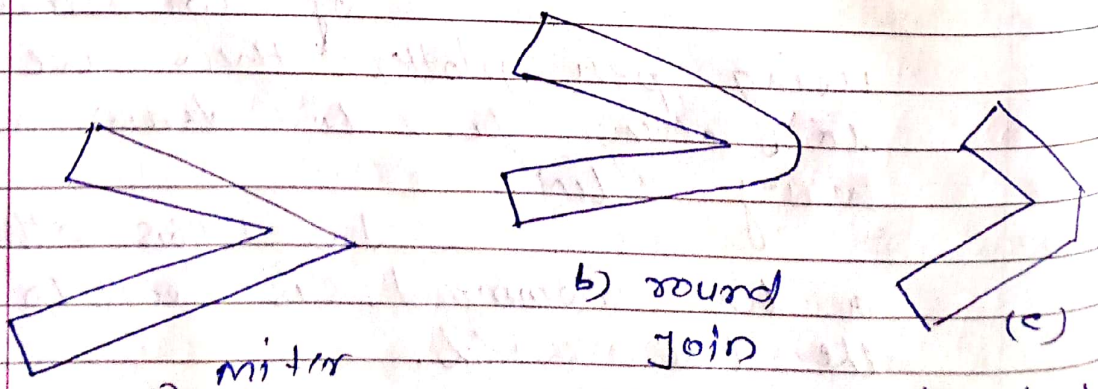
a) butt caps



b) Round caps



c) Projecting square caps



a) mitre join

b) round join

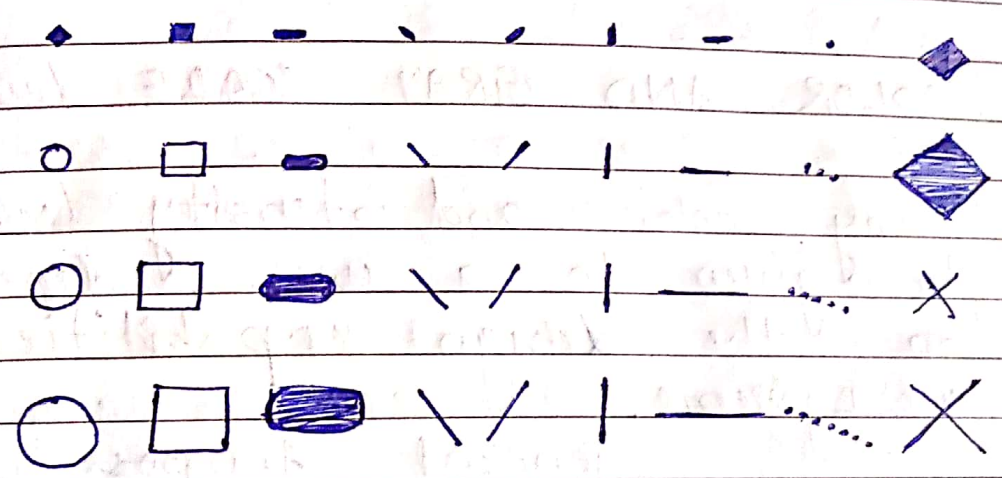
(c) bevel join

Page No.
Date

Pen and Brush options

* lines can be drawn with pen or brush option.
This category includes shap, size and pattern.

* some possible pen or brush shapes are given —



Curve lines are drawn using different brush option.

Line color

We set line color value in PHIGS with the function

setpolylinecolorIndex (lc)

where lc = any color value which can be defined using any +ve integer value.

COLOR AND GRAY SCALE levels.

Many color and intensity level can be given to a user, depending on the design capabilities of a systems.

general purpose raster scan system usually provide a wide range of colors, while random scan monitors offer only a few color choices.

COLOR TABLE

color table is used to store color values for a color scheme

fig shows possible scheme for storing color values in a color lookup table, where frame buffer

values are now used as indices into a color table.

color code	stored color values in frame buffer RED	GREEN	BLUE	Displayed color
0	0	0	0	Black
1	0	0	1	Blue
2	0	1	0	Green
3	0	1	1	Cyan
4	1	0	0	Red
5	1	0	1	Magenta
6	1	1	0	Yellow
7	1	1	1	White

A user can set values in color table in a PHIGS application program with function -

setColorRepresentation (ws, ci, colorptr)

ws → workstation o/p device
ci → specifies color index
colorptr → points to a trio of RGB color values

→ Advantages -

use of a color table can provide a reasonable no. of simultaneous colors without requiring large frame buffer.

- table entries can be changed at any time.

GRAY SCALE

- With monitors that have no color capabilities, color functions can be used in an application program to set the shades of gray or grayscale, for displaying primitives.

Numeric values over the range from 0 to 1 can be used to specify gray scale level.

These gray scale levels are then converted to appropriate binary code for storage in the raster.

This allows intensity settings to be easily adapted to systems with differing gray scale capabilities.

INTENSITY CODES FOR A GRAY SCALE SYSTEMS

INTENSITY codes	stored intensity values in frame buffer		Displayed Gray Scale
0.0	0	00	Black
0.33	1	01	Dark gray
0.67	2	10	Light gray
1.00	3	11	White

AREA FILL ATTRIBUTES

Whenever we want to fill an area, we have choice between a solid color or a patterned fill.

Also we can choice a particular color and patterns to fill the area.

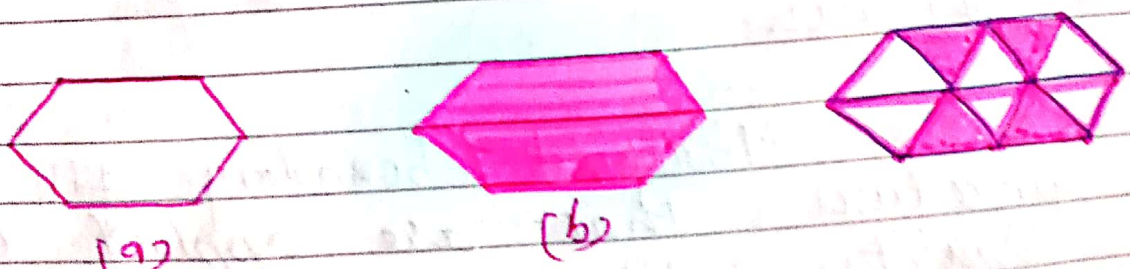
FILL style :-

Areas are displayed with these different fill style.

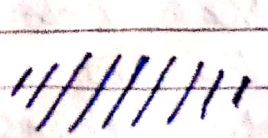
i) Hollow with a color border

ii) Filled with a solid color

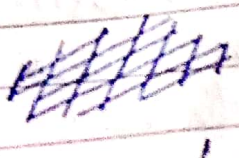
iii) filled with a specified pattern or design



(a) (b)



hatch pattern



cross hatch pattern

Pattern fill

We select fill patterns with

setInteriorStyleIndex (pi)

where pi → specifies a table position

Our fill style patterns, table entries can be created on individual output devices with -

setPatternRepresentation (ws, pi, nx, ny, ...)

pi → pattern index no.

ws → workstation code
cp → 2-D array of color codes
nx → column in cp
ny → row in cp

SOFT FILL

Modified boundary-fill procedures that are applied to repaint areas so that fill color is combined with the background color are referred to as soft-fill or tint fill algorithms.

Page No.
Date

CHARCTER ATTRIBUTES

With the help of character attributes we can set font, size, color, and orientation of characters.

Attributes can be set both for entire characters strings and for individual characters.

Text- Attributes

Text- attributes include i) font
ii) color
iii) size.

choice of font or typeface is a set- of characters with a particular design style such as New York, courier, Times Roman etc.

The characters in a selected font can also be displayed bold italic and underlined.

A particular font and associated style is selected in PHIGS by setting an integer code for the text- font parameter tf in the function -

setTextFont(tf)

eg - a

* serif type is more readable i.e it is easier to read in longer block of text.
* But in sans-serif it is not easier to read in longer block of text. they are not easy readable they are legible.
* Sans-serif are good for writing headings.
* Serifs are good for writing other than heading i.e paragraph or main text.

Two Different Representations are used for storing computer fonts

i) Bitmapped font

It is a very simple method in which we use rectangular grid pattern to store information of characters.

Bitmap fonts are simplest to define and display. The character grid only needs to be mapped

in a frame buffer. But bitmap fonts are require more space because each variation must be stored in a separate font cache.

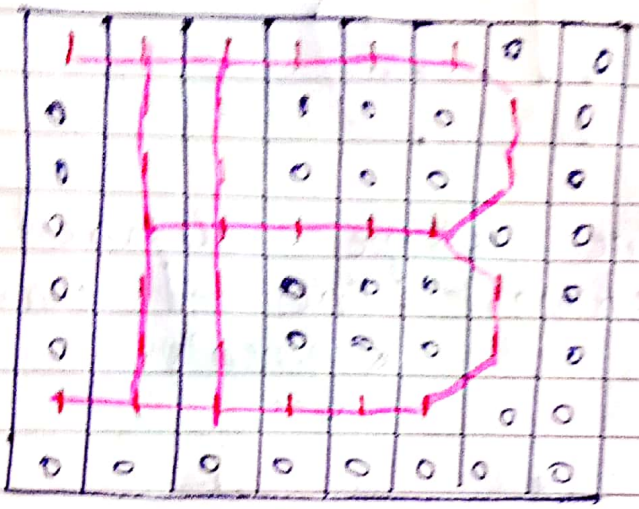


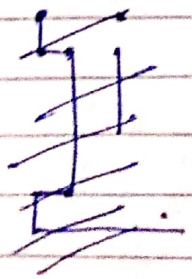
fig (a)

The letter 'B' represented in fig (a) with 8x8 bilevel bitmap pattern

(b) Outline - font :-

In outline font- we use display file to keep information of the font- in form of instructions.

It requires less storage space with comparision to bitmap font- but- it take more time in processing.



Page No.
Date

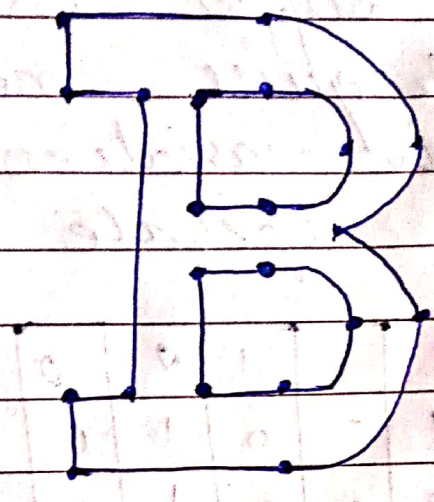


fig 1b)

- outline shape defined with straight-line and curve segments.