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. Pixel:
   Each screen point is referred to as a pixel or pel (shortened form 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 line algorithms used are - Bresenham's Algorithm and circle drawing.

4. Character Generation:
   These letters, numbers, and other characters are generated and displayed to label and annotate drawing and to give instructions and information to the user. There are three basic 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 full pattern of line segments are used to generate characters. Out of these 24 line segments, segments required to display for particular character are highlighted.

\[
\text{Aspect Ratio} = \frac{H}{V} = \frac{\text{Row}}{\text{Column}}
\]

\[
\text{Page of frame buffer} = \text{Resolution} \times \text{Bits/pixel}
\]

1 byte = 8 bits
Bitmap method:

Bitmap method is 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 5x7 array is commonly used to represent characters.

Raster scan system:

In raster scan display systems, the purpose of display processor is to free the CPU from the graphics routine task. The display processor is provided in 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 processors are also designed to perform a number of additional operations. These operations include:

- Generating various line styles (dashed, dotted, or solid)
- Displaying colour areas
- Performing certain transformations
- Manipulations on displayed objects
Random (Vector) Scan System

CPU

System memory

Display processor

Monitor

System Port

I/O devices

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.

Scan Conversion Techniques:
1. Simple DDA line algorithm
2. Bresenham's Algorithm for line and circle
3. Midpoint Circle algorithm

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) -

Connector tubes

Horizontal deflection plates

Focus gun

Vertical deflection plates

Electron beam

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

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

Two basic techniques used for producing colour displays are Beam Penetration Technique and Shadow-mask Technique.

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 phosphors, usually red and green. The displayed colour depends on how for the electron beam penetrates into the phosphor layers:

Outer layer → red

Inner layer → green.
Mist → inexpensive technique to produce colour in random scan monitors

Display only few colours, quality of picture is not good.

→ Shadow Mask Technique

![Diagram of Shadow Mask Technique]

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

→ Direct View Storage Tube (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 part 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 redraw process can take several seconds for a complete picture.

Flat Panel display → reduced volume, weight and power requirement

Liquid crystal displays (LCD) → crystalline arrangement of molecules

Light emitting diode (LED)
Line Drawing -

1. Simple DDA (Digital Differential Analyzer) Algorithm:
   When step \( m \leq 1 \) then \( \Delta x = 1 \) (assume) (at \( n \) unit intervals)
   \[
   m = \frac{\Delta y}{\Delta x} = \frac{y_{f} - y_{0}}{1} \quad \Rightarrow \quad y_{2} = y_{1} + m \quad \Rightarrow \quad y_{k+1} = y_{k} + m
   \]
   \[\text{If } m > 1 \text{ then } \Delta y = 1 \text{ (assume) (at } y \text{ unit intervals)}
   \[
   m = \frac{\Delta y}{\Delta x} = \frac{1}{x_{2} - x_{1}} \quad \Rightarrow \quad x_{2} = x_{1} + \frac{1}{m} \quad \Rightarrow \quad x_{k+1} = x_{k} + \frac{1}{m}
   \]

   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 \( \Delta x = -1 \Rightarrow y_{k+1} = y_{k} - m \) & if \( \Delta y = -1 \) then \[x_{k+1} = x_{k} - 1\]

   Procedure-
   \[\text{STEP-1} \quad \text{Create or get the end point coordinate } (x_{0}, y_{0}) \text{ and } (x_{f}, y_{f}).\]
   \[\text{STEP-2} \quad \text{Calculate } \Delta x = x_{f} - x_{0} \text{ and } \Delta y = y_{f} - y_{0}.\]
   \[\text{STEP-3} \quad \text{If } \Delta x \geq \Delta y \text{ then } \text{Step } = \Delta x, \text{ else } \text{Step } = \Delta y.\]
   \[\text{STEP-4} \quad \text{Calculate } x_{\text{inter}} = \Delta x / \text{Step} \text{ and } y_{\text{inter}} = \Delta y / \text{Step}\]
   \[\text{STEP-5} \quad \text{Set pixel with color setpixel } (x_{0}, y_{0}, \text{color});\]
   \[\text{STEP-6} \quad \text{Calculate next coordinate until } (x_{f}, y_{f}) \text{ is reached using the formula,}\]
   \[
   x_{k+1} = x_{k} + x_{\text{inter}} \quad \text{and } y_{k+1} = y_{k} + y_{\text{inter}}
   \]

2. Bresenham's Line Algorithm -
   Since, \( y = mx + b \)
   \[
   d_{1} = y_{f} - y_{0} \quad \text{and} \quad d_{2} = y_{f+1} - y_{0}
   \]
   \[\Rightarrow \quad d_{1} - d_{2} = (y_{f} - y_{0}) - (y_{f+1} - y_{0}) \Rightarrow \quad d_{1} - d_{2} = 2y - y_{f+1} - y_{0}\]
   Decision Parameter, \( P_{k} = \Delta x (d_{1} - d_{2}) \)
   \[
   \Rightarrow P_{k} = \Delta x (2y - y_{f+1} - y_{0}) = 2\Delta x y - \Delta x y_{f+1} - \Delta x y_{0}
   \]
   \[
   = 2\Delta x (y_{k+1} - y_{0}) - \Delta x y_{f+1} - \Delta x y_{0}
   \]
   \[\Rightarrow P_{k+1} = \Delta x (2y_{k+1} - y_{0}) - \Delta x y_{f+1} - \Delta x y_{0}\]
   \[\text{Therefore, } \frac{d_{1} - d_{2}}{2} \geq \frac{P_{k+1}}{\Delta x}\]
   (10)
\[ P_k = 2A \gamma \nu_k \]

\[ P_k = 2A \gamma (n_{k+1} + b) = A \gamma (y_{k+1} + \Delta y) \]

\[ P_k = 2A \gamma (\Delta y_{k+1} + b) = A \gamma (y_{k+1}) - A \gamma y_k \]

\[ P_k = 2A \gamma y_k + 2A \gamma + 2A \gamma b - 2A \gamma y_k - A \gamma = 0 \]

\[ P_0 = 2A \gamma y_0 - A \gamma \]

\[ \begin{align*}
P_{k+1} &= 2A \gamma y_{k+1} + 2A \gamma + 2A \gamma b - 2A \gamma y_{k+1} - A \gamma \\
&= 2A \gamma y_k + 2A \gamma - 2A \gamma
\end{align*} \]

Therefore,

\[ P_{k+1} = P_k + 2A \gamma - 2A \gamma \]

**Procedure**

**Step 1** - Input end point coordinates \((x_1, y_1)\)

**Step 2** - Calculate \(\Delta x, \Delta y, 2A \gamma, 2A \gamma\) where \( \Delta x = x_2 - x_1, \Delta y = y_2 - y_1 \)

**Step 3** - \( P_0 = 2A \gamma - A \gamma \).

**Step 4** - Repeat this step until \((x_2, y_2)\) is reached.

If \( P_k < 0 \), then plot \((x_{k+1}, y_{k+1})\) & calculate \( P_{k+1} = P_k + 2A \gamma \) unless \( P_{k+1} < 0 \), then plot \((x_{k+1}, y_{k+1})\) & calculate \( P_{k+1} = P_k + 2A \gamma - 2A \gamma \).
Circle Drawing Algorithms:

1. **Midpoint Circle Drawing Algorithm**
   
   Two points can be plotted next that is $y_k$ $(x_{k+1}, y_{k+1})$ and $(x_{k+1}, y_{k-1})$
   
   \[
   \text{Midpoint} = \left( \frac{x_{k+1} + x_k}{2}, \frac{y_{k+1} + y_k}{2} \right) = (x_k, y_k)
   \]
   
   \[
   y_{k+1} = y_k - 1
   \]
   
   \[
   f_{\text{circle}} = x^2 + y^2 - r^2
   \]
   
   \[
   \Rightarrow \left( \frac{x_k + 1}{2} \right)^2 + \left( \frac{y_k - 1}{2} \right)^2 - r^2 = 0
   \]
   
   Putting $x_k = 0$ & $y_k = r$
   
   \[
   P_0 = 1 - \frac{r^2}{4}
   \]
   
   For $n$ an integer $\Rightarrow P_0 = 1 - r$
   
   **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 $n = y$

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

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

   Plot calculated point $(x, y)$ onto the circular path centered on $(x_c, y_c)$

   $\Rightarrow P(\frac{x+x_c}{2}, \frac{y+y_c}{2})$
Bresenham’s Algorithm -

\[ \frac{y_0^2}{x_0} = x_0^2 - r^2 \] 

when \( n = n_{k+1} \) \[ \frac{y_k^2}{x_k} = \frac{y_k^2}{x_k} \] 

Min. \( d_1 = y_k^2 - y_k^2 \) and \( d_2 = y_k^2 - y_{k-1}^2 \)

\[ d_1 = d_2 = y_k^2 - y_k^2 - \left( y_{k-1}^2 - y_{k-1}^2 \right) \]

\[ = y_k^2 + y_{k-1}^2 - 2y_k^2 \]

\[ = y_k^2 + y_{k-1}^2 - 2y_k^2 = 2 \left( y_k^2 - y_{k-1}^2 \right) \]

\[ = y_k^2 + y_{k-1}^2 + 2y_k^2 - 2y_k^2 + 2y_{k-1}^2 \]

\[ \Rightarrow d_1 - d_2 = -2y_k + 2x_{k-1} - 2r + 2 + 12y_k^2 \]

Therefore, \( P_k = \frac{2y_k^2 + 2x_{k-1}^2 + 1 + 2y_k^2}{4} \)

\[ \Rightarrow P_k = -2y_k + 2(x_{k-1} + 1)^2 - 2r^2 + 2 + 12y_k^2 \]

Putting \( k = 0 \) and \( x_0 = 0, y_0 = r \)

\[ P_0 = -2r + 2 - 2r^2 + 2 + 12y_k^2 \]

\[ \Rightarrow P_0 = 3 - 2r \]

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 \( n = y \)

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

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

Plot calculated point \((x, y)\) onto the circumeference path continued on \((x, y)\) to \( P = \text{point}(x + x_c, y + y_c) \).
Polygon fill algorithm:

1. **Boundary-fill algorithm**
   - Start at a point inside a region and paint the interior outward toward the boundary. If the boundary is specified in a single color, the fill algorithm proceeds outward pixel by pixel until the boundary color 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);
       }
   }
   ```