NOTES

SUBJECT: COMPUTER GRAPHICS

SUBJECT CODE: NCS-403

BRANCH: CSE SEM: 4th

SESSION: 2016-17

Asst. Prof. Ashutosh Pandey (CSE Department)
SIET, Allahabad.

CONTENT

2.7	Line clipping against non rectangular clip windows
2.8 2.8.1 2.8.2	 Polygon clipping - Sutherland Hodgeman polygon clipping Weiler and Atherton polygon clipping,
2.9	Curve clipping
2.10	Text clipping
UNIT III	
3.1 3.1.1 3.1.2 3.1.3 3.1.4 3.1.5	Three- Dimensional Concepts: Three Dimensional Display Methods Parallel Projection Perspective Projection Depth Cueing Visible Line and Surface Identification Surface Rendering Exploded and Cutaway Views Three-Dimensional and Stereoscopic views
3.2 3.2.1 3.2.2 3.2.3	3-D Object representation Polyfon Surfaces Polygon Tables Plane Equations Plane Meshes
3.3	3-D Geometric primitives and Modeling Transformation Translation, Rotation, Scaling, Reflection, Shearing
3.4 3.4.1 3.4.2 3.4.3	3-D Viewing Viewing Pipeline Viewing Cordinates Transformation from world to viewing Coordinates
3.5 3.5.1 3.5.2	Projections: Parallel Projection, Perspective Projection

3.6	3-D Clipping
UNIT IV	
4.1	Quadric Surfaces, Spheres, Ellipsoid, Blobby Objects
4.2	Introductory concepts of Spline,
4.3	Bezier curves and surfaces
4.4	Bspline curves and surfaces
4.5	Hidden Lines and Surfaces: Back Face Detection algorithm
4.6	Depth Buffer Method
4.7	A- Buffer Method
4.8	Scan line Method
4.9	Basic Illumination Models - Ambient light ,Diffuse reflection, Specular reflection and Phong model, Combined approach, Warn model
4.10	Intensity Attenuation
4.11	Color Consideration
4.12	Transparency and Shadows

SYLLABUS

L T P
2 1 0

Objectives : To study the concepts of Computer Graphics like graphics displays, the various areas where CG is applicable, brief idea about Line and circle algorithms. This course also explains about the 2D and 3D and Transformation, Curves and surfaces, which is the emerging area of computer science and IT..

Unit - I

Introduction and Line Generation: Types of computer graphics, Graphic Displays- Random scan displays, Raster scan displays, Frame buffer and video controller, Points and lines, Line drawing algorithms, Circle generating algorithms, Mid point circle generating algorithm, and parallel version of these algorithms.

Unit - II

Transformations: Basic transformation, Matrix representations and homogenous coordinates, Composite transformations, Reflections and shearing.Windowing and Clipping: Viewing pipeline, Viewing transformations, 2-D Clipping algorithms- Line clipping algorithms such as Cohen Sutherland line clipping algorithm, Liang Barsky algorithm, Line clipping against non rectangular clip windows; Polygon clipping - Sutherland Hodgeman polygon clipping, Weiler and Atherton polygon clipping, Curve clipping, Text clipping.

Unit - III

Three Dimensional: 3-D geometric primitives, 3-D Object representation, 3-D Transformation, 3-D viewing, projections, 3-D Clipping.

Unit - IV

Curves and Surfaces: Quadric surfaces, Spheres, Ellipsoid, Blobby objects, Introductory concepts of Spline, Bspline and Bezier curves and surfaces.Hidden Lines and Surfaces: Back Face Detection algorithm, Depth buffer method, A- buffer method, Scan line method, basic illumination models - Ambient light, Diffuse reflection, Specular reflection and Phong model, Combined approach, Warn model, Intensity Attenuation, Color consideration, Transparency and Shadows.

References:

1. Donald Hearn and M Pauline Baker, “Computer Graphics C Version”, Pearson Education
2. Amrendra N Sinha and Arun D Udai,” Computer Graphics”, TMH
3. Donald Hearn and M Pauline Baker, “Computer Graphics with OpenGL”, Pearson Education
4. Steven Harrington, “Computer Graphics: A Programming Approach” , TMH

1

Introduction to Computer Graphics

1.1 Introduction

The computer is an information processing machine. It is a tool for storing, manipulating and correlating data. There are many ways to communicate the processed information to the user. The computer graphics is one of the most effective and commonly used way to communicate the processed information to the user. It displays the information in the form of graphics objects such as pictures, charts, graphs and diagrams instead of simple text. Thus we can say that computer graphics makes it possible to express data in pictorial form. The picture or graphics object may be an engineering drawing, business graphs, architectural structures, a single frame from an animated movie or a machine parts illustrated for a service manual. It is the fundamental cohesive concept in computer graphics. Therefore, it is important to understand -

- How pictures or graphics objects are presented in computer graphics ?
- How pictures or graphics objects are prepared for presentation ?
- How previously prepared pictures or graphics objects are presented ?
- How interaction with the picture or graphics object is accomplished ?

In computer graphics, pictures or graphics objects are presented as a collection of discrete picture elements called pixels. The pixel is the smallest addressable screen element. It is the smallest piece of the display screen which we can control. The control is achieved by setting the intensity and colour of the pixel which compose the screen. This is illustrated in Fig. 1.1.

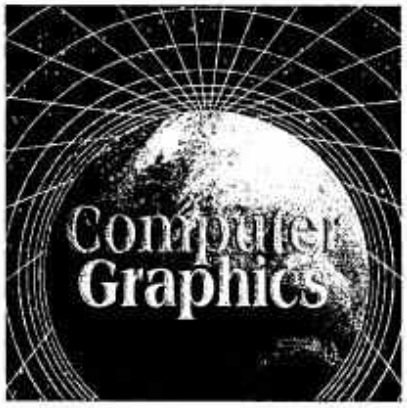


Fig. 1.1 Representation of picture

Each pixel on the graphics display does not represent mathematical point. Rather, it represents a region which theoretically can contain an infinite number of points. For example, if we want to display point P_1 whose coordinates are (4.2, 3.8) and point P_2

whose coordinates are (4.8, 3.1) then P_1 and P_2 are represented by only one pixel (4, 3), as shown in the Fig. 1.2. In general, a point is represented by the integer part of x and integer part of y, i.e., pixel (int (x), int (y)).

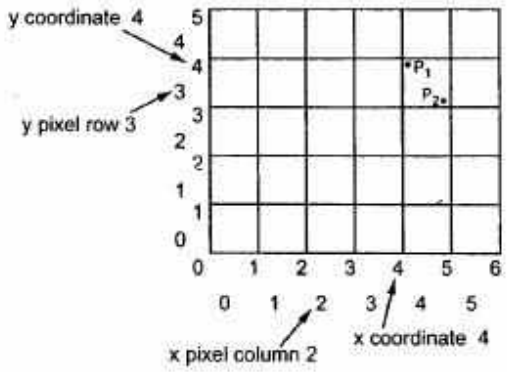


Fig. 1.2 Pixel display area of 6×5

The special procedures determine which pixel will provide the best approximation to the desired picture or graphics object. The process of determining the appropriate pixels for representing picture or graphics object is known as **rasterization**, and the process of representing continuous picture or graphics object as a collection of discrete pixels is called **scan conversion**.

The computer graphics allows rotation, translation, scaling and performing various projections on the picture before displaying it. It also allows to add effects such as hidden surface removal, shading or transparency to the picture before final representation. It provides user the control to modify contents, structure, and appearance of pictures or graphics objects using input devices such as a keyboard, mouse, or touch-sensitive panel on the screen. There is a close relationship between the input devices and display devices. Therefore, graphics devices includes both input devices and display devices.

1.2 Image Processing as Picture Analysis

The computer graphics is a collection, combination and representation of real or imaginary objects from their computer-based models. Thus we can say that computer graphics concerns the pictorial synthesis of real or imaginary objects. However, the related field image processing or sometimes called picture analysis concerns the analysis of scenes, or the reconstruction of models of 2D or 3D objects from their picture. This is exactly the reverse process. The image processing can be classified as

- Image enhancement
- Pattern detection and recognition
- Scene analysis and computer vision

The image enhancement deals with the improvement in the image quality by eliminating noise or by increasing image contrast. Pattern detection and recognition deal

with the detection and clarification of standard patterns and finding deviations from these patterns. The optical character recognition (OCR) technology is an practical example of pattern detection and recognition. Scene analysis and computer vision deals with the recognition and reconstruction of 3D model of scene from several 2D images.

The above three fields of image processing proved their importance in many area such as finger print detection and recognition, modeling of buildings, ships, automobiles etc., and so on.

We have discussed the two fields : computer graphics and image processing of computer processing of pictures. In the initial stages they were quite separate disciplines. But now a days they use some common features, and overlap between them is growing. They both use raster displays.

1.3 The Advantages of Interactive Graphics

Let us discuss the advantages of interactive graphics.

- Today, a high quality graphics displays of personal computer provide one of the most natural means of communicating with a computer.
- It provides tools for producing pictures not only of concrete, "real-world" objects but also of abstract, synthetic objects, such as mathematical surfaces in 4D and of data that have no inherent geometry, such as survey results.
- It has an ability to show moving pictures, and thus it is possible to produce animations with interactive graphics.
- With interactive graphics use can also control the animation by adjusting the speed, the portion of the total scene in view, the geometric relationship of the objects in the scene to one another, the amount of detail shown and so on.
- The interactive graphics provides tool called **motion dynamics**. With this tool user can move and tumble objects with respect to a stationary observer, or he can make objects stationary and the viewer moving around them. A typical example is walk throughs made by builder to show flat interior and building surroundings. In many case it is also possible to move both objects and viewer.
- The interactive graphics also provides facility called **update dynamics**. With update dynamics it is possible to change the shape, colour or other properties of the objects being viewed.
- With the recent development of digital signal processing (DSP) and audio synthesis chip the interactive graphics can now provide audio feedback alongwith the graphical feedbacks to make the simulated environment even more realistic.

In short, interactive graphics permits extensive, high-bandwidth user-computer interaction. It significantly enhances the ability to understand information, to perceive trends and to visualize real or imaginary objects either moving or stationary in a realistic environment. It also makes it possible to get high quality and more precise results and products with lower analysis and design cost.

1.4 Applications of Computer Graphics

The use of computer graphics is wide spread. It is used in various areas such as industry, business, government organisations, education, entertainment and most recently the home. Let us discuss the representative uses of computer graphics in brief.

- **User Interfaces :** User friendliness is one of the main factors underlying the success and popularity of any system. It is now a well established fact that graphical interfaces provide an attractive and easy interaction between users and computers. The built-in graphics provided with user interfaces use visual control items such as buttons, menus, icons, scroll bar etc, which allows user to interact with computer only by mouse-click. Typing is necessary only to input text to be stored and manipulated.
- **Plotting of graphics and chart :** In industry, business, government and educational organisations, computer graphics is most commonly used to create 2D and 3D graphs of mathematical, physical and economic functions in form of histograms, bars and pie-charts. These graphs and charts are very useful for decision making.
- **Office automation and Desktop Publishing :** The desktop publishing on personal computers allow the use of graphics for the creation and dissemination of information. Many organisations does the in-house creation and printing of documents. The desktop publishing allows user to create documents which contain text, tables, graphs and other forms of drawn or scanned images or pictures. This is one approach towards the office automation.
- **Computer-aided Drafting and Design :** The computer-aided drafting uses graphics to design components and systems electrical, mechanical, electromechanical and electronic devices such as automobile bodies, structures of building, airplane, ships, very large-scale integrated (VLSI) chips, optical systems and computer networks.
- **Simulation and Animation :** Use of graphics in simulation makes mathematic models and mechanical systems more realistic and easy to study. The interactive graphics supported by animation software proved their use in production of animated movies and cartoons films.
- **Art and Commerce :** There is a lot of development in the tools provided by computer graphics. This allows user to create artistic pictures which express messages and attract attentions. Such pictures are very useful in advertising.
- **Process Control :** By the use of computer now it is possible to control various processes in the industry from a remote control room. In such cases, process systems and processing parameters are shown on the computer with graphic symbols and identifications. This makes it easy for operator to monitor and control various processing parameters at a time.
- **Cartography :** Computer graphics is also used to represent geographic maps, weather maps, oceanographic charts, contour maps, population density maps and so on.

1.5 Classification of Applications

In the last section we have seen various uses of computer graphics. These uses can be classified as shown in the Fig. 1.3. As shown in the Fig. 1.3, the use of computer graphics can be classified according to dimensionality of the object to be drawn : 2D or 3D. It can also be classified according to kind of picture : Symbolic or Realistic. Many computer graphics

applications are classified by the type of interaction. The type of interaction determines the user's degree of control over the object and its image. In controllable interaction user can change the attributes of the images. Role of picture gives the another classification. Computer graphics is either used for representation or it can be an end product such as drawings. Pictorial representation gives the final classification of use computer graphics. It classifies the use of computer graphics to represent pictures such as line drawing, black and white, colour and so on.

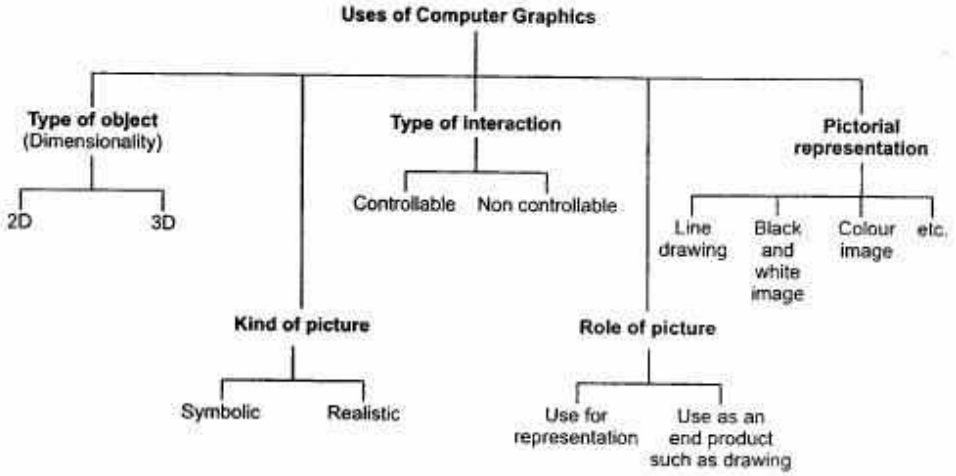


Fig. 1.3

1.6 Input Devices

Number of devices are available for data input in the graphics systems. These include keyboard, mouse, trackball, spaceball, joystick, digitizers, scanners and so on. Let us discuss them.

1.6.1 Keyboard

The keyboard is a primary input device for any graphics system. It is used for entering text and numbers, i.e. on graphics data associated with pictures such as labels x-y coordinates etc.

Keyboards are available in various sizes, shapes and styles. Fig. 1.4 shows standard keyboard. It consists of

- Alphanumeric key
- Function keys
- Modifier keys
- Cursor movement keys
- Numeric keypad

speech as opposed to single words or phrases. The speaker independent recognizers have very limited vocabularies. Usually, they include only the digits and 50 to 100 words.

The Fig. 1.13 shows the typical voice recognition system. In such systems microphone is also included to minimize input of other backgrond sounds.

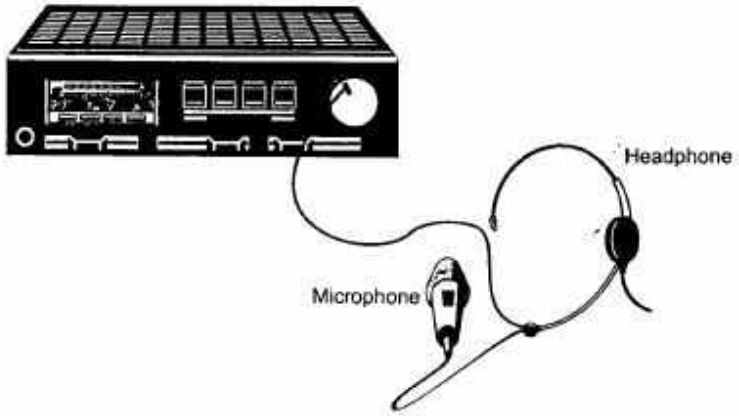


Fig. 1.13

The one advantage of voice system over other devices is that in voice systems the attention of the operator does not have to be switched from one device to another to enter a command.

1.7 Output Devices

The output devices can be classified as display devices and hardcopy devices. Let us see some of them.

1.7.1 Video Display Devices

The most commonly used output device in a graphics system is a video monitor. The operation of most video monitors is based on the standard **cathode-ray-tube** (CRT) design. Let us see the basics of the CRT.

1.7.1.1 Cathode-Ray-Tubes

A CRT is an evacuated glass tube. An electron gun at the rear of the tube produces a beam of electrons which is directed towards the front of the tube (screen). The inner side of the screen is coated with phosphor substance which gives off light when it is stroked by electrons. This is illustrated in Fig. 1.14. It is possible to control the point at which the electron beam strikes the screen, and therefore the position of the dot upon the screen, by deflecting the electron beam. The Fig. 1.15 shows the electrostatic deflection of the electron beam in a CRT.

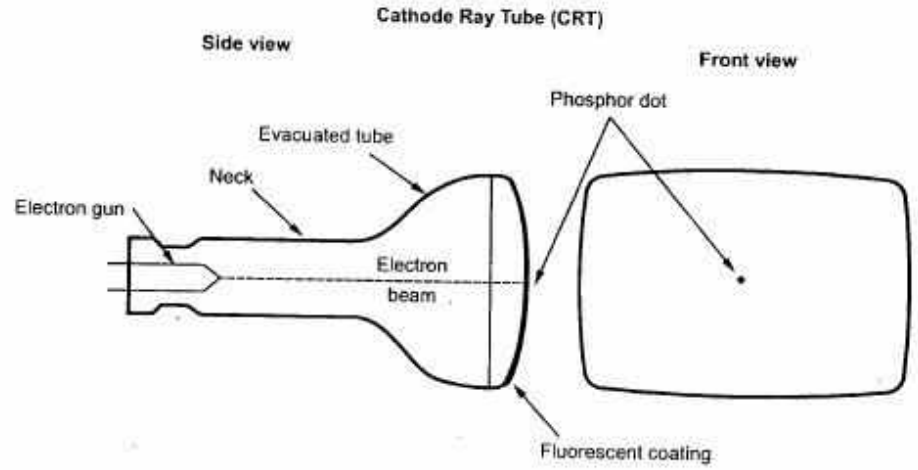


Fig. 1.14 Simplified representation of CRT

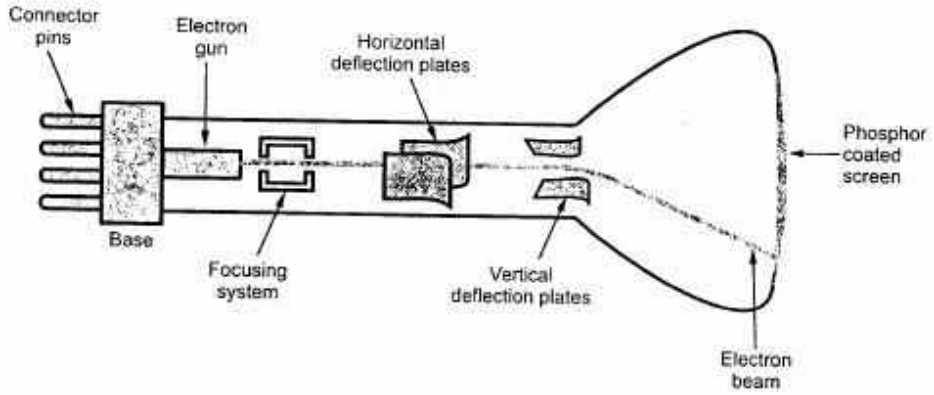


Fig. 1.15 Cathode Ray Tube

The deflection system of the cathode-ray-tube consists of two pairs of parallel plates, referred to as the vertical and horizontal deflection plates. The voltage applied to vertical plates controls the vertical deflection of the electron beam and voltage applied to the horizontal deflection plates controls the horizontal deflection of the electron beam. There are two techniques used for producing images on the CRT screen : Vector scan/random scan and Raster scan.

1.7.1.2 Vector Scan/Random Scan Display

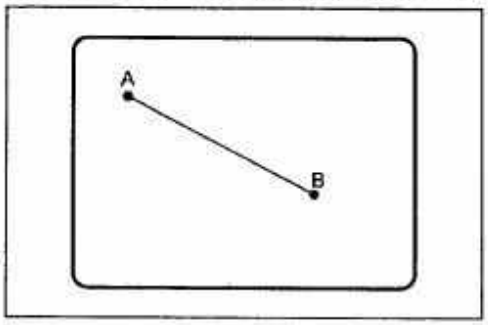


Fig. 1.16 Vector scan CRT

As shown in Fig. 1.16, vector scan CRT display directly traces out only the desired lines on CRT i.e. If we want a line connecting point A with point B on the vector graphics display, we simply drive the beam deflection circuitry, which will cause beam to go directly from point A to B. If we want to move the beam from point A to point B without showing a line between points, we can blank the beam as we move it. To move the beam across the CRT, the information about both, magnitude and direction is required. This information is generated with the help of vector graphics generator.

The Fig. 1.17 shows the typical vector display architecture. It consists of display controller, Central Processing Unit (CPU), display buffer memory and a CRT. A display controller is connected as an I/O peripheral to the central processing unit (CPU). The display buffer memory stores the computer produced display list or display program. The program contains point and line plotting commands with (x, y) or (x, y, z) end point coordinates, as well as character plotting commands. The display controller interprets commands for plotting points, lines and characters and sends digital and point coordinates to a vector generator. The vector generator then converts the digital coordinate values to analog voltages for beam-deflection circuits that displace an electron beam writing on the CRT's phosphor coating.

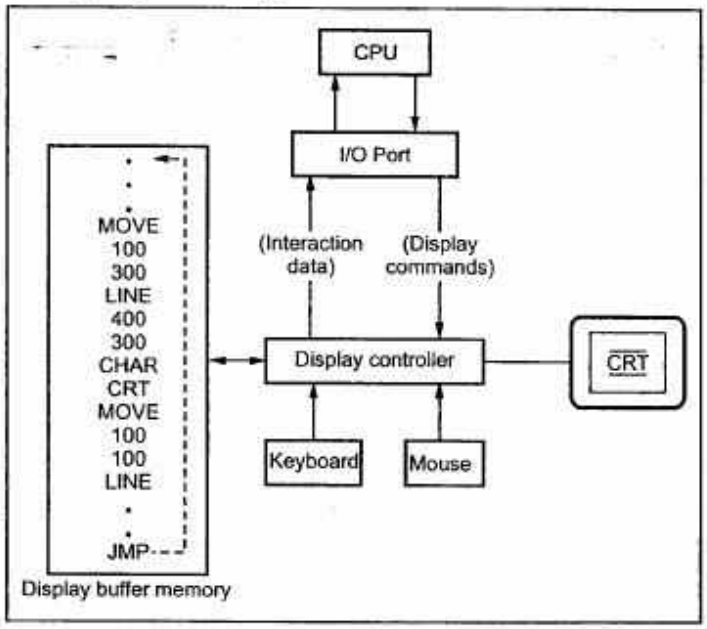


Fig. 1.17 Architecture of a vector display

In vector displays beam is deflected from end point to end point, hence this technique is also called **random scan**. We know as beam, strikes phosphor it emits light. But phosphor light decays after few milliseconds and therefore it is necessary to repeat through the display list to refresh the phosphor at least 30 times per second to avoid flicker. As display buffer is used to store display list and it is used for refreshing, the display buffer memory is also called **refresh buffer**.

1.7.1.3 Raster Scan Display

The Fig. 1.18 shows the architecture of a raster display. It consists of display controller, central processing unit (CPU), video controller, refresh buffer, keyboard, mouse and the CRT.

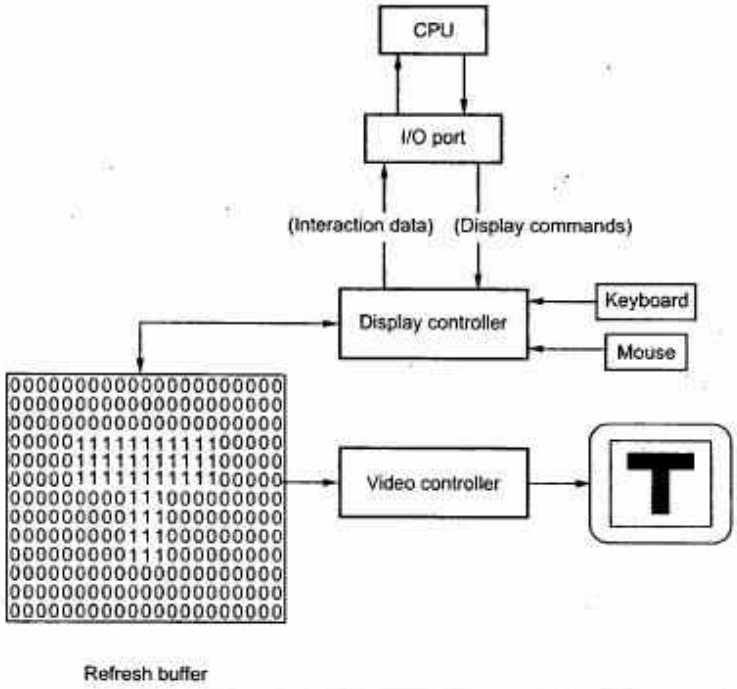


Fig. 1.18 Architecture of a raster display

As shown in the Fig. 1.18, the display image is stored in the form of 1s and 0s in the refresh buffer. The video controller reads this refresh buffer and produces the actual image on the screen. It does this by scanning one scan line at a time, from top to bottom and then back to the top, as shown in the Fig. 1.18.

Raster scan is the most common method of displaying images on the CRT screen. In this method, the horizontal and vertical deflection signals are generated to move the beam all over the screen in a pattern shown in the Fig. 1.19

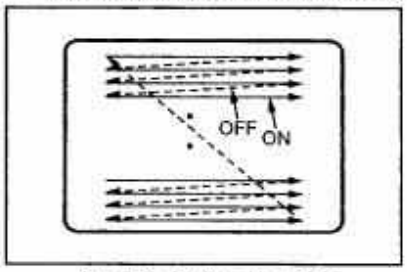


Fig. 1.19 Raster scan CRT

Here, the beam is swept back and forth from the left to the right across the screen. When the beam is moved from the left to the right, it is ON. The beam is OFF, when it is moved from the right to the left as shown by dotted line in Fig. 1.19

When the beam reaches the bottom of the screen, it is made OFF and rapidly retraced back to the top left to start again. A display produced in this way is called **raster scan display**. Raster scanning process is similar to reading different lines on the page of a book. After completion of scanning of one line, the electron beam flies back to the start of next line and process repeats. In the raster scan display, the screen image is maintained by repeatedly scanning the same image. This process is known as **refreshing of** screen.

In raster scan displays a special area of memory is dedicated to graphics only. This memory area is called **frame buffer**. It holds the set of intensity values for all the screen points. The stored intensity values are retrieved from frame buffer and displayed on the screen one row (scan line) at a time. Each screen point is referred 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. Thus by specifying row and column number we can specify the **pixel position** on the screen.

Intensity range for pixel positions depends on the capability of the raster system. It can be a simple black and white system or colour system. In a simple black and white system, each pixel position is either on or off, so only one bit per pixel is needed to control the intensity of the pixel positions. Additional bits are required when colour and intensity variations can be displayed. Upto 24 bits per pixel are included in high quality display systems, which can require several megabytes of storage space for the frame buffer. On a black and white system with one bit per pixel, the frame buffer is commonly called a **bitmap**. For systems with multiple bits per pixel, the frame buffer is often referred to as a **pixmap**.

Vector Scan Display	Raster Scan Display
1. In vector scan display the beam is moved between the end points of the graphics primitives.	1. In raster scan display the beam is moved all over the screen one scan line at a time, from top to bottom and then back to top.
2. Vector display flickers when the number of primitives in the buffer becomes too large.	2. In raster display, the refresh process is independent of the complexity of the image.
3. Scan conversion is not required.	3. Graphics primitives are specified in terms of their endpoints and must be scan converted into their corresponding pixels in the frame buffer.
4. Scan conversion hardware is not required.	4. Because each primitive must be scan-converted, real time dynamics is far more computational and requires separate scan conversion hardware.
5. Vector display draws a continuous and smooth lines.	5. Raster display can display mathematically smooth lines, polygons, and boundaries of curved primitives only by approximating them with pixels on the raster grid.
6. Cost is more.	6. Cost is low.
7. Vector display only draws lines and characters.	7. Raster display has ability to display areas filled with solid colours or patterns.

Table 1.1

Frame Buffer Organization

In raster scan displays a special area of memory is dedicated to graphics only. This memory area is called frame buffer. It holds the set of intensity values for all the screen points. The stored intensity values are retrieved from frame buffer and displayed on the screen one row (scan line) at a time.

Usually, frame buffer is implemented using rotating random access semiconductor memory. However, frame buffer also can be implemented using shift registers. Conceptually, shift register is operated as first-in, first-out (FIFO) fashion, i.e. similar to stack. We know that, when stack is full and if we want to add new data bit then first data bit is pushed out from the bottom and then the new data bit is added at the top. Here, the data pushed out of the stack can be interpreted as the intensity of a pixel on a scan line.

Fig. 1.20 shows the implementation of frame buffer using shift register. As shown in the Fig. 1.20, one shift register is required per pixel on a scan line and the length of shift register in bits is equal to number of scan lines. Here, there are 8 pixels per scan line and there are in all 5 scan lines. Therefore, 8 shift registers, each of 5 bit length are used to implement frame buffer. The synchronization between the output of the shift register and the video scan rate is maintained data corresponding to particular scan line is displayed correctly.

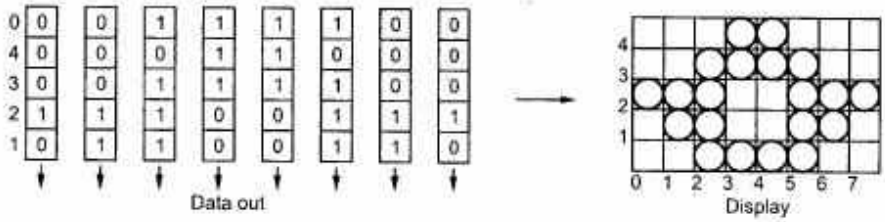


Fig. 1.20 Frame buffer using eight 5-bit shift registers

Both rotating memory and shift register frame buffer implementations have low levels of interactivity. The interactivity in rotating memory is limited due to disk access time and it is reduced in shift register implementations because changes can only be made as bits are being added to the register.

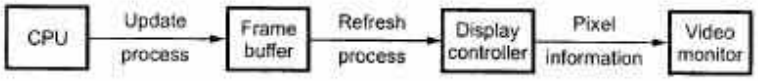
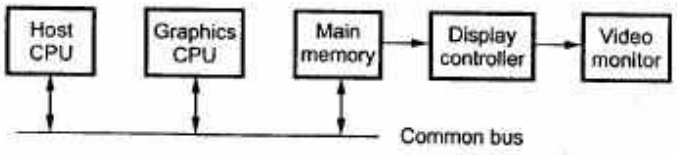


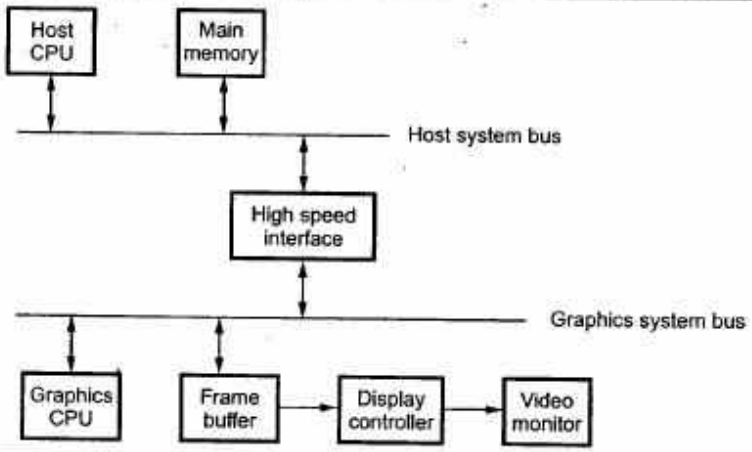
Fig. 1.21 Frame buffer graphics system

Fig. 1.21 shows the frame buffer graphics system. It consists of CPU, frame buffer, display controller and video monitor. An application program running in the computer updates the frame buffer as per the picture information. The display controller cycles through the frame buffer in scan line order (top to bottom) and passes the corresponding information to the video monitor to refresh the display. The frame buffer can be part of

computer memory itself or it can be implemented with separate memory as shown in the Fig. 1.22.



(a) Frame buffer as a part of computer memory



(b) Separate frame buffer

Fig. 1.22 Frame buffer architectures

Generally, separate graphics processor is used to improve the performance of graphics system. The graphics processor manipulates the frame buffer as per commands issued by main processor.

The performance of the graphics system is also affected by sharing of single memory done by two processors. The performance of the graphics system thus can be improved by having separate frame buffer memory as shown in the Fig. 1.22 (b).

Display File and its Structure

We know that in raster scan displays image information is stored in the frame buffer. It includes information of all pixels. On the other hand, the vector refresh displays store only the commands necessary for drawing the line segments. Here, input to the vector generator is stored instead of the output. The file used to store the commands necessary for drawing the line segments is called **display file**.

In vector refresh display system, display processor uses the information in the display file to draw lines with the help of vector generating algorithms. This is illustrated in Fig. 1.23. Therefore, we can say that display files provides an interface between the image specification process and the image display process. It also describes image in a compact

format. The concept of display file may be applied to devices other than refresh displays. Such files are called pseudo display files, or metafiles.

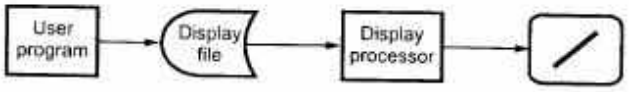


Fig. 1.23 Vector refresh display system

The Fig. 1.23 shows the structure of display file. We know that it contains series of commands. Each display file command contains two fields, an **operation code (opcode)** and **operands**. Opcode identifies the command such as draw line, move cursor etc., and operand provides the coordinates of a point to process the command.

One way to store opcode and operands of series of commands is to use three separate arrays. One for operation code, one for operand 1, i.e. x coordinate and one for operand2, i.e. y coordinate. This is illustrated in Fig. 1.24. The display file stores all commands to be needed to create a specified image. It is necessary to assign meaning to the possible operation codes before we can proceed to interpret them. Let us consider three commands MOVE, LINE and PLOT, and assign opcodes to these commands as shown in table 1.2.

Command	Opcode
MOVE	1
LINE	2
PLOT	3

Table 1.2

Once the opcodes are defined we can write commands needed to draw image into the display file. The following algorithm gives the steps required to insert command in the display file.

Algorithm

1. Read opcode, x and y coordinates.
2. Search for empty space in the display file. Let i be the empty row.
3. DF_OP [i] ← Opcode;
   DF_x [i] ← x ;
   DF_y [i] ← y ;
4. Stop

The table 1.3 shows the structure of display file with five commands in it, and Fig. 1.24 shows how these commands are interpreted and plotted.

DF_OP	DF_x	DF_y
Opcode	Operand 1	Operand 2
	x	y
1	30	30
2	50	80
2	70	30
1	40	55
2	60	55

Table 1.3 Display file structure

Command 1 : 1, 30, 30

It moves current cursor position to point (30, 30)

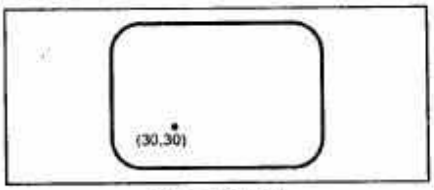


Fig. 1.24 (a)

Command 2 : 2, 50, 80

It draws the line between point (50, 80) and the current cursor position (30, 30)

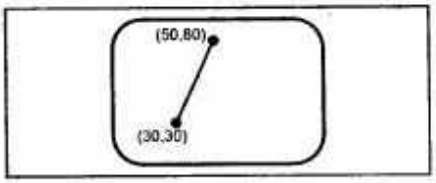


Fig. 1.24 (b)

Command 3 : 2, 70, 30

It draws the line between point (70, 30) and the current cursor position (50, 80)

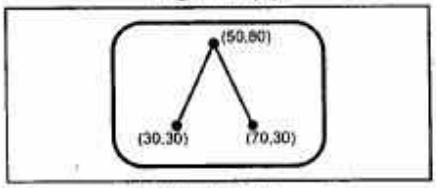


Fig. 1.24 (c)

Command 4 : 1, 40, 55

It moves current cursor position to point (40, 55)

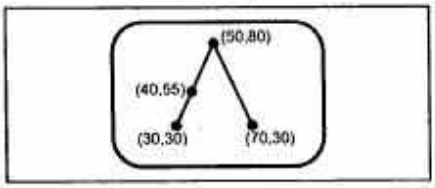


Fig. 1.24 (d)

Command 5 : 2, 60, 55

It draws the line between point (60, 55) and the current cursor position

(40, 55)

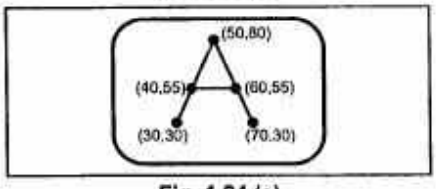


Fig. 1.24 (e)

Display File Interpreter

We have seen that display file contains information necessary to construct the picture. This information is in the form of commands. The program which converts these commands into actual picture is called display file interpreter. It is an interface between graphics representation in the display file and the display device, as shown in the Fig. 1.25.

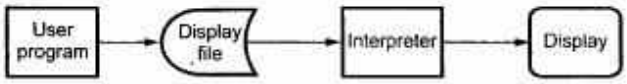


Fig. 1.25 Display file and interpreter

As shown the Fig. 1.25, the display process is divided into two steps : first the image is stored in the display file structure and then it is interpreted by an appropriate interpreter to get the actual image.

Instead of this, if we store actual image for particular display device it may not run on other displays. To achieve the device independence the image is stored in the raw format, i.e. in the display file format and then it is interpreted by an appropriate interpreter to run on required display.

Another advantage of using interpreter is that saving raw image takes much less storage than saving the picture itself.

Display Controller

In some graphics systems a separate computer is used to interpret the commands in the display file. Such computer is known as display controller. Display controller access display file and it cycles through each command in the display file once during every refresh cycle Fig. 1.26 shows the vector scan system with display controller.

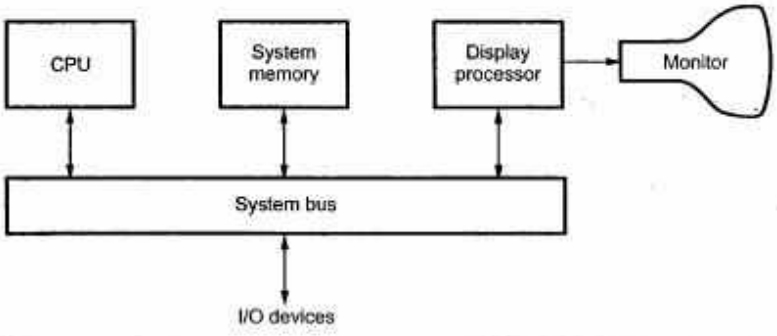


Fig. 1.26 Vector scan system

In the raster scan display systems, the purpose of display controller is to free the CPU from the graphics routine task. Here, display controller is provided with separate memory area as shown in the Fig. 1.26. The main task of display controller is to digitize a picture definition given in an application program into a set of pixel-intensity values for storage in the frame buffer. This digitization process is known as **scan conversion**.

Display controller 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 and
- Manipulations on displayed objects

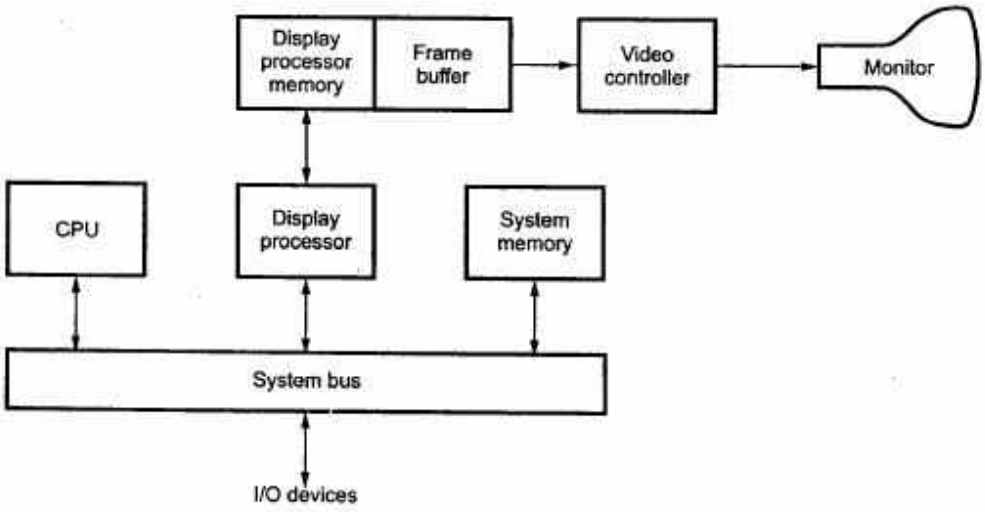


Fig. 1.27 Raster scan system with a display processor

1.7.1.4 Colour CRT Monitors

A CRT monitor displays colour pictures by using a combination of phosphors that emit different-coloured light. It generates a range of colours by combining the emitted light from the different phosphors. There are two basic techniques used for producing colour displays :

- 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 phosphor, usually red and green. The displayed colour depends on how far the electron beam penetrates into the phosphor layers. The outer layer is of red phosphor and inner layer is of green phosphor. A beam of slow electrons excites only the outer red layer. A beam of very fast electrons penetrates through the red layer and excites the inner green layer. At intermediate beam speeds, combinations of red and green light are emitted and two additional colours, orange and yellow displayed. The beam acceleration voltage controls the speed of the electrons and hence the screen colour at any point on the screen.

Merits and Demerits

- It is an inexpensive technique to produce colour in random scan monitors.
- It can display only four colours.
- The quality of picture produced by this technique is not as good as compared to other techniques.

Shadow Mask Technique

The shadow mask technique produces a much wider range of colours than the beam penetration technique. Hence this technique is commonly used in raster-scan displays including colour TV. In a shadow mask technique, CRT has three phosphor colour dots at each pixel position. One phosphor dot emits a red light, another emits a green light, and the third emits a blue light. The Fig. 1.28 shows the shadow mask CRT. It has three electron guns, one for each colour dot, and a shadow mask grid just behind the phosphor coated screen.

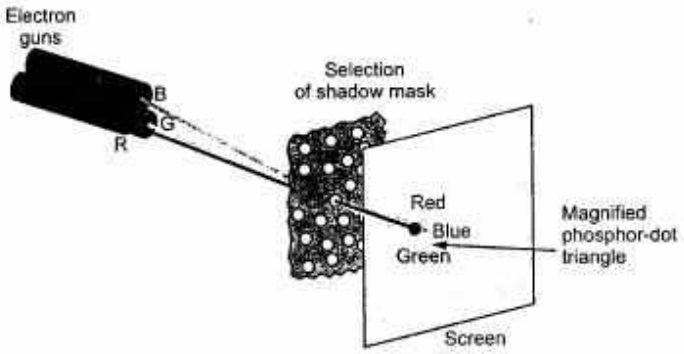


Fig. 1.28

The shadow mask grid consists of series of holes aligned with the phosphor dot pattern. As shown in the Fig. 1.28, three electron beams are deflected and focused as a group onto the shadow mask and when they pass through a hole in the shadow mask, they excite a dot triangle. A dot triangle consists of three small phosphor dots of red, green and blue colour. These phosphor dots are arranged so that each electron beam can activate only its

corresponding colour dot when it passes through the shadow mask. A dot triangle when activated appears as a small dot on the screen which has colour of combination of three small dots in the dot triangle. By varying the intensity of the three electron beams we can obtain different colours in the shadow mask CRT.

1.7.1.5 Direct-view Storage Tubes

We know that, in raster scan display we do refreshing of the screen to maintain a screen image. The direct-view storage tubes give the alternative method of maintaining the screen image. A direct-view storage tube (DVST) uses the storage grid which stores the picture information as a charge distribution just behind the phosphor-coated screen.

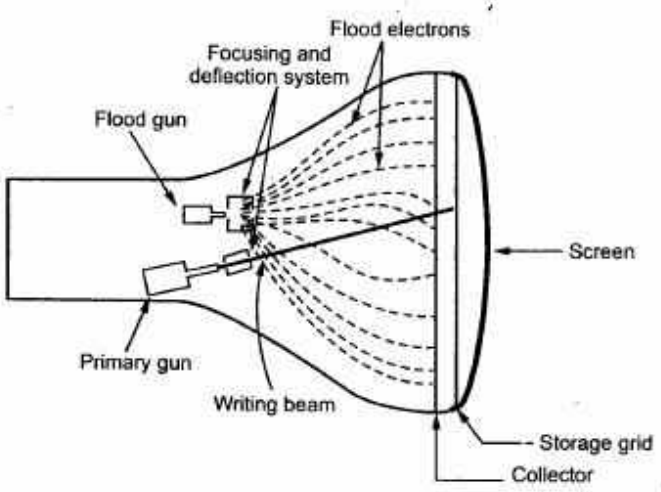


Fig. 1.29 Arrangement of the DVST

The Fig. 1.29 shows the general arrangement of the DVST. It consists of two electron guns: a primary gun and a flood gun.

A primary gun stores the picture pattern and the flood gun maintains the picture display.

A primary gun produces high speed electrons which strike on the storage grid to draw the picture pattern. As electron beam strikes on the storage grid with high speed, it knocks out electrons from the storage grid keeping the net positive charge. The knocked out electrons are attracted towards the collector. The net positive charge on the storage grid is nothing but the picture pattern. The continuous low speed electrons from flood gun pass through the control grid and are attracted to the positive charged areas of the storage grid. The low speed electrons then penetrate the storage grid and strike the phosphor coating without affecting the positive charge pattern on the storage grid. During this process the collector just behind the storage grid smooths out the flow of flood electrons.

Advantages of DVST

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.
3. It has flat screen.

Disadvantages of DVST

1. They do not display colours and are available with single level of line intensity.
2. Erasing requires removal of charge on the storage grid. Thus erasing and redrawing process takes several seconds.
3. Selective or part erasing of screen is not possible.
4. Erasing of screen produces unpleasant flash over the entire screen surface which prevents its use of dynamic graphics applications.
5. It has poor contrast as a result of the comparatively low accelerating potential applied to the flood electrons.
6. The performance of DVST is some what inferior to the refresh CRT.

1.7.1.6 Flat Panel Displays

The term flat-panel display refers to a class of video devices that have reduced volume, weight, and power requirements compared to a CRT. The important feature of flat-panel display is that they are thinner than CRTs. There are two types of flat panel displays : emissive displays and nonemissive displays.

Emissive displays : They convert electrical energy into light energy. Plasma panels, thin-film electro luminescent displays, and light emitting diodes are examples of emissive displays.

Nonemissive displays : They use optical effects to convert sunlight or light from some other source into graphics patterns. Liquid crystal display is an example of nonemissive flat panel display.

1.7.1.7 Plasma Panel Display

Plasma panel display writes images on the display surface point by point, each point remains bright after it has been intensified. This makes the plasma panel functionally very similar to the DVST eventhough its construction is markedly different.

The Fig. 1.30 shows the construction of plasma panel display. It consists of two plates of glass with thin, closely spaced gold electrodes. The gold electrodes are attached to the inner faces and covered with a dielectric material. These are attached as a series of vertical conducting ribbons on one glass plate, and a set of horizontal ribbons to the other glass plate. The space between two glass plates is filled with neon-based gas and sealed. By applying voltages between the electrodes the gas within the panel is made to behave as if it were divided into tiny cells, each one independent of its neighbours. These independent cells are made to glow by placing a firing voltage of about 120 volts across it by means of the electrodes. The glow can be sustained by maintaining a high frequency alternating voltage of about 90 volts across the cell. Due to this refreshing is not required.

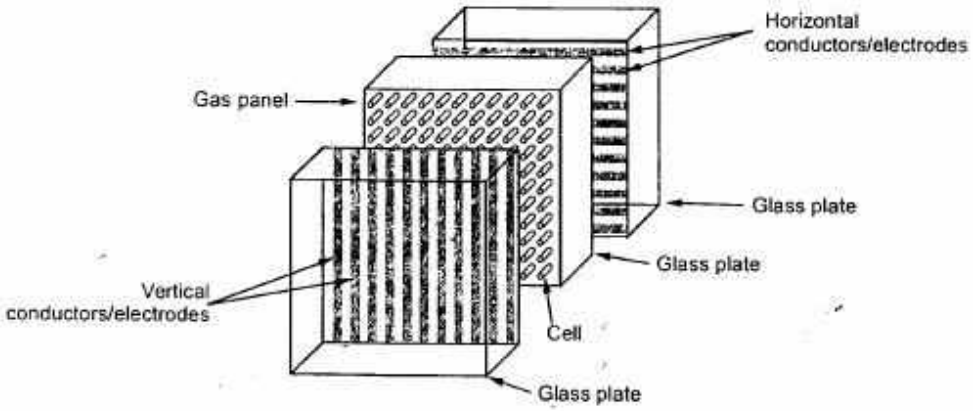


Fig. 1.30 Construction of plasma panel display

Advantages

1. Refreshing is not required.
2. Produces a very steady image, totally free of flicker.
3. Less bulky than a CRT.
4. Allows selective writing and selective erasing, at speed of about 20 μsec per cell.
5. It has the flat screen and is transparent, so the displayed image can be superimposed with pictures from slides or other media projected through the rear panel.

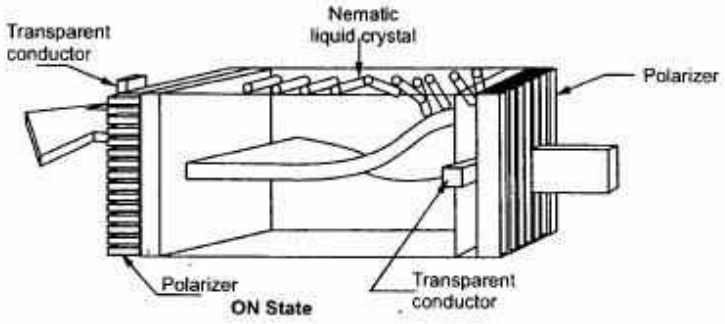
Disadvantages

1. Relatively poor resolution of about 60 dots per inch.
2. It requires complex addressing and wiring.
3. Costlier than the CRTs.

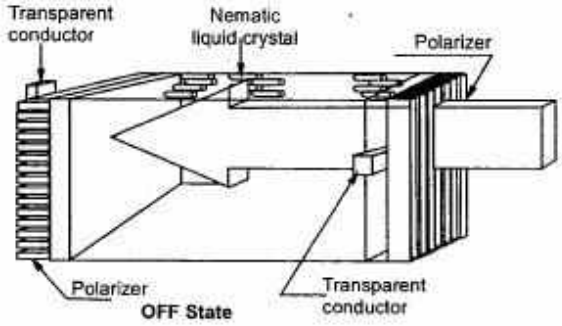
1.7.1.8 Liquid Crystal Monitors

The term liquid crystal refers to the fact that these compounds have a crystalline arrangement of molecules, yet they flow like a liquid. Flat panel displays commonly use nematic (thread like) liquid-crystal compounds that tend to keep the long axes of the rod-shaped molecules aligned.

Two glass plates, each containing a light polarizer at right angles to the other plate sandwich the liquid-crystal material. Rows of horizontal transparent conductors are built into one glass plate, and columns of vertical conductors are put into the other plate. The intersection of two conductors defines a pixel position. In the ON state, polarized light passing through material is twisted so that it will pass through the opposite polarizer. It is then reflected back to the viewer. To turn OFF the pixel, we apply a voltage to the two intersecting conductors to align the molecules so that the light is not twisted as shown in the Fig. 1.31. This type of flat panel device is referred to as a passive matrix LCD.



(a) Field effect display 'ON State'



(b) Field effect display 'OFF State'

Fig. 1.31

1.7.1.9 Important Characteristics of Video Display Devices

Persistence : The major difference between phosphors is their persistence. It decides how long they continue to emit light after the electron beam is removed. Persistence is defined as the time it takes the emitted light from the screen to decay to one-tenth of its original intensity. Lower persistence phosphors require higher refreshing rates to maintain a picture on the screen without flicker. However it is useful for displaying animations. On the other hand higher persistence phosphors are useful for displaying static and highly complex pictures.

Resolution : Resolution indicates the maximum number of points that can be displayed without overlap on the CRT. It is defined as the number of points per centimeter that can be plotted horizontally and vertically.

Resolution depends on the type of phosphor, the intensity to be displayed and the focusing and deflection systems used in the CRT.

Aspect Ratio : It is the ratio of vertical points to horizontal points to produce equal length lines in both directions on the screen. An aspect ratio of 4/5 means that a vertical line plotted with four points has the same length as a horizontal line plotted with five points.

1.7.2 Hardcopy Devices

We can obtain hard-copy output for our image in several formats using printer or plotter. Therefore, printers and plotters are also called hard-copy devices. The quality of pictures obtained from a hard-copy device depends on dot size and the number of dots per inch, or lines per inch, that can be printed, i.e. it depends on the resolution of printer or plotter. Before going to see working principle of various plotters and printers we see the important characteristics of hardcopy devices.

1.7.2.1 Important Characteristics of Hardcopy Devices

Dot Size : It is the diameter of a single dot on the device's output. It is also referred to as **spot size**.

Addressability : It is the number of individual dots (not necessarily distinguishable) per inch that can be created. If the address of current dot is (x, y), then address of next dot in the horizontal direction is given as (x+1, y). Similarly, the address of next dot in vertical direction is (x, y+1).

Interdot distance : It is the reciprocal of addressability. If addressability is large the interdot distance is less. The interdot distance should be less to get smooth shapes, as shown in the Fig. 1.32.

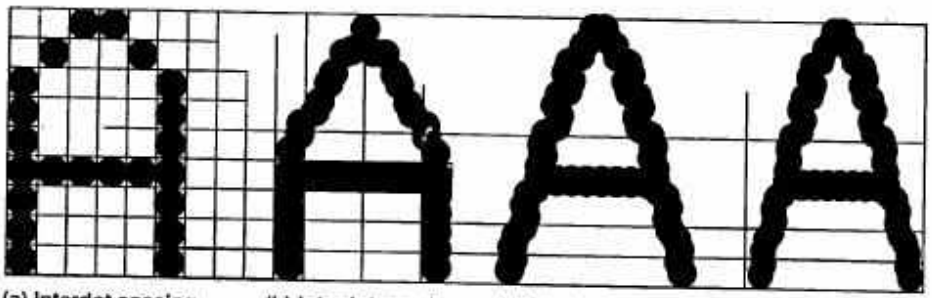


Fig. 1.32

Resolution : It is the number of distinguishable lines per inch that a device can create. It depends on a dot size and the cross-sectional intensity distribution of a spot. A spot with sharply delineated edges yields higher resolution than does one with edge that trail off, as shown in the Fig. 1.33.

Solved Examples

Ex. 1.1: *A video monitor has a display area measuring 12 inch by 9.6 inch. If the resolution is 1280 by 1024 and the aspect ratio is 1. What is the diameter of each screen point ?*

(Dec.-2001)

Sol. : An aspect ratio of 1 means that a vertical line plotted and horizontal line plotted with equal number of points have the same length. Therefore the diameter of each screen point can be given as

$$d = \frac{\text{horizontal display length}}{\text{horizontal resolution}}$$

$$= \frac{\text{vertical display length}}{\text{vertical resolution}}$$

$$= \frac{12}{1280} = \frac{9.6}{1024} = 9.375 \times 10^{-3} \text{ inch}$$

Ex. 1.2 : *How long it will take to load a 640 by 480 frame buffer with 12 bits per pixel if 10^5 bits can be transferred per second ? How long it will take to load a 24-bits per pixel frame buffer with a resolution of 1280 by 1024 using the same transfer rate ?* **(Dec-2001)**

Sol. : i) Total number of bits required to load the frame buffer with a resolution of 640 × 480 and with 12-bits per pixel can be given as

$$B = 640 \times 480 \times 12 = 3.6864 \times 10^6$$

Transfer rate is 10^5 bits/sec,

Therefore, time required to load the frame buffer is

$$T = \frac{3.6864 \times 10^6}{10^5} = 36.864 \text{ seconds}$$

ii) Total number of bits required to load the frame buffer with a resolution of 1280 × 1024 and with 24-bits per pixel can be given as

$$B = 1280 \times 1024 \times 24 = 31.45728 \times 10^6$$

Transfer rate is 10^5 bits/sec,

Therefore, time required to load the frame buffer is

$$T = \frac{31.45728 \times 10^6}{10^5}$$

$$= 314.5728 \text{ seconds}$$

Ex. 1.3 : *What is the fraction of the total refresh time per frame spent in retrace of the electron beam for a non-interlaced raster system with a resolution of 1280 by 1024, a refresh rate of 60 Hz, a horizontal retrace time of 5μsec and a vertical retrace time of 500 microseconds (μ sec) ?* **(Dec-99)**

Sol. : Total horizontal retrace time = $1024 \times 5 \times 10^{-6}$

Total vertical retrace time = 500×10^{-6}

∴ Total retrace time = $1024 \times 5 \times 10^{-6} + 500 \times 10^{-6}$

∴ The fraction of the total refresh time per frame spent in retrace of the electron beam can be given as

$$T = \frac{\text{Total retrace time}}{\text{refresh time}} = \frac{1024 \times 5 \times 10^{-6} + 500 \times 10^{-6}}{1 / 60}$$

$$= 0.3372$$

Ex. 1.4 : *For an electrostatic plotter 18-inch-wide paper, a resolution of 200 units to the inch in each direction and a paper speed of 3 inches per second, how many bits per second must be provided to allow the paper to move at full speed ?* **(Dec-99)**

Sol. : Total dots in the horizontal direction = 18 × 200 = 3600

Total dots in the 3 inch length of paper in one column = 3 × 200 = 600

∴ Total number of dot information required per second = 3600 × 600

$$= 2.16 \times 10^6$$

If we assume 8-bits are required for each dot, then total number of bits per second required to plot are given as

$$B = 2.16 \times 10^6 \times 8$$

$$= 17.28 \times 10^6$$

Review Questions

1. Discuss on topic image processing as picture analysis.
2. List the advantages of interactive graphics.
3. Explain the representative uses of computer graphics.
4. Explain the classification of use of computer graphics.
5. What do you mean by rasterization ?
6. Define scan conversion.
7. Write a short note on
   a) Keyboard
   b) Mouse
   c) Trackball and spaceball
   d) Joystick
   e) Digitizer
   f) Light pen
   g) Touch panels
   h) Scanner
8. Write a short note on
   a) Cathode -Ray Tubes
   b) Vector scan display
   c) Raster scan display
   d) Beam penetration technique
   e) Shadow mask technique
9. Explain the working of direct-view storage tubes.
10. List the advantages and disadvantages of DVST.

2 Raster Graphics Algorithms for Drawing 2-D Primitives

2.1 Introduction

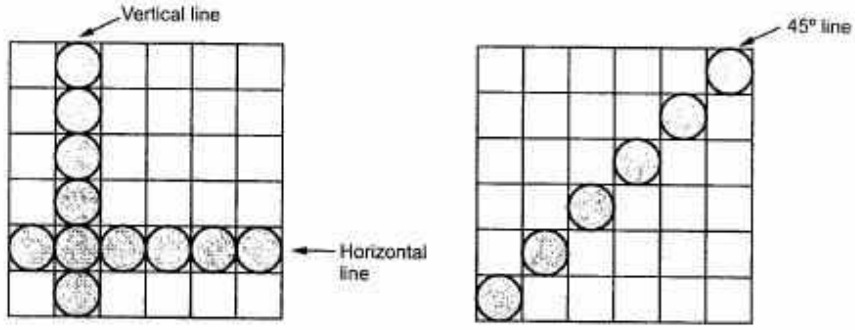
In the previous chapter we have seen that the raster scan graphics devices require special procedures for displaying graphics objects such as line, circle, polygons, curves and even characters. In this chapter we will examine the procedures for line, circle and character generation.

2.2 Basic Concepts in Line Drawing

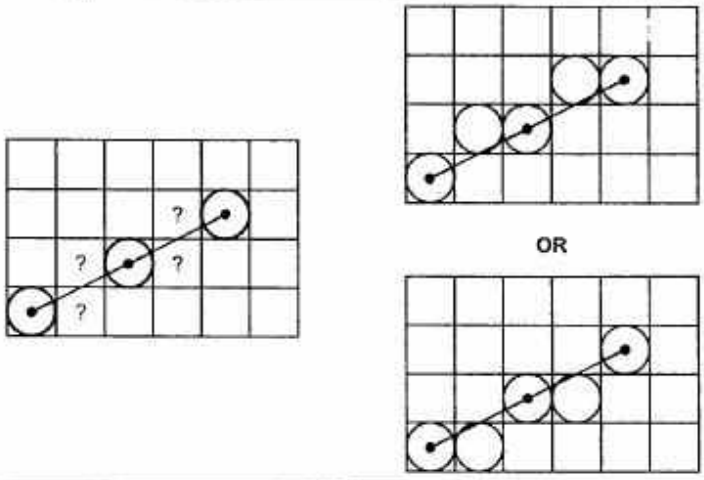
Before discussing specific line drawing algorithms it is useful to note the general requirements for such algorithms. These requirements specify the desired characteristics of line.

- The line should appear as a straight line and it should start and end accurately.
- The line should be displayed with constant brightness along its length independent of its length and orientation.
- The line should be drawn rapidly.

Let us see the different lines drawn in Fig. 2.1.



(a) Vertical and Horizontal lines (b) 45° line



(c) Line with other orientation

Fig. 2.1

As shown in Fig. 2.1 (a), horizontal and vertical lines are straight and have same width. The 45° line is straight but its width is not constant. On the other hand, the line with any other orientation is neither straight nor has same width. Such cases are due to the finite resolution of display and we have to accept approximate pixels in such situations, shown in Fig. 2.1 (c).

The brightness of the line is dependent on the orientation of the line. We can observe that the effective spacing between pixels for the 45° line is greater than for the vertical and horizontal lines. This will make the vertical and horizontal lines appear brighter than the 45° line. Complex calculations are required to provide equal brightness along lines of varying length and orientation. Therefore, to draw line rapidly some compromises are made such as

- Calculate only an approximate line length
- Reduce the calculations using simple integer arithmetic
- Implement the result in hardware or firmware

2.3 Line Drawing Algorithms

Considering the assumptions made in the previous section most line drawing algorithms use incremental methods. In these methods line starts with the starting point. Then a fix increment is added to the current point to get the next point on the line. This is continued till the end of line. Let us see the incremental algorithm.

Incremental Algorithm

1. CurrPosition = Start
   Step = Increment
2. if (| CurrPosition – End | < Accuracy) then go to step 5

[This checks whether the current position is reached upto approximate end point. If yes, line drawing is completed.]

if (CurrPosition < End) then go to step 3

[Here start < End]

if (CurrPosition > End) then go to step 4

[Here start > End]

3. CurrPosition = CurrPosition + Step

   go to step 2

4. CurrPosition = CurrPosition – Step

   go to step 2

5. Stop.

In the following sections we discuss the line rasterizing algorithms based on the incremental algorithm.

2.3.1 Digital Differential Analyzer

We know that the slope of a straight line is given as

$$m = \frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1} \quad \ldots (2.1)$$

The above differential equation can be used to obtain a rasterized straight line. For any given x interval Δx along a line, we can compute the corresponding y interval Δy from equation 2.1 as

$$\Delta y = \frac{y_2 - y_1}{x_2 - x_1} \Delta x \quad \ldots (2.2)$$

Similarly, we can obtain the x interval Δx corresponding to a specified Δy as

$$\Delta x = \frac{x_2 - x_1}{y_2 - y_1} \Delta y \quad \ldots (2.3)$$

Once the intervals are known the values for next x and next y on the straight line can be obtained as follows

$$x_{i+1} = x_i + \Delta x$$

$$= x_i + \frac{x_2 - x_1}{y_2 - y_1} \Delta y \quad \ldots (2.4)$$

and

$$y_{i+1} = y_i + \Delta y$$

$$= y_i + \frac{y_2 - y_1}{x_2 - x_1} \Delta x \quad \ldots (2.5)$$

The equations 2.4 and 2.5 represent a recursion relation for successive values of x and y along the required line. Such a way of rasterizing a line is called a **digital differential analyzer** (DDA). For simple DDA either Δx or Δy, whichever is larger, is chosen as one raster unit, i.e.

if $|\Delta x| \geq |\Delta y|$ then

else $\Delta x = 1$

$\Delta y = 1$

With this simplification, if $\Delta x = 1$ then

we have $y_{i+1} = y_i + \frac{y_2 - y_1}{x_2 - x_1}$ and

$x_{i+1} = x_i + 1$

If $\Delta y = 1$ then

we have

$y_{i+1} = y_i + 1$ and

$x_{i+1} = x_i + \frac{x_2 - x_1}{y_2 - y_1}$

Let us see the digital differential analyzer (DDA) routine for rasterizing a line

DDA Line Algorithm

1. Read the line end points (x_1, y_1) and (x_2, y_2) such that they are not equal.
   [if equal then plot that point and exit]
2. $\Delta x = |x_2 - x_1|$ and $\Delta y = |y_2 - y_1|$
3. if $(\Delta x \geq \Delta y)$ then
   length = Δx
   else
   length = Δy
   end if
4. $\Delta x = (x_2 - x_1) / \text{length}$
   $\Delta y = (y_2 - y_1) / \text{length}$
   [This makes either Δx or Δy equal to 1 because length is either $|x_2 - x_1|$ or $|y_2 - y_1|$. Therefore, the incremental value for either x or y is one.]
5. $x = x_1 + 0.5 * \text{Sign}(\Delta x)$
   $y = y_1 + 0.5 * \text{Sign}(\Delta y)$
   [Here, Sign function makes the algorithm work in all quadrant. It returns - 1, 0, 1 depending on whether its argument is < 0, = 0, > 0 respectively. The factor 0.5 makes it possible to round the values in the integer function rather than truncating them.]
6. i = 1 [Begins the loop, in this loop points are plotted]
   While (i ≤ length)
   {
   Plot (Integer (x), Integer (y))
   $x = x + \Delta x$
   $y = y + \Delta y$
   i = i + 1

7. Stop

Let us see few examples to illustrate this algorithm.

Ex. 2.1 : *Consider the line from (0, 0) to (4, 6). Use the simple DDA algorithm to rasterize this line*

Sol. : Evaluating steps 1 to 5 in the DDA algorithm we have

$x_1 = 0 \qquad y_1 = 0$

$x_2 = 4 \qquad y_2 = 6$

$\therefore \quad \text{Length} = |y_2 - y_1| = 6$

$\therefore \quad \Delta x = |x_2 - x_1| / \text{length}$

$= \frac{4}{6}$

and $\quad \Delta y = |y_2 - y_1| / \text{length}$

$= 6 / 6 = 1$

Initial value for

$$x = 0 + 0.5 * \text{Sign}\left(\frac{4}{6}\right) = 0.5$$

$$y = 0 + 0.5 * \text{Sign}\,(1) = 0.5$$

Tabulating the results of each iteration in the step 6 we get,

i	Plot	x	y
		0.5	0.5
1	(0, 0)		
		1.167	1.5
2	(1, 1)		
		1.833	2.5
3	(1, 2)		
		2.5	3.5
4	(2, 3)		
		3.167	4.5
5	(3, 4)		
		3.833	5.5
6	(3, 5)		
		4.5	6.5

Table 2.1

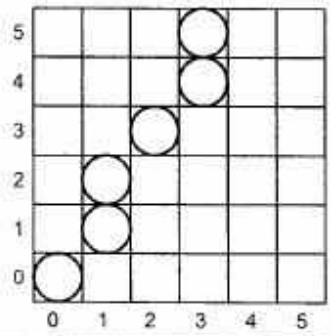


Fig. 2.2 Result for a simple DDA

The results are plotted as shown in the Fig. 2.2. It shows that the rasterized line lies to both sides of the actual line, i.e. the algorithm is **orientation dependent**.

Ex. 2.2 : *Consider the line from (0, 0) to (– 6, – 6). Use the simple DDA algorithm to rasterize this line.*

Sol. : Evaluating steps 1 to 5 in the DDA algorithm we have

$x_1 = 0 \qquad y_1 = 0$

$x_2 = -6 \qquad y_2 = -6$

$\therefore$ Length $= |x_2 - x_1| = |y_2 - y_1| = 6$

$\therefore \qquad \Delta x = \Delta y = -1$

Initial values for

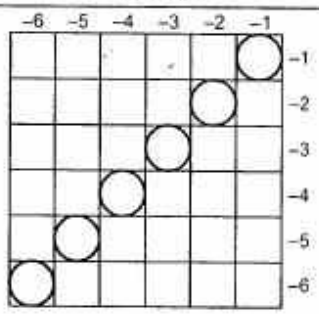
$$x = 0 + 0.5 * \text{Sign}(-1) = -0.5$$
$$y = 0 + 0.5 * \text{Sign}(-1) = -0.5$$

Tabulating the results of each iteration in the step 6 we get,

i	Plot	x	y
		– 0.5	– 0.5
1	(– 1, – 1)		
		– 1.5	– 1.5
2	(– 2, – 2)		
		– 2.5	– 2.5
3	(– 3, – 3)		
		– 3.5	– 3.5
4	(– 4, – 4)		

		-4.5	-4.5
5	(-5, -5)		
		-5.5	-5.5
6	(-6, -6)		
		-6.5	-6.5

Table 2.2



★ -ve pixel values are with reference to pixel at the center of screen

Fig. 2.3 Result for a simple DDA

The results are plotted as shown in the Fig. 2.3. It shows that the rasterized line lies on the actual line and it is 45° line.

'C' code for DDA Line Drawing Algorithm

(Softcopy of this program is available at vtubooks.com)

```
#include<stdio.h>
#include<graphics.h>
#include<math.h>
main()
{
float x,y,x1,y1,x2,y2,dx,dy,length;
int i,gd,gm;
clrscr();

/* Read two end points of line
--------------------------------- */
printf("Enter the value of x1 :\t");
scanf("%f",&x1);
```

```
printf("Enter the value of y1 :\t");
scanf("%f",&y1);
printf("Enter the value of x2 :\t");
scanf("%f",&x2);
printf("Enter the value of y2 :\t");
scanf("%f",&y2);

/* Initialise graphics mode
---------------------------------- */
detectgraph(&gd,&gm);
initgraph(&gd,&gm,"");

dx=abs(x2-x1);
dy=abs(y2-y1);

if (dx >= dy)
      {
      length = dx;
      }
      else
      {
      length = dy;
      }
dx = (x2-x1)/length;
dy = (y2-y1)/length;

x = x1 + 0.5;  /* Factor 0.5 is added to round the values */
y = y1 + 0.5;  /* Factor 0.5 is added to round the values */

i = 1;         /* Initialise loop counter */
while(i <= length)
   {
    putpixel(x,y,15);
    x = x + dx;
    y = y + dy;
    i = i + 1;
    delay(100); /* Delay is purposely inserted to see
                   observe the line drawing process */
```

```
    }
getch();
closegraph();
}
```

Advantages of DDA Algorithm

1. It is the simplest algorithm and it does not require special skills for implementation.
2. It is a faster method for calculating pixel positions than the direct use of equation $y = mx + b$. It eliminates the multiplication in the equation by making use of raster characteristics, so that appropriate increments are applied in the x or y direction to find the pixel positions along the line path.

Disadvantages of DDA Algorithm

1. Floating point arithmetic in DDA algorithm is still time-consuming.
2. The algorithm is orientation dependent. Hence end point accuracy is poor.

2.3.2 Bresenham's Line Algorithm

Bresenham's line algorithm uses only integer addition and subtraction and multiplication by 2, and we know that the computer can perform the operations of integer addition and subtraction very rapidly. The computer is also time-efficient when performing integer multiplication by powers of 2. Therefore, it is an efficient method for scan-converting straight lines.

The basic principle of Bresenham's line algorithm is to select the optimum raster locations to represent a straight line. To accomplish this the algorithm always increments either x or y by one unit depending on the slope of line. The increment in the other variable is determined by examining the distance between the actual line location and the nearest pixel. This distance is called **decision variable** or the **error**. This is illustrated in the Fig. 2.4.

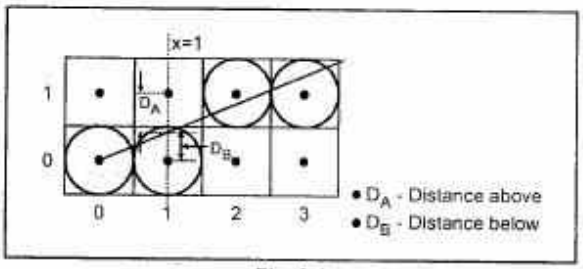


Fig. 2.4

As shown in the Fig. 2.4, the line does not pass through all raster points (pixels). It passes through raster point (0, 0) and subsequently crosses three pixels. It is seen that the intercept of line with the line $x = 1$ is closer to the line $y = 0$, i.e. pixel (1, 0) than to the line $y = 1$ i.e. pixel (1, 1). Hence, in this case, the raster point at (1, 0) better represents the path of the line than that at (1, 1). The intercept of the line with the line $x = 2$ is close to the line $y = 1$, i.e. pixel (2, 1) than to the line $y = 0$, i.e. pixel (2, 0). Hence, the raster point at (2, 1) better represents the path of the line, as shown in the Fig. 2.4

In mathematical terms error or decision variable is defined as

$$e = D_B - D_A \quad \text{or} \quad D_A - D_B$$

Let us define $e = D_B - D_A$. Now if $e > 0$, then it implies that $D_B > D_A$, i.e., the pixel above the line is closer to the true line. If $D_B < D_A$ (i.e. $e < 0$) then we can say that the pixel below the line is closer to the true line. Thus by checking only the sign of error term it is possible to determine the better pixel to represent the line path.

The error term is initially set as

$$e = 2\Delta y - \Delta x$$

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

Then according to value of e following actions are taken.

```
while ( e ≥ 0)
    {
        y = y + 1
        e = e - 2 * Δx
    }
x = x + 1
e = e + 2 * Δy
```

When $e \geq 0$, error is initialized with $e = e - 2\Delta x$. This is continued till error is negative. In each iteration y is incremented by 1. When $e < 0$, error is initialized to $e = e + 2\Delta y$. In both the cases x is incremented by 1. Let us see the Bresenham's line drawing algorithm.

Bresenham's Line Algorithm

1. Read the line end points (x_1, y_1) and (x_2, y_2) such that they are not equal.
   [if equal then plot that point and exit]
2. $\Delta x = |x_2 - x_1|$ and $\Delta y = |y_2 - y_1|$
3. [Initialize starting point]
   $x = x_1$
   $y = y_1$
4. $e = 2 * \Delta y - \Delta x$
   [Initialize value of decision variable or error to compensate for nonzero intercepts]
5. i = 1 [Initialize counter]
6. Plot (x, y)
7. while ($e \geq 0$)
   {
   $y = y + 1$
   $e = e - 2 * \Delta x$
   }
   $x = x + 1$
   $e = e + 2 * \Delta y$
8. $i = i + 1$
9. if ($i \leq \Delta x$) then go to step 6.
10. Stop

Ex. 2.3 : *Consider the line from (5, 5) to (13, 9). Use the Bresenham's algorithm to rasterize the line.*

Sol. : Evaluating steps 1 through 4 in the Bresenham's algorithm we have,

$$\Delta x = |13-5| = 8$$
$$\Delta y = |9-5| = 4$$
$$x = 5$$
$$y = 5$$
$$e = 2 * \Delta y - \Delta x = 2 * 4 - 8$$
$$= 0$$

Tabulating the results of each iteration in the step 5 through 10.

i	Plot	x	y	e
		5	5	0
1	(5, 5)	6	6	−8
2	(6, 6)	7	6	0
3	(7, 6)	8	7	−8
4	(8, 7)	9	7	0
5	(9, 7)	10	8	−8
6	(10, 8)	11	8	0
7	(11, 8)	12	9	−8
8	(12, 9)	13	9	0
9	(13, 9)	14	10	−8

Table 2.3

The results are plotted as shown in Fig. 2.5.

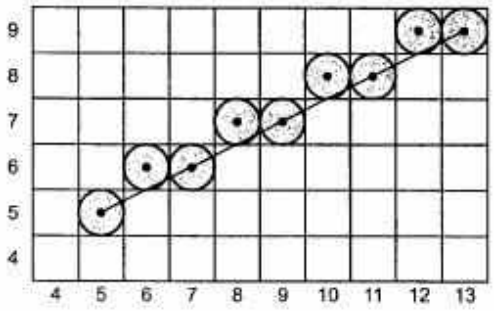


Fig. 2.5

```
/* Enter the thickness of the line
-----------------------------------*/
printf("Enter the required thickness: ");
scanf("%d", &thickness);
cleardevice();
line (x1, y1, x2, y2);
if ((y2 - y1) / (x2 - x1) < 1)
{
wy =(thickness-1)*sqrt(pow((x2-x1),2)+pow((y2-y1),2))/(2*fabs(x2-x1));
 for(i = 0; i < wy; i++)
 {
 line(x1, y1 - i, x2, y2 - i);
 line(x1, y1 + i, x2, y2 + i);
 }
}
else
{
wx = (thickness-1)*sqrt(pow((x2-x1),2)+pow((y2-y1),2))/(2*fabs(y2-y1));
 for(i = 0; i < wx; i++)
 {
 line(x1 - i, y1, x2 - i, y2);
 line(x1 + i, y1, x2 + i, y2);
 }
}
printf("This is the line of thickness %d units.\n", thickness);
getch();
}
```

2.7 Basic Concepts in Circle Drawing

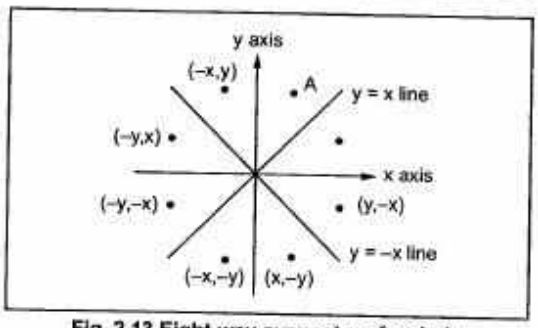


Fig. 2.13 Eight-way symmetry of a circle

A circle is a symmetrical figure. It has eight-way symmetry as shown in the Fig. 2.13. Thus, any circle generating algorithm can take advantage of the circle symmetry to plot eight points by calculating the coordinates of any one point. For example, if point A in the Fig. 2.13 is calculated with a circle algorithm, seven more points could be found just by reflection.

2.9 Circle Drawing Algorithms

In this section we are going to discuss two efficient circle drawing algorithms :

- DDA algorithm and
- Bresenham's algorithm
- Midpoint algorithm

2.9.1 DDA Circle Drawing Algorithm

We know that, the equation of circle, with origin as the center of the circle is given as

$$x^2 + y^2 = r^2$$

The digital differential analyser algorithm can be used to draw the circle by defining circle as a differential equation. It is as given below

$$2x\,dx + 2y\,dy = 0 \quad \text{where } r \text{ is constant}$$

$$\therefore \quad x\,dx + y\,dy = 0$$

$$\therefore \quad y\,dy = -x\,dx$$

$$\therefore \quad \frac{dy}{dx} = \frac{-x}{y}$$

From above equation, we can construct the circle by using incremental x value, $\Delta x = \varepsilon y$ and incremental y value, $\Delta y = -\varepsilon x$, where ε is calculated from the radius of the circle as given below

$$2^{n-1} \leq r < 2^n \quad r : \text{radius of the circle}$$

$$\varepsilon = 2^{-n}$$

For example, if $r = 50$ then $n = 6$ so that $32 \leq 50 < 64$

$$\therefore \quad \varepsilon = 2^{-6}$$

$$= 0.0156$$

Applying these incremental steps we have,

$$x_{n+1} = x_n + \varepsilon y_n$$

$$y_{n+1} = y_n - \varepsilon x_n$$

The points plotted using above equations give the spiral instead of the circle. To get the circle we have to make one correction in the equation; we have to replace x_n by x_{n+1} in the equation of y_{n+1}.

Therefore, now we have

$$x_{n+1} = x_n + \varepsilon y_n$$

$$y_{n+1} = y_n - \varepsilon x_{n+1}$$

Algorithm

1. Read the radius (r), of the circle and calculate value of ε
2. start_x = 0

   start_y = r
3. x_1 = start_x

   y_1 = start_y

4. do

   { $x_2 = x_1 + \varepsilon y_1$

   $y_2 = y_1 - \varepsilon x_2$

   [x_2 represents x_{n+1} and x_1 represents x_n]

   Plot (int (x_2) , int (y_2))

   $x_1 = x_2$;

   $y_1 = y_2$;

   [Reinitialize the current point]

   } while (y_1 – start_y) < ε or (start_x – x_1) > ε

   [check if the current point is the starting point or not. If current point is not starting point repeat step 4 ; otherwise stop]

5. Stop.

'C' code for DDA Circle Drawing Algorithm

(Softcopy of this program is available at vtubooks.com)

```
#include<stdio.h>
#include<graphics.h>
#include<math.h>
main()
{
float x1,y1,x2,y2,startx,starty,epsilon;
int gd,gm,i,val;
int r;
clrscr();

/* Read two end points of line
---------------------------------- */
printf("Enter the radius of a circle :");
scanf("%d",&r);

/* Initialise graphics mode
---------------------------------- */
detectgraph(&gd,&gm);
initgraph(&gd,&gm,"");

/* Initialise starting point
---------------------------------- */
x1=r*cos(0);
```

```
y1=r*sin(0);
startx = x1;
starty = y1;

/*Calculations for epsilon
-----------------------------*/
i=0;
do
{
val = pow(2,i);
i++;
}while(val<r);
epsilon = 1/pow(2,i-1);

do
   {
    x2= x1 + y1*epsilon;
    y2 = y1 - epsilon*x2;
    putpixel(200+x2,200+y2,15);

/* Reinitialise the current point
---------------------------------- */
    x1=x2;
    y1=y2;
    delay(1000);   /* Delay is purposely inserted to see
                      observe the line drawing process */
   }
  while( (y1 - starty ) < epsilon || (startx - x1) > epsilon);
getch();
closegraph();
}
```

2.9.2 Bresenham's Circle Drawing Algorithm

The Bresenham's circle drawing algorithm considers the eight-way symmetry of the circle to generate it. It plots $1/8^{th}$ part of the circle, i.e. from 90° to 45°, as shown in the Fig. 2.16. As circle is drawn from 90° to 45°, the x moves in positive direction and y moves in the negative direction.

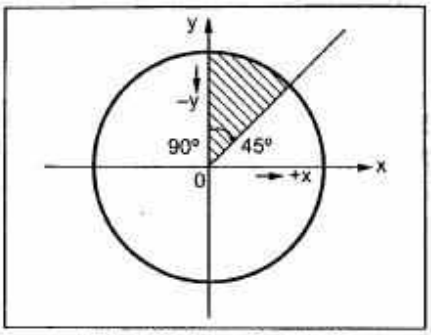


Fig. 2.16 1/8 part of circle

To achieve best approximation to the true circle we have to select those pixels in the raster that fall the least distance from the true circle. Refer Fig. 2.17. Let us observe the 90° to 45° portion of the circle. It can be noticed that if points are generated from 90° to 45°, each new point closest to the true circle can be found by applying either of the two options :

- Increment in positive x direction by one unit or
- Increment in positive x direction and negative y direction both by one unit

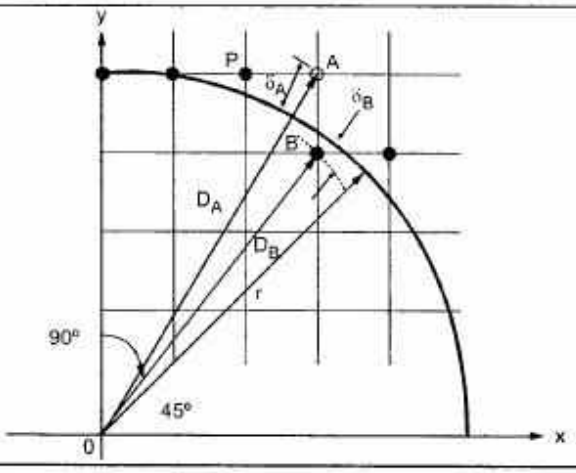


Fig. 2.17 Scan conversion with Bresenham's algorithm

Let us assume point P in Fig. 2.17 as a last scan converted pixel. Now we have two options either to choose pixel A or pixel B. The closer pixel amongst these two can be determined as follows

The distances of pixels A and B from the origin are given as

$$D_A = \sqrt{(x_{i+1})^2 + (y_i)^2} \quad \text{and}$$

$$D_B = \sqrt{(x_{i+1})^2 + (y_i - 1)^2}$$

Now, the distances of pixels A and B from the true circle are given as

$$\delta_A = D_A - r \text{ and } \delta_B = D_B - r$$

However, to avoid square root term in derivation of decision variable, i.e. to simplify the computation and to make algorithm more efficient the δ_A and δ_B are defined as

$$\delta_A = D_A^2 - r^2 \quad \text{and}$$

$$\delta_B = D_B^2 - r^2$$

From Fig. 2.17, we can observe that δ_A is always positive and δ_B always negative. Therefore, we can define **decision variable** d_i as

$$d_i = \delta_A + \delta_B$$

and we can say that, if $d_i < 0$, i.e., $\delta_A < \delta_B$ then only x is incremented; otherwise x is incremented in positive direction and y is incremented in negative direction. In other words we can write,

For $d_i < 0$, $x_{i+1} = x_i + 1$ and

For $d_i \geq 0$, $x_{i+1} = x_i + 1$ and $y_{i+1} = y_i - 1$

The equation for d_i at starting point, i.e. at $x = 0$ and $y = r$ can be simplified as follows

$$\begin{aligned} d_i &= \delta_A + \delta_B \\ &= (x_i + 1)^2 + (y_i)^2 - r^2 + (x_i + 1)^2 + (y_i - 1)^2 - r^2 \\ &= (0 + 1)^2 + (r)^2 - r^2 + (0 + 1)^2 + (r - 1)^2 - r^2 \\ &= 1 + r^2 - r^2 + 1 + r^2 - 2r + 1 - r^2 \\ &= 3 - 2r \end{aligned}$$

Similarly, the equations for d_{i+1} for both the cases are given as

For $d_i < 0$, $d_{i+1} = d_i + 4\,x_i + 6$ and

For $d_i \leq 0$, $d_{i+1} = d_i + 4\,(x_i - y_i) + 10$

Algorithm to plot 1/8 of the circle

1. Read the radius (r) of the circle.
2. d = 3 – 2 r

   [Initialize the decision variable]
3. x = 0, y = r

   [Initialize starting point]
4. do

   {

   plot (x, y)

   if (d < 0) then

   {

   d = d + 4x + 6

   }

   else

   { d = d + 4 (x – y) + 10

   y = y – 1

   }

   x = x + 1

   } while (x < y)
5. Stop

The remaining part of circle can be drawn by reflecting point about y axis, x axis and about origin as shown in Fig. 2.18.

Therefore, by adding seven more plot commands after the plot command in the step 4 of the algorithm, the circle can be plotted. The remaining seven plot commands are :

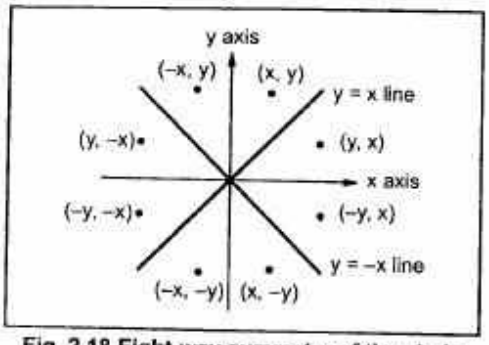


Fig. 2.18 Eight-way symmetry of the circle

plot (y, x)
plot (y, – x)
plot (x, – y)
plot (– x, – y)
plot (– y, – x)
plot (– y, x) and
plot (– x, y)

'C' code for Bresenham's Circle Drawing Algorithm

(Softcopy of this program is available at vtubooks.com)

```
#include<stdio.h>
#include<graphics.h>
#include<math.h>
main()
{
float d;
int gd,gm,x,y;
int r;
clrscr();

/* Read the radius of the circle
-----------------------------*------- */
printf("Enter the radius of a circle :");
scanf("%d",&r);

/* Initialise graphics mode
------------------------------*/
detectgraph(&gd,&gm);
initgraph(&gd,&gm,"");

/* Initialise starting points
------------------------------*/
x = 0;
y = r;
```

```
/* initialise the decision variable
---------------------------------------*/
d = 3 - 2 * r;

do
{       putpixel(200+x,200+y,15);
        putpixel(200+y,200+x,15);
        putpixel(200+y,200-x,15);
        putpixel(200+x,200-y,15);
        putpixel(200-x,200-y,15);
        putpixel(200-y,200-x,15);
        putpixel(200-y,200+x,15);
        putpixel(200-x,200+y,15);
        if (d <= 0)
        {
        d = d + 4*x + 6;
        }
        else
        {
        d = d + 4*(x-y) + 10;
        y = y - 1;
        }
        x = x + 1;
        delay(1000);  /* Delay is purposely inserted to see
                         observe the line drawing process */
        }
        while(x < y);
getch();
closegraph();
}
```

2.9.3 Midpoint Circle Drawing Algorithm

The midpoint circle drawing algorithm also uses the eight-way symmetry of the circle to generate it. It plots 1/8 part of the circle, i.e. from 90° to 45°, as shown in the Fig. 2.19. As circle is drawn from 90 to 45°, the x moves in the positive direction and y moves in the negative direction. To draw a 1/8 part of a circle we take unit steps in the positive x direction and make use of decision parameter to determine which of the two possible y positions is closer to the circle path at each step. The Fig. 2.19 shows the two possible y positions (y_i and $y_i + 1$) at sampling position $x_i + 1$. Therefore, we have to determine whether the pixel at position

$(x_i + 1, y_i)$ or at position $(x_i + 1, y_i - 1)$ is closer to the circle. For this purpose decision parameter is used. It uses the circle function ($f_{circle}(x, y) = x^2 + y^2 - r^2$) evaluated at the midpoint between these two pixels.

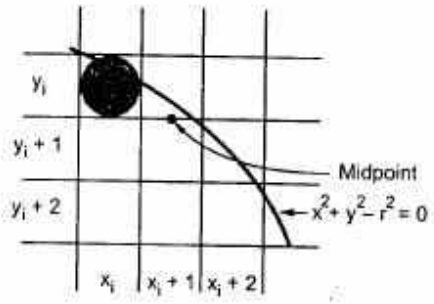


Fig. 2.19 Decision parameter to select correct pixel in circle generation algorithm

$$d_i = f_{circle}(x_i + 1, y_i - \frac{1}{2})$$

$$= (x_i + 1)^2 + \left(y_i - \frac{1}{2}\right)^2 - r^2$$

$$= (x_i + 1)^2 + y_i^2 - y_i + \frac{1}{4} - r^2 \quad \ldots (2.6)$$

If $d_i < 0$, this midpoint is inside the circle and the pixel on the scan line y_i is closer to the circle boundary. If $d_i \geq 0$, the midposition is outside or on the circle boundary, and $y_i - 1$ is closer to the circle boundary. The incremental calculation can be determined to obtain the successive decision parameters. We can obtain a recursive expression for the next decision parameter by evaluating the circle function at sampling position $x_{i+1} + 1 = x_i + 2$.

$$d_{i+1} = f_{circle}(x_{i+1} + 1, y_{i+1} - \frac{1}{2})$$

$$= [(x_i + 1) + 1]^2 + \left(y_{i+1} - \frac{1}{2}\right)^2 - r^2$$

$$= (x_i + 1)^2 + 2(x_i + 1) + 1 + y_{i+1}^2 - \left(y_{i+1}\right) + \frac{1}{4} - r^2 \quad \ldots (2.7)$$

Looking at equations 2.6 and 2.7 we can write

$$d_{i+1} = d_i + 2(x_i + 1) + \left(y_{i+1}^2 - y_i^2\right) - (y_{i+1} - y_i) + 1$$

where y_{i+1} is either y_i or y_{i-1}, depending on the sign of d_i.

If d_i is negative, $y_{i+1} = y_i$

$\therefore$ $d_{i+1} = d_i + 2(x_i + 1) + 1$

$= d_i + 2x_{i+1} + 1 \quad \ldots (2.8)$

If d_i is positive, $y_{i+1} = y_{i-1}$

$\therefore$ $d_{i+1} = d_i + 2(x_i + 1) + 1 - 2y_{i+1} \quad \ldots (2.9)$

The terms $2x_{i+1}$ and $-2y_{i+1}$ in equations (2.8) and (2.9) can be incrementally calculated as

$$2x_{i+1} = 2x_i + 2$$
$$2y_{i+1} = 2y_i - 2$$

The initial value of decision parameter can be obtained by evaluating circle function at the start position $(x_0, y_0) = (0, r)$.

$$d_0 = f_{circle}\left((0+1)^2 + \left(r - \frac{1}{2}\right)^2 - r^2\right)$$
$$= 1 + \left(r - \frac{1}{2}\right)^2 - r^2$$
$$= 1.25 - r$$

Algorithm

1. Read the radius (r) of the circle
2. Initialize starting position as
   x = 0
   y = r
3. Calculate initial value of decision parameter as
   P = 1.25 – r
4. do
```
{   plot (x, y)
    if (d < 0)
      {  x = x + 1
         y = y
         d = d + 2x + 1
      }
    else
      {  x = x + 1
         y = y – 1
         d = d + 2x + 2y + 1
}
    while (x < y)
```
5. Determine symmetry points
6. Stop.

'C' code for Midpoint Circle Drawing Algorithm

(Softcopy of this program is available at vtubooks.com)

```
#include<stdio.h>
#include<graphics.h>
```

```
#include<math.h>
main()
{
float  p;
int i,gd,gm,x,y;
int r;

/* initialise graphics
----------------------- */
detectgraph(&gd,&gm);
initgraph(&gd,&gm,"");

/* Read the radius
----------------------- */
printf("Enter the radius of the circle :");
scanf("%d",&r);

x=0;
y=r;
p = 1.25 - r;
do
{
      putpixel(200+x,200+y,15);
      putpixel(200+y,200+x,15);
      putpixel(200+x,200-y,15);
      putpixel(200+y,200-x,15);
      putpixel(200-x,200-y,15);
      putpixel(200-x,200+y,15);
      putpixel(200-y,200+x,15);
      putpixel(200-y,200-x,15);

      if (p < 0)
      {
      x = x+1;
      y = y;
      p = p + 2*x + 1;
      }
      else
```

u