UNIT-1

① **Frame Buffer:-**

A frame buffer is a large contiguous piece of computer memory which stores the pixel value and place into an array. The frame buffer is commonly called a bitmap.

② **Pixels:-**

Each screen point is refered ~~as~~ to as a pixel or pel (shortened forms of picture element). Each pixel on the screen can be specified by its row and column number.

③ **Vector Generation:-**

The process of 'turning on' the pixels for a line segment is called vector generation. ~~In this~~ For vector generation different algorithms used are - Bresenham's line Algorithm and circle drawing

④ **Character Generation:-**

~~There~~ letters, numbers, and other characters are generate or displayed to label and annotate drawing and to give instructions and information to the user. There are basic three methods of generating characters are:-

Stroke method, Starburst method and bitmap method.

Stroke method:-

This method uses small line segments to generate a character. The small series of ~~the~~ line segments are drawn like a stroke of pen to form a character.

Starburst method:-

In this method, a fix pattern of line segments are used to generate characters. Out of these 24 line segments, segments required to display for particular character are highlighted.

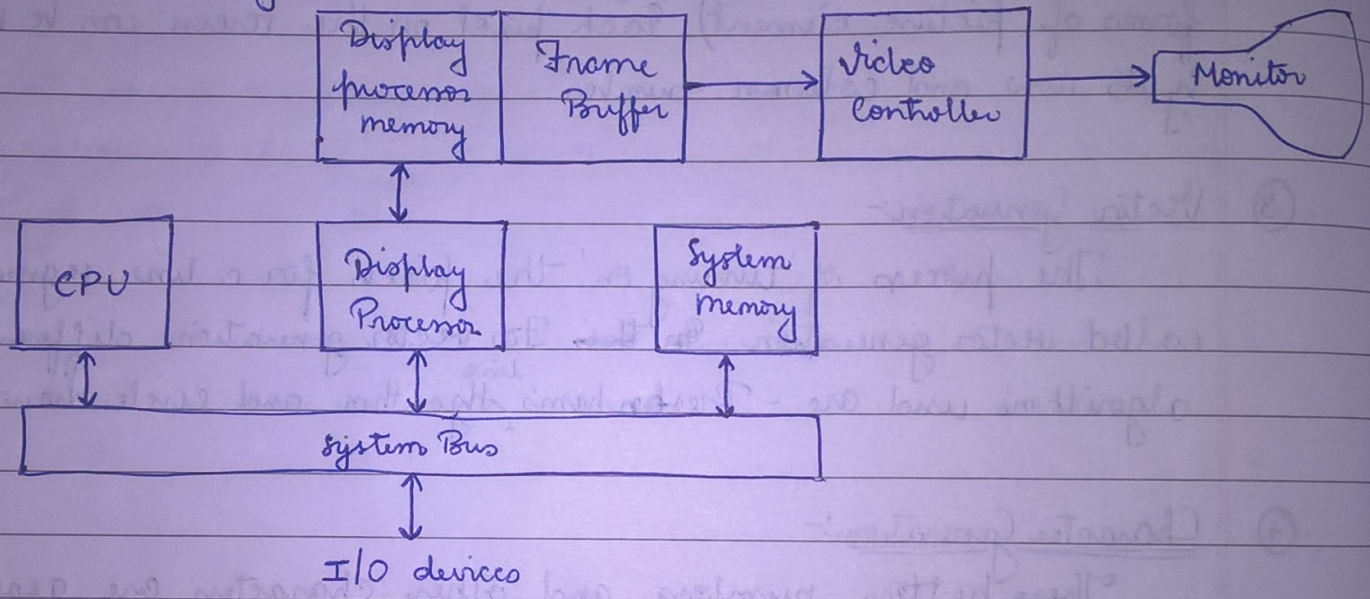
Aspect Ratio = $\frac{H}{V} = \frac{\text{Row}}{\text{Column}}$; Size of frame buffer = Resolution × Bits/pixel

1 byte = 8 bits

Bitmap method :-

Bitmap method also called dot matrix method because in this method characters are represented by an array of dots in the matrix form. It is a two dimensional array having columns and rows. An 5×7 array is commonly used to represent characters.

(5) Raster scan System :-

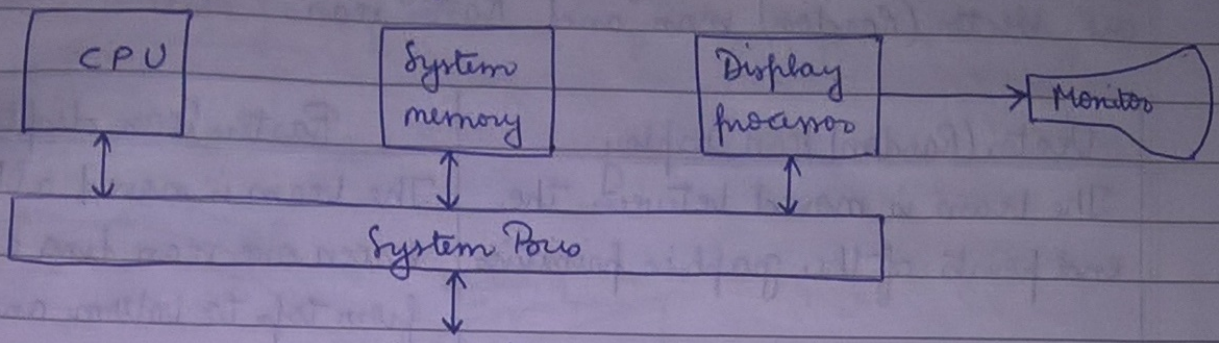


In raster scan display systems, the purpose of display processor is to free the CPU from the graphics routine task. Here, display processor is provided with separate memory area. The main task of display processor is to digitize a picture definition given in an application program into a set of pixel-intensity values for storage in the frame buffer. The digitization process is known as scan conversion.

Display processor are also designed to perform a number of additional operations. These operations include:

→ Generating various line styles (dashed, dotted, or solid)
→ Display colour areas
→ Performing certain transformations
→ Manipulations on displayed objects.

⑥ Random (Vector) Scan System :-



An application program is input and stored in the system memory along with a graphics package. Graphics commands in the application program are translated by the graphics package into a display file stored in the system memory. This display file is then accessed by the display processor to refresh the screen. The display processor cycles through each command in the display file program once during every refresh cycle.

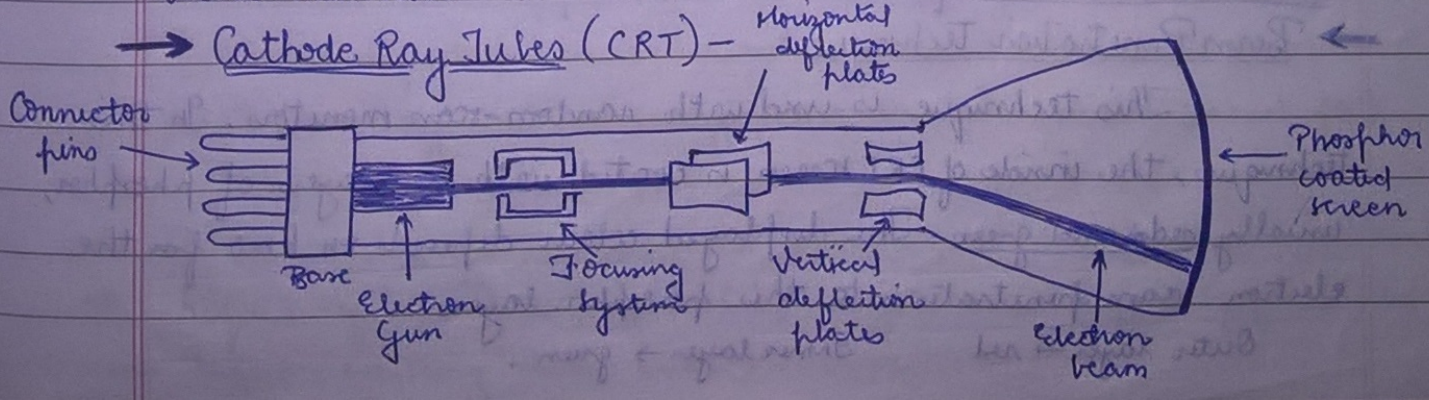
⑦ Scan Conversion Techniques :-

(1) Simple DDA line algorithm
(2) Bresenham's Algorithm for line and circle
(3) Midpoint Circle algorithm

⑧ Display Devices :-

The display device are also known as output devices. The most commonly used output device in a graphic system is a video monitor.

→ Cathode Ray Tubes (CRT) –



Two techniques used for producing images on the CRT screen are Vector (Random) scan and Raster scan.

Vector (Random) Scan Display	Raster Scan display
The beam is moved between the end points of the graphics primitives	The beam is moved all over the screen one scan line at a time, from top to bottom and then back to top.
It flickers when the number of primitives in the buffer becomes too large	The refresh process is independent of the complexity of the image.
Scan conversion is not required.	Scan conversion is required.
It draws a continuous and smooth lines	It can display mathematically smooth lines, polygons, and boundaries of curved primitives only by approximating them with pixels on the raster grid.
Cost is more	Cost is low
It only draws lines and characters	It has ability to display areas filled with solid colours or patterns

Two basic techniques used for producing colour displays are beam penetration technique and shadow-mask technique.

~~Beam Penetration technique — see mobile pic~~

~~Shadow Mask Technique — see mobile pic~~

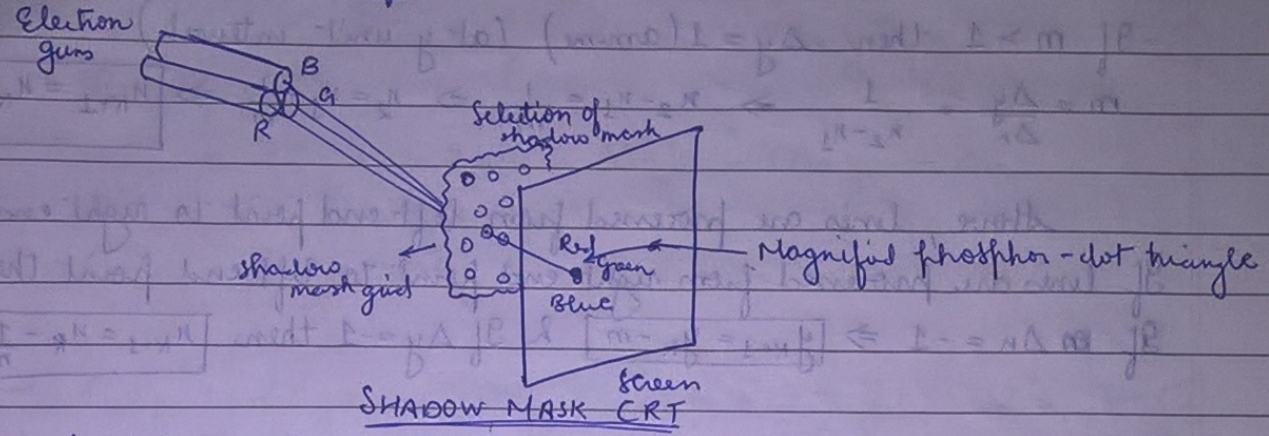
→ Beam Penetration Technique :

This technique is used with random-scan monitors. In this technique, the inside of CRT screen is coated with two layers of phosphor, usually red and green. The displayed colour depends on how far the electron beam penetrates into the phosphor layers

Outer layer → red Inner layer → green.

Merits → inexpensive technique to produce colour in random scan monitors
Demerits → display only four colours, quality of picture is not good.

→ Shadow Mask Technique -



SHADOW MASK CRT

Used in raster-scan display, have wide range of colours

→ Direct View Storage tubes - (DVST)

It stores the picture information as a charge distribution just behind the phosphor-coated screen. Two electron guns are used in a DVST. One, the primary gun, is used to store the picture pattern; the second, the flood gun, maintains the picture display.

Advantages -
(1) Refreshing of CRT is not required
(2) Because no refreshing is required, very complex pictures can be displayed at very high resolution without flicker

Disadvantages -
(1) They ordinarily do not display colour and that selected parts of a picture cannot be erased
(2) To eliminate a picture section, the entire screen must be erased and the modified picture redrawn
(3) The erasing and redrawing process can take several seconds for a complex picture

→ Flat Panel display → reduced volume, weight and power requirements
→ liquid-crystal displays (LCD) → crystalline arrangement of molecules
→ light-emitting diode (LED)

Line Drawing -

① Simple DDA (Digital Differential Analyzer) Algorithm -

When slop $m \leq 1$ then $\Delta x = 1$ (assume) (at x unit intervals)

$$m = \frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{1} \Rightarrow y_2 = y_1 + m \Rightarrow \boxed{y_{k+1} = y_k + m}$$

If $m > 1$ then $\Delta y = 1$ (assume) (at y unit intervals)

$$m = \frac{\Delta y}{\Delta x} = \frac{1}{x_2 - x_1} \Rightarrow x_2 - x_1 = \frac{1}{m} \Rightarrow x_2 = x_1 + \frac{1}{m} \Rightarrow \boxed{x_{k+1} = x_k + \frac{1}{m}}$$

Above, lines are processed from left end point to right end point.
If lines are processed from right end point to left end point then,
If $\Delta x = -1 \Rightarrow \boxed{y_{k+1} = y_k - m}$ & If $\Delta y = -1$ then $\boxed{x_{k+1} = x_k - \frac{1}{m}}$

Procedure -

STEP-1 - Create or get the end point coordinate (x_1, y_1) and (x_2, y_2).

STEP-2 - Calculate $\Delta x = x_2 - x_1$ and $\Delta y = y_2 - y_1$.

STEP-3 - If $\Delta x \geq \Delta y$ then step $= \Delta x$
else step $= \Delta y$.

STEP-4 - Calculate $x_{inc} = \Delta x / step$ and $y_{inc} = \Delta y / step$

STEP-5 - Set pixel with colour setpixel (x, y, color);

STEP-6 - Calculate next coordinate until (x_2, y_2) is reached using the formula,

$\boxed{x_{k+1} = x_k + x_{inc}}$ and $\boxed{y_{k+1} = y_k + y_{inc}}$

② Bresenham's line Algorithm -

Since, $y = m x_{k+1} + b$

Also, $d_1 = y - y_k$, & $d_2 = y_{k+1} - y$.

separation ← $\Rightarrow d_1 - d_2 = (y - y_k) - (y_{k+1} - y) \Rightarrow d_1 - d_2 = 2y - y_{k+1} - y_k$

Decision Parameter, $P_k = \Delta x (d_1 - d_2)$

$\Rightarrow P_k = \Delta x (2y - y_{k+1} - y_k)$

$= 2\Delta x y - \Delta x y_{k+1} - \Delta x y_k$

$= 2\Delta x (m x_{k+1} + b) - \Delta x y_{k+1} - \Delta x y_k$

$= 2\Delta x \left[\frac{\Delta y}{\Delta x}(x_k + 1) + b\right] - \Delta x (y_k + 1) - \Delta x y_k$

Therefore, $P_{k+1} = \Delta x (2 \cdot y \Delta x -$

$\Rightarrow$ ~~$P_K = 2\Delta y\, x_K$~~

$\Rightarrow P_K = 2\Delta x\,(m x_{K+1} + b) - \Delta x\, y_{K+1} - \Delta x\, y_K$

$\Rightarrow P_K = 2\Delta x\left(\frac{\Delta y}{\Delta x}(x_K + 1) + b\right) - \Delta x\,(y_{K+1}) - \Delta x\, y_K$

$\Rightarrow P_K = 2\Delta y\, x_K + 2\Delta y + 2\Delta x b - 2\Delta x\, y_K - \Delta x$ — (1)

Putting $K = 0$,

$\Rightarrow P_0 = 2\Delta y\, x_0 + 2\Delta y + 2\Delta x b - 2\Delta x\, y_0 - \Delta x$

Also, $x_0 = 0$, $y_0 = 0$, $b = 0$.

$\Rightarrow \boxed{P_0 = 2\Delta y - \Delta x}$

$\rightarrow P_{K+1} = 2\Delta y\, x_{K+1} + 2\Delta y + 2\Delta x b - 2\Delta x\, y_{K+1} - \Delta x$ — (2)

Therefore,

$P_{K+1} - P_K = 2\Delta y\, x_{K+1} - 2\Delta x\, y_{K+1} - 2\Delta y\, x_K + 2\Delta x\, y_K$ (from (1) & (2))

$\Rightarrow P_{K+1} - P_K = 2\Delta y(x_K + 1) - 2\Delta x(y_K + 1) - 2\Delta y\, x_K + 2\Delta x\, y_K$

$\Rightarrow P_{K+1} - P_K = 2\Delta y - 2\Delta x$

$\Rightarrow \boxed{P_{K+1} = P_K + 2\Delta y - 2\Delta x}$

Procedure -

STEP-1 - Input end point coordinates (x_1, y_1)

STEP-2 - Calculate Δx, Δy, $2\Delta x$ & $2\Delta y$ where $\Delta x = x_2 - x_1$ & $\Delta y = y_2 - y_1$

STEP-3 - $P_0 = 2\Delta y - \Delta x$.

STEP-4 - Repeat this step until (x_2, y_2) is reached,

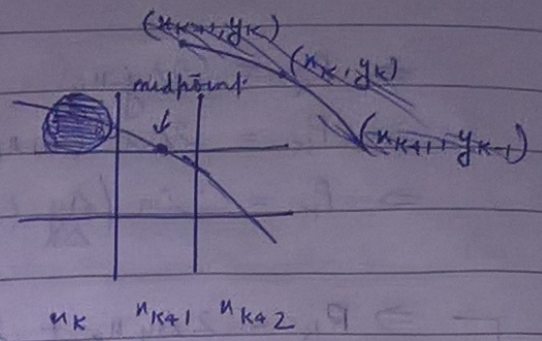
If $P_K < 0$ then plot (x_{K+1}, y_K) & calculate $\boxed{P_{K+1} = P_K + 2\Delta y}$

If $P_K \geq 0$ then plot (x_{K+1}, y_{K+1}) & calculate $\boxed{P_{K+1} = P_K + 2\Delta y - 2\Delta x}$

Circle Drawing Algorithms -

① Midpoint Circle Drawing Algorithm -

Two point can plotted next that is (x_{k+1}, y_k) and (x_{k+1}, y_{k-1})



$$\text{Midpoint} \equiv \left(\frac{x_{k+1}+x_{k+1}}{2}, \frac{y_k + y_{k-1}}{2}\right)$$

$$\equiv \left(x_k + 1, y_k - \frac{1}{2}\right)$$

$$f_{circle} \equiv x^2 + y^2 - r^2$$

$$\Rightarrow (x_k+1)^2 + \left(y_k - \frac{1}{2}\right)^2 - r^2$$

Putting $x_k = 0$ & $y_k = r$

$$P_0 \equiv (1)^2 + \left(r - \frac{1}{2}\right)^2 - r^2$$

$$P_0 = 1 + r^2 - r + \frac{1}{4} - r^2$$

$$\Rightarrow \boxed{P_0 = \frac{5}{4} - r}$$ For r an integer $\Rightarrow \boxed{P_0 = 1 - r}$

Procedure -

STEP 1 - Input radius r and circle centre (x_c, y_c) and obtain the first point on the circumference of a circle centered on the origin as

$$(x_0, y_0) = (0, r)$$

STEP 2 - Calculate $P_0 = \frac{5}{4} - r$

STEP 3 - Repeat this step until $x \geq y$.

If $P_k < 0$ then next point along the circle centered on $(0,0)$ is (x_{k+1}, y_k) and $\boxed{P_{k+1} = P_k + 2x_k + 3}$

else if $P_k \geq 0$ then next point along the circle centered on $(0,0)$ is (x_{k+1}, y_{k-1}) and $\boxed{P_{k+1} = P_k + 2x_k - 2y_k + 5}$

& Plot calculated point (x, y) onto the circular path centered on (x_c, y_c) is : PLOT $(x + x_c, y + y_c)$

② Bresenham's Algorithm -

$f_{circle} \equiv x^2 + y^2 - r^2 = 0 \Rightarrow y^2 = r^2 - x^2$ where $x = x_{k+1} \Rightarrow \boxed{y^2 = r^2 - x_{k+1}^2}$

Also, $\boxed{d_1 = y_k^2 - y^2}$ and $\boxed{d_2 = y^2 - y_{k-1}^2}$

$$d_1 - d_2 = y_k^2 - y^2 - (y^2 - y_{k-1}^2)$$
$$= y_k^2 + y_{k-1}^2 - 2y^2$$
$$= y_k^2 + (y_k - 1)^2 - 2(r^2 - x_{k+1}^2)$$
$$= y_k^2 + y_k^2 + 1 - 2y_k - 2r^2 + 2x_{k+1}^2$$

$$\Rightarrow d_1 - d_2 = -2y_k + 2x_{k+1}^2 - 2r^2 + 1 + 2y_k^2$$

Therefore, $P_k = -2y_k + 2x_{k+1}^2 - 2r^2 + 1 + 2y_k^2$

$$\Rightarrow P_k = -2y_k + 2(x_k + 1)^2 - 2r^2 + 1 + 2y_k^2$$

Putting $k = 0$ and $x_0 = 0$, $y_0 = r$

$$\Rightarrow P_0 = -2r + 2 - 2r^2 + 1 + 2r^2$$

$$\Rightarrow \boxed{P_0 = 3 - 2r}$$

Procedure -

STEP 1 - Input radius r and circle centre (x_c, y_c) and obtain the first point on the circumference of a circle centered on the origin as

$$(x_0, y_0) = (0, r)$$

STEP 2 - Calculate $P_0 = 3 - 2r$

STEP 3 - Repeat this step until $x \geq y$

If $P_k < 0$ then next point along the circle centered on $(0,0)$ is (x_{k+1}, y_k) and $\boxed{P_{k+1} = P_k + 4x_k + 6}$

else if $P_k \geq 0$ then next point along the circle centered on $(0,0)$ is (x_{k+1}, y_{k-1}) and $\boxed{P_{k+1} = P_k + 4x_k - 4y_k + 10}$

Plot calculated point (x, y) onto the circular path centered on (x_c, y_c) is $\text{PLOT}(x + x_c, y + y_c)$.

Polygon fill algorithm -

① Boundary-fill algorithm -

In filling start at a point inside a region and paint the interior outward toward the boundary. If the boundary is specified in a single colour, the fill algorithm proceeds outward pixel by pixel until the boundary colour is encountered.

```
void boundaryFill (int x, int y, int fill, int boundary)
{
  int current = getpixel (x, y);
  if ((current != boundary) && (current != fill))
    {
      setColor (fill);
      setPixel (x, y);
      boundaryFill (x+1, y, fill, boundary);
      boundaryFill (x-1, y, fill, boundary);
      boundaryFill (x, y+1, fill, boundary);
      boundaryFill (x, y-1, fill, boundary);
    }
}
```

② Flood-Fill algorithm -

We can paint areas by replacing a specified interior color instead of searching for a boundary color value.

```
void floodFill (int x, int y, int fillColor, int oldColor)
{
  if (getpixel (x, y) == oldColor)
    {
      setColor (fillColor);
      setPixel (x, y);
      floodFill (x+1, y, fillColor, oldColor);
      floodFill (x-1, y, fillColor, oldColor);
      floodFill (x, y+1, fillColor, oldColor);
      floodFill (x, y-1, fillColor, oldColor);
    }
}
```