INTRODUCTION TO COMPUTER GRAPHICS [5HR]

1.1. Introduction to Computer Graphics

The term computer graphics describes any use of computers to create and manipulate images. It is the algorithmic and mathematical tools that can be used to create all kinds of images realistic visual effects, informative technical illustrations, or beautiful computer animations. Graphics can be two- or three-dimensional; images can be completely synthetic or can be produced by manipulating photographs. Actually doing computer graphics inevitably requires knowing about specific hardware, file formats, and usually a graphics API or two. Computer graphics is a rapidly evolving field, so the specifics of that knowledge API: application program interface are a moving target. Fortunately, the culture of computer graphics has enough standard terminology and concepts. This chapter defines some basic terminology, and provides some historical background as well as information sources related to computer graphics.

1.1.1 Overview and Definition Computer graphics

Computer graphics is a field related to the generation of graphics using computers. It includes the creation, storage and manipulation of images of objects. These objects come from diverse fields such as physical, mathematical engineering, architectural abstract structures and natural phenomenon. Computer graphics today is largely interactive; that is largely interactive; that is the user controls the contents structure and appearance of images of the objects by using input devices such as a keyboard, mouse, or touch sensitive panel on the screen.

Until the early 1980‘s computer graphics was a small specialized field, largely because the hardware was expensive and graphics based application programs that were easy to use and cost effective were few. Then personal computers with built in raster graphics displays such as the Xerox Star Apple Macintosh and the IBM PC – popularized the use of bitmap graphics for user computer interaction. A bitmap is a ones and zeros representation of the rectangular array of points on the screen. Each point is called a pixel, short for ―picture elements. Once the bitmap graphics became affordable an explosion of easy to use and inexpensive graphics based user interfaces allowed millions of new users to control simple low cost application programs such as word processors, spreadsheets and drawing programs.

Early History of Computer Graphics

- **Prehistory**
    - Whirlwind: Defensive radar system (1951). Computer Graphics origin.
    - DAC-1: IBM & General Motors, 3D representation of a car.
- **Advances in the 60’s**
    - Skechpad: Ivan Sutherland, considered as the father of computer gaphics. created an interactive drawing program.(1961)
    - SpaceWar: Steve Russell (MIT) designed the first video-game on a DEC PDP-11. (1961)

- First animation shorts to simulate physical effects (gravity, movement, etc.) (1963)
- Sutherland (MIT) made up the first head-mounted display with stereoscopy vision (1966)
- First algorithm of hidden surfaces. by Catmull et al. at the Utah University. At the end of 60's.
- The same team began to have interest in surface shading using color.

- **Advances in the 70's**
  - Introduction of computer graphics in television.
  - Gouraud presented his famous polygonal surface smoothing method.(1971)
  - Microprocessor on the market (1971)
  - Atari was born in 1972. It is the computer game pioneer.
  - First uses of CG (Computer Graphics) in movies.
  - Newell at the University of Utah create the famous teapot, a classical benchmark for visualization algorithms.
  - Texturing and Z-Buffer: Catmull's thesis in 1974.
  - Phong developed his polygonal surface smoothing method (1974).
  - 1975 Baum and Wozniak founded Apple in a garage.
  - Gates founded Microsoft (1975).
  - Lucasfilm created the computer graphics division with the best gurus of the moment (1979).
- **Advances in the 80's**
  - SIGGRAPH is the most important event in this field.
  - Whitted published an article about ray tracing technique (1980)
  - Carpenter, at Lucasfilm, developed the first rendering engine: REYES, the Renderman precursor.(1981)
  - TRON film by Lisberger and Kushner at Disney (beginning of the 80's)
  - Massive sales of graphics terminals: IBM, Tektronix.
  - The first ISO and ANSI standard for graphics libraries: GKS.
  - IBM created the Personal Computer PC.
- **Advances in the 90's and nowadays:**
  - Operative system based on windows for PC (Windows 3.0 at 1990).
  - 3D-Studio from Autodesk (1990).
  - Massive use of computers to produce special effects: Terminator 2 (1991), Disney-Pixar (Toy Story, Bugs, Monsters, inc.), Forrest Gump, Jurassic Park, Lord of the Rings, Starwars episodes I, II and III etc.

- Internet success and 2D and 3D applications for the web.
- 3D graphics cards for PC (Voodoo, Nvidia Gforce etc.). Unstoppable 3D games evolution.
- Virtual Reality. A reality.
- Nowadays: a must for any application.

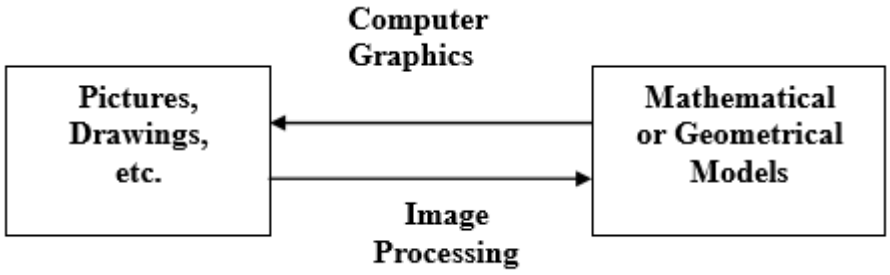
1.1.2 Relation between computer graphics and image processing

Computer Graphics is about drawing things on the screen with pixels, using mathematics and physics (trigonometry, lighting, shading, curvature, etc.) to give the impression of objects to a human viewer.

The output requirements can be simple e.g. arcade games, or complex e.g. realistic rendition for movies.

Image Processing is about taking a digital input (black & white photo or colour photo or scanned image or Xerox copy etc.) and using mathematics and physics (trigonometry, lighting, shading, curvature, etc.) to extract details of objects in that input.

The output requirements can be simple e.g. finding lines or detecting colours (which can be for non-AI purposes) or complex e.g. finding faces or detecting emotions (which can be for AI purposes)



Computer Graphics: Synthesize pictures form mathematical or geometrical models.

Image processing: Analyse pictures to derive descriptions (often in mathematical or geometrical forms) of objects appeared in the pictures.

1.1.3 Advantages of Computer Graphics

The main advantages of computer graphics are as follows: -

1. It provides tools for producing pictures not only of concrete real world objects but also of abstract, synthetic objects such as mathematical surface in 4D and of data that have no inherent geometry such as survey results.
2. It has ability to show moving pictures and thus it is possible to produce animations with computer graphics.
3. With computer graphics user can also control the animation speed, portion of the view, the geometric relationship the object in the scene to one another, the amount of detail shown and on.
4. The computer 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 through made by builder show flat interior and building surroundings. In many case it is also possible to move both objects and viewer.
5. The computer graphics also provides facility called **update dynamics**. With update dynamics it is possible to change this shape, colour or other properties of the objects being viewed.

6. A high quality graphics displays of personal computer provide one of the most natural means of communicating with a computer.
7. It has an ability to show moving pictures, and thus it is possible to produce animations with computer graphics.
8. With computer 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.
9. The computer graphics also provides facility called update dynamics. With update dynamics it is possible to change the shape, color or other properties of the objects being viewed.
10. With the recent development of digital signal processing (DSP) and audio synthesis chip the interactive graphics can now provide audio feedback along with the graphical feedbacks to make the simulated environment even more realistic.

1.1.4 Areas of applications and its classification

1. **User interfaces:** 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.

2. **Plotting of graphics and chart:** In industry, business, government, and educational organizations, 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.

3. **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 chips, optical systems and computer networks.

4. **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.

5. **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 message and attract attentions. Such pictures are very useful in advertising.

6. **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.

7. **Cartography:** Computer graphics is also used to represent geographic maps, weather maps, oceanographic charts, counter maps, population density maps and so on.

8. **Education and training:** computer graphics can be used to generate models of physical aids. Models of physical systems, physiological systems, population trends, or equipment, such as color –coded diagram can help trainees to understand the operation of the system.

9. **Image processing:** in computer graphics, a computer is used to create pictures. Image processing, on the other hands, applies techniques to modify or interpret existing picture such as photographs and scanned image. Image processing and computer graphics are typically combined in many applications such as to model and study physical functions, to design artificial limbs, and to plan and practice surgery. Image processing techniques are most commonly used for picture enhancements to analyze satellite photos, X-ray photography and so on.

Graphics Applications:

1. **Entertainment : Movies**

Pixar: Monstor's Inc.

Final Fantasy

A Bug's Life

2. **Medical Visualization**

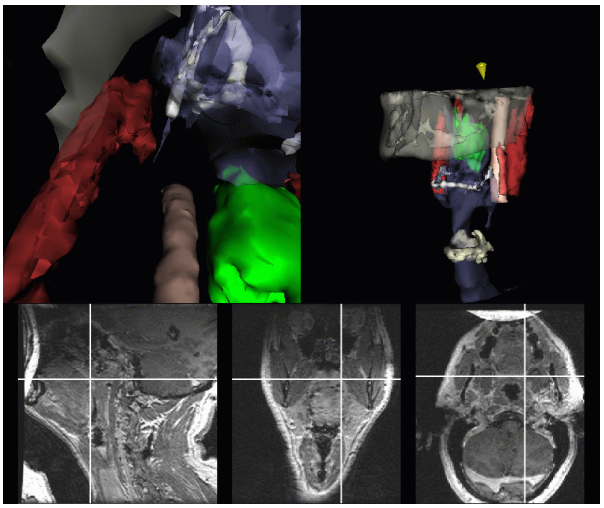
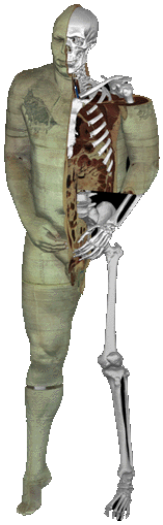
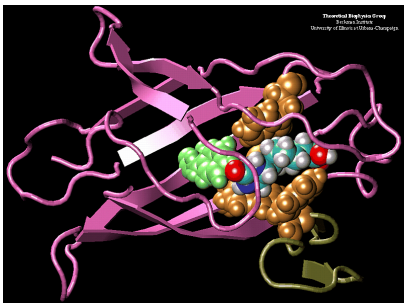


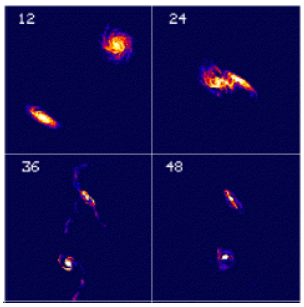
Image guided surgery project

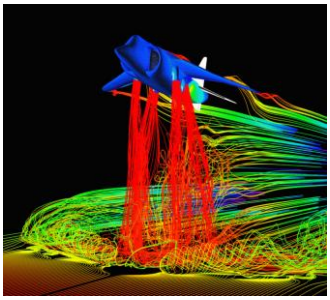


The visible Human Project

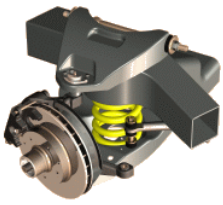
3. Scientific Visualization

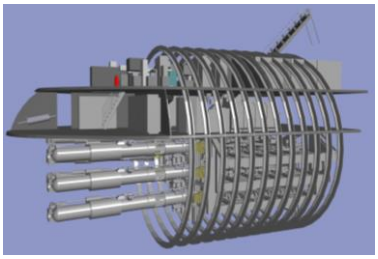




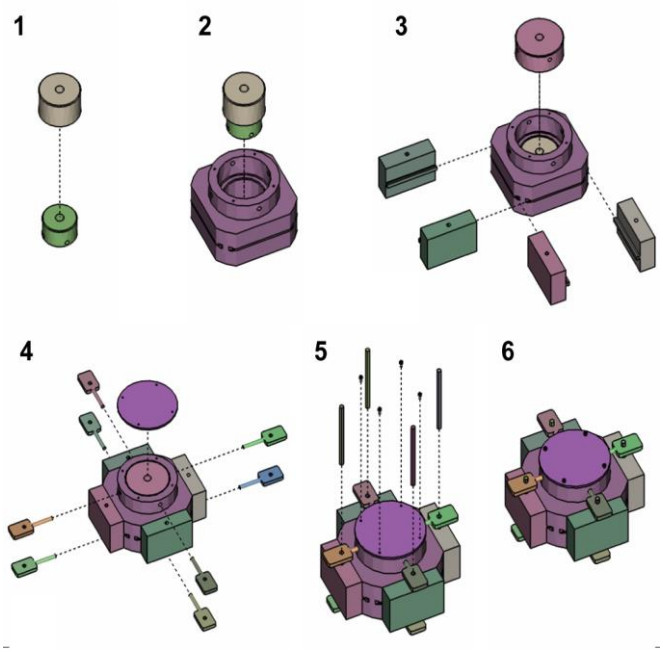


4. Computer Aided Design (CAD)





5. Training



1.2. Graphics Hardware and Software

1.2.1 Graphics Software

Graphics Software:

General classifications of graphics software are:

- **General Programming package:** It provides an extensive set of graphics functions that can be used in a high-level programming language, such as C or FORTRAN. Eg. GL (Graphics Library) system on Silicon Graphics equipment.
    - Include basic functions for generating picture components (straight lines, polygons, circles, and other figures), setting colors and intensity values, selecting views, and applying transformations.
- **Special purpose applications packages:** They are, in contrast designed for nonprogrammers, so that users can generate displays without worrying about how graphics operations work. The interface to the graphics routines in such packages allows users to communicate with the programs in their own terms. Eg. the artist's painting programs and various business, medical, and Computer-Aided Design (CAD) systems.

Computer Aided Design(CAD):

Computer-Aided Design (CAD) is the use of computer technology to aid in the design of a product, particularly the drafting of a part or the product—a part visual (drawing) and part symbol method of communications particular to a specific technical field. It is in origination, the use of computers to aid the art of drafting—the integral communications of technical drawings — which for a three dimensional object are typically represented by three projected views at right angles —drafting is the Industrial arts sub-discipline which underlies all involved technical endeavours. Current CAD software packages range from 2D vector base drafting systems to 3D solid and surface modellers. Modern CAD packages can also frequently allow rotations in three dimensions, allowing viewing of a designed object from any desired angle, even from the inside looking out. Some CAD software is capable of dynamic mathematic modelling, in which case it may be marketed as CADD — Computer Aided Design and Drafting.

Software Standards:

The primary goal of standardized graphics software is portability. When packages are designed with standard graphics functions, software can be moved easily from one hardware system to another and used in different implementations and applications. Without standards, programs designed for one hardware system often cannot be transferred to another system without extensive rewriting of the programs.

- The International Standards Organization (ISO) and American Nation Standards Institute (ANSI) adopted General Kernel System (GKS) as the first graphics software standard.
- PHIGS (Programmer's Hierarchical Interactive Graphics Standard), which is an extension of GKS, increased capabilities for object modeling, color specifications, surface rendering, and picture manipulations. PHIGS+ was developed to provide three-dimensional surface-shading capabilities not available in PHIGS.

Although PHIGS presents a specification for basic graphics functions, it does not provide a standard methodology for a graphics interface to output devices. Nor does it specify methods for storing and transmitting pictures. Separate standards have been developed for these areas. Standardization for device interface methods is given in the Computer Graphics Interface (CGI) system. And Computer Graphics Metafile (CGM) system specifies standards for archiving and transporting pictures.

PHIGS (Programmer's Hierarchical Interactive Graphics System) is an API standard for rendering 3D computer graphics, at one time considered to be the 3D graphics standard for the 1990s. Instead a combination of features and power led to the rise of OpenGL, which remains the de facto 3D standard to this day. PHIGS is no longer used.

PHIGS was available as a standalone implementation (examples: Digital Equipment Corporation's DEC PHIGS, IBM's graPHIGS, Sun's SunPHIGS) and also used with the X Window system, supported via PEX, the "PHIGS Extension to X". PEX consisted of an extension to X, adding commands that would be forwarded from the X server to the PEX system for rendering. Workstations were placed in windows typically, but could also be forwarded to take over the whole screen, or to various printer-output devices.

PHIGS originally lacked the capability to render illuminated scenes, and was superseded by PHIGS+. **PHIGS+** works in essentially the same manner, but added methods for lighting a 3D scene, and was often referred to as PHIGS PLUS (where the PLUS was a slightly tongue-in-cheek acronym for "Plus Lumière Und Shading"). PHIGS+ also introduced more advanced graphics primitives.

OpenGL (Open Graphics Library) is a standard specification defining a cross-language cross-platform API for writing applications that produce 2D and 3D computer graphics. The interface consists of over 250 different function calls, which can be used to draw complex three-dimensional scenes from simple primitives. OpenGL was developed by Silicon Graphics Inc. (SGI) in 1992[1] and is widely used in CAD, virtual reality, scientific visualization, information visualization, and flight simulation. It is also used in video games, where it competes with Direct3D on Microsoft Windows platforms (see Direct3D vs. OpenGL). OpenGL serves two main purposes:

- To hide the complexities of interfacing with different 3D accelerators, by presenting the programmer with a single, uniform API.
- To hide the differing capabilities of hardware platforms, by requiring that all implementations support the full OpenGL feature set (using software emulation if necessary).

The rise of OpenGL and the decline of PHIGS:

OpenGL, unlike PHIGS, is an immediate-mode rendering system with no "state"; once an object is sent to a view to be rendered it essentially disappears. Changes to the model have to be re-sent into the system and re-rendered, dramatically increasing programmer workload. For simple projects, PHIGS was considerably easier to use and work with.

However, OpenGL's "low-level" API allowed the programmer to make dramatic improvements in rendering performance by first examining the data on the CPU-side before trying to send it over the bus to the graphics engine. For instance, the programmer could "cull" the objects by examining which objects were actually visible in the scene, and sending only those objects that would actually end up on the screen. This was kept private in PHIGS, making it much more difficult to tune performance.

Given the low performance systems of the era and the need for high-performance rendering, OpenGL was generally considered to be much more "powerful" for 3D programming. PHIGS fell into disuse. Version 6.0 of the PEX protocol was designed to support other 3D programming models as well, but did not regain popularity. PEX was mostly removed from XFree86 4.2.x (2002) and finally removed from the X Window System altogether in X11R6.7.0.

Major Graphics File Formats:

Image file formats provide a standardized method of organizing and storing image data. Image files are made up of either pixel or vector (geometric) data, which is rasterized to pixels in the display process, with a few exceptions in vector graphic display. The pixels that make up an image are in the form of a grid of columns and rows. Each pixel in an image consists of numbers representing brightness and color.

Image file sizes, expressed in bytes, increase with the number of pixels in the image, and the color depth of the pixels. The more rows and columns, the greater the image resolution and the greater the file size. Also, each pixel making up the image increases in size as color depth is increased. An 8-bit pixel (1 byte) can store 256 colors and a 24-bit pixel (3 bytes) can store 16 million colors. The latter is known as truecolor.

Image compression is a method of using algorithms to decrease file size. High resolution cameras produce large image files. Files sizes may range from hundreds of kilobytes to many megabytes depending on the resolution of the camera and the format used to save the images. High resolution digital cameras record 8 megapixels (MP) (1MP= 1000000 pixels/ 1 million) images, or more, in truecolor. Consider an image taken by an 8 MP camera. Since each of the pixels uses 3 bytes to record true color, the uncompressed image would occupy 24,000,000 bytes of memory. That is a lot of storage space for just one image, and cameras must store many images to be practical. Faced with large file sizes, both within the camera, and later on disc, image file formats have been developed to address the storage problem. An overview of the major graphic file formats is given below:

There are two types of image file compression algorithms: Lossy and Lossless.

Lossless compression algorithms reduce file size with no loss in image quality, though they usually do not compress to as small a file as a lossy method does. When image quality is valued above file size, lossless algorithms are typically chosen.

Lossy compression algorithms take advantage of the inherent limitations of the human eye and discard information that cannot be seen. Most lossy compression algorithms allow for variable levels of quality (compression) and as these levels are increased, file size is reduced. At the highest compression levels, image deterioration becomes noticeable. This deterioration is known as compression artifacting.

- **JPEG (Joint Photographic Experts Group)** files are a lossy format (in most cases). The DOS filename extension is JPG, although other operating systems may use JPEG. Nearly all digital cameras have the option to save images in JPEG format. The JPEG format supports 8 bits per color – red, green, and blue, for 24bit total – and produces relatively small file sizes. The compression when not too severe does not detract noticeably from the image. But JPEG files can suffer generational degradation when repeatedly edited and saved. Photographic images may be better stored in a lossless non-JPEG format if they will be re-edited in future, or if the presence of small "artifacts" (blemishes), due to the nature of the JPEG compression algorithm, is unacceptable. JPEG is also used as the image compression algorithm in many Adobe PDF files.

- **TIFF (Tagged Image File Format)** is a flexible image format that normally saves 8 or 16 bits per color – red, green and blue – for a total of 24 or 48 bits, and uses a filename extension of TIFF or TIF. TIFF's flexibility is both a feature and a curse, with no single reader capable of handling all the different varieties of TIFF files. TIFF can be lossy or lossless. Some types of TIFF files offer relatively good lossless compression for bi-level (black and white, no grey) images. Some highend digital cameras have the option to save images in the TIFF format, using the LZW compression algorithm for lossless storage. The TIFF image format is not widely supported by web browsers. TIFF is still widely accepted as a photograph file standard in the printing industry. TIFF is capable of handling device-specific color spaces, such as the CMYK defined by a particular set of printing press inks.

- **Raw** refers to a family of raw image formats that are options available on some digital cameras. These formats usually use a lossless or nearly-lossless compression, and produce file sizes much smaller than the TIFF formats of fullsize processed images from the same cameras. The raw formats are not standardized or documented, and differ among camera manufacturers. Many graphic programs and image editors may not accept some or all of them, and some older ones have been effectively orphaned already. Adobe's Digital Negative specification is an attempt at standardizing a raw image format to be used by cameras, or for archival storage of image data converted from proprietary raw image formats.

- **PNG (Portable Network Graphics)** file format is regarded, and was made, as the free and open-source successor to the GIF file format. The PNG file format supports true color (16 million colors) whereas the GIF file format only allows 256 colors. PNG excels when the image has large areas of uniform color. The lossless PNG format is best suited for editing pictures, and the lossy formats like JPG are best for final distribution of photographic-type images because of smaller file size. Many older browsers do not yet support the PNG file format, however with the release of Internet Explorer 7 all popular modern browsers fully support PNG. The Adam7-interlacing allows an early preview even when only a small percentage of the data of the image has been transmitted.

- **GIF (Graphics Interchange Format)** is limited to an 8-bit palette, or 256 colors. This makes the GIF format suitable for storing graphics with relatively few colors such as simple diagrams, shapes, logos and cartoon style images. The GIF format supports animation and is still widely used to provide image animation effects. It also uses a lossless compression that is more effective when large areas have a single color, and ineffective for detailed images or dithered images.

- **BMP file format (Windows bitmap)** is used internally in the Microsoft Windows operating system to handle graphics images. These files are typically not compressed, resulting in large files.

The main advantage of BMP files is their wide acceptance, simplicity, and use in Windows programs.

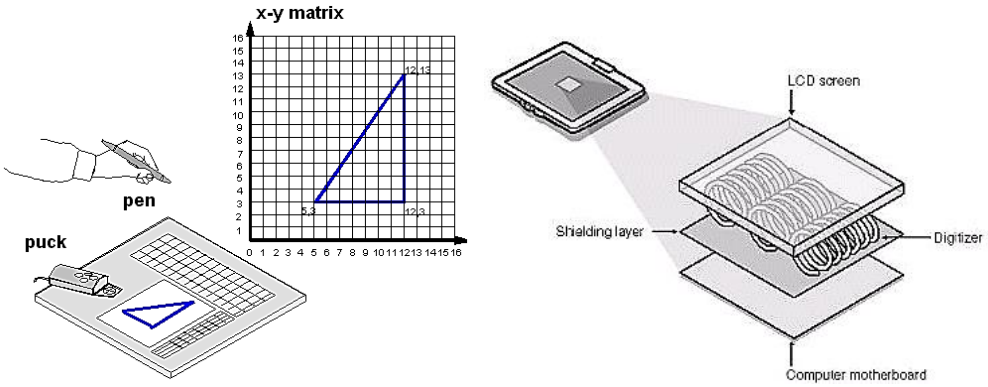
S.N.	Property	JPG	TIF	PNG	GIF
1	Lossy Compression	✓			
2	Lossless compression		✓	✓	✓
3	Uncompressed option		✓		
4	Grayscale	✓	✓	✓	✓
5	RGB color	✓	✓	✓	
6	8 bit color	✓	✓	✓	
7	16 bit color option		✓	✓	
8	CMYK or Lab color		✓		
9	Indexed color option		✓	✓	✓
10	Transparency option			✓	✓
11	Animation option				✓

1.2.2 Graphics Hardware

Input Devices:

1. **Tablet:**

A tablet is digitizer. In general, a digitizer is a device which is used to scan over an object, and to input a set of discrete coordinate positions. These positions can then be joined with straight-line segments to approximate the shape of the original object. A tablet digitizes an object detecting the position of a movable stylus (pencil-shaped device) or puck (link mouse with cross hairs for sighting positions) held in the user's hand. A tablet is flat surface, and its size of the tablet varies from about 6 by 6 inches up to 48 by 72 inches or more. The accuracy of the tablets usually falls below 0.2 mm. There are mainly three types of tablets.



a. Electrical tablet:

A grid of wires on ¼ to ½ inch centres is embedded in the tablet surface and electromagnetic signals generated by electrical pulses applied in sequence to the wires in the grid induce an electrical signal in a wire coil in the stylus (or puck). The strength of the signal induced by each pulse is used to determine the position of the stylus. The signal strength is also used to determine roughly how far the stylus is from the tablet. When the stylus is within ½ inch from the tablet, it is taken as "near" otherwise it is either "far" or "touching". When the stylus is "near" or "touching", a cursor is usually shown on the display to provide visual feedback to the user. A signal is sent to the computer when the tip of the stylus is pressed against the tablet, or when any button on the puck is pressed. The information provided by the tablet repeats 30 to 60 time per second.

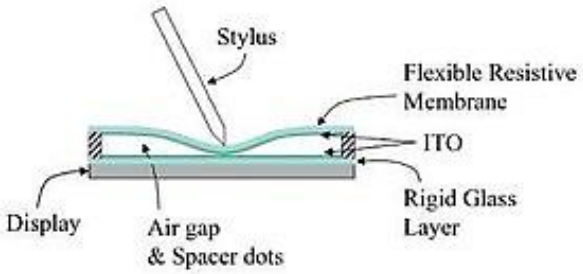
b. Sonic tablet:

The sonic tablet uses sound waves to couple the stylus to microphones positioned on the periphery of the digitizing area. An electrical spark at the tip of the stylus creates sound bursts. The position of the stylus or the coordinate values is calculated using the delay between when the spark occurs and when its sound arrives at each microphone. the main advantage of sonic tablet is that it does not require a dedicated working area for the microphones can be placed on any surface to form the "tablet" work area. This facilitates digitizing drawing on thick books. Because in an electrical tablet this is not convenient for the stylus cannot get closer to the tablet surface.

c. Resistive tablet:

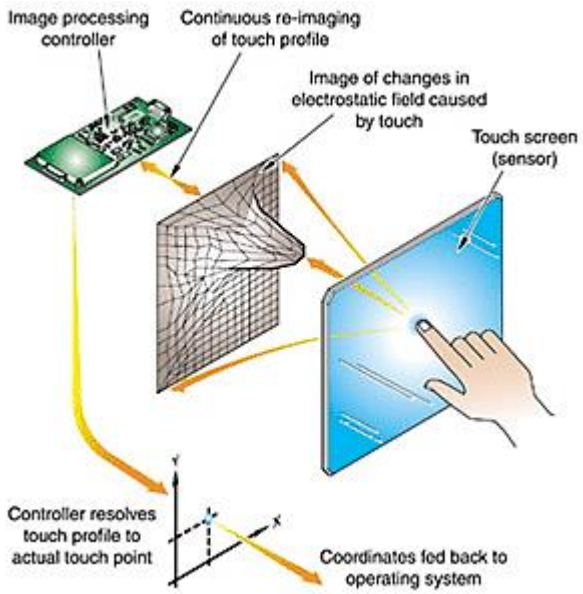
The tablet is just a piece of glass coated with a thin layer of conducting material. When a buttery-powered stylus is activated at certain position, it emits high-frequency radio signals, which induces the radio signals on the conducting layer. The strength of the signal received at the edges of the tablet is used to calculate the position of the stylus.

Several types of tablets are transparent, and thus can be backlit for digitizing x-rays films and photographic negatives. The resistive tablet can be used to digitize the objects on CRT because it can be curved to the shape of the CRT. The mechanism used in the electrical or sonic tablets can also be used to digitize the 3D objects.



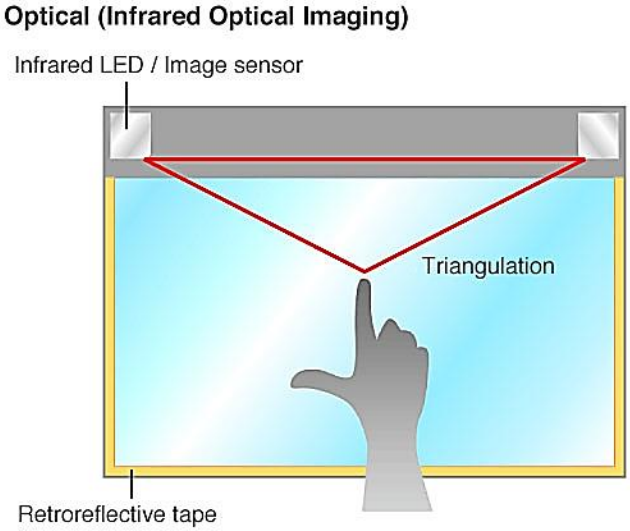
2. Touch Panel

The touch panel allows the users to point at the screen directly with a finger to move the cursor around the screen, or to select the icons. Following are the mostly used touch panels.



a. Optical touch panel

It uses a series of infra-red light emitting diodes (LED) along one vertical edge and along one horizontal edge of the panel. The opposite vertical and horizontal edges contain photo-detectors to form a grid of invisible infrared light beams over the display area. Touching the screen breaks one or two vertical and horizontal light beams, thereby indicating the finger's position. The cursor is then moved to this position, or the icon at this position is selected. It two parallel beams are broken, the finger is presumed to be centered between them; if one is broken, the finger is presumed to be on the beam. There is a low-resolution panel, which offers 10 to 50 positions in each direction.

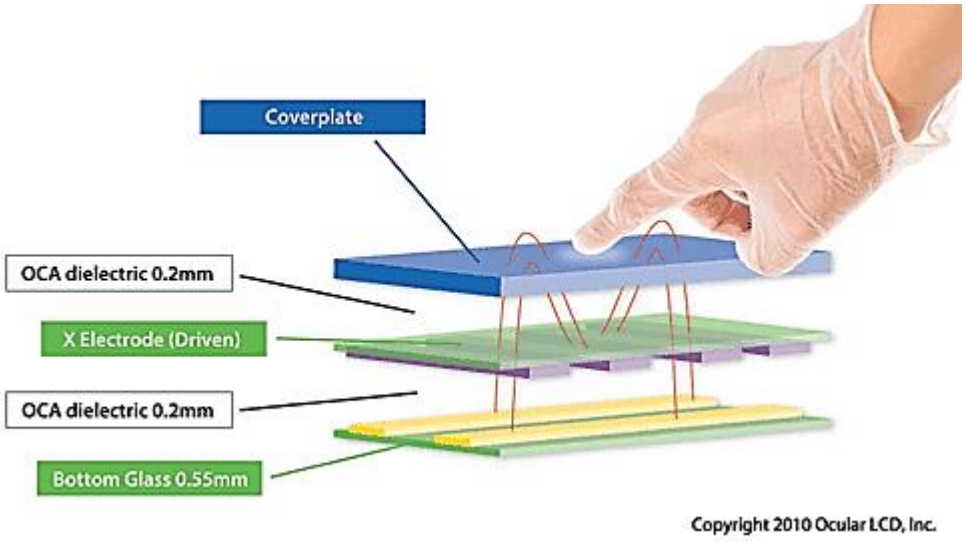


b. Sonic panel:

Bursts of high-frequency sound waves traveling alternately horizontally and vertically are generated at the edge of the panel. Touching the screen causes part of each wave to be reflected back to its source. The screen position at the point of contact is then calculated using the time elapsed between when the wave is emitted and when it arrives back at the source. This is a high-resolution touch panel having about 500 positions in each direction.

c. Electrical touch panel:

It consists of slightly separated two transparent plates one coated with a thin layer of conducting material and the other with resistive material. When the panel is touched with a finger, the two plates are forced to touch at the point of contact thereby creating the touched position. The resolution of this touch panel is similar to that of sonic touch panel.



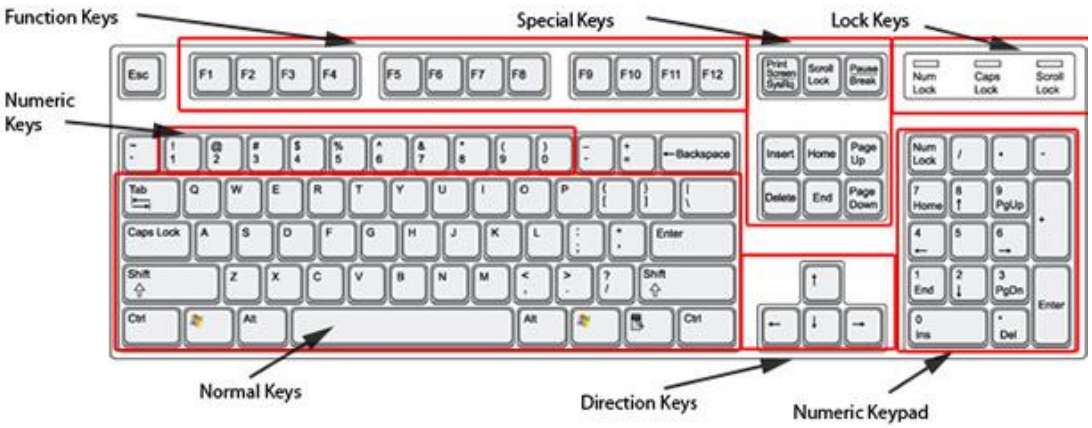
3. Light Pen

It is a pencil-shaped device to determine the coordinates of a point on the screen where it is activated such as pressing the button. In raster display, Y is set at Y_{max} and X changes from 0 to X_{max} for the first scanning line. For second line, Y decreases by one and X again changes from 0 to X_{max}, and so on. When the activated light pen "sees" a burst of light at certain position as the electron beam hits the phosphor coating at that position, it generates a electric pulse, which is used to save the video controller's X and Y registers and interrupt the computer. By reading the saved values, the graphics package can determine the coordinates of the position seen by the light pen. Because of the following drawbacks the light pens are not popular now a days.

- Light pen obscures the screen image as it is pointed to the required spot
- Prolong use of it can cause arm fatigue
- It can not report the coordinates of a point that is completely black. As a remedy one can display a dark blue field in place of the regular image for a single frame time
- It gives sometimes false reading due to background lighting in a room

4. Keyboard

A keyboard creates a code such as ASCII uniquely corresponding to a pressed key. It usually consists of alphanumeric keys, function keys, cursor-control keys, and separate numeric pad. It is used to move the cursor, to select he menu item, pre-defined functions. In computer graphics keyboard is mainly used for entering screen coordinates and text, to invoke certain functions. Now-a-days ergonomically designed keyboard (Ergonomic keyboard) with removable palm rests is available. The slope of each half of the keyboard can be adjusted separately.

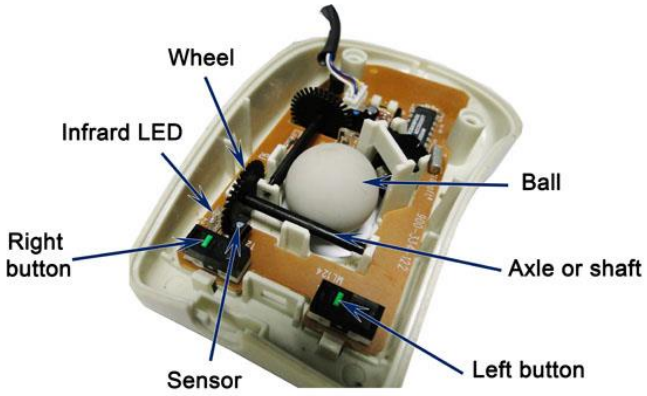


5. Mouse

A mouse is a small hand-held device used to position the cursor on the screen. Mice are relative devices, that is, they can be picked up, moved in space, and then put down gain without any change in the reported position. For this, the computer maintains the current mouse position, which is incremented or decremented by the mouse movements. Following are the mice, which are mostly used in computer graphics.

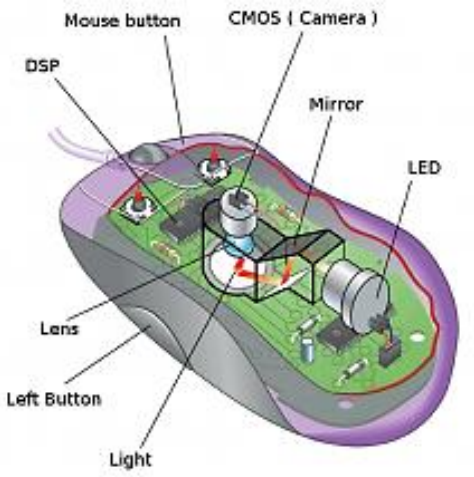
a. Mechanical mouse

When a roller in the base of this mechanical mouse is moved, a pair of orthogonally arranged toothed wheels, each placed in between a LED and a photo detector, interrupts the light path. the number of interrupts so generated are used to report the mouse movements to the computer.



b. Optical mouse

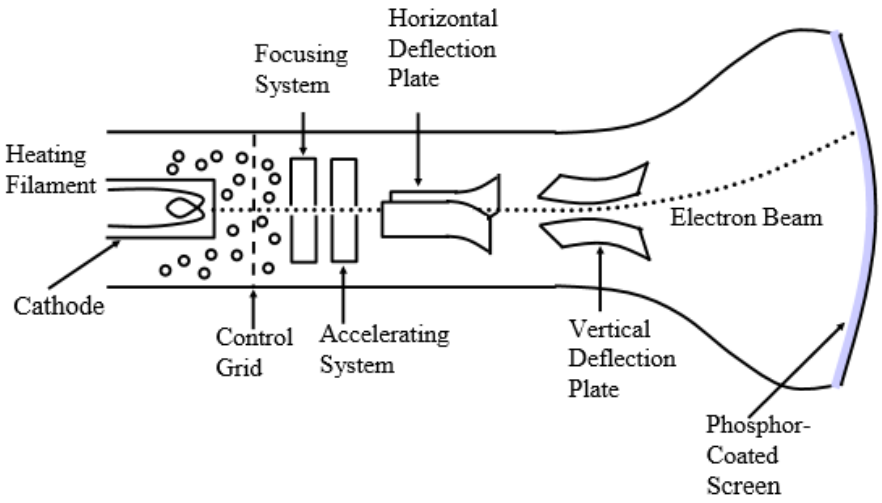
The optical mouse is used on a special pad having a grid of alternating light and dark lines. A LED on the bottom of the mouse directs a beam of light down onto the pad, from which it is reflected and sensed by the detectors on the bottom of the mouse. As the mouse is moved, the reflected light beam is broken each time a dark line is crossed. The number of pulses so generated, which is equal to the number of lines crossed, are used to report mouse movements to the computer.



1.2.3 Display technologies, CRT, Raster-Scan displays

Cathode Ray Tube (CRT):

- CRT are the most common display devices on computer today. A CRT is an evacuated glass tube, with a heating element on one end and a phosphor-coated screen on the other end.
- When a current flow through this heating element (filament) the conductivity of metal is reduced due to high temperature. These cause electrons to pile up on the filament.
- These electrons are attracted to a strong positive charge from the outer surface of the focusing anode cylinder.
- Due to the weaker negative charge inside the cylinder, the electrons head towards the anode forced into a beam and accelerated by the inner cylinder walls in just the way that water is speeds up when its flow through a small diameter pipe.
- The forwarding fast electron beam is called Cathode Ray. A cathode ray tube is shown in figure below.



- There are two sets of weakly charged deflection plates with oppositely charged, one positive and another negative. The first set displaces the beam up and down and the second displaces the beam left and right.
- The electrons are sent flying out of the neck of bottle (tube) until the smash into the phosphor coating on the other end.
- When electrons strike on phosphor coating, the phosphor then emits a small spot of light at each position contacted by electron beam. The glowing positions are used to represent the picture in the screen.
- The amount of light emitted by the phosphor coating depends on the no of electrons striking the screen. The brightness of the display is controlled by varying the voltage on the control grid.

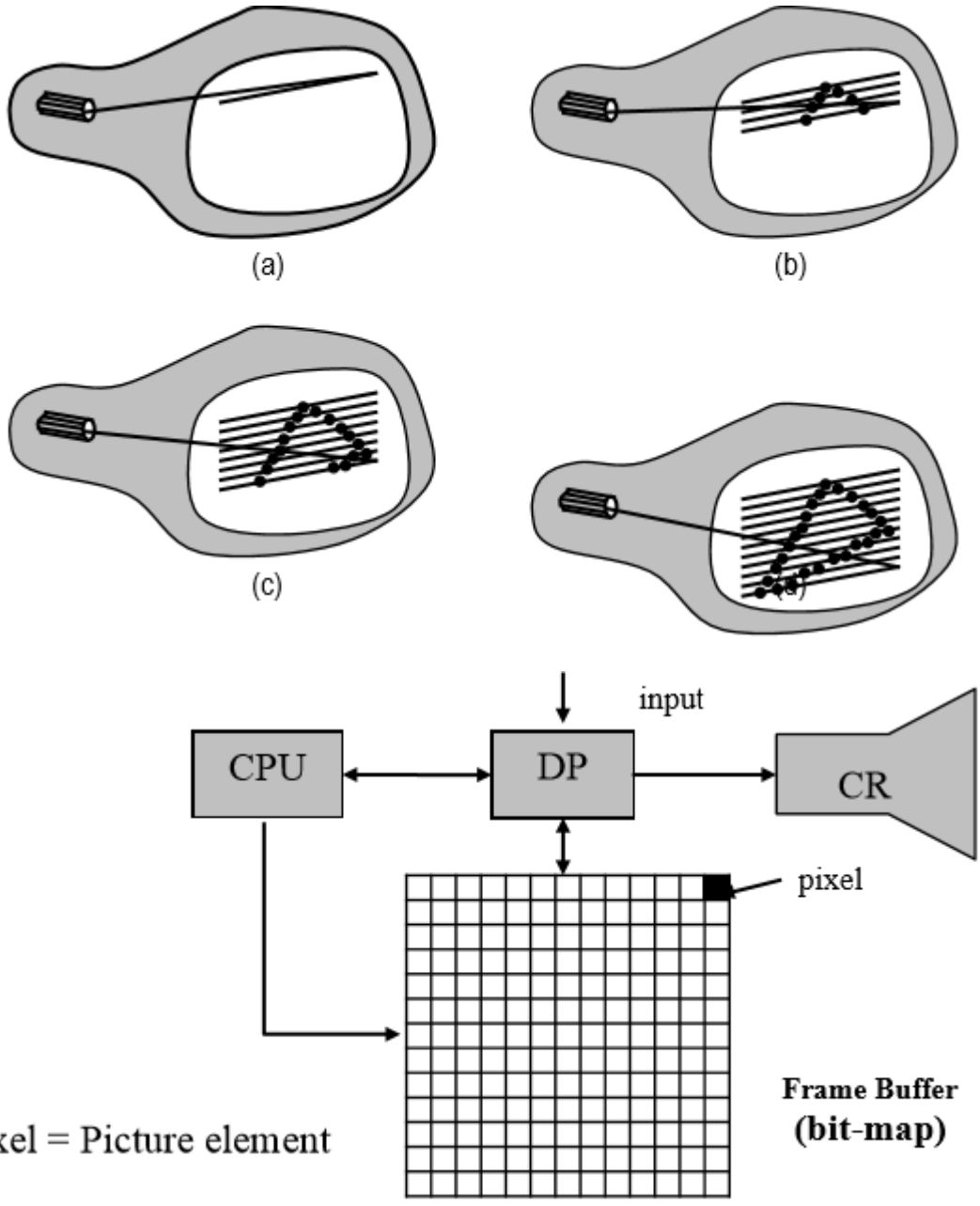
- **Persistence:**
  - How long a phosphor continues to emit light after the electron beam is removed
  - Persistence of phosphor is defined as the time it takes for emitted light to decay to 1/10 (10%) of its original intensity. Range of persistence of different phosphors can react many seconds.
  - Phosphors for graphical display have persistence of 10 to 60 microseconds. Phosphors with low persistence are useful for animation whereas high persistence phosphor is useful for highly complex, static pictures.
- **Refresh Rate:**
  - Light emitted by phosphor fades very rapidly, so to keep the drawn picture glowing constantly, it is required to redraw the picture repeatedly and quickly directing the electron beam back over the some point. The no of times/sec the image is redrawn to give a feeling of non-flickering pictures is called refresh-rate.
  - If Refresh rate decreases, flicker develops.
  - For refresh displays, it depends on picture complexity

- ❖ Refresh rate above which flickering stops and steady it may be called as critical fusion frequency(CFF).

- **Resolution:**
  - ❖ Maximum number of points displayed horizontally and vertically without overlap on a display screen is called resolution. In other ways, resolution is referred as the no of points per inch (dpi/pixel per inch).

1.2.4 Raster-Scan System

- ❖ Raster Scan Display is based on television technology. In raster-scan the electron beam is swept across the screen, one row at a time from top to bottom. No of scan line per second is called horizontal scan rate.
- ❖ As electron beam moves across each row, the beam intensity is turned on and off to create a pattern of illuminated spots. Picture definition is stored in a memory called frame buffer or refresh buffer. Frame buffer holds all the intensity value for screen points.



- The stored intensity value is retrieved from frame buffer and painted on the scan line at a time. Home television are common examples using raster display
- Intensity range for pixel position depends on capability of raster system. For B/W system each point on screen are either on or off, so only one bit per pixel is needed to control the pixel intensity. To display color with varying intensity level, additional bits are needed. Up to 24 to 32 bit per pixel are included in high quality systems, which require more space of storage for the frame buffer, depending upon the resolution of the system.
- A system with 24 bit pixel and screen resolution 1024 * 1024 require 3 megabyte of storage in frame buffer. 1024*1024 pixel = 1024*1024*24 bits = 3 MB
- The frame butter in B/W system stores a pixel with one bit per pixel so it is termed as bitmap. The frame buffer in multi bit per pixel storage, is called pixmap.
- Refreshing on Raster-Scan display is carried out at the rate of 60 or higher frames per second. 60 frames per second is also termed as 60 cycle per second usually used unit Hertz (HZ)
- Returning of electron beam from right end to deft end after refreshing each scan line is horizontal retrace. At the end of each frame, the electron beam returns to the top left corner to begin next frame called vertical retrace.

Interlaced: Display in two pass with interlacing.

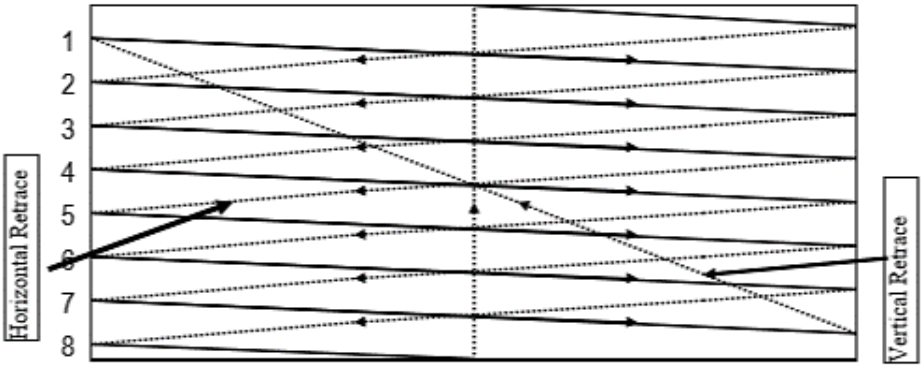


Figure: Horizontal retrace and Vertical retrace

Question: Consider a RGB raster system is to be designed using 8 inch by 10 inch screen with a resolution of 100 pixels per inch in each direction. If we want to store 6 bits per pixel in the frame buffer, how much storage (in bytes) do we need for the frame buffer?

Solution:

Size of screen = 8 inch x 10 inch.

Pixel per inch(Resolution) = 100.

Then, Total no of pixels = 8x100 x 10 x 100 pixels

Bit per pixel storage = 8

Therefore Total storage required in frame buffer = (800 x1000x8) bits

= (800 x1000x8)/8 Bytes

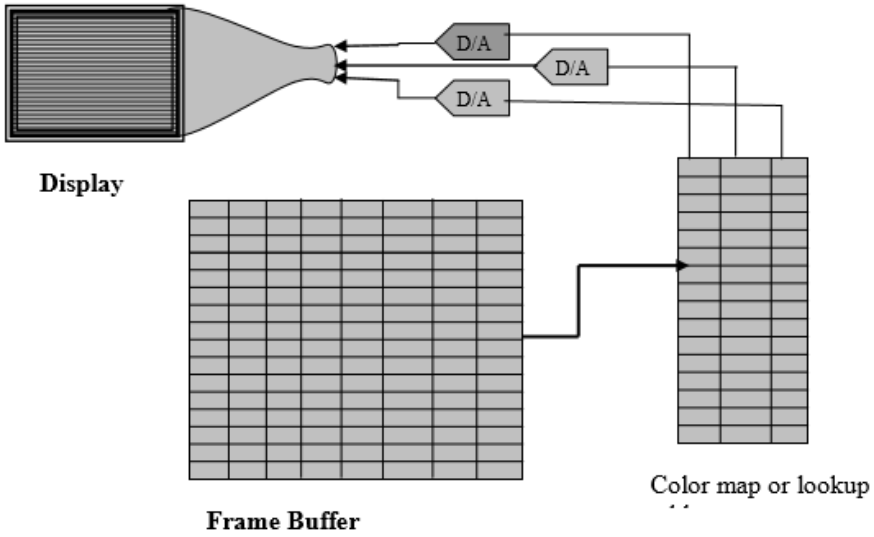
= 800000 Bytes.

Frame Buffer Architecture of Raster Display

1. **Indexed-color frame buffer:**

   In indexed-color frame buffer,

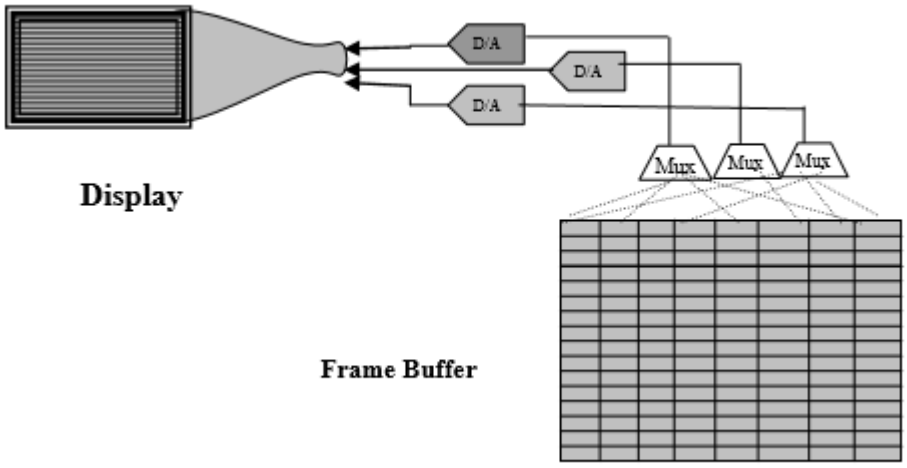
   - Each pixel uses one byte in frame buffer.
   - Each byte is an index into a color map.
   - Each pixel may be one of 324 colors, but only 256 color can be displayed at a time.
   - There is a look-up table which has as many entries as there are pixel values.
   - The table entry value is used to control the intensity or color of the CRT.



2. **True-color Frame buffer (24 bit or above):**

   In true color frame buffer,

   - Each pixel requires at least 3-bytes, one for each primary color (R,G,B)
   - Sometimes combined with a look-up table per primary.
   - Each pixel can be one of 2^{24} colors.



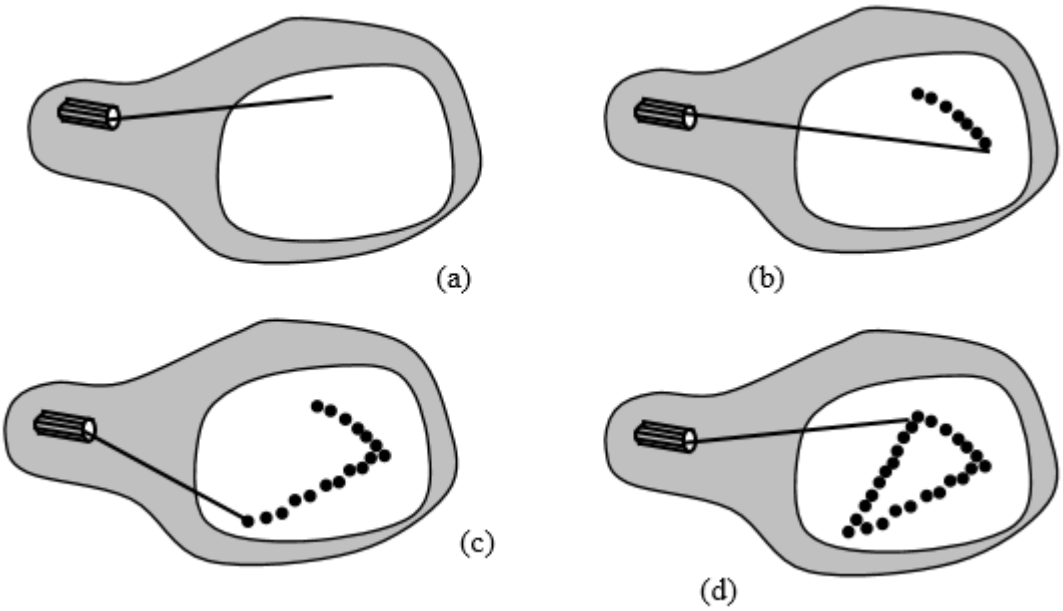
3. High-color Frame buffer:

In high-color frame buffer,

- It is popular PC/SVGA standard
- Pixels are packed in a short i.e. each primary color use 5 bit.
- Each pixel can be one of 2^{15} colors

1.2.5 Random scan display (Vector display)

In random scan system, the CRT has the electron beam that is directed only to the parts of the screen where the picture is to be drawn. It draws a picture one line at a time, so it is also called vector display (or stroke writing or calligraphic display). The component lines of a picture are drawn and refreshed by random scan system in any specified order.



- The refresh rate of vector display depends upon the no of lines to be displayed for any image. Picture definition is stored as a set of line drawing instructions in an area of memory called the refresh display file (Display list or display file)
- To display a picture, the system cycles through the set of commands (line drawing) in the display file. After all commands have been processed, the system cycles back to the first line command in the list.
- Random scan systems are designed for drawing all component lines 30 to60 times per second. Such systems are designed for line-drawing applications and can not display realistic shaded scenes. Since CRT beam directly follows the line path, the vector display system produce smooth line.

Color CRT:

In color CRT, the phosphor on the face of CRT screen are laid into different fashion. Depending on the technology of CRT there are two methods for displaying the color pictures into the screen.

1. Beam penetration method

2. Shadow mask method

Beam Penetration method: This method is commonly used for random scan display or vector display. In random scan display CRT, the two layers of phosphor usually red and green are coated on CRT screen. Display color depends upon how far electrons beam penetrate the phosphor layers.

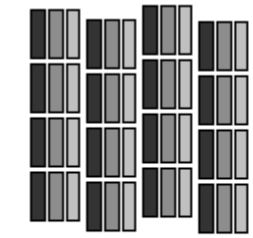
Slow electron excite only red layer so that we can see red color displayed on the screen pixel where the beam strikes. Fast electron beam excite green layer penetrating the red layer and we can see the green color displayed at the corresponding position. Intermediate is combination of red and green so two additional colors are possible – orange and yellow.

So only four colors are possible so no good quality picture in this type of display method.

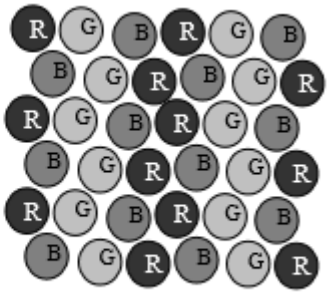
Shadow Mask Method: Shadow mask method is used for raster scan system so they can produce wide range of colors. In shadow mask color CRT, the phosphor on the face of the screen are laid out in a precise geometric pattern. There are two primary variations.

1. The stripe pattern of inline tube

2. The delta pattern of delta tube



Stripe pattern



Delta Pattern

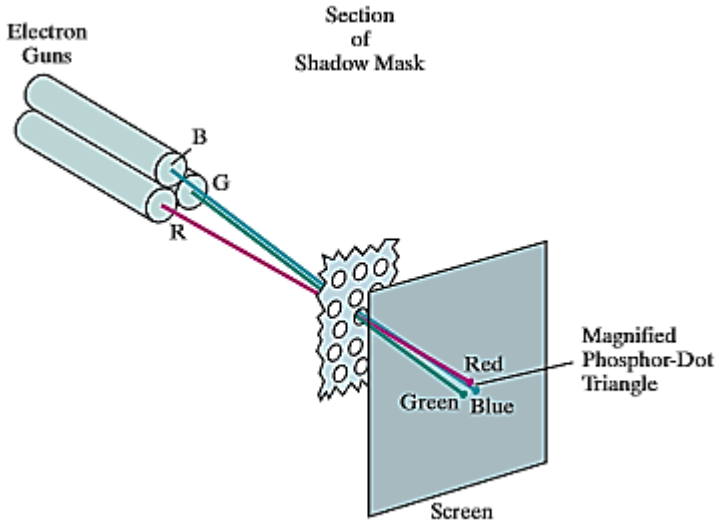
- In color CRT, the neck of tube, there are three electron guns, one for each red, green and blue colors. In phosphor coating there may be either strips one for each primary color, for a single pixel or there may be three dots one for each pixel in delta fashion.
- Special metal plate called a shadow mask is placed just behind the phosphor coating to cover front face.
- The mask is aligned so that it simultaneously allow each electron beam to see only the phosphor of its assigned color and block the phosphor of other two color.

Depending on the pattern of coating of phosphor, two types of raster scan color CRT are commonly used using shadow mask method.

1. **Delta – Delta CRT:**

- In delta-delta CRT, three electron beams one for each R,G,B colors are deflected and focused as a group onto shadow mask, which contains a series of holes aligned with the phosphor dots.
- Inner side of viewing has several groups of closely spaced red ,green and blue phosphor dot called triad in delta fashion.

- Thin metal plate adjusted with many holes near to inner surface called shadow mask which is mounted in such a way that each hole aligned with respective triad.
- Triad are so small that is perceived as a mixture of colors. When three beams pass through a hole in shadow mask, they activate the dot triangle to illuminate an small spot colored on the screen.
- The color variation in shadow mask CRT can be obtained by varying the intensity level of the three electron guns.

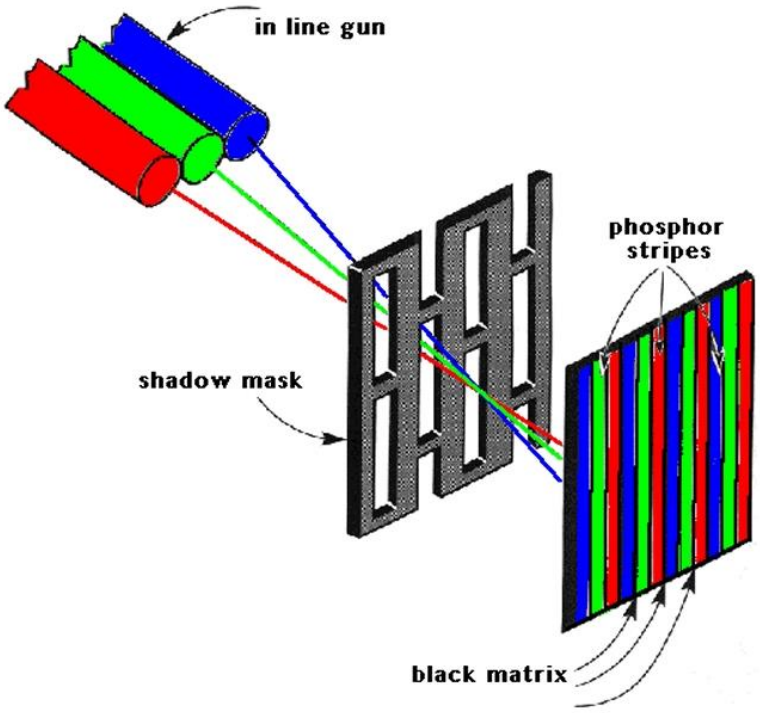


The main draw back of this CRT is due to difficulty for the alignment of shadow mask hole and respective triads.

2. A precision inline CRT:

This CRT uses strips pattern instead of delta pattern. Three strips one for each R, G, B color are used for a single pixel along a scan line so called inline. This eliminates the drawbacks of delta-delta CRT at the cost of slight reduction of image sharpness at the edge of the tube.

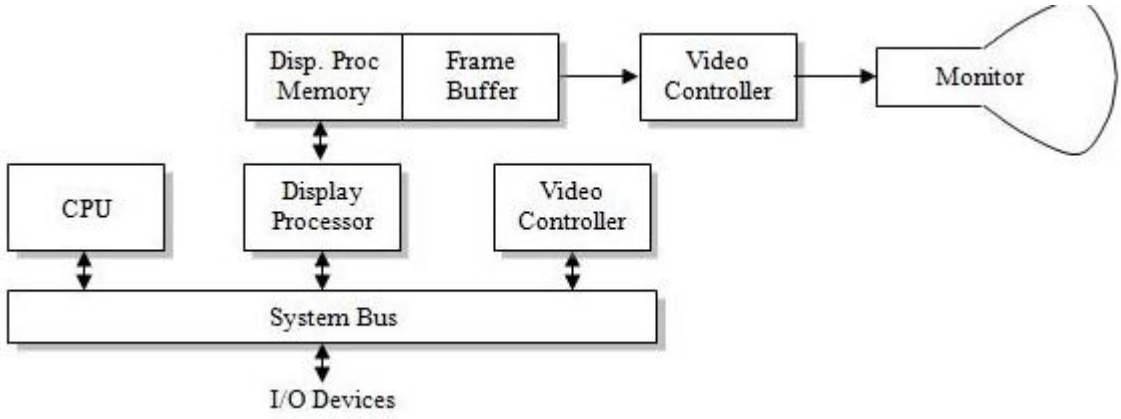
Normally 1000 scan lines are displayed in this method. Three beams simultaneously expose three inline phosphor dots along scan line.



1.2.6 Architecture of Vector and Raster Scan System

1. Architecture of Raster Scan System

The raster graphics systems typically consists of several processing units. CPU is the main processing unit of computer systems. Besides CPU, graphics system consists of a special purpose processor called video controller or display processor. The display processor controls the operation of the display device.

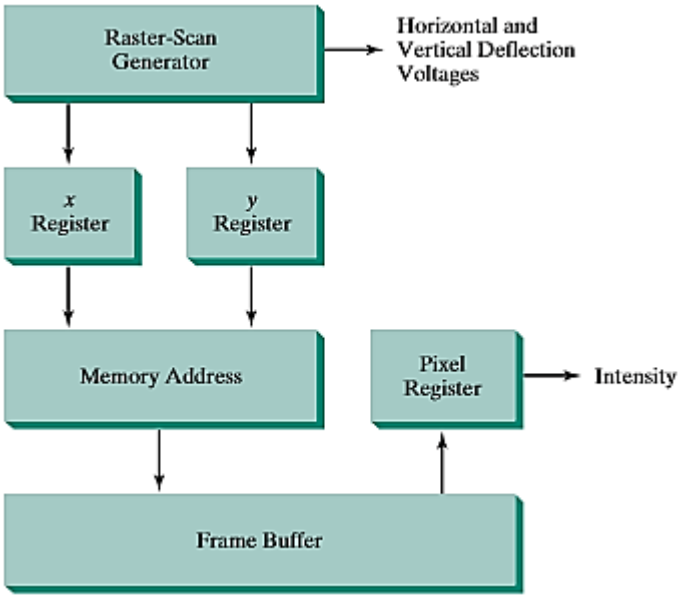


- A fixed area of system memory is reserved for the frame buffer. The video controller has the direct access to the frame buffer for refreshing the screen.
- The video controller cycles through the frame buffer, one scan line at a time, typically at 60 times per second or higher. The contents of frame buffer are used to control the CRT beam's intensity or color.

The Video Controller:

The video controller is organized as in figure below. The raster-scan generator produces deflection signals that generate the raster scan and also controls the X and Y address registers, which in turn defines memory

location to be accessed next. Assume that the frame buffer is addressed in X from 0 to X_{max} and in Y from 0 to Y_{max} then, at the start of each refresh cycle, X address register is set to 0 and Y register is set to 0.



As first scan line is generated, the X address is incremented up to X_{max}. Each pixel value is fetched and used to control the intensity of CRT beam. After first scan line X address is reset to 0 and Y address is incremented by 1. The process is continued until the last scan line ($Y=Y_{max}$) is generated.

Raster-scan Display Processor:

- The raster scan with a peripheral display processor is a common architecture that avoids the disadvantage of simple raster scan system. It includes a separate graphics processor to perform graphics functions such as scan conversion and raster operation and a separate frame buffer for image refresh.
- The display processor has its own separate memory called display processor memory.
- System memory holds data and those programs that execute on the CPU, and the application program, graphics packages and OS.
- The display processor memory holds data plus the program that perform scan conversion and raster operations.
- The frame buffer stores displayable image created by scan conversion and raster operations.

Advantages:

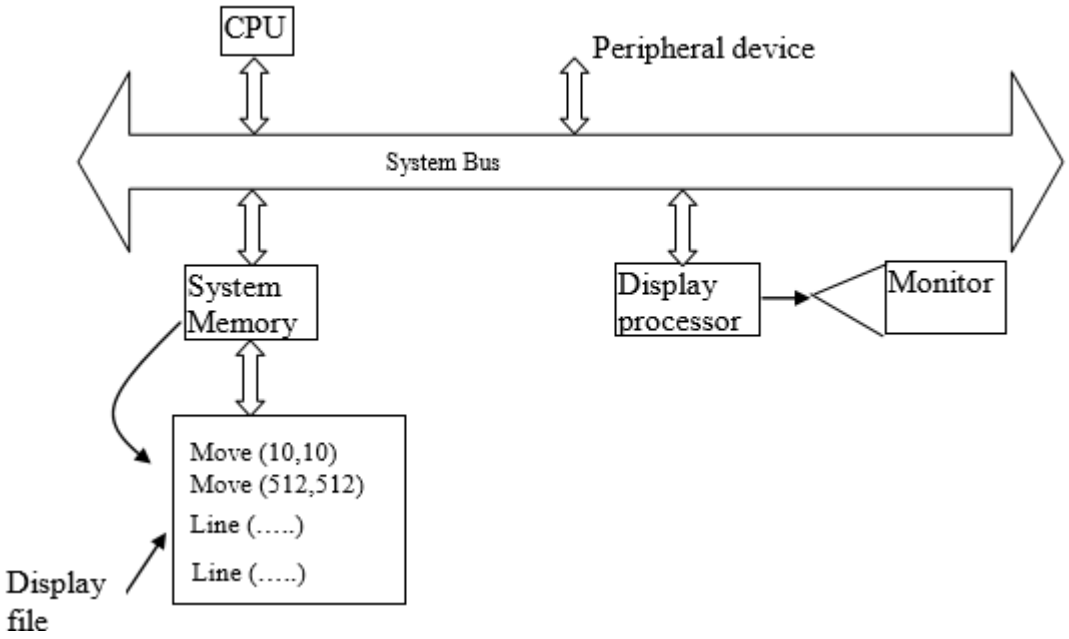
- It has an ability to fill the areas with solid colors or patterns.
- The time required for refreshing is independent of the complexity of the image.
- Low cost.

Disadvantages:

- For Real-Time dynamics not only the end points are required to move but all the pixels in between the moved end points have to be scan converted with appropriate algorithms Which might slow down the dynamic process
- Due to scan conversion "*jaggies*" or "*stair-casing*" are unavoidable

3. Architecture of Vector Scan System

- Vector display system consists of several units along with peripheral devices. The display processor is also called as graphics controller.
- Graphics package creates a display list and stores in systems memory (consists of points and line drawing commands) called display list or display file.
- Refresh time around so cycle per second.
- Vector display technology is used in monochromatic or beam penetration color CRT.
- Graphics are drawn on a vector display system by directing the electron beam along component line.



Advantages:

- Can produce output with high resolutions.
- Better for animation than raster system since only end point information is needed.

Disadvantages:

- Cannot fill area with pattern and manipulate bits.
- Refreshing image depends upon its complexity.

Difference between vector scan display and Raster scan display

S.N.	Vector Scan Display	Raster Scan Display
1	In vector scan display the beam is moved between the end points of the graphics primitives.	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.	In raster display, the refresh process is independent of the complexity of the image.
3	Scan conversion is not required.	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.	Raster display can display mathematically smooth lines, polygons, and boundaries of curved primitives only by approximating them with pixels on the raster grid.
5	Cost is more.	Cost is low.
6	Vector display draws continuous and smooth lines.	Raster display can display mathematically smooth lines, polygons, and boundaries of curved primitives only by approximating them with pixels on the raster grid.
7	Vector display only draws line characters.	Raster display has ability to display areas filled with solid colors or patterns.

1.2.7 Flat Panel Displays:

Although most graphics monitors are still constructed with CRTs, other technologies are emerging that may soon replace CRT monitors. The term flat-panel display refers to a class of video devices that have reduced volume, weight, and power requirements compared to a CRT. A significant feature of flat-panel displays is that they are thinner than CRTs, and we can hang them on walls or wear them on our wrists. Since we can even write on some flat-panel displays, they are also available as pocket note pads. Some additional uses for flat-panel displays are as small TV monitors, calculator screens, pocket video-game screens, laptop computer screens, armrest movie-viewing stations on airlines, advertisement boards in elevators, and graphics displays in applications requiring rugged, portable monitors.

We can separate flat-panel displays into two categories: emissive displays and non-emissive displays. The emissive displays (or emitters) are devices that convert electrical energy into light. Plasma panels, thin-film electroluminescent displays, and light-emitting diodes are examples of emissive displays. Flat CRTs have also been devised, in which electron beams are accelerated parallel to the screen and then deflected 90^0 onto the screen. But flat CRTs have not proved to be as successful as other emissive devices. Non emissive displays (or no emitters) use optical effects to convert sunlight or light from some other source in to graphics patterns. The most important example of a non-emissive flat-panel display is a liquid-crystal device.

Plasma panels, also called gas-discharge displays, are constructed by filling the region between two glass plates with a mixture of gases that usually includes neon. A series of vertical conducting ribbons is placed on one glass panel, and a set of horizontal conducting ribbons is built into the other glass panel. Firing voltages applied to an intersecting pair of horizontal and vertical conductors cause the gas at the intersection of the two conductors to break down into a glowing plasma of electrons and ions. Picture definition is stored in a refresh buffer, and the firing voltages are applied to refresh the pixel positions (at the intersections of the conductors) 60 times per second. Alternating-current methods are used to provide faster application of the firing voltages and, thus, brighter displays. Separation between pixels is provided

by the electric field of the conductors. One disadvantage of plasma panels has been that they were strictly monochromatic devices, but systems are now available with multicolour capabilities.

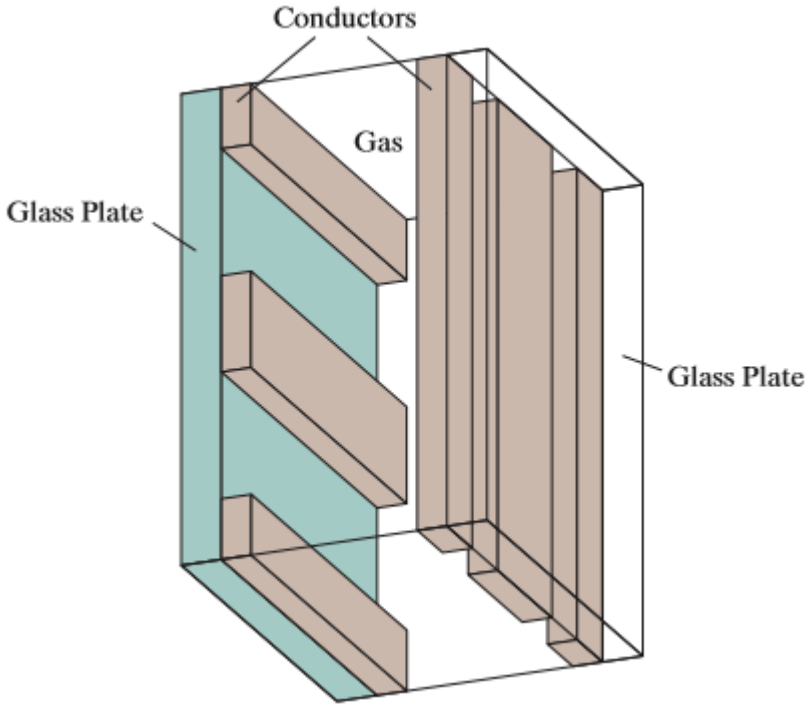


Fig: Basic design of plasma panel display

Thin-film electroluminescent displays are similar in construction to plasma panels. The difference is that the region between the glass plates is filled with a phosphor, such as zinc sulfide doped with manganese, instead of a gas. When a sufficiently high voltage is applied to a pair of crossing electrodes, the phosphor becomes a conductor in the area of the intersection of the two electrodes. Electrical energy is absorbed by the manganese atoms, which then release the energy as a spot of light similar to the glowing plasma effect in a plasma panel. Electroluminescent displays require more power than plasma panels, and good color displays are harder to achieve.

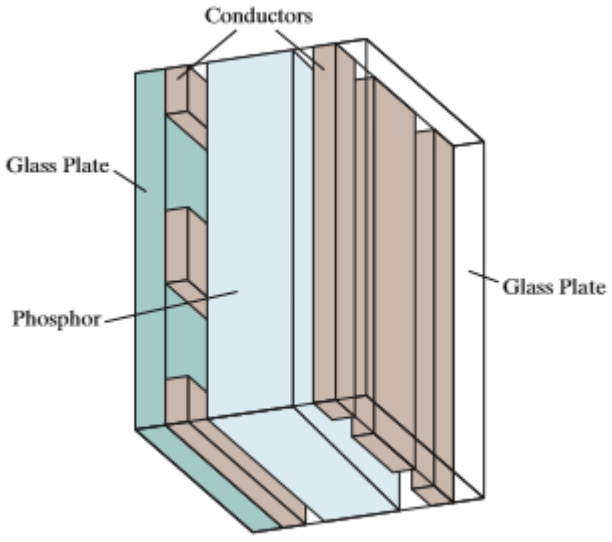


Fig: Basic design of a thin-film electroluminescent display device

Light Emitting Diode (LED): A third type of emissive device is the light-emitting diode (LED). A matrix of diodes is arranged to form the pixel positions in the display, and picture definition is stored in a refresh buffer. A sin scan-line refreshing of a CRT, information is read from the refresh buffer and converted to voltage levels that are applied to the diodes to produce the light patterns in the display.

Liquid-crystal displays (LCDs) are commonly used in small systems, such as laptop computers and calculators. These non-emissive devices produce a picture by passing polarized light from the surroundings or from an internal light source through a liquid-crystal material that can be aligned to either block or transmit the light. 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 (threadlike) liquid-crystal compounds that tend to keep the long axes of the rod-shaped molecules aligned. A flat-panel display can then be constructed with a nematic liquid crystal, as demonstrated in Fig. Two glass plates, each containing a light polarizer that is aligned at a right angle to the other plate, sandwich the liquid-crystal material. Rows of horizontal, transparent conductors are built in to one glass plate, and columns of vertical conductors are put into the other plate. The intersection of two conductors defines a pixel position. Normally, the molecules are aligned as shown in the "on state" of Fig. Polarized light passing through the material is twisted so that it will pass through the opposite polarizer. The light 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. This type of flat-panel device is referred to as a passive-matrix LCD.

.Picture definitions are stored in a refresh buffer, and the screen is refreshed at the rate of 60 frames per second, as in the emissive devices. Backlighting is also commonly applied using solid-state electronic devices, so that the system is not completely dependent on outside light sources. Colors can be displayed by using different materials or dyes and by placing a triad of color pixels at each screen location. Another method for constructing LCDs is to place a transistor at each pixel location, using thin-film transistor technology. The transistors are used to control the voltage at pixel locations and to prevent charge from gradually leaking out of the liquid-crystal cells. These devices are called active-matrix displays.

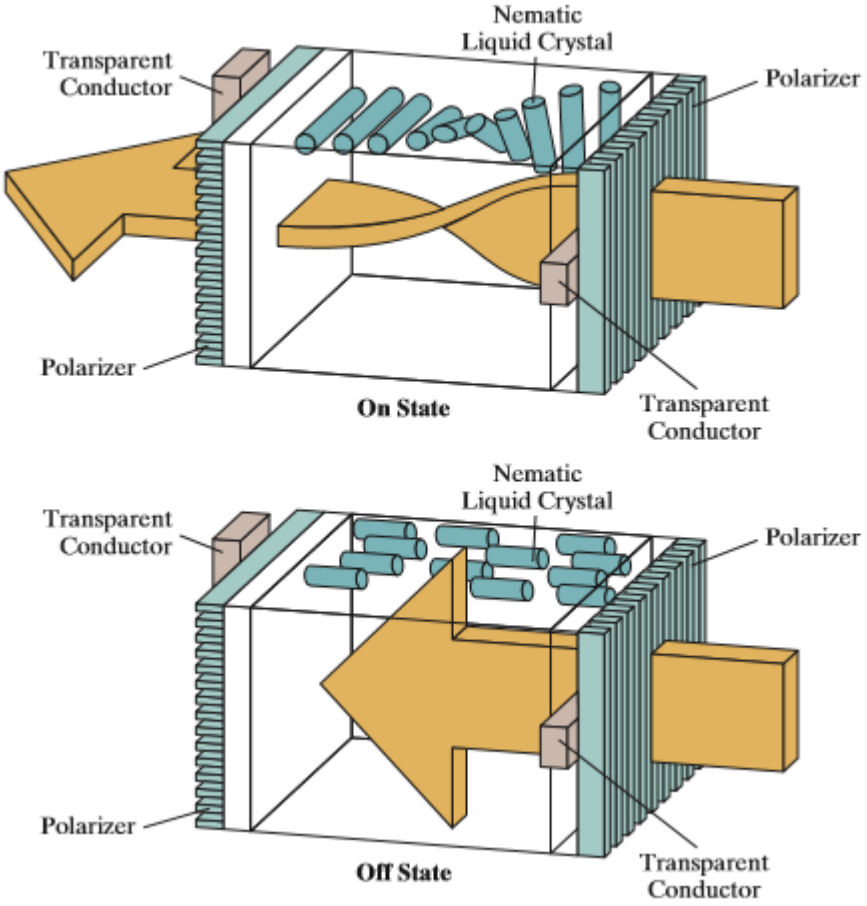


Figure: The light-twisting, shutter effect used in the design of most liquid-crystal display devices.

1.3. Line Drawing Algorithms

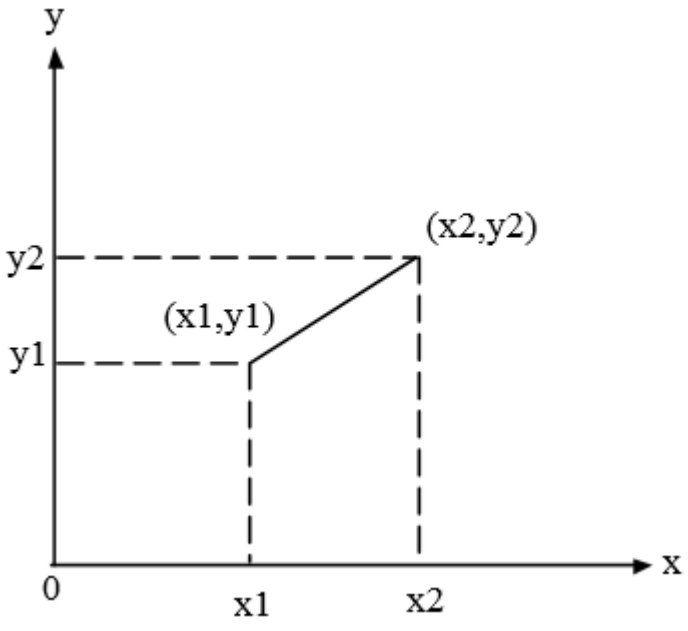
1.3.1 Scan line algorithms:

The Cartesian slope equation of a straight line as:

$$y=mx+b \quad \ldots\ldots\ldots\ldots(i)$$

Where, *m* represents the slope of the line and *b* as the y-intercept.

Suppose two end points of a line segment at positions (x_1, y_1) and (x_2, y_2) are shown in figure.



$$m = \frac{y_2 - y_1}{x_2 - x_1} \ldots\ldots\ldots\ldots \text{(ii)}$$

$$b = y - mx \ldots\ldots\ldots\ldots \text{(iii)}$$

For any given x-interval Δx along the line, we compute the corresponding y-intercept Δy from equation (ii)

$$\Delta y = m\Delta x \ldots\ldots\ldots\ldots \text{(iv)}$$

$$x = \frac{m}{y} \ldots\ldots\ldots\ldots \text{(v)}$$

These equations form the basis of determining deflection voltage in analog devices.

Case – I

For |m| < 1: Then Δx can be proportional to a small horizontal deflection voltage and the corresponding vertical deflection is set to Δy as calculated from equation (iv).

Case – II

For |m|>1: Then Δy can be set proportional to a small vertical deflection voltage with the corresponding horizontal voltage set proportional to Δx calculated from equation (v).

Case – III

For |m| = 1: Then Δx = Δy and then horizontal and vertical voltages are equal.

DDA (Digital Differential Analyzer) Line drawing algorithm:

This algorithm samples the line at unit interval in one-coordinate and determines corresponding integer values nearest the line path for other co-ordinates.

The equation of the line is,

$$y = mx + b \ldots\ldots\ldots\ldots \text{(i)}$$

$$m = \frac{y_2 - y_1}{x_2 - x_1} \ldots\ldots\ldots\ldots \text{(ii)}$$

For any interval Δx, corresponding interval is given by $\Delta y = m\Delta x$

Case – I

$|m|<1$, we sample at unit x interval i.e. Δx=1.

$$x_{k+1} = x_k + 1 \ldots\ldots\ldots\ldots \text{(iii)}$$

Then we compute each successive y-values.

Δy = m

$$y_{k+1} = y_k + m \ldots\ldots\ldots\ldots \text{(iv)}$$

Case – II

$|m| > 1$, we sample at unit y-interval i.e. $\Delta y = 1$ and compute each successive x-values.

Therefore, 1 = mΔx

$$x = \frac{1}{m}$$

$$x_{k+1} - x_k = \frac{1}{m} \ldots\ldots\ldots\ldots \text{(v)}$$

$$y_{k+1} = y_k + 1 \ldots\ldots\ldots\ldots \text{(vi)}$$

Above equation hold for the lines processed from left end to right end. For the reverse process i.e. if the line is to be processed from right to left then,

For $|m| < 1$, $\Delta x = -1$

$$x_{k+1} = x_k - 1 \ldots\ldots\ldots\ldots \text{(vii)}$$

$$y_{k+1} = y_k - m \ldots\ldots\ldots\ldots \text{(viii)}$$

For $|m| > 1$, $\Delta y = -1$

$$y_{k+1} = y_k + 1 \ldots\ldots\ldots\ldots \text{(ix)}$$

$$x_{k+1} = x_k - \frac{1}{m} \ldots\ldots\ldots\ldots \text{(x)}$$

Therefore, in general

$$y_{k+1} = y_k + m \quad \text{if } |m| < 1$$

$$x_{k+1} = x_k + \frac{1}{m} \quad \text{if } |m| > 1$$

DDA Algorithm:

1. Input the two line endpoints and store the left endpoints in (x_0, y_0).
2. Plot the first point (x_0, y_0).
3. Calculate constants Δx, Δy.
4. If $|\Delta x| > |\Delta y|$, steps = $|\Delta x|$ else steps = $|\Delta y|$
5. Calculate $x_{k+1} = |\Delta x|/\text{step}$ and $y_{k+1} = |\Delta y|/\text{steps}$
6. At each x_k along the line, starting at k=0, plot the next pixel at (x_k+x_{k+1}, y_k+y_{k+1})
7. Repeat step 6 steps times.

A complete C function for DDA algorithm:

```
void lineDDA (int x1, int y1, int x2, int y2)
{
  int dx, dy, steps, k;
  float incrx, incry, x,y;
  dx=x2-x1;
  dy=y2-y1;
   if (abs(dx)>abs(dy))
   steps=abs(dx);
  else
   steps=abs(dy);
  incrx=dx/steps;
  incry=dy/steps;
  x=x1;    /* first point to plot */
  y=y1;
  putpixel(round(x), round(y),1); //1 is used for color
        for (k=1;k<=steps;k++)
  {
   x = x + incrx;
   y = y + incry;
   putpixel(round(x),round(y),1);
}
}
```

Example: Digitize a line with end points (2, 3) and (12, 8).

Solution:

The slope of line is: $m = \frac{y_2-y_1}{x_2-x_1} = \frac{8-3}{12-2} = \frac{5}{10} = 0.5$

Here, |m| < 1

So, we sample at x interval.

The formula is given by: $y_{k+1} = y_k + m$

S.N.	x	y	$x_{plotted}$	$y_{plotted}$
1	2	3	2	3

2	3	3.5	3	4
3	4	4	4	4
4	5	4.5	5	5
5	6	5	6	5
6	7	5.5	7	6
7	8	6	8	6
8	9	6.5	9	7
9	10	7	10	7
10	11	7.5	11	8
11	12	8	12	8

Example 2: Digitize a line with end points (10,15) and (15,30).

The slope of line is: $m = \frac{y_2-y_1}{x_2-x_1} = \frac{30-15}{15-10} = \frac{15}{5} = 3$

Here, |m| > 1

So, we sample at y interval.

The formula is given by: $x_{k+1} = x_k + \frac{1}{m}$

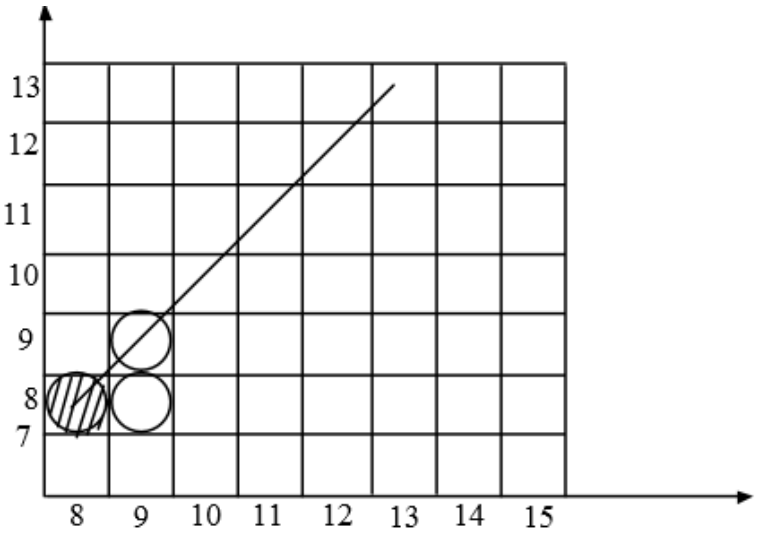
S.N.	x	y	$x_{plotted}$	$y_{plotted}$
1	10	15	10	15
2	10.3	16	10	16
3	10.6	17	11	17
4	11	18	11	18
5	11.3	19	11	19
6	11.6	20	12	20
7	12	21	12	21
8	12.3	22	12	22
9	12.6	23	13	23
10	13	24	13	24
11	13.3	25	13	25
12	13.6	26	14	26
13	14	27	14	27
14	14.3	28	14	28
15	14.6	29	15	29
16	15	30	15	30

Comments:

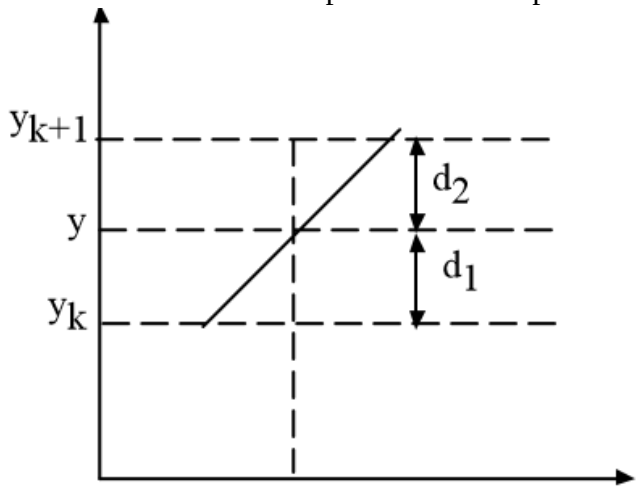
1. It is faster to calculate pixel position.
2. Due to propagation of round off errors due to successive addition the calculated pixel may shift among from the true line path.

Bresenhan's Line Algorithm (BLA):

DDA includes calculation related to m and 1/m which is little complicated. Bresenhan's improves DDA algorithm by only involving integer calculation.



Bresenhan's algorithm selects the best pixel co-ordinate by testing the sign of an integer parameter whose value is proportional to the difference between the separation of two-pixel actual line path.



Case – I: (0<m<1)

$0 < |m| < 1$

Let, (x_k, y_k) be the pixel position determined then the next pixel to be plotted is either (x_{k+1}, y_k) or (x_{k+1}, y_{k+1}).

Let, d_1 and d_2 be the separation of pixel position (x_{k+1}, y_k) and (x_{k+1}, y_{k+1}) from the actual line path.

$$y=mx+b$$

Then, at sampling position (x_{k+1})

$$y=m(x_{k+1})+b$$

From the figure,

$$d_1 = y - y_k$$

$$d_2 = y_{k+1} - y$$

$$d_1 - d_2 = (y-y_k) - (y_{k+1} - y)$$

Let us define a decision parameter P_k for the k^{th} step by

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

$\Delta x > 0$,

Therefore,

$P_k < 0$ *if* $d_1 < d_2$

$P_k >= 0$, *if* $d_1 > d_2$

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

$= \Delta x\{ y-yk- y_k-1+y\}$ since, $y_{k+1} = y_k+1$

$= \Delta x\{ 2\{ m(x_k+1)+b\} – 2\ y_k – 1\}$

$P_k = \Delta x\{2mx_k + 2m + 2b-2y_k-1\}$ since, $x_{k+1}= x_k+1$

$P_k = 2mx_k\ \Delta x – 2y_k\ \Delta x +(2m+2b-1)\ \Delta x$

$P_k = 2.\ (\Delta y/\Delta x).x_k\ \Delta x -2y_k\ \Delta x +(2m+2b-1)\ \Delta x$

$P_k = 2\ \Delta y\ x_k – 2\ \Delta x\ y_k +c$ …….. (i)

$\boxed{P_k = 2\ \Delta y\ x_k – 2\ \Delta x\ y_k +c}$

Where, $c = (2m+2b-1)\ \Delta x$ is a constant.

Now, for next step,

$P_{k+1} = 2\Delta x.\ x_{k+1} – 2\ \Delta x\ y_{k+1} + c$ ..... *(ii)*

From (i) and (ii)

$P_{k+1} – P_k = 2\ \Delta y\ (x_{k+1}- x_k) – 2\ \Delta x\ (y_{k+1}- y_k)$

i.e $p_{k+1} = p_k +2\ \Delta y – 2\ \Delta x\ (y_{k+1} –y_k)$

Where, $y_{k+1} – y_k =$ *'0' or '1'*

If $p_k < 0$, then we plot lower pixel

$y_{k+1} = y_k$ ……*(iii)*

$P_{k+1} = p_k +2\Delta y$ ……*(iv)*

If $p_k \geq 0$ then we plot upper pixel.

Therefore, $y_{k+1} = y_k+1$……*(v)*

$P_{k+1} = p_k + 2\ \Delta y – 2\ \Delta x$ ……*(vi)*

Therefore, Initial decision parameter.

$P_0 = 2\ \Delta y\ x_0 – 2\ \Delta x\ y_0 + c$ *[from (i)]*

$= 2\ \Delta y\ x_0- 2\ \Delta xy_0 +(2m+ 2b-1)\Delta x$

$= 2\Delta y\ x_0 – 2\ \Delta x\ y_0 + 2\ m\ \Delta x + 2b\ \Delta x – \Delta x$

$= 2\ \Delta y\ x_0 - 2\Delta x\ y_0 + 2.\ (\Delta y/\Delta x).\Delta x + 2\ (y_0 – mx_0)\Delta x – \Delta x$

$= 2\ \Delta y\ x_0 - 2\ \Delta x\ y_0 + 2\ \Delta y + 2\ \Delta xy_0 + 2\ \Delta y/\Delta x.\ x_0\ \Delta x – \Delta x$

Therefore,

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

Algorithm:

1. Input the two line endpoint and store the left endpoint at (x_0, y_0).
2. Load (x_0, y_0) in to frame buffer, i.e. plot the first point.
3. Calculate constants $2\Delta x$, $2\Delta y$ calculating Δx, Δy and obtain first decision parameter value as: $p_0 = 2\Delta y - \Delta x$
4. At each x_k along the line, starting at k=0, perform the following test:
   If $p_k < 0$, next point is $(x_k +1, y_k)$
   $P_{k+1} = p_k + 2\ \Delta y$
   Otherwise,
   Next point to plot is $(x_k + 1, y_k + 1)$
   $P_{k+1} = p_k + 2\Delta y - 2\Delta x$
5. Repeat step 4 Δx times.

Function implementation in C:

```
void lineBresenham (int x1, int y1, int x2, int y2)
  {
   int x, y, dx, dy, pk, k xEnd;
   dx=abs(x2-x1);
  dy=abs(y2-y1);
   if(x1>x2)
   {
    x = x2;
    y = y2;
    xEnd = x1;
   }
   Else
   {
    x = x1;
    y = y1;
    xEnd = x2;
  }
  putpixel (x,y,1);
   pk=2*dy-dx;
  while (x<=xEnd)
  {
   if(pk<0)
   {
    x=x+1;
    y=y;
    pk=pk+2*dy;
   }
   Else
   {
    x=x+1;
    y=y+1;
    pk= pk+2*dy-2*dx
    }
   putpixel (x,y,1);
  }
 }
```

Example: Digitize a line with end points (15, 18) and (10, 15) using BLA.

Solution:

$\Delta y = |15-18| = 3$

$\Delta x = |10-15| = 5$

$|m| < 1$, we sample at x direction.

First pixel is: (10, 15)

$P_0 = 2\ \Delta y - \Delta x$

$= 2 \times 3 - 5$

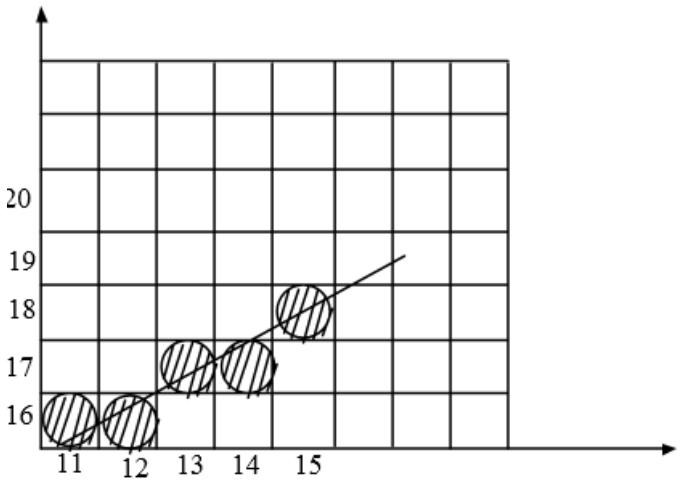
$= 1$

We know that,

If $P_k < 0$ then
$P_{k+1} = p_k + 2\Delta y$
$y_{k+1} = y_k$
if $p \geq 0$
$P_{k+1} = p_k + 2\,\Delta y - 2\,\Delta x$
$y_{k+1} = y_k + 1$

We have $p_k \geq 0$,
So, $p_1 = p_0 + 2\,\Delta y - 2\,\Delta x$
$= 1 + 2 \times 3 - 2 \times 5$
$= -3$
$P_2 = p_1 + 2\,\Delta y$
$= -3 + 2 \times 3$
$= 3$
$P_3 = p_2 + 2\Delta y - 2\,\Delta x$
$= 3 + 2 \times 3 - 2 \times 5$
$= 9 - 10$
$= -1$

k	p_k	x_{k+1}	y_{k+1}
0	1	11	16
1	-3	12	16
2	3	13	17
3	-1	14	17
4	5	15	18



Example: Digitize a line with end points (15, 15) and (10, 18) use BLE and DDA.
Solution:
$\Delta y = |15-18| = 3$
$\Delta x = |10-15| = 5$
$m = 3/5 = 0.6$
$m < 1$, we sample at x direction,

First pixel , (10, 15)
$P_0 = 2\,\Delta y - \Delta x$
$= 2 \times 3 - 5$
$= 1$
We know that,

If $p_k \geq 0$

$P_1 = p_0 + 2\,\Delta y - 2\,\Delta x$

$= 1 + 2 \times 3 - 2 \times 5$

$= -3$

$P_k < 0$

$P_2 = p_1 + 2\,\Delta y.$

$= -3 + 2 \times 3$

$= 3$

$P_3 = p_2 + 2\,\Delta y - 2\,\Delta x$

$= 3 + 2\times3 - 2\times 5$

$= -1$

$Pk < 0$

$P_4 = p_3 + 2\,\Delta y$

$= -1+ 2\times3$

$= 5$

Note:

For, $P_k \geq 0$,

$P_{k+1} = p_k + 2\Delta y - 2\Delta x$

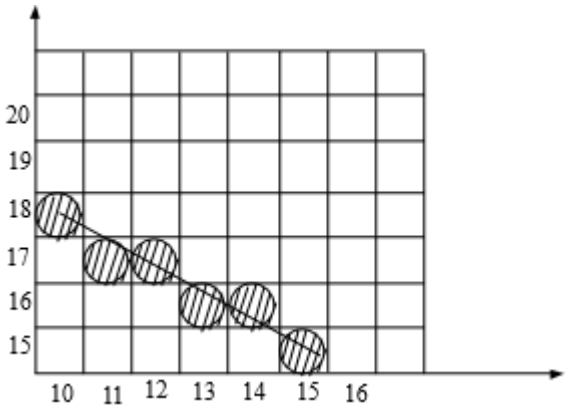
$y_{k+1} = y_k - 1$ *(when the slope is –ve)*

For, $P_k < 0$

$P_{k+1} = p_k + 2\Delta y$

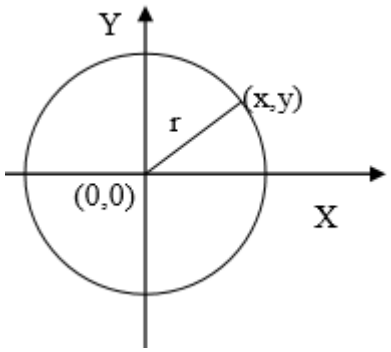
$Y_{k+1} = y_k$

k	p_k	x_{k+1}	y_{k+1}
0	1	11	17
1	-3	12	17
2	3	13	16
3	-1	14	16
4	5	15	15



1.3.2 Algorithm for Circle:

Simple Algorithm:



The equation of circle centered at origin and radius r is given by $x^2 + y^2 = r^2$

$$y = \pm\sqrt{r^2 - x^2}$$

- Increment x in unit steps and determine corresponding value of y from the equation above. Then set pixel at position (x,y).
- The steps are taken from -r to +r.
- In computer graphics, we take origin at upper left corner point on the display screen i.e. first pixel of the screen so any visible circle drawn would be centered at point other than (0,0). If center of circle is (xc, yc) then the calculated point from origin center should be moved to pixel position by (x+xc, y+yc).

In general the equation of circle centered at (xc, yc) and radius r is :

$$(x-xc)^2 + (y-yc)^2 = r^2$$

$$y = yc \pm \sqrt{r^2 - (x-xc)^2}$$

Use this equation to calculate the position of points on the circle. Take unit step from xc-r to xc+r for x value and calculate the corresponding value of y-position for pixel position (x,y).

Comments:

- Time consuming for square root and square computations.
- Non-uniform spacing, due to changing slope of curve.
- If non-uniform spacing is avoided by interchanging x and y for slope |m| >1, this leads to more computation.

C function:

```
void drawcircle(int xc,int yc,int r)
 {
  int i,x,y,y1;
  for(i=xc-r;i<=xc+r;i++)
  {
   x=i;
   y=yc+sqrt(SQUARE(r)-SQUARE(x-xc));
   y1=yc-sqrt(SQUARE(r)-SQUARE(x-xc));
   putpixel(x,y,1);
   putpixel(x,y1,1);
  }
```

Drawing circle using Polar equations:

If (x,y) be any point on the circle boundary with center (0,0) and radius r, then

$x = r\cos\theta$

$y = r\sin\theta$

i.e. $(x,y) = (r\cos\theta, r\sin\theta)$

To draw circle using these co-ordinates approach, just increment angle starting from 0 to 360. Computer (x,y) position corresponding to increment angle. Which draws circle centered at origin, but the circle centered at origin is not visible completely on the screen since (0,0) is the starting pixel of the screen. If center of circle is given by (xc, yc) then the pixel position (x,y) on the circle path will be computed as:

$x = xc + r\cos\theta$

$y = yc + r\sin\theta$

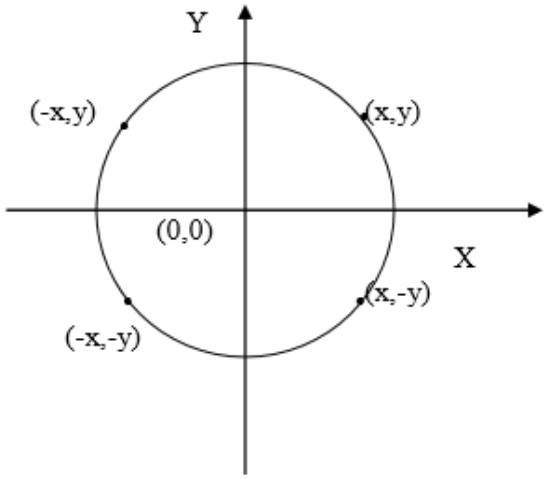
C function to draw circle using the polar transformation:

```
void drawcircle(int xc,int yc,int r)
 {
  int x,y;
  float theta;
  const float PI=3.14;
 for(theta=0.0;theta<=360;theta+=1)
```

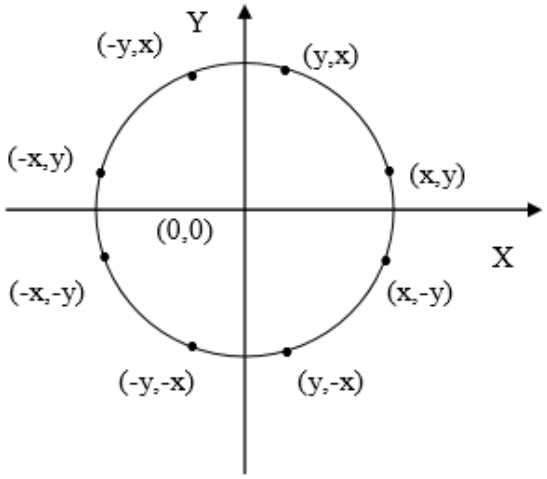
```
 {
   x= xc+r*cos(theta*PI/180.0);
   y= yc+r*sin(theta*PI/180.0);
  putpixel(x,y,1);
}
}
```

Symmetry in circle scan conversion:

We can reduce the time required for circle generation by using the symmetries in a circle. For example 4-way or 8-way symmetry. So, we only need to generate the points for one quardrant or octants and then use the symmetry to determine all the other points.



4-way symmetry



8-way symmetry

Comments:

Problem of computation still persists using symmetry since there are square roots, trigonometric functions are still not eliminated in above algorithms.

Midpoint circle algorithm:

In midpoint circle algorithm, we sample at unit intervals and determine the closest pixel position to the specified circle path at each step.

For a given radius r, and screen center position (xc, yc), we can set up our algorithm to calculate pixel positions around a circle path centered at (0,0) and then each calculated pixel position (x,y) is moved to its proper position by adding xc to x and yc to y.

$$\text{i.e. } x = x + xc$$
$$y = y+yc$$

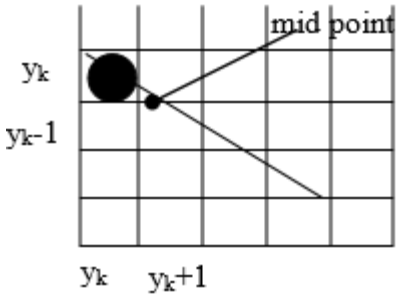
To apply the mid point method, we define a circle function as:

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

To summarize the relative position of point (x,y) by checking sign of f_{circle} function,

$$f_{circle}(x,y) \begin{cases} <0, \text{ if } (x,y) \text{ lies inside the circle boundary} \\ =0, \text{ if } (x,y) \text{ lies on the circle boundary} \\ >0, \text{ if } (x,y) \text{ lies outside the circle boundary.} \end{cases}$$

The circle function tests are performed for the mid positions between pixels near the circle path at each sampling step. Thus the circle function is decision parameter in mid point algorithm.



The figure shows the mid point between the two candidate pixel at sampling position x_k+1. Assuming we have just plotted the pixel (x_k, y_k), we next need to determine whether the pixel at postion (x_k+1, y_k) or (x_k+1, y_k-1) is closer to the circle.

Our decision parameter is circle function evaluated at the mid point,

$$p_k = f_{circle}(x_k + 1, y_k - \frac{1}{2})$$
$$= (x_k + 1)^2 + (y_k - \tfrac{1}{2})^2 - r^2$$
$$= (x_k + 1)^2 + y_k^{\,2} - y_k + \tfrac{1}{4} - r^2$$

If $p_k<0$, then mid-point lies inside the circle, so point at y_k is close to boundary otherwise, y_k-1 closer to choose next pixel position.

Successive decision parameters are obtained by incremental calculation. The decision parameter for next position is calculated by evaluating circle function at sampling position $x_{k+1}+1$ i.e. x_k+2 as:

$$p_{k+1} = f_{circle}(x_{k+1} + 1, y_{k+1} - \frac{1}{2})$$
$$= (x_{k+1} + 1)^2 + (y_{k+1} - \frac{1}{2})^2 - r^2$$
$$= (x_{k+1})^2 + 2x_{k+1} + 1 + (y_{k+1})^2 - y_{k+1} + \frac{1}{4} - r^2$$

Now, $p_{k+1} - p_k = 2x_{k+1} + (y_{k+1}^2 - y_k^2) - (y_{k+1} - y_k) + 1$
i.e. $p_{k+1} = p_k + 2x_{k+1} + (y_{k+1}^2 - y_k^2) - (y_{k+1} - y_k) + 1$
Where, y_{k+1} is either y_k or y_k-1 depending upon sign of p_k and $x_{k+1} = x_k + 1$
If p_k is negative, $y_{k+1} = y_k$ so we get,

$$p_{k+1} = p_k + 2x_{k+1} + 1$$

If p_k is positive, $y_{k+1} = y_k - 1$ so, we get,
$P_{k+1} = p_k + 2x_{k+1} + 1 - 2y_{k+1}$
Where, $2x_{k+1} = 2x_k + 2$
$2y_{k+1} = 2y_k - 2$

At the start position (0, r), these two terms have the values 0 and 2r respectively. Each successive values are obtained by adding 2 to the previous value of 2x and subtracting 2 from previous value of 2y.

The initial decision parameter is obtained by evaluating the circle function at starting position $(x_0, y_0) = (0, r)$.

$P_0 = f_{circle}(1, r - \frac{1}{2})$
$= 1 + (r - \frac{1}{2})^2 - r^2$
$= 1 + r^2 - r + ¼ - r^2$
$= 5/4 - r$

If p_0 is specified in integer,

$$P_0 = 1 - r$$

The Algorithm:

1. Input radius r and circle centre (x_c, y_c), and obtain the first point on circle centered at origin as: $(x_0, y_0) = (0, r)$.
2. Calculate initial decision parameter
   $P_0 = \frac{5}{4} - r$
3. At each x_k position, starting k = 0, perform the tests:
   If $p_k < 0$ next point along the circle centre at (0,0) is (x_k+1, y_k)
   $P_{k+1} = p_k + 2x_{k+1} + 1$
   Otherwise, the next point along circle is $(x_k+1, y_k - 1)$
   $P_{k+1} = p_k + 2x_{k+1} + 1 - 2y_{k+1}$
   Where, $2x_{k+1} = 2x_k + 2$ and $2y_{k+1} = 2y_k-2$
4. Determine symmetry point on the other seven octants.
5. Move each calculated pixels positions (x,y) in to circle path centered at (x_c, y_c) as:
   $x = x+x_c$, $y = y+y_c$
6. Repeat 3 through 5 until $x \geq y$.

C function for Circle drawing algorithm

```
void drawcircle(int xc,int yc,int r)
 {
  int p,x,y;
  x=0;
  y=r;
drawpoints(x,y,xc,yc);
  p=1-r;
  while(x<y)
  {
   if(p<0)
   {
    x=x+1;
    p=p+2*x+1;
  }
   Else
   {
    x=x+1;
    y=y-1;
    p=p+2*(x-y)+1;
   }
  drawpoints(x,y,xc,yc);
}
}
```

Example: Digitize $x^2 + y^2 = 100$ in first octant.

Solution:

Given, Centre = (0,0)

Radius (r) = 10

We demonstrate the mid point circle algorithm by determining positions along the circle octant in the first quadrant from x=0, to x=y.

The initial value of the decision parameter is:

$P_0 = 1 - r$
$= 1 - 10$
$= -9$

Successive decision parameter values are positions along the circle path are calculated using mid-point method as,

$P_{k+1} = p_k + 2x_{k+1}+1 \ (p_k < 0)$

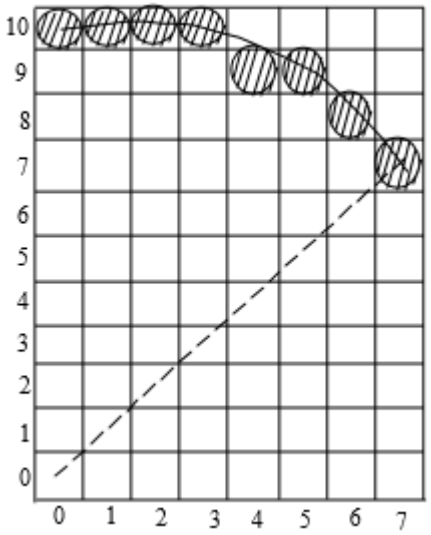
Set : $x_{k+1} = x_k+1$, $y_{k+1} = y_k$

$P_{k+1} = p_k + 2x_{k+1} - 2y_{k+1} + 1 \ (p_k>0)$

Set : $x_k+1 = x_k+1$, $y_{k+1} = y_k - 1$

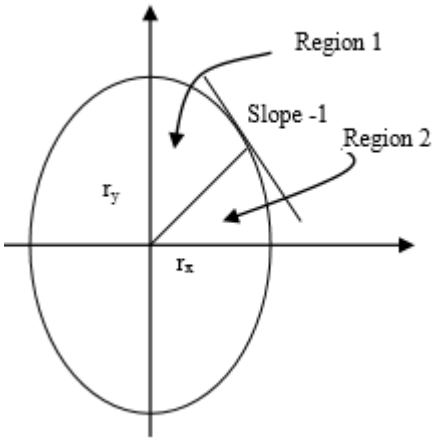
Do until, (x>=y).

K	p_k	x_{k+1}, y_{k+1}	$2x_{k+1}$	$2y_{k+1}$
1	-9	(1,10)	2	20
1	-6	(2,10)	4	20
2	-1	(3,10)	6	20
3	6	(4,9)	8	18
4	-3	(5,9)	10	18
5	8	(6,8)	12	16
6	5	(7,7)	14	14



1.3.3 Ellipse Algorithm:

The basic algorithm for drawing ellipse is same as circle computing x and y position at the boundary of the ellipse from the equation of ellipse directly.



We have equation of ellipse centered at origin (0,0) is

$$\frac{x^2}{r_x^2}+\frac{y^2}{r_y^2}=1 \text{ which gives}$$

$$y=\pm\frac{r_y}{r_x}\sqrt{(r_x^2-x^2)} \quad \text{.....(1)}$$

Stepping unit interval in x direction from $-r_x$ to r_x we can get corresponding y value at each x position which gives the ellipse boundary co-ordinates. Plotting these computed points we can get he ellipse.
If center of ellipse is any arbitrary point (xc, yc) then the equation of ellipse can be written as:

$$\frac{(x-xc)^2}{r_x^2}+\frac{(y-yc)^2}{r_y^2}=1$$

$$\text{i.e. } y=yc\pm\frac{r_y}{r_x}\sqrt{r_x^2-(x-xc)^2} \quad \text{----------(2)}$$

For any point (x,y) on the boundary of the ellipse if major axis of ellipse with major axis along x-axis the algorithm based on the direct computation of ellipse boundary points can be summarized as:

1. Input the center of ellipse (xc, yc) x-radius xr and y-radius yr.
2. For each x position starting from xc-r and stepping unit interval along x-direction, compute corresponding y positions as:

$$y=yc\pm\frac{r_y}{r_x}\sqrt{r_x^2-(x-xc)^2}$$

3. Plot the point (x,y).
4. Repeat step 2 to 3 until x > = xc+xr

Computation of ellipse using polar co-ordinate:

Using the polar co-ordinates for ellipse, we can compute the (x,y) position of the ellipse boundary using the following parametric equations:

$x=xc+r\cos\theta$

$y=yc+r\sin\theta$

The algorithm based on these parametric equation on polar co-ordinates can be summarized as below:

1. Input center of ellipse (xc, yc) and radii xr and yr.
2. Starting θ from angle 0^0 step minimum increments and compute boundary point of ellipse as:
   $x = xc + r\cos\theta$
   $y = yc + r\sin\theta$
3. Plot the point at position (round(x), round(y))
4. Repeat until θ is greater or equal to 360^0.

C function for drawing ellipse:

```
void drawellipse(int xc, int yc, int rx, in try)
{
int x, y;
float theta;
const float PI=3.14;
for(theta=0.0;theta<=360;theta+=1)
  {
   x= xc+rx*cos(theta*PI/180.0);
   y= yc+ry*sin(theta*PI/180.0);
   putpixel(x,y,1);
 }
}
```

The methods of drawing ellipses explained above are not efficient. The method based on direct equation of ellipse must perform the square and square root operations due to which there may be floating point number computation which cause rounding off to plot the pixels. Also the square root causes the domain error. Due to the changing slope of curve along the path of ellipse, there may be un-uniform separation of pixel when slope changes. To eliminate this problem, extra computation is needed. Although, the method based on polar co-ordinate parametric equation gives the uniform spacing of pixel due to uniform increment of angle but it also take extra computation to evaluate the trigonometric functions. So these algorithms are not efficient to construct the ellipse. We have another algorithm called mid- point ellipse algorithm similar to mid-point circle algorithm which is efficient algorithm for computing ellipse.

Mid-point Ellipse Algorithm:

The mid-point ellipse algorithm decides which point near the boundary (i.e. path of the ellipse) is closer to the actual ellipse path described by the ellipse equation. That point is taken as next point.

It is applied to the first quadrant in two parts as in figure. Region 1 and Region 2. We process by taking unit steps in x-coordinates direction and finding the closest value for y for each x-steps in region 1.

In first quadrant at region 1, we start at position (0, xy) and incrementing x and calculating y closet to the path along clockwise direction. When slope becomes -1 then shift unit step in x to y and compute corresponding x closest to ellipse path at Region 2 in same direction.

Alternatively, we can start at position (rx, 0) and select point in counter clockwise order shifting unit steps in y to unit step in x when slope becomes greater than -1.

Here, to implement mid-point ellipse algorithm, we take start position at (0, ry) and step along the ellipse path in clockwise position throughout the first quadrant.

We define ellipse function center at origin i.e. (xc, yc) = (0,0) as

$$f_{ellipse}(x,y) = r_y^2 x^2 + r_x^2 y^2 - r_x^2 r_y^2 y$$

$$f_{ellipse}(x,y) = \begin{cases} < 0, \text{ if (x,y) lies inside boundary of ellipse.} \\ = 0 \text{ if (x,y lies on the boundary of ellipse.} \\ >0 \quad \text{if} \quad \text{(x,y)} \quad \text{lies} \quad \text{outside} \quad \text{the} \quad \text{boundary} \quad \text{of} \quad \text{ellipse.} \end{cases}$$

So $f_{ellipse}$ function serves as decision parameter in ellipse parameter in ellipse algorithm at each sampling positon. We select the next pixel position according to the sign of decision parameter.

Starting at $(0, r_y)$, we take unit step in x-direction until we reach the boundary between the region 1 and region 2. Then we switch unit steps in y over the remainder of the curve in first quadrant. At each step, we need to test the slope of curve. The slope of curve is calculated as;

$$\frac{dy}{dx} = -\frac{2r_y^2 x}{2r_x^2 y}$$

At the boundary between region 1 and region 2,

$$\frac{dy}{dx} = -1 \text{ and } 2r_y^2 = 2r_x^2 \quad \text{Therefore , we move out of region 1 when } 2r_y^2 x \geq 2r_x^2$$

Assuming the position (x_k, y_k) is filled, we move x_{k+1} to determine next pixel. The corresponding y value for x_{k+1} position will be either y_k or y_k-1 depending upon the sign of decision parameter. So the decision parameter for region 1 is tested at mid point of $(x_k + 1, y_k)$ and $(x_k+1, y_k – 1)$ i.e.

$$p_{1k} = f_{ellipse}(x_{k+1}, y_k - \frac{1}{2})$$

$$\text{or } p_{1k} = r_y^2 (x_{k+1})^2 + r_x^2 (y_k - \frac{1}{2})^2 - r_x^2 r_y^2$$

$$\text{or } p_{1k} = r_y^2 (x_{k+1})^2 + r_x^2 y_k^2 - r_x^2 y_k + \frac{r_x^2}{4} - r_x^2 r_y^2 \quad \text{.....(1)}$$

If $p_{1k}<0$, the mid point lies inside boundary, so next point to plot is (x_k+1, y_k) otherwise, next point to plot will be (x_k+1, y_k-1)

The successive decision parameter is computed as:

$$p_{1k+1} = f_{ellipse}(x_{k+1} + 1, y_{k+1} - \frac{1}{2})$$

$$= r_y^2 (x_{k+1} + 1)^2 + r_x^2 (y_{k+1} - \frac{1}{2})^2 - r_x^2 r_y^2$$

$$\text{or } p_{1k+1} = r_y^2 (x_{k+1}^2 + 2x_{k+1} + 1) + r_x^2 (y_{k+1}^2 - y_{k+1} + \frac{1}{4}) - r_x^2 r_y^2$$

$$\text{or } p_{1k+1} = r_y^2 x_{k+1}^2 + 2r_y^2 x_{k+1} + r_y^2 + r_x^2 y_{k+1}^2 - r_x^2 y_{k+1} + \frac{r_x^2}{4} - r_x^2 r_y^2 \quad \text{.....(2)}$$

Subtracting (2) - (1)

$$p_{1k+1} - p_{1k} = 2r_y^2 x_{k+1} + r_y^2 + r_x^2 (y_{k+1}^2 - y_k^2) - r_x^2 (y_{k+1} - y_k)$$

if $p_{1k} < 0$, $y_{k+1} = y_k$ then,

$\therefore p_{1k+1} = p_{1k} + 2r_y{}^2 x_{k+1} + r_y{}^2$

Otherwise $y_{k+1} = y_k - 1$ then we get,

$$p_{1k+1} = p_{1k} + 2r_y{}^2 x_{k+1} + r_y{}^2 - 2r_x{}^2 y_{k+1}$$

At the initial position, $(0,r_y)$ $2r_y{}^2 x = 0$ and $2r_x{}^2 y = 2r_x^2 r_y$

In region 1, initial decision parameter is obtained by evaluating ellipse function at $(0,r_y)$ as

$$p_{10} = f_{ellipse}(1, r_y - \frac{1}{2})$$

or $$p_{10} = f_{ellipse}(1, r_y - \frac{1}{2})$$

$$= r_y^2 - r_x^2 r_y + \frac{1}{4} r_x{}^2$$

Similarly, over the region 2, the decision parameter is tested at mid point of $(x_k, y_k - 1)$ and $(x_k + 1, y_k - 1)$ i.e.

$$p_{2k} = f_{ellipse}(x_k + \frac{1}{2}, y_k - 1)$$

$$= r_y{}^2 (x_k + \frac{1}{2})^2 + r_x{}^2 (y_k - 1)^2 - r_x{}^2 r_y{}^2$$

$$\therefore p_{2k} = r_y{}^2 x_k{}^2 + r_y^2 x_k + \frac{r_y^2}{4} + r_x{}^2 (y_k - 1)^2 - r_x{}^2 r_y{}^2 \quad \text{.....(3)}$$

If $p_{2k} > 0$, the mid point lies outside the boundary, so next point to plot is (x_k, y_k-1) otherwise, next point to plot will be (x_k+1, y_k-1)

The successive decision parameter is computed as evaluating ellipse function at mid point of

$$p_{2k+1} = f_{ellipse}(x_{k+1} + \frac{1}{2}, y_{k+1} - 1) \text{ with } y_{k+1} = y_k - 1$$

$$p_{2k+1} = r_y^2 (x_{k+1} + \frac{1}{2})^2 + r_x^2 [(y_k - 1) - 1]^2 - r_x^2 r_y^2$$

or $p_{2k+1} = r_y^2 x_{k+1}^2 + r_y^2 x_{k+1} + \frac{r_y^2}{4} + r_x^2 (y_k - 1)^2 - 2r_x^2 (y_k - 1) + r_x^2 - r_x^2 r_y^2$ .....(4)

subtracting (4)-(3)

$$p_{2k+1} - p_{2k} = r_y^2 (x_{k+1}^2 - x_k^2) + r_y^2 (x_{k+1} - x_k) - 2r_x^2 (y_k - 1) + r_x^2$$

or $p_{2k+1} = p_{2k} + r_y^2 (x_{k+1}^2 - x_k^2) + r_y^2 (x_{k+1} - x_k) - 2r_x^2 (y_k - 1) + r_x^2$

if $p_{2k} > 0$, $x_{k+1} = x_k$ then

$$p_{2k+1} = p_{2k} - 2r_x^2 (y_k - 1) + r_x^2$$

otherwise $x_{k+1} = x_k + 1$ then

$$p_{2k+1} = p_{2k} + r_y^2 [(x_k + 1)^2 - x_k^2)] + r_y^2 (x_k + 1 - x_k) - 2r_x^2 (y_k - 1) + r_x^2$$

or $p_{2k+1} = p_{2k} + r_y^2 (2x_k + 1) + r_y^2 - 2r_x^2 (y_k - 1) + r_x^2$

or $p_{2k+1} = p_{2k} + r_y^2 (2x_k + 2) - 2r_x^2 (y_k - 1) + r_x^2$

or $p_{2k+1} = p_{2k} + 2r_y^2 x_{k+1} - 2r_x^2 y_{k+1} + r_x^2$ where $x_{k+1} = x_k + 1$ and $y_{k+1} = y_k - 1$

The initial position for region 2 is taken as last position selected in region 1 say which is (x_0, y_0) then initial decision parameter in region 2 is obtained by evaluating ellipse function at mid point of (x_0, y_0-1) and (x_0+1, y_0-1) as:

$$p_{20} = f_{ellipse}(x_0 + \frac{1}{2}, y_0 - 1)$$

$$= r_y^2 (x_0 + \frac{1}{2})^2 + r_x^2 (y_0 - 1)^2 - r_x^2 r_y^2$$

Mid point ellipse algorithm:

1. Input center (xc, yc) and r_x and r_y for the ellipse and obtain the first point as $(x_0, y_0) = (0, r_y)$
2. Calculate initial decision parameter value in Region 1 as:
$$P_{10} = r_y^2 - r_x^2 r_y + \frac{1}{4} r_x^2$$
3. At each x_k position, in Region 1, starting at k=0, compute
   $x_{k+1} = x_k + 1$
   If $p_{1k} < 0$, then the next point to plot is
   $P_{1k+1} = p_{1k} + 2r_y^2 x_{k+1} + r_y^2$
   $y_{k+1} = y_k - 1$
   $p_{1k+1} = p_{1k} + 2r_y^2 x_{k+1} + r_y^2 - 2r_x^2 y_{k+1}$ with $x_{k+1} = x_k + 1$ and $y_{k+1} = y_k - 1$
4. Calculate the initial value of decision parameter at region 2 using last calculated point say (x_0, y_0) in region 1 as:
$$p_{20} = r_y^2 (x_0 + \frac{1}{2})^2 + r_x^2 (y_0 - 1)^2 - r_x^2 r_y^2$$

5. At each y_k position in Region 2 starting k = 0, perform computation:
   $y_{k+1} = y-1;$
   if $p_{2k} > 0$, then
   $x_{k+1} = x_k$
   $p_{2k+1} = p_{2k} - 2r_x^2(y_k - 1) + r_x^2$
   Otherwise
   $x_{k+1} = x_k + 1$
   $p_{2k+1} = p_{2k} + 2r_y^2 x_{k+1} - 2r_x^2 y_{k+1} + r_x^2$ where $x_{k+1} = x_k + 1$ and $y_{k+1} = y_k - 1$
6. Determine the symmetry points in other 3 quadrants.
7. Move each calculated point (x_k, y_k) on to the centered (xc, yc) ellipse path as
   $x_k = x_k + xc$
   $y_k = y_k + yc$
8. Repeat the process for region 1 until $2r_y^2 x_k >= 2r_x^2 y_k$ and region until (x_k, y_k) = (r_x, 0)

For example: Digitize an ellipse $(x-2)^2/64 + (y+5)^2/36 = 1$ using Mid point algorithm.
Solution:
Given input ellipse parameters $r_x = 8$ and $r_y = 6$, we illustrate the steps in the midpoint ellipse algorithm by determining raster positons along the ellipse path in the first quadrant. Initial values and increments for the decision parameter calculations are:
Let a= r_x and b= r_y
Center = (2, -5)
a=8
b=6
For region R_1
First pixel is (0,6)
$P_0 = b^2 - a^2b + a^2/4$
$= 36 - 64 \times 6 + 64/4$
$= -332$
Successive decision parameter values at positions along the ellipse path are calculated using the midpoint method as:
If $p_k < 0$, $p_{k+1} = p_k + 2b^2x_{k+1} + b^2$
If p > = 0, $p_{k+1} = p_k + 2b^2x_{k+1} - 2a^2y_{k+1} + b^2$ where $x_{k+1} = x_k + 1$, $y_{k+1} = y_k - 1$

K	p_k	(x_{k+1}, y_{k+1})	$2b^2x_{k+1}$	$2a^2y_{k+1}$
0	-332	(1,6)	72	768
1	-224	(2,6)	144	768
2	-44	(3,6)	216	768
3	-208	(4,5)	288	640
4	-108	(5,5)	360	640
5	288	(6,4)	432	512
6	244	(7,3)	504	384

We now move out of region 1, since $2b^2x > 2a^2y$
For region R2,
The initial point is (x_0, y_0) = (7,3) and the initial decision parameter is:
$P_0 = b^2 (x_0+1/2) + a^2 (y-1)^2 + a^2b^2$
$= 36 (7+1/2)^2 + 64 (3-1)^2 - 36 \times 64$
$= 36 \times 225/4 + 64 \times 4 - 36 \times 64$
$= -151$

The remaining positions along the ellipse path in the first quadrant are then calculated as:

If $p_k > 0$

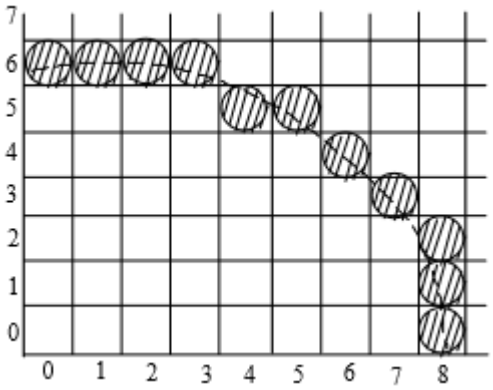
$P_{k+1} = p_k - 2a^2y_{k+1} + a^2$ where, $x_{k+1} = x_k$, $y_{k+1} = y_k - 1$

If $p_k >= 0$

$P_{k+1} = p_k + 2b^2x_{k+1} - 2a^2y_{k+1} + a^2$ where, $x_{k+1} = x_k + 1$, $y_{k+1} = y_k - 1$

K	p_k	(x_{k+1}, y_{k+1})	$2b^2x_{k+1}$	$2a^2y_{k+1}$
0	-151	(8,2)	576	256
1	233	(8,1)	576	128
	745	(8,0)	-	-

A plot of the selected positions around the ellipse boundary within the first quadrant is shown in fig:



1.4 General Scan-line Polygon fill algorithm:

A scan line fill of a region is performed by first determining the intersection positions of the boundaries of the fill region with the screen scan lines. Then the fill colors are applied to each section of a scan line that lies within the interior of the fill region. The scan line fill algorithm identifies the same interior regions as the odd-even rule. The simplest area to fill is a polygon, because each scan-line intersection point with a polygon boundary is obtained by solving a pair of simultaneous linear equations, where the equation for the scan line is simply y = constant.

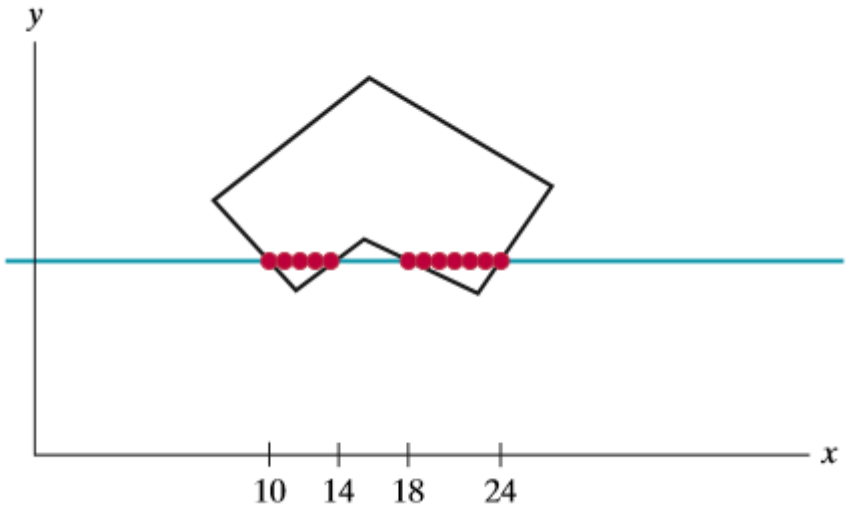


Fig: Interior pixels along a scan line passing through a polygon fill area

The above figure illustrates the basic scan-line procedure for a solid-color fill of a polygon. For each scan line that crosses the polygon, the edge intersections are sorted from left to right, and then the pixel positions between, and including, each intersection pair are set to the specified fill color. As an example of above figure, the four pixel intersection positions with the polygon boundaries define two stretches of

interior pixels. Thus, the fill color is applied to the five pixels from x=10 to x=14 and to the seven pixels from x=18 to x=24. If a pattern fill is to be applied to the polygon, then the color for each pixel along a scan line is determined from its overlap position with the fill pattern.

However, the scan-line fill algorithm for a polygon is not quite as simple as the above figure. Whenever a scan line passes through a vertex, it intersects two polygon edges at the point. In some cases, this can result in an odd number of boundary intersections for a scan line. Following figure shows two scan lines that cross a polygon fill area and intersect a vertex.

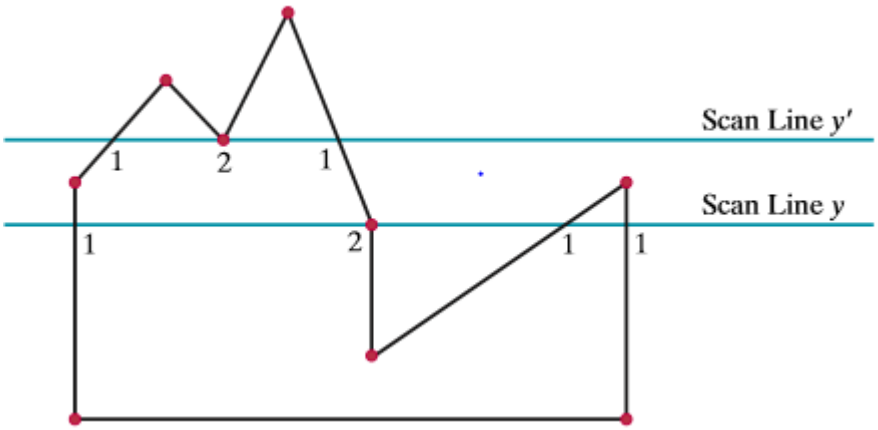
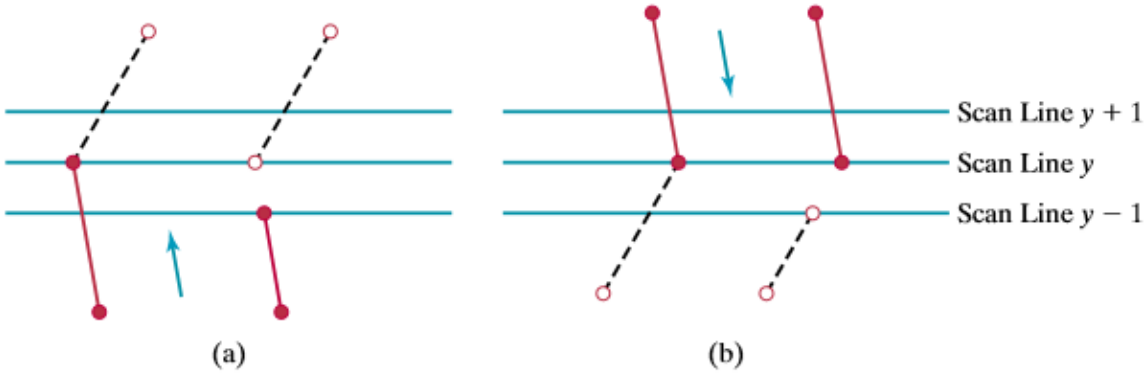


Fig: Intersection points along scan lines that intersect polygon vertices.

Scan line y' intersects an even number of edges, and the two pairs of intersection points along this scan line correctly identify the interior pixel spans. But scan line y intersects five polygon edges. To identify the interior pixels for scan line y, we must count the vertex intersection as only one point. Thus, as we process scan lines, we need to distinguish between these cases.

We can detect the topological difference between scan line y and scan line y' in above figure by noting the position of the intersecting edges relative to the scan line. For scan line y, the two edges sharing an intersection vertex are on opposite sides of the scan line.. But for scan line y', the two intersecting edges are both above the scan line. Thus, a vertex that has adjoining edges on opposite sides of an intersecting scan line should be counted as just one boundary intersection point. We can identify these vertices by tracing around the polygon boundary in either clockwise or counter clockwise order and observing the relative changes in vertex y coordinates as we move from one edge to the next. If the three endpoint y values of two consecutive edges monotonically increase or decrease, we need to count the shared (middle) vertex as a single intersection point for the scan line passing through that vertex. Otherwise, the shared vertex represents a local extremum (minimum or maximum) on the polygon boundary, and the two edge intersections with the scan line passing through that vertex can be added to the intersection list.

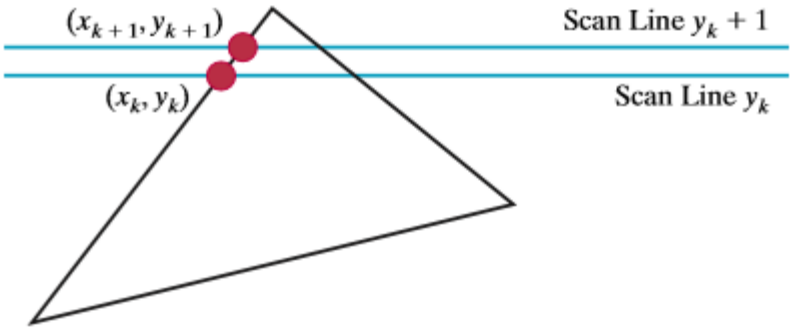
One method for implementing the adjustment to the vertex-intersection count is to shorten some polygon edges to split those vertices that should be counted as one intersection. We can process nonhorizontal edges around the polygon boundary in the order specified, either clockwise or counterclockwise. As we process each edge, we can check to determine whether that edge and the next nonhorizontal edge have either monotonically increasing or decreasing endpoint y values. If so, the lower edge can be shortened to ensure that only one intersection point is generated for the scan line going through the common vertex joining the two edges.



The above figure illustrates shortening of an edge. When the endpoint y coordinates of the two edges are increasing, the y value of the upper endpoint for the current edge is decreased by 1, as in fig (a). When the endpoint y values are monotonically decreasing, as in fig (b), we decrease the y coordinate of the upper endpoint of the edge following the current edge.

Coherence Properties:

Typically, certain properties of one part of a scene are related in some way to the properties in other parts of the scene, and these coherence properties can be used in computer-graphics algorithms to reduce processing. Coherence methods often involve incremental calculations applied along a single scan line or between successive scan lines. For example, in determining fill-area edge intersections, we can set up incremental coordinate calculations along any edge by exploiting the fact that the slope of the edge is constant from one scan line to the next. Following figure shows two successive scan lines crossing the left edge of a triangle.



The slope of this edge can be expressed in terms of the scan-line intersection coordinates:

$$m = \frac{y_{k+1} - y_k}{x_{k+1} - x_k}$$

Since the change in y coordinates between the two scan lines is simply,

$$y_{k+1} - y_k = 1$$

the x-intersection value x_{k+1} on the upper scan line can be determined from the x-intersection value x_k on the preceding scan line as

$$x_{k+1} = x_k + \frac{1}{m}$$

Each successive x intercept can thus be calculated by adding the inverse of the slope and rounding to the nearest integer.

An obvious parallel implementation of the fill algorithm is to assign each scan line that crosses the polygon to a separate processor. Edge intersection calculations are then performed independently. Along an edge with slope m, the intersection x_k value for scan line k above the initial scan line can be calculated as:

$$x_k = x_0 + \frac{k}{m}$$

In a sequential fill algorithm, the increment of x values by the amount $\frac{1}{m}$ along an edge can be accomplished with integer operations by recalling that the slope m is the ratio of two integers:

$$m = \frac{\Delta y}{\Delta x}$$

Where Δx and Δy are the difference between the edge endpoint x and y coordinate values. Thus, incremental calculations of x intercepts along an edge for successive scan lines can be expressed as

$$x_{k+1} = x_k + \frac{\Delta y}{\Delta x}$$

Using this equation, we can perform integer evaluation of the x intercepts by initializing a counter to 0, then incrementing the counter by the value of Δx each time we move up to a new scan line. Whenever the counter value becomes equal to or greater than Δy, we increment the current x intersection value by 1 and decrease the counter by the value Δy. This procedure is equivalent to maintaining integer and fractional parts for x intercepts and incrementing the fractional part until we reach the next integer value.

1.4.1 Boundary fill:

- Start at a point inside a region and paint the interior outward toward the boundary. If boundary is specified in a single color the fill algorithm proceeds outward pixel by pixel until the boundary color is encountered.
- Accept as input coordinates of an interior point (x,y) a fill color and boundary color.
- Starting from (x,y) the procedure tests neighboring position to determine whether they are of the boundary color.
- If not they are painted with fill color and their neighbors are tested.
- Process continues until all pixels up to boundary color for the area have been tested.

C function for boundary fill algorithm:

```
boundaryFill (int x, y, fill_color, boundary_color)
{
int color;
getpixel(x,y,color);
if(color!= boundary_color && color !=fill_color)
{
Setpixel(x,y,fill_color);
boundaryFill(x+1, y, fill_color, boundary_color);
boundaryFill(x, y+1, fill_color);
boundaryFill(x-1, y, fill_color, boundary_color);
boundaryFill(x, y-1, fill_color, boundary_color)
}
}
```

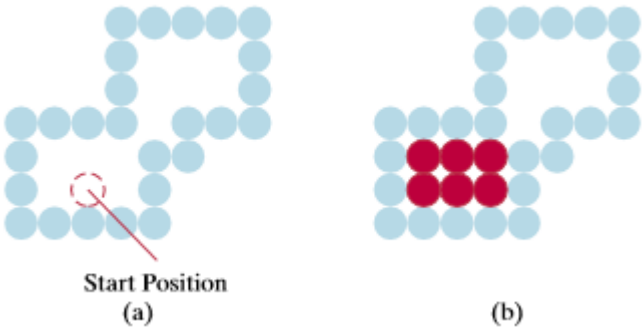


Fig: The area defined within the color boundary (a) is only partially filled in (b) using a 4-connected boundary-fill algorithm.

1.4.2 Flood fill algorithm:

Sometimes we want to fill in (or recolour) an area that is not defined within a single color boundary. Following figure shows an area bordered by several different color regions.

Fig: An area defined within multiple color boundaries

We can paint such areas by replacing a specified interior color instead of searching for a particular boundary color. This fill procedure is called a flood-fill algorithm. We start from a specified interior point (x,y) and reassign all pixel values that are currently set to a given interior color with the desired fill color. If the area we want to paint has more than one interior color, we can first reassign pixel values so that all interior points have the same color. Using either a 4-connected or 8-connected approach, we then step through pixel positions until all interior points have been repainted. The following procedure flood fills a 4-connected region recursively starting from the input position.

```
void floodFill4(int x, int y, int fillcolor, int interiorcolor)
{
Int color;
Getpixel(x,y,color);
If(color= interiorcolor)
{
Setpixel(x,y);
floodFill4(x+1, y, fillcolor, interiorcolor);
floodFill4(x-1, y, fillcolor, interiorcolor);
floodFill4(x, y+1, fillcolor, interiorcolor);
floodFill4(x, y-1, fillcolor, interiorcolor);
}
}
```

Exercises

1. What is computer graphics? Explain in detail about the application of computer graphics.
2. What is a digital differential analyser (DDA)? How can you draw the line using this algorithm?
3. What is a random scan display system? Draw its block diagram and explain it in detail.
4. How can you draw circle? Explain with algorithm.
5. What is raster scan display system? Draw its block diagram and explain it in detail.
6. Explain the random scan display system with its advantages and disadvantages.
7. Differentiate between incremental algorithm over DDA with example.
8. Define the term: Video controller, Raster graphics.

9. What is a random scan system? Explain the operation of random scan with architecture.
10. Write a procedure to fill the interior of a given ellipse with a specified pattern.
11. Define computer graphics and its applications.
12. Explain the scan line algorithm with example.