MODULE-1

1. APPLICATIONS OF COMPUTER GRAPHICS:

1.1 Graphs and Charts

- An early application for computer graphics is the display of simple data graphs usually plotted on a character printer. Data plotting is still one of the most common graphics application.
- Graphs & charts are commonly used to summarize functional, statistical, mathematical, engineering and economic data for research reports, managerial summaries and other types of publications.
- Typically examples of data plots are line graphs, bar charts, pie charts, surface graphs, contour plots and other displays showing relationships between multiple parameters in two dimensions, three dimensions, or higher-dimensional spaces.
- Three dimensional graphs and charts are used to display additional parameter information, sometimes they are used for effect, providing more dramatic or more attractive presentations of the data relationships.

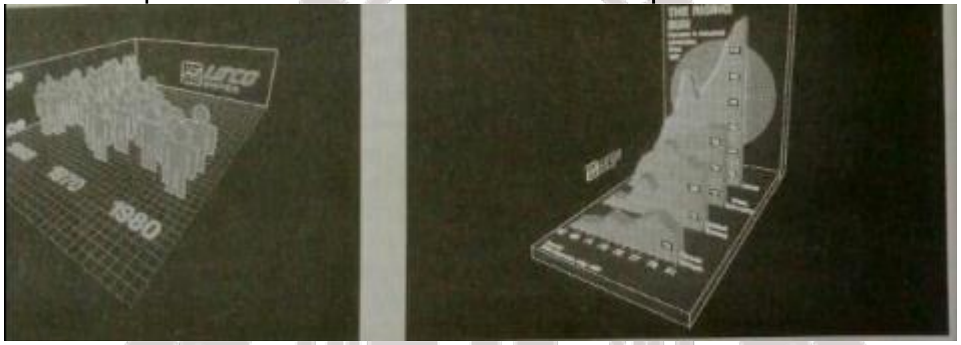


Fig: Two Three dimensional graphs designed for dramatic effect.

1.2 Computer-Aided Design

- A major use of computer graphics is in design processes-particularly for engineering and architectural systems.
- CAD, computer-aided design or CADD, computer-aided drafting and design methods are now routinely used in the automobiles, aircraft, spacecraft, computers, home appliances.
- Circuits and networks for communications, water supply or other utilities are constructed with repeated placement of a few geographical shapes.
- Animations are often used in CAD applications. Real-time, computer animations using wire-frame shapes are useful for quickly testing the performance of a vehicle or system.
- When object designs are complete, or nearly complete, realistic lighting conditions and surface rendering are applied to produce displays that will show the appearance of the final product.
- A circuit board layout, for example, can be transformed into a description of the individual processes needed to construct the electronics network.

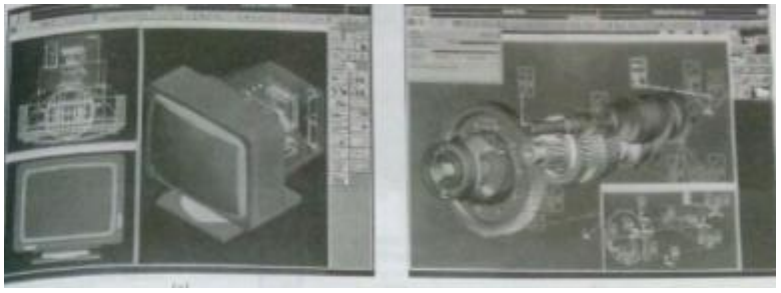


Fig: Multiple-window, color-coded CAD workstation displays.

1.3 Virtual-Reality Environments

- A more recent application of computer graphics is in the creation of virtual-reality environments in which a user can interact with the objects in a three dimensinal scene.
- Animations in virtual-reality environments are often used to train heavy-equipment operators or to analyze the effectiveness of various cabin configurations and control placements.
- With virtual-reality systems, designers and others can move about and interact with objects in various ways. Architectural designs can be examined by taking simulated "walk" through the rooms or around the outsides of buildings to better appreciate the overall effect of a particular design.
- With a special glove, we can even "grasp" objects in a scene and turn them over or move them from one place to another.

Fig: View of the tractor displayed on a standard monitor.

1.4 Data Visualizations:

- Producing graphical representations for scientific, engineering and medical data sets and processes is another fairly new application of computer graphics, which is generally referred to as **scientific visualization.** And the term **business visualization** is used in connection with data sets related to commerce, industry and other nonscientific areas.
- There are many different kinds of data sets and effective visualization schemes depend on the characteristics of the data. A collection of data can contain scalar values, vectors or higher-order tensors.

- Visual techniques are also used to aid in the understanding and analysis of complex processes and mathematical functions.
- A color plot of mathematical curve functions in fig a) and a surface plot of a function is shown in fig b).

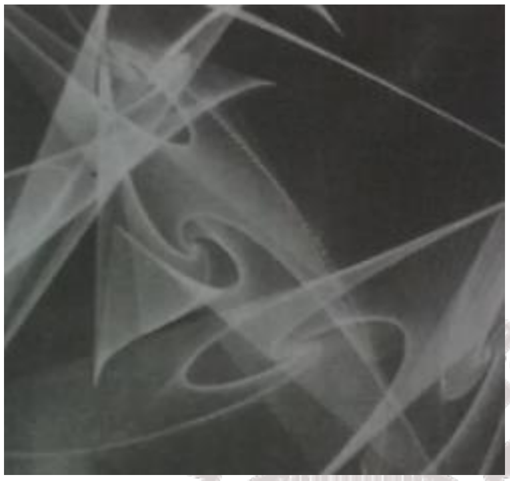


Fig a: Mathematical curve functions plotted in various color combinations.

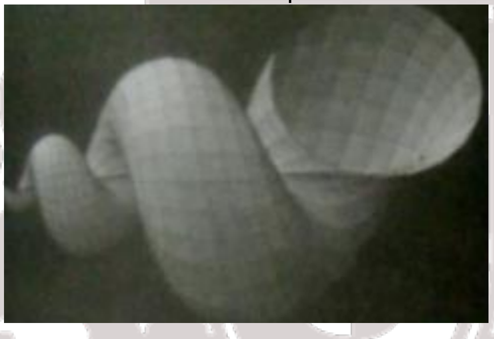


Fig b: Lighting effects & surface-rendering techniques were applied to produce this surface representaion for a 3D function.

1.5 Education and Training

- Computer generated models of physical,financial,political,social,economic & other systems are often used as educational aids.
- Models of physical processes physiological functions,equipment, such as the color coded diagram as shown in the figure, can help trainees to understand the operation of a system.
- For some training applications,special hardware systems are designed.Examples of such specialized systems are the simulators for practice sessions ,aircraft pilots,air traffic- control personnel.
- Some simulators have no video screens,for eg: flight simulator with only a control panel for instrument flying.
- But most simulators provide screens for visual displays of the external environment.

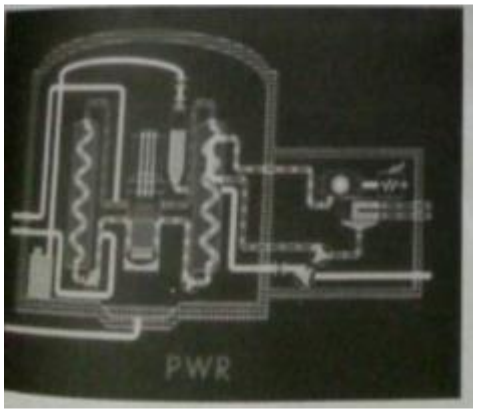


Fig: Color-coded diagram used to explain the operation of a nuclear reactor.

1.6 Computer Art

Fig: Cartoon drawing produced with a paint brush program, symbolically illustrating an artist at work on video monitor.

- Figure gives a figurative representation of the use of a **paintbrush program** that allows an artist to "paint" pictures on the screen of a video monitor.
  The picture is usually painted electronically on a graphics tablet using a stylus, which can simulate different brush strokes, brush widths and colors.
- Fine artists use a variety of other computer technologies to produce images. To create pictures the artist uses a combination of 3D modeling packages, texture mapping, drawing programs and CAD software etc.
- Commercial art also uses theses "painting" techniques for generating logos & other designs, page layouts combining text & graphics, TV advertising spots & other applications.
- A common graphics method employed in many television commercials is morphing, where one object is transformed into another.

1.7 Entertainment

- Television production, motion pictures, and music videos routinely a computer graphics methods.
- Sometimes graphics images are combined a live actors and scenes and sometimes the films are completely generated a computer rendering and animation techniques.
- Some television programs also use animation techniques to combine computer generated figures of people, animals, or cartoon characters with the actor in a scene or to transform an actor’s face into another shape.
- Advanced computer-modeling & surface-rendering methods were employed in 2 award-winning short films to produce the scenes shown in the figure.

Fig: Computer-generated film scenes. a) Red’s Dream b)Knickknack

1.8 Image Processing:

- The modification or interpretation of existing pictures, such as photographs and TV scans is called **image processing.**
- Methods used in computer graphics and image processing overlap, the two areas are concerned with fundamentally different operations.
- In computer graphics, a computer is used to create a pictures.
- Image processing methods are used to improve picture quality, analyze images, or recognize visual patterns for robotics applications.
- Image processing methods are often used in computer graphics, and computer graphics methods are frequently applied in image processing.
- The digital methods can be used to rearrange picture parts, to enhance color separation.
- Medical applications also make extensive use of image processing techniques for picture enhancements in tomography and in simulations and surgical operations.
- It is also used in computed X-ray tomography(CT), position emission tomography(PET),and computed axial tomography(CAT).

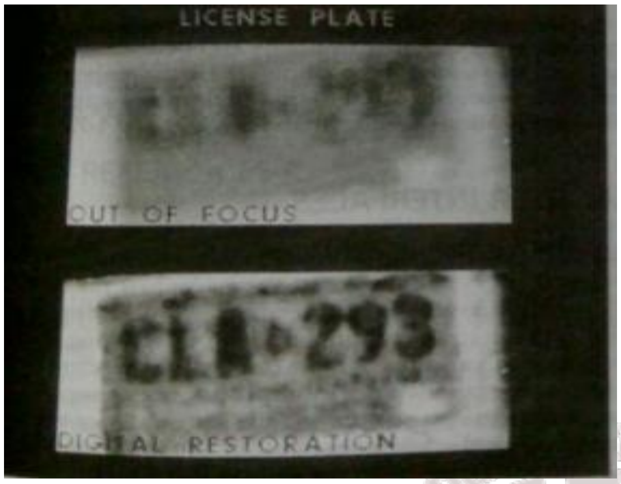


Fig: A blurred photograph of a license plate becomes legible after the application of image processing techniques.

1.9 Graphical User Interfaces

- It is common now for applications software to provide **graphical user interface** (GUI) .
- A major component of graphical interface is a window manager that allows a user to display multiple, rectangular screen areas called *display windows.*
- Each screen display area can contain a different process, showing graphical or non-graphical information, and various methods can be used to activate a display window.
- Using an interactive pointing device, such as mouse, we can active a display window on some systems by positioning the screen cursor within the window display area and pressing the left mouse button.

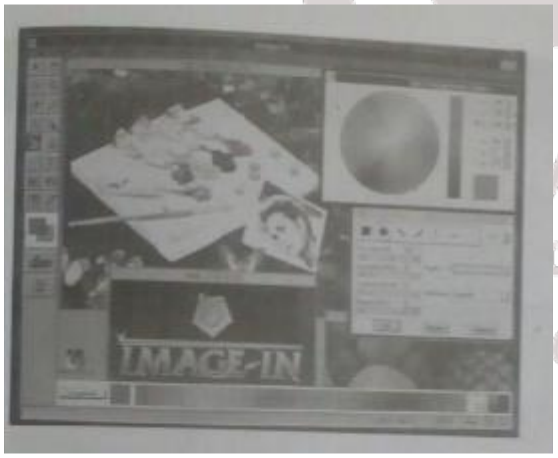
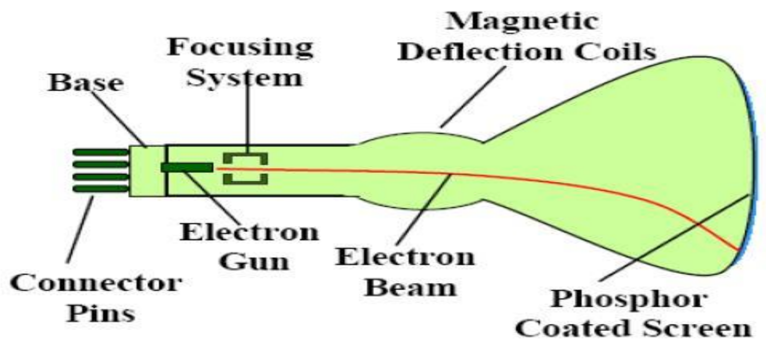


Fig: A graphical user interface, showing multiple display windows, menus, & icons.

2. **VIDEO DISPLAY DEVICES**:

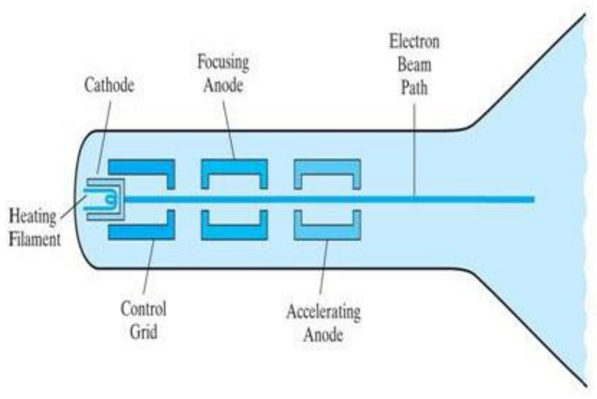
- The primary output device in a graphics system is a video monitor.
- The operation of most video monitors is based on the standard **cathode-ray tube (CRT)**.

2.1 Refresh Cathode-Ray Tubes:-



Basic design of a magnetic deflection CRT

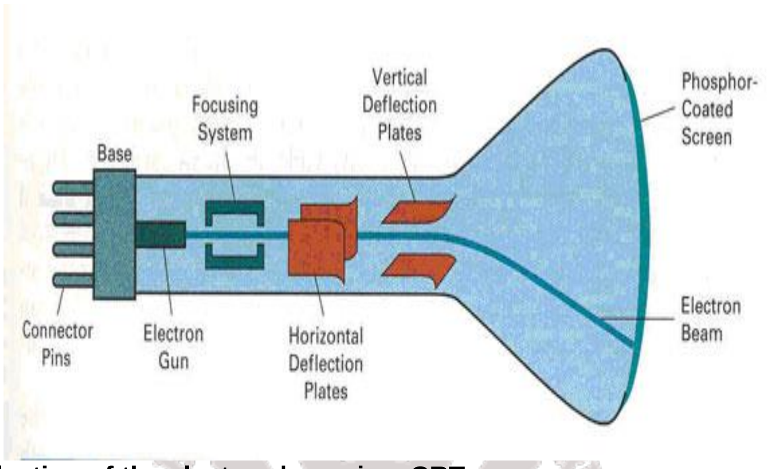
- A beam of electrons, emitted by an electron gun, passes through focusing and deflection systems that direct the beam toward specified positions on the phosphor-coated screen.
- The phosphor then emits a small spot of light at each position contacted by the electron beam and the light emitted by the phosphor fades very rapidly.
- One way to maintain the screen picture is to store the picture information as a charge distribution within the CRT in order to keep the phosphors activated.
- The most common method now employed for maintaining phosphor glow is to redraw the picture repeatedly by quickly directing the electron beam back over the same screen points. This type of display is called a **refresh CRT.**
- The frequency at which a picture is redrawn on the screen is referred to as the **refresh rate.**



Operation of an electron gun with an accelerating anode

- The primary components of an electron gun in a CRT are the heated metal cathode and a control grid.
- The heat is supplied to the cathode by directing a current through a coil of wire, called the filament, inside the cylindrical cathode structure.
- This causes electrons to be “boiled off” the hot cathode surface.
- Inside the CRT envelope, the free, negatively charged electrons are then accelerated toward the phosphor coating by a high positive voltage.
- Intensity of the electron beam is controlled by the voltage at the control grid.
- Since the amount of light emitted by the phosphor coating depends on the number of electrons striking the screen, the brightness of a display point is controlled by varying the voltage on the control grid.
- The focusing system in a CRT forces the electron beam to converge to a small cross section as it strikes the phosphor and it is accomplished with either electric or magnetic fields.
- With electrostatic focusing, the electron beam is passed through a positively charged metal cylinder so that electrons along the center line of the cylinder are in equilibrium position.
- Deflection of the electron beam can be controlled with either electric or magnetic fields.
- Cathode-ray tubes are commonly constructed with two pairs of magnetic-deflection coils.
- One pair is mounted on the top and bottom of the CRT neck, and the other pair is mounted on opposite sides of the neck.
- The magnetic field produced by each pair of coils results in a traverse deflection force that is perpendicular to both the direction of the magnetic field and the direction of travel of the electron beam.

- Horizontal and vertical deflections are accomplished with these pair of coils.

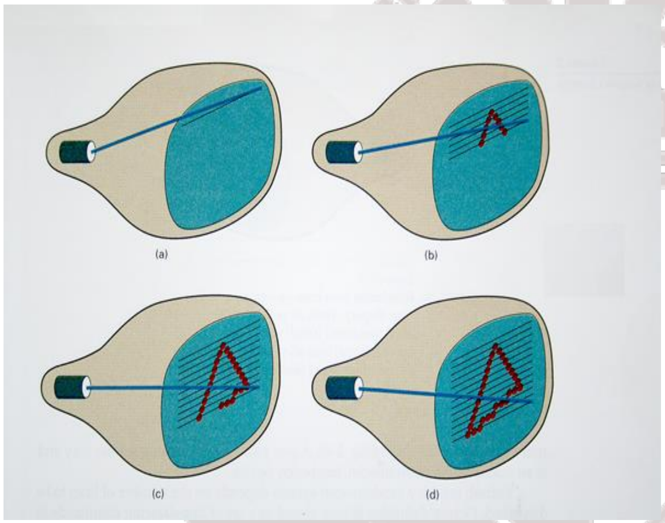


Electrostatic deflection of the electron beam in a CRT

- When electrostatic deflection is used, two pairs of parallel plates are mounted inside the CRT envelope where, one pair of plates is mounted horizontally to control vertical deflection, and the other pair is mounted vertically to control horizontal deflection.
- Spots of light are produced on the screen by the transfer of the CRT beam energy to the phosphor.
- When the electrons in the beam collide with the phosphor coating, they are stopped and their kinetic energy is absorbed by the phosphor.
- Part of the beam energy is converted by the friction in to the heat energy, and the remainder causes electros in the phosphor atoms to move up to higher quantum-energy levels.
- After a short time, the "excited" phosphor electrons begin dropping back to their stable ground state, giving up their extra energy as small quantum of light energy called photons.
- What we see on the screen is the combined effect of all the electrons light emissions: a glowing spot that quickly fades after all the excited phosphor electrons have returned to their ground energy level.
- The frequency of the light emitted by the phosphor is proportional to the energy difference between the excited quantum state and the ground state.
- Lower persistence phosphors required higher refresh rates to maintain a picture on the screen without flicker.
- The maximum number of points that can be displayed without overlap on a CRT is referred to as a **resolution.**
- Resolution of a CRT is dependent on the type of phosphor, the intensity to be displayed, and the focusing and deflection systems.
- High-resolution systems are often referred to as high-definition systems.

2.2 Raster-Scan Displays:-

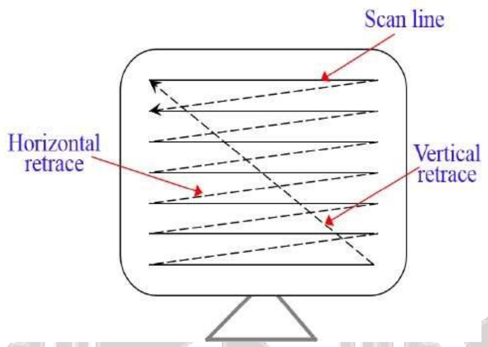
- The most common type of graphics monitor employing CRT is the **raster-scan display**, based on television technology.
- In a raster-scan system, the electron beam is swept across the screen, one row at a time, from top to bottom, where each row is referred to as a **scan line.**
- As the electron beam moves across a scan line, the beam intensity is turned on and off to create a pattern of illuminated spots.
- Picture definition is stored in a memory area called the **refresh buffer** or **frame buffer**, where **frame** refers to the total screen area.
- The memory area holds the set of color values for the screen points. These stored color values are then retrieved from the refresh buffer and used to control the intensity of the electron beam as it moves from spot to spot across the screen.
- The refresh buffer is used to store the set of screen color values, it is also called as **color buffer.**



A raster-scan system displays an object as a set of discrete points across each scan line

- Each screen spot that can be illuminated by the electron beam is referred to as a **pixel** or **pel** (**picture element**).
- Raster systems are commonly characterized by their resolution, which is the number of pixel positions that can be plotted.
- Another property of video monitors is **aspect ratio,** which is often defined as the number of pixel columns divided by the number of scan lines that can be displayed by the system.
- Aspect ratio can also be described as the number of horizontal points to vertical points (or vice versa).
- The number of bits per pixel in a frame buffer is referred to as either the **depth** of the buffer area or the number of **bit planes.**
- A frame buffer with one bit per pixel is called as **bitmap,** and a frame buffer with multiple bits per pixel is called as **pixmap.**
- The terms bitmap and pixmap are also used to describe other rectangular arrays, where a bitmap is any pattern of binary values and a pixmap is a multicolor pattern.

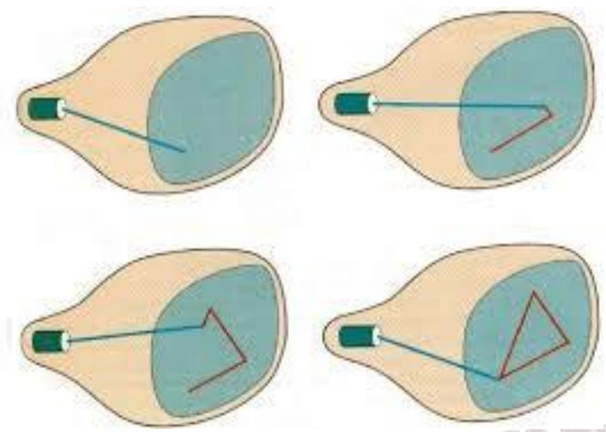
- As each screen refresh takes place, we tend to see each frame as a smooth continuation of the patterns in the previous frame, as long as the refresh rate is not too low.
- Refresh rates are described in units of cycles per second, or Hertz (Hz), where a cycle corresponds to one frame.
- At the end of each scan line, the electron beam returns to the left side of the screen to begin displaying the next scan line.
- The return to the left of the screen, after refreshing each scan line, is called the **horizontal retrace** of the electron beam.
- At the end of each frame, the electron beam returns to the top left corner of the screen (**vertical retrace**) to begin the next frame.



- On some raster-scan systems and TV sets, each frame is displayed in two passes using an interlaced refresh procedure.
- In the first pass, the beam sweeps across every other scan line from top to bottom.
- After the vertical retrace, the beam then sweeps out the remaining scan lines.
- Interlacing of the scan lines in this way allows us to see the entire screen displayed in one-half the time it would have taken to sweep across all the lines at once from top to bottom.
- This technique is primarily used with slow refresh rates.
- This is an effective technique for avoiding flicker-provided that adjacent scan lines contains similar display information.

2.3 Random-Scan Displays:-

- In a **random-scan display** unit, a CRT has the electron beam directed only to those parts of the screen where a picture is to be displayed.
- Pictures are generated as line drawings, with the electron beam tracing out the component lines one after the other. Hence random-scan monitors are also referred to as **vector displays** or **stroke-writing displays** or **calligraphic displays.**

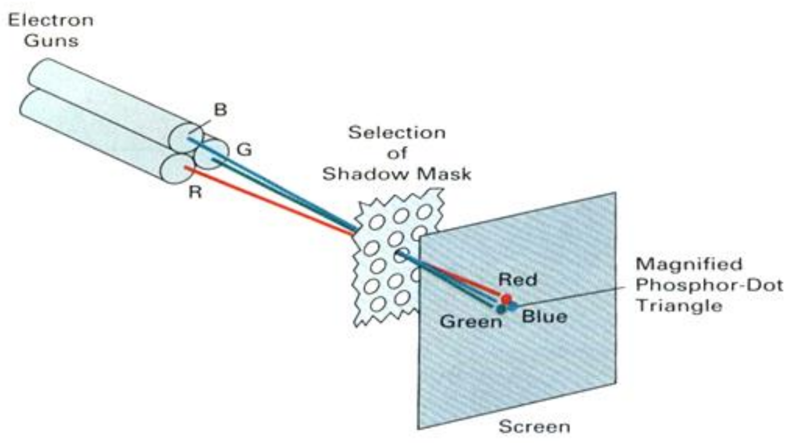


A random-scan system draws the component lines of an object in any specified order

- The component lines of a picture can be drawn and refreshed by a random-scan system in any specified order.
- Refresh rate on a random-scan system depends on the number of lines to be displayed on that system.
- Picture definition is now stored as a set of line-drawing commands in an area of memory referred to as the **display list, refresh display file, vector file,** or **display program.**
- To display a specified picture, the system cycles through the set of commands in the display file, drawing each component line in turn.
- After all line-drawing commands have been processed, the system cycles back to the first line command in the first list.
- When a small set of lines is to be displayed, each refresh cycle is delayed to avoid very high refresh rates, which could burn out the phosphor.
- Since picture definition is stored as a set of line-drawing instructions rather than as a set of intensity values for all screen points, vector displays generally have higher resolutions than raster systems.

2.4 Color CRT Monitors:

- A CRT monitor displays color pictures by using a combination of phosphors that emit different-colored light.
- **Beam-penetration** method uses only two phosphor layers: red and green.
- A beam of slow electrons excites only the outer red layer, but a beam of very fast electrons penetrates through the red layer and excites the inner green layer.
- At intermediate beam speeds, combinations of red and green light are emitted to show two additional colors, orange and yellow.
- **Shadow-mask** methods are commonly used in raster-scan systems since they produce a much wider range of colors than the beam penetration method.



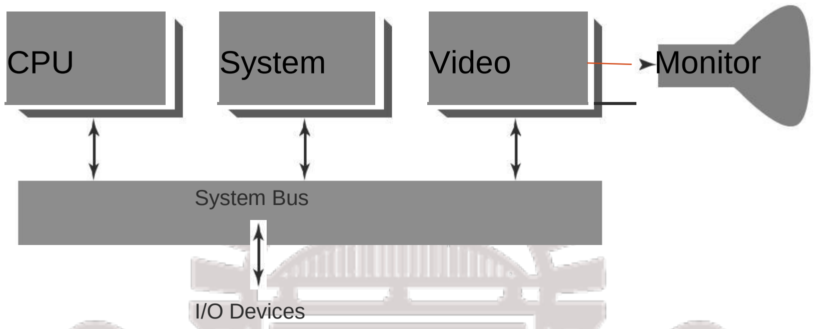
Operation of a delta-delta, shadow-mask CRT

- This approach is based on the way that we seem to perceive colors as combinations of red, green, and blue components, called the **RGB color model.**
- Thus, a shadow-mask CRT uses three phosphor color dots at each pixel position. One phosphor dot emits a red light, another emits a green light, and the third emits a blue light.
- This type of CRT has three electron guns, one for each color dot, and a shadow-mask grid just behind the phosphor-coated screen.
- The light emitted from the three phosphors results in a small spot of color at each pixel position, since our eyes tend to merge the light emitted from the three dots into one composite color.
- In the figure above, three electron beams are deflected and focused as a group onto the shadow mask, which contains a series of hoes aligned with the phosphor-dot patterns.
- When the three beams pass through a hole in the shadow mask, they activate a dot triangle, which appears as a small color spot on the screen.
- The phosphor dots in the triangles are arranged so that each electron beam can activate only its corresponding color dot when it passes through the shadow mask.
- Another configuration for the three electron guns is an in-line arrangement in which the three electron guns, and the corresponding red-green-blue color dots on the screen, are aligned along one scan line instead of in a triangular pattern.
- We obtain color variations in a shadow-mask CRT by varying the intensity levels of the three electron beams (red, green, or blue).
- When all three dots are activated with equal beam intensities, white color is obtained.
- Yellow is produced with equal intensities from the green and red dots.
- Magenta is produced with equal blue and red intensities and cyan when blue and green are activated equally and more sophisticated systems can allow intermediate intensity levels, so that several million colors are possible.
- Color graphics systems can be used with several types of CRT display devices.
- Color CRTs in graphics systems are designed as **RGB monitors**. These monitors use shadow-mask methods and take the intensity level for each electron gun directly from the computer system without any intermediate processing.

- An RGB color system with 24 bits of storage per pixel is generally referred to as a **full-color system** or a **true-color system.**

3.RASTER-SCAN SYSTEMS:

Interactive raster-graphics system typically employ several processing units. In addition to the central processing unit, or CPU, a special-purpose processor, called the video controller or display controller, is used to control the operation of the display device. Organization of a simple raster system is shown as below.

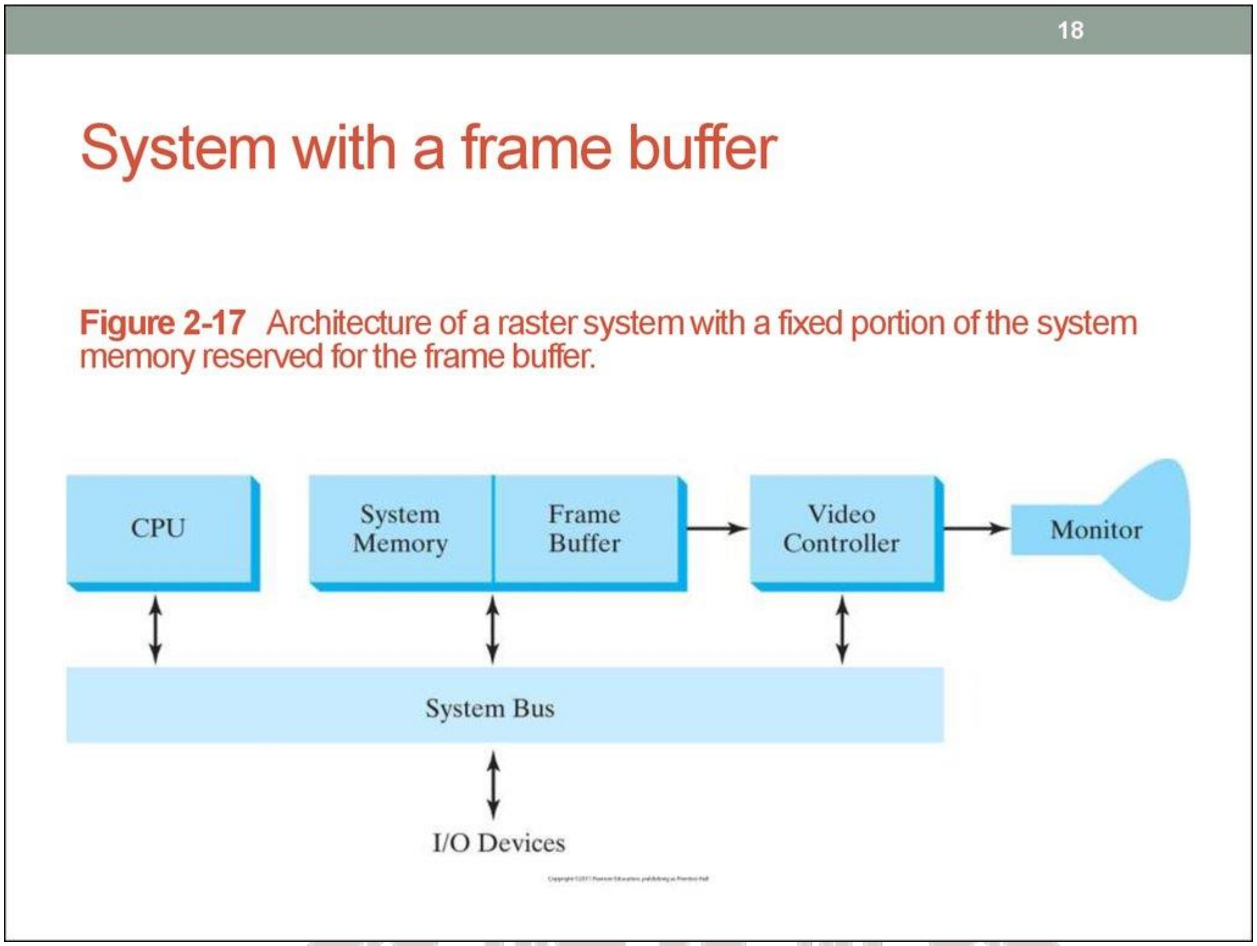


F I G U R E : Architecture of a simple raster-graphics system.

Here the frame buffer can be anywhere in the system memory, and the video controller accesses the frame buffer to refresh the screen. In addition to the video controller, more sophisticated raster systems employ other processors as coprocessors and accelerators to implement various graphic operations.

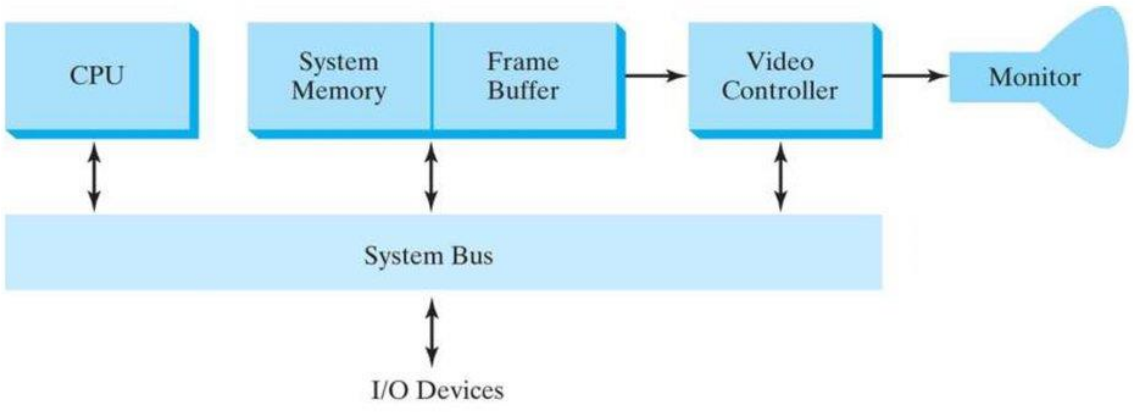
3.1 Video Controller

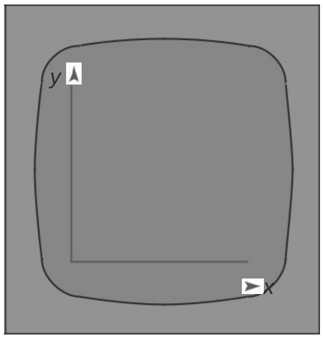
The figure below shows a commonly used organization for raster systems. A fixed area of the system memory is reserved for the frame buffer, and the video controller is given direct access to the frame-buffer memory. Frame-buffer locations, and the corresponding screen positions, are referenced in the Cartesian coordinates.



System with a frame buffer

Figure 2-17 Architecture of a raster system with a fixed portion of the system memory reserved for the frame buffer.

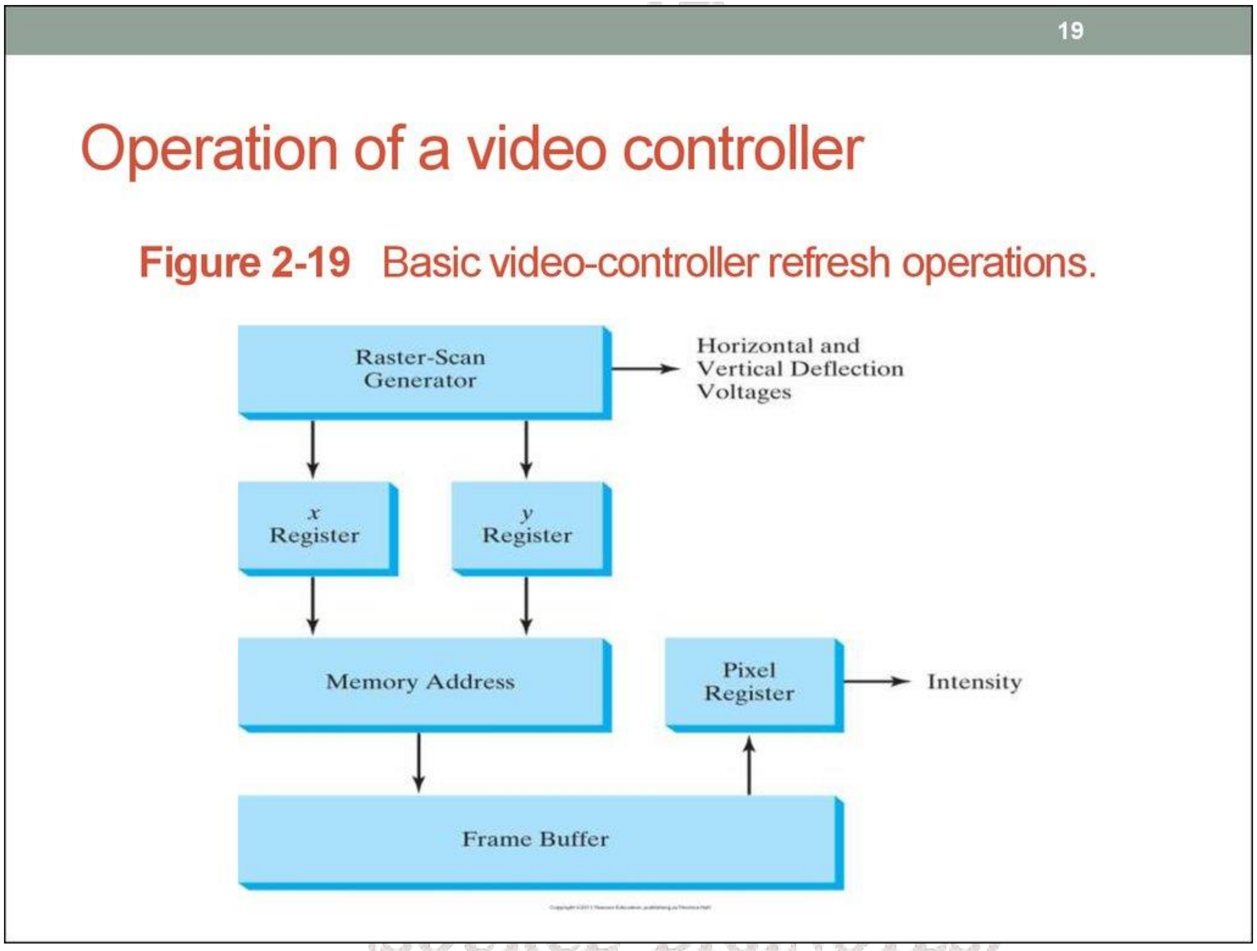




F I G U R E 1 8

A Cartesian reference frame with origin at the lower-left corner of a video monitor.

The figure above shows a two-dimensional Cartesian reference frame with the origin at the lower-left screen corner. The screen surface is then represented as the first quadrant of a two-dimensional system, with positive X values increasing from left to right and positive Y values increasing from the bottom screen to top


19

Operation of a video controller

Figure 2-19 Basic video-controller refresh operations.

In the figure above, the basic refresh operations of the video controller are diagrammed. Two registers are used to store the coordinate values for the screen pixels. Initially, the X register is set to 0 and the Y register is set to the value for the top scan line. The contents of the frame buffer at this pixel position are then retrieved and used to set the intensity of the CRT beam.

A video controller can be designed to perform a number of other operations. For various applications, the video controller can retrieve pixel values from different memory areas on different refresh cycles.

3.2 Raster-Scan Display Processor:

The following figure shows one way to organize the components of a raster system that contains a separate display processor, sometimes referred to as a graphics controller or a display coprocessor. The purpose of the display processor is to free the CPU from the graphics chores. In addition to the system memory, a separate display-processor memory area can be provided.

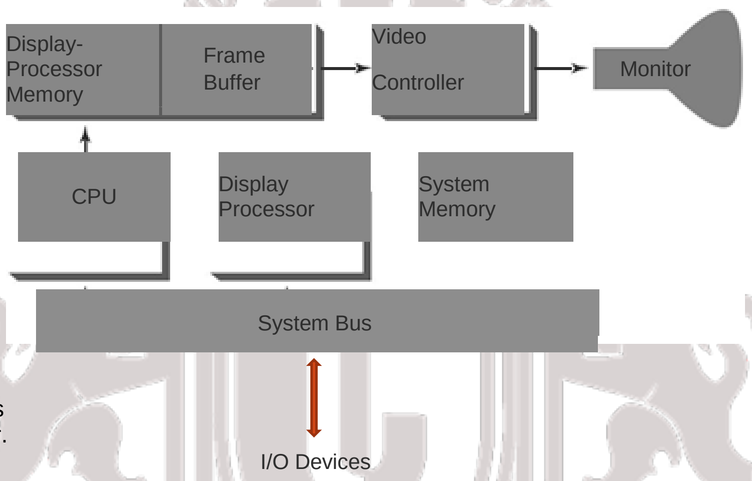


FIGURE 20
Architecture of a raster-graphics system with a display processor.

A major task of the display processor is digitizing a picture definition given in an application program into a set of pixel values for storage in the frame buffer. This digitization process is called Scan Conversion. Graphics commands specifying straight lines and other geometric objects are scan converted into a set of discrete points, corresponding to screen pixel positions.

Display processors are also designed to perform a number of additional operations. These Function include generating various line styles (dashed, dotted or solid), displaying color areas, and applying transformations to the objects in a scene. Also, display processors are typically designed to interface with interactive input devices, such as a mouse.

In an effort to reduce memory requirements in raster systems, methods have been devised for organizing the frame buffer as a linked list and encoding the color information. One organization scheme is to store each scan line as a set of numbered pairs.

Run-length encoding is a technique in which the first number in each pair can be a reference to a color value, and the second number specifies the number of adjacent pixels on the scan line that are to be displayed in that color. This technique result in a considerable saving in storage space if a picture is to be constructed mostly with long runs of a single color each.

Cell encoding is the approach of encoding the raster as a set of rectangular areas

Disadvantage of encoding runs:

- Color changes are difficult to record
- Storage requirements increases as the lengths of run decreases
- Difficult for the display controller to process raster when short runs are involved

Advantages of encoding methods:

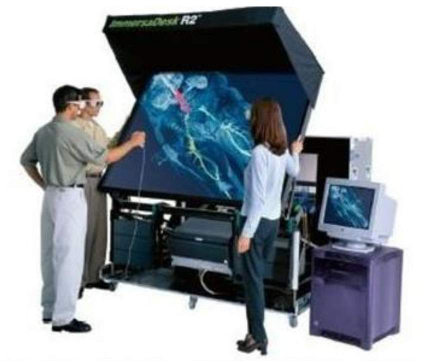
Useful in digital storage and transmission of picture information

4. Graphics workstations and viewing systems

- Most graphics monitors today operate as raster-scan displays, and both CRT and flat panel systems are in common use.
- Graphics workstation range from small general-purpose computer systems to multi monitor facilities, often with ultra –large viewing screens.
- High-definition graphics systems, with resolutions up to 2560 by 2048, are commonly used in medical imaging, air-traffic control, simulation, and CAD.

Fig 1.1-A high-resolution (2048 by 2048) graphics monitor.

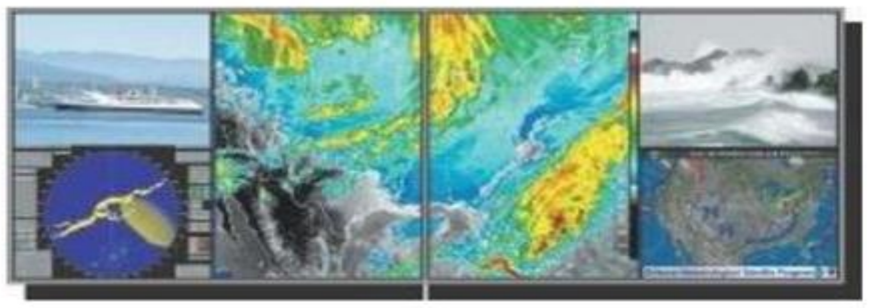
- Many high-end graphics workstations also include large viewing screens, often with specialized features.



Large-screen stereoscopic view

Fig 1.2- A large screen stereoscopic view of pressure contours in a vascular blood-flow simulation.

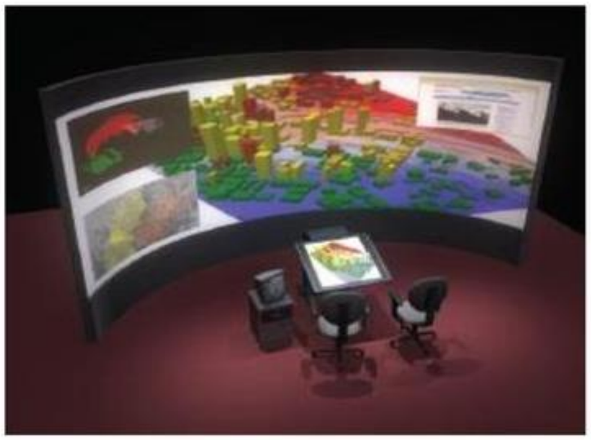
- Multi-panel display screens are used in a variety of applications that require "wall-sized" viewing areas. These systems are designed for presenting graphics displays at meetings, conferences, conventions, trade shows, retail stores etc.
- A multi-panel display can be used to show a large view of a single scene or several individual images. Each panel in the system displays one section of the overall picture.



Multi-panel display

Fig 1.3- A multi-panel display system called the "Super Wall".

- Large graphics displays can also be presented on curved viewing screens.



Curved viewing screen

Fig 1.4- A homeland security study displayed using a system with large curved viewing screen.

- ➢ A large, curved-screen system can be useful for viewing by a group of people studying a particular graphics application.

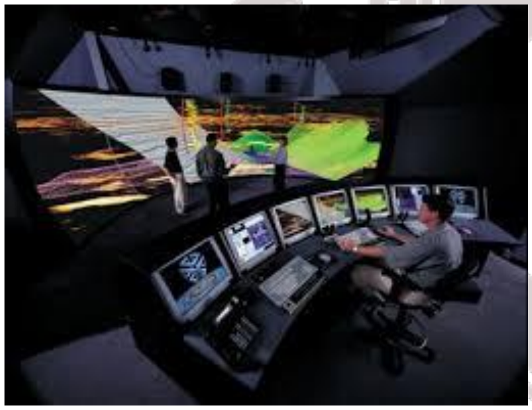


Fig 1.5-A geophysical visualization presented on a 25 foot semicircular screen, which provides a 160 degree horizontal and 40 degree vertical field of view.

- ➢ A control center, featuring a battery of standard monitors, allows an operator to view sections of the large display and to control the audio, video, lighting, and projection systems using a touch-screen menu.
- ➢ A 360 degree paneled viewing system in the NASA control-tower simulator, which is used for training and for testing ways to solve air-traffic and runway problems at airports.

Fig 1.6-The 360 degree viewing screen in the NASA airport control tower simulator called the Future Flight Central Facility.

5. Input Devices

Graphics workstations make use of various devices for data input.Most systems have keyboards and mouses,while some other systems have trackball,spaceball,joystick,button boxes,touch panels,image scanners and voice systems.

Keyboard:

- Keyboard on graphics system is used for entering text strings,issuing certain commands and selecting menu options.
- Keyboards can also be provided with features for entry of screen coordinates,menu selections or graphics functions.
- General purpose keyboard uses function keys and cursor-control keys.
- Function keys allow user to select frequently accessed operations with a single keystroke.Cursor-control keys are used for selecting a displayed object or a location by positioning the screen cursor.

Button Boxes and Dials:

- Buttons are often used to input predefined functions .Dials are common devices for entering scalar values.
- Numerical values within some defined range are selected for input with dial rotations.

Mouse Devices:

- Mouse is a hand-held device,usually moved around on a flat surface to position the screen cursor.wheeler or roolers on the bottom of the mouse used to record the amount and direction of movement.
- Some of the mouses uses optical sensors,which detects movement across the horizontal and vertical grid lines.
- Since a mouse can be picked up and put down,it is used for making relative changes in the position of the screen.
- Most general purpose graphics systems now include a mouse and a keyboard as the primary input devices.

Trackballs and Spaceballs:

- A trackball is a ball device that can be rotated with the fingers or palm of the hand to produce screen cursor movement.
- Laptop keyboards are equipped with a trackball to eliminate the extra space required by a mouse.
- Spaceball is an extension of two-dimensional trackball concept.
- Spaceballs are used for three-dimensional positioning and selection operations in virtual-reality systems,modeling,animation,CAD and other applications.

Joysticks:

- Joystick is used as a positioning device,which uses a small vertical lever(stick) mounded on a base.It is used to steer the screen cursor around and select screen position with the stick movement.

- A push or pull on the stick is measured with strain gauges and converted to movement of the screen cursor in the direction of the applied pressure.

Data Gloves:

- Data glove can be used to grasp a virtual object.The glove is constructed with a series of sensors that detect hand and finger motions.

- Input from the glove is used to position or manipulate objects in a virtual scene.

Digitizers:

- Digitizer is a common device for drawing,painting or selecting positions.

- Graphics tablet is one type of digitizer,which is used to input 2-dimensional coordinates by activating a hand cursor or stylus at selected positions on a flat surface.

- A hand cursor contains cross hairs for sighting positions and stylus is a pencil-shaped device that is pointed at positions on the tablet.

Image Scanners:

- Drawings,graphs,photographs or text can be stored for computer processing with an image scanner by passing an optical scanning mechanism over the information to be stored.

- Once we have the representation of the picture, then we can apply various image-processing method to modify the representation of the picture and various editing operations can be performed on the stored documents.

Touch Panels:

- Touch panels allow displayed objects or screen positions to be selected with the touch of a finger.
- Touch panel is used for the selection of processing options that are represented as a menu of graphical icons.
- Optical touch panel-uses LEDs along one vertical and horizontal edge of the frame.
- Acoustical touch panels generates high-frequency sound waves in horizontal and vertical directions across a glass plate.

Light Pens:

- Light pens are pencil-shaped devices used to select positions by detecting the light coming from points on the CRT screen.
- To select positions in any screen area with a light pen,we must have some nonzero light intensity emitted from each pixel within that area.
- Light pens sometimes give false readings due to background lighting in a room.

Voice Systems:

- Speech recognizers are used with some graphics workstations as input devices for voice commands.The voice system input can be used to initiate operations or to enter data.
- A dictionary is set up by speaking command words several times,then the system analyses each word and matches with the voice command to match the pattern.

7. GRAPHICS NETWORKS:

So far, we have mainly considered graphics applications on an isolated system with a single user.

Multiuser environments & computer networks are now common elements in many graphics applications.

Various resources, such as processors, printers, plotters and data files can be distributed on a network & shared by multiple users.

A graphics monitor on a network is generally referred to as a graphics **server.**

The computer on a network that is executing a graphics application is called the **client.**

A workstation that includes processors, as well as a monitor and input devices can function as both a server and a client.

8. GRAPHICS ON INTERNET

A great deal of graphics development is now done on the **Internet**.
Computers on the Internet communicate using **TCP/IP**.
Resources such as graphics files are identified by **URL** (Uniform resource locator).

The **World Wide Web** provides a hypertext system that allows users to loacate and view documents, audio and graphics.

Each URL sometimes also called as universal resource locator.
The **URL** contains two parts

- **Protocol**- for transferring the document, and
- **Server**- contains the document.

9. Graphics Software

There are two broad classifications for computer-graphics software

➢ Special-purpose packages: Special-purpose packages are designed for nonprogrammers
Example: generate pictures, graphs, charts, painting programs or CAD systems in some application area without worrying about the graphics procedures

➢ General programming packages: general programming package provides a library of graphics functions that can be used in a programming language such as C, C++, Java, or FORTRAN.
Example: GL (Graphics Library), OpenGL, VRML (Virtual-Reality Modeling Language), Java 2D,

And Java 3D

<u>**NOTE**</u>: A set of graphics functions is often called a computer-graphics application programming interface (CG API)

Coordinate Representations

- To generate a picture using a programming package we first need to give the geometric descriptions of the objects that are to be displayed known as coordinates.
- If coordinate values for a picture are given in some other reference frame (spherical, hyperbolic, etc.), they must be converted to Cartesian coordinates.

Several different Cartesian reference frames are used in the process of constructing and displaying

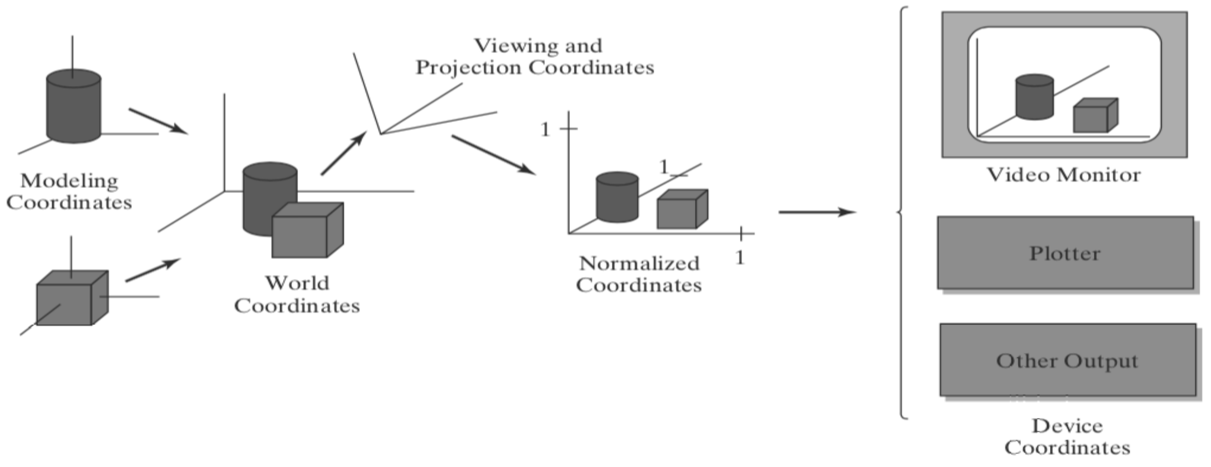
- First we define the shapes of individual objects, such as trees or furniture, These reference frames are called **modeling coordinates** or **local coordinates**
- Then we place the objects into appropriate locations within a scene reference frame called **world coordinates**.
- After all parts of a scene have been specified, it is processed through various output-device reference frames for display. This process is called the **viewing pipeline.**
- The scene is then stored in **normalized coordinates.** Which range from −1 to 1 or from 0 to 1 Normalized coordinates are also referred to as **normalized device coordinates**.
- .The coordinate systems for display devices are generally called **device coordinates**, or **screen coordinates.**

NOTE: Geometric descriptions in modeling coordinates and world coordinates can be given in floating-point or integer values.

Example: Figure briefly illustrates the sequence of coordinate transformations from modeling coordinates to device coordinates for a display

(xmc, ymc, zmc) → (xwc, ywc, zwc) → (xvc, yvc, zvc) → (xpc, ypc, zpc) → (xnc, ync, znc) → (xdc, ydc)

Computer Graphics Software



Graphics Functions

- It provides users with a variety of functions for creating and manipulating pictures
- The basic building blocks for pictures are referred to as **graphics output primitives**
- **Attributes** are properties of the output primitives
- We can change the size, position, or orientation of an object using **geometric transformations**
- **Modeling transformations**, which are used to construct a scene.
- **Viewing transformations** are used to select a view of the scene, the type of projection to be used and the location where the view is to be displayed.
- **Input functions** are used to control and process the data flow from these interactive devices(mouse, tablet and joystick)
- Graphics package contains a number of tasks .We can lump the functions for carrying out many tasks by under the heading **control operations.**

Software Standards

➢ The primary goal of standardized graphics software is **portability.**

➢ In 1984, **Graphical Kernel System (GKS)** was adopted as the first graphics software standard by the International Standards Organization (ISO)

➢ The second software standard to be developed and approved by the standards organizations was **Programmer's Hierarchical Interactive Graphics System (PHIGS).**

- Extension of PHIGS, called PHIGS+, was developed to provide **3-D** surface rendering capabilities not available in PHIGS.
- The graphics workstations from **Silicon Graphics, Inc. (SGI)**, came with a set of routines called **GL (Graphics Library)**

Other Graphics Packages

Many other computer-graphics programming libraries have been developed for

- general graphics routines
- Some are aimed at specific applications (animation, virtual reality, etc.)

Example: **Open Inventor**

Virtual-Reality Modeling Language (VRML)

We can create 2-D scenes with in Java applets (java2D, Java 3D)

10. Introduction To OpenGL:

OpenGL basic(core) library :-A basic library of functions is provided in OpenGL for specifying graphics primitives, attributes, geometric transformations, viewing transformations, and many other operations.

Basic OpenGL Syntax

-Function names in the **OpenGL basic library** (also called the **OpenGL core library**) are prefixed with **gl.** The component word first letter is capitalized.

For eg:- **glBegin, glClear, glCopyPixels, glPolygonMode**

-Symbolic constants that are used with certain functions as parameters are all in capital letters, preceded by "GL", and component are separated by underscore. For eg:-

GL_2D, GL_RGB, GL_CCW, GL_POLYGON, GL_AMBIENT_AND_DIFFUSE

-The OpenGL functions also expect specific data types. For example, an OpenGL function parameter might expect a value that is specified as a 32-bit integer. But the size of an integer specification can be different on different machines.

To indicate a specific data type, OpenGL uses special built-in, data-type names, such as **GLbyte, GLshort, GLint, GLfloat, GLdouble, GLboolean**

Related Libraries

-In addition to OpenGL basic(core) library(prefixed with gl), there are a number of associated libraries for handling special operations:-

1) OpenGL Utility(GLU):- Prefixed with “glu”. It provides routines for setting up viewing and projection matrices, describing complex objects with line and polygon approximations, displaying quadrics and B-splines using linear approximations, processing the surface-rendering operations, and other complex tasks.

-Every OpenGL implementation includes the GLU library

2) Open Inventor:- provides routines and predefined object shapes for interactive three-dimensional applications which are written in C++.

3) Window-system libraries:- To create graphics we need display window. We cannot create the display window directly with the basic OpenGL functions since it contains only device-independent graphics functions, and window-management operations are device-dependent. However, there are several window-system libraries that supports OpenGL functions for a variety of machines.

Eg:- Apple GL(AGL), Windows-to-OpenGL(WGL), Presentation Manager to OpenGL(PGL), GLX.

4) OpenGL Utility Toolkit(GLUT):- provides a library of functions which acts as interface for interacting with any device specific screen-windowing system, thus making our program device-independent. The GLUT library functions are prefixed with “**glut**”.

Header Files

In all graphics programs, we will need to include the header file for the OpenGL core library.

-In windows to include OpenGL core libraries and GLU we can use the following header files:-

#include <windows.h> //precedes other header files for including Microsoft windows ver ofOpenGLlibraries
#include<GL/gl.h>
#include <GL/glu.h>

The above lines can be replaced by using GLUT header file which ensures gl.h and glu.h are included correctly,

#include <GL/glut.h> //GL in windows

In Apple OS X systems, the header file inclusion statement will be,

#include <GLUT/glut.h>

Display-Window Management Using GLUT

Steps for displaying a picture:-

1) Initialization of GLUT:- the initialization function can also process command line arguments.

glutInit (&argc, argv);

2) Create a display window:-

glutCreateWindow ("An Example OpenGL Program");

The above function accepts a string which will be the title of display-window.

3) Specify content of display window:- For this, we create a picture using OpenGL functions and pass the picture definition to the GLUT routine glutDisplayFunc, which assigns our picture to the display window.

glutDisplayFunc (lineSegment); //passes the line-segment description to the display window.

4) Activate the display window:- the following line activates all the display windows, including their graphic content:

glutMainLoop ( );

This function must be the last one in our program. It displays the initial graphics and puts the program into an infinite loop that checks for input from devices such as a mouse or keyboard.

Additional GLUT functions:-

-We can specify location for the window using glutInitWindowPosition function

glutInitWindowPosition (50, 100);

The above statement specifies location that is 50 pixels to the right of the left edge of screen and 100 pixels down from top edge. The origin is at upper-left corner of the screen.

-Size of the display window can be specified using:

glutInitWindowSize (400, 300); //width of 400 pixels and height of 300 pixels

- We can also set a number of other options for the display window, such as buffering and a choice of color modes, with the **glutInitDisplayMode** function. The argument is a GLUT constant.

glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB); // logical or('|') used to combine constants

The above statement specifies a single refresh buffer to be used and to use red, green, and blue components to select color values.

-Background color of window can be set using:

glClearColor (1.0, 1.0, 1.0, 0.0);

This statement set the background color to white. The first three parameter are for color where 1.0 is white and 0.0 is black. The fourth is called alpha value where 1.0 indicates opaque object and 0.0 indicates transparent object.

-To display the assigned color we use:

glClear (GL_COLOR_BUFFER_BIT);

The argument **GL_COLOR_BUFFER_BIT** is an OpenGL symbolic constant specifying that it is the bit values in the color buffer (refresh buffer) that are to be set to the values indicated in the **glClearColor** function.

-We can choose a variety of color schemes for the objects we want to display in a scene.

glColor3f (0.0, 0.4, 0.2);

The suffix **3f** on the **glColor** function indicates that we are specifying the three RGB color components using floating-point (**f**) values. This function requires that the values be in the range from 0.0 to 1.0, and we have set red = 0.0, green = 0.4, and blue = 0.2.

-We need to tell OpenGL how we want to "project" our picture onto the display window because generating a two-dimensional picture is treated by OpenGL as a special case of three-dimensional viewing.

glMatrixMode (GL_PROJECTION);
gluOrtho2D (0.0, 200.0, 0.0, 150.0);

This specifies that an orthogonal projection is to be used to map the contents of a two-dimensional rectangular area of world coordinates to the screen, and that the *x*-coordinate values within this rectangle range from 0.0 to 200.0 with *y*-coordinate values ranging from 0.0 to 150.0. Whatever objects we define within this world-coordinate rectangle will be shown within the display window. Anything outside this coordinate range will not be displayed.

Therefore, the GLU function **gluOrtho2D** defines the coordinate reference frame within the display window to be (0.0, 0.0) at the lower-left corner of the display window and (200.0, 150.0) at the upper-right window corner.

The orthogonal projection just pastes our picture onto the screen.

The following code defines a two-dimensional, straight-line segment with integer, Cartesian endpoint coordinates (180, 15) and (10, 145).
glBegin(GL_LINES);
glVertex2i(180,15);
glVertex2i(10,145);
glEnd ( );

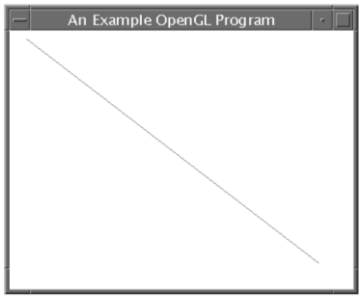
-glFlush:- routine to force execution of our OpenGL functions, which are stored by computer systems in buffers in different locations

Q) Program to display a two-dimensional line segment.

```
#include<GL/glut.h> // (or others, depending on the system in use)
void init (void)
{
glClearColor(1.0, 1.0, 1.0, 0.0); // Set display-window color to white.
glMatrixMode(GL_PROJECTION); // Set projection parameters.
gluOrtho2D(0.0,200.0,0.0,150.0);
}
voidlineSegment(void)
{
glClear(GL_COLOR_BUFFER_BIT);//Clear display window.
glColor3f(0.0,0.4,0.2);// Set line segment color to green.
glBegin(GL_LINES);
glVertex2i (180, 15); // Specify line-segment geometry.
glVertex2i (10, 145);
glEnd ( );
glFlush ( ); // Process all OpenGL routines as quickly as possible.
}
void main (int argc, char** argv)
{
glutInit (&argc, argv); // Initialize GLUT.
glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB); // Set display mode.
glutInitWindowPosition (50, 100); // Set top-left display-window position.
glutInitWindowSize (400, 300); // Set display-window width and height.
glutCreateWindow ("An Example OpenGL Program"); // Create display window.
init ( ); // Execute initialization procedure.
glutDisplayFunc (lineSegment); // Send graphics to display window.
glutMainLoop ( ); // Display everything and wait.

}
```

OUTPUT:-



-The procedure **lineSegment** is referred to as a *display callback function.* And this procedure is described as being "registered" by **glutDisplayFunc** as the routine to invoke whenever the display window might need to be redisplayed. This can occur, for example, if the display window is moved.

11. Coordinate Reference Frames.

To describe a picture, we first decide upon a convenient Cartesian coordinate system, called the **world-coordinate reference frame** which could be either Two-dimensional or Three-dimensional.

Coordinate positions are stored along with other information about the objects, such as their color and their **coordinate extents.**
Coordinate extents are the minimum and maximum x, y and z values for each object.
A set of Coordinate Extents is also described as a **bounding box** for an object

The scan-conversion process stores information about the scene, such as color values, at the appropriate locations in the frame buffer, and the objects in the scene are displayed on the output device.

Screen Coordinates

Locations on a video monitor are referenced in integer screen coordinates, which correspond to the pixel positions in the frame buffer.

Pixel Coordinate values give the scan line number and the column number.
Scan lines are referenced from 0, at the top of the screen, to some integer value, y_{max} at the bottom of the screen and pixel positions along each scan line are numbered from 0 to x_{max} , left to right.

A display algorithm must calculate the positions for those pixels that lie along the line path between the endpoints. A pixel position occupies a finite area of the screen,

Once a pixel positions have been identified for an object, the appropriate color values must be stored in the frame buffer.

setPixel (x, y) ;

This procedure stores the current color setting into the frame buffer at integer position (x, y) relative to the selected position of the screen-coordinate origin.

getPixel (x, y, color) ;

Screen coordinates are stored as three-dimensional values, where the third dimension references the depth of object positions relative to a viewing position. For a two-dimensional scene, all depth values are 0.

Absolute and Relative Coordinate Specifications

Absolute coordinate values means that the values specified are the actual positions within the coordinate system in use.

Some graphics packages also allow positions to be specified using relative coordinates. This method is useful for various graphics applications, such as producing drawings with pen plotters, artist's drawing and painting systems, and graphics packages for publishing and printing applications.

We can specify a coordinate position as an offset from the last position that was referenced called as the **current position.**

12. Specifying a two-dimensional world-coordinate reference frame in OpenGL:

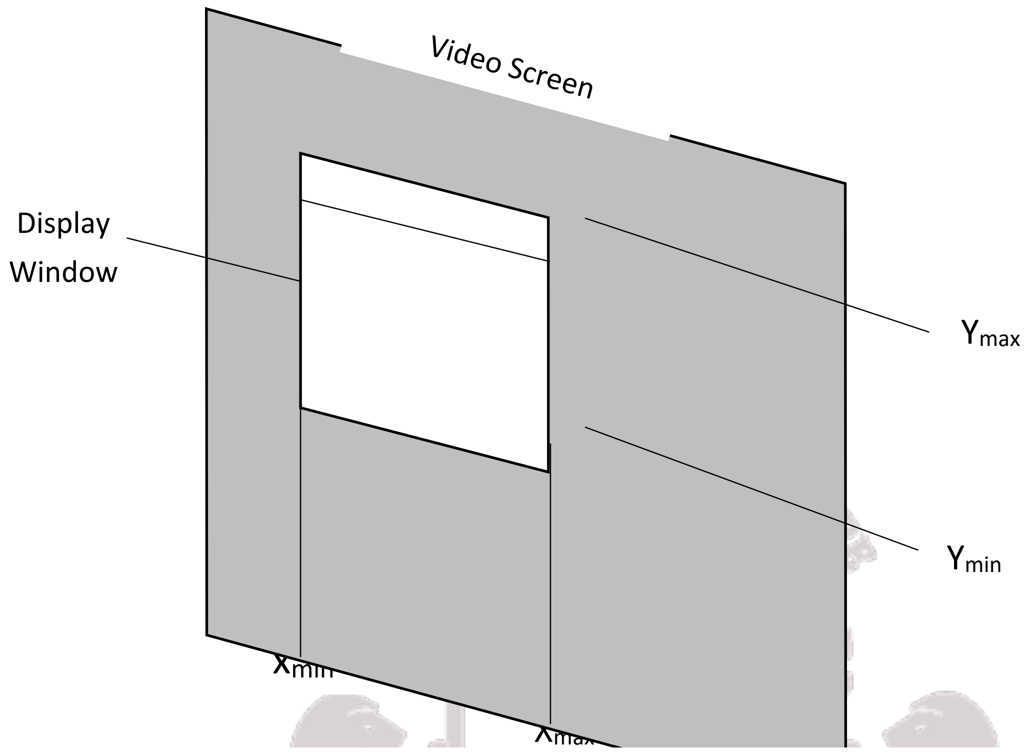
The **gluOrtho2D** command is a function used to set up any two-dimensional Cartesian reference frame. The arguments for this function are the four values defining the x and y coordinate limits for the picture to display.

The **gluOrtho2D** function specifies an orthogonal projection and the coordinate values are placed in the OpenGL projection matrix and then assign the identity matrix as the projection matrix before defining the old coordinate range.

It ensures that the coordinate values are not accumulated with any values which have been set for previous projection matrix.

glMatrixMode (GL_PROJECTION) ;
glLoadIdentity ( ) ;
gluOrtho2D (x_{min}, x_{max}, y_{min}, y_{max}) *;*

The display window will then be referenced by coordinates (x_{min}, y_{min}) at the lower left-corner and by coordinates (x_{max}, y_{max}) at the upper-right corner as shown below.



World coordinate limits for a display window, as specified in the gluOrtho2D

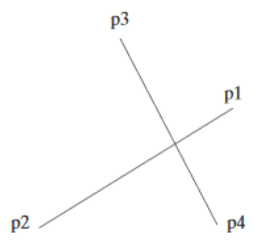
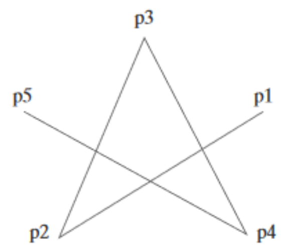
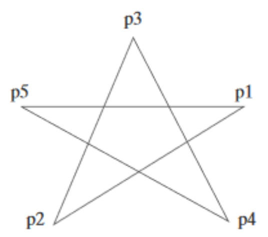
If the coordinate extents are within the coordinate range of display window, all primitives will be displayed otherwise, only parts of the primitives within the display window and all positions for the OpenGL primitives must be given in absolute coordinates with respect to the reference frame defined in the gluOrtho2D function.

13. OpenGL Line Functions

- Graphics packages typically provide a function for specifying one or more straight-line segments, where each line segment is defined by two endpoints coordinate positions.
- We use a symbolic constant as the argument for the glBegin function that interprets a
  list of positions as the endpoint coordinates for line segments.
- There are three symbolic constants in OpenGL that we can use to specify how a list of endpoint positions should be connected to form a set of straight-line segments.

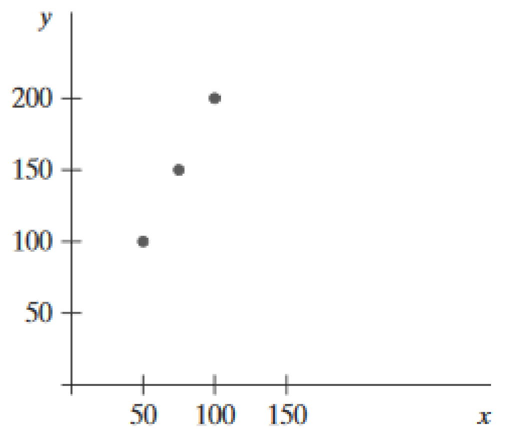
GL_LINES	GL_LINE_STRIP	GL_LINE_LOOP
A set of straight-line segments between each	The display is a sequence of connected line segments	An additional line is

successive pair of endpoints in a list is generated using this primitive line constant.	between the first endpoint in the list and the last endpoint which is a **polyline**.	drawn to connect the last coordinate position and the first coordinate position which produces a **closed polyline**.
A set of unconnected lines are formed unless some coordinate positions are repeated. Lines that cross but do not share a vertex are still considered to be unconnected.	The first line segment in the polyline is displayed between the first endpoint and the second endpoint; the second line segment is between the second and third endpoints; and so on, up to the last line endpoint.	The first line segment in the polyline is displayed between the first endpoint and the second endpoint; the second line segment is between the second and third endpoints; and so on, up to the last line endpoint which in turn is connected to the first endpoint.
Nothing is displayed if only one endpoint is specified, and the last endpoint is not processed if the number of endpoints listed is odd.	Nothing is displayed if at least two coordinate positions are not listed.	Nothing is displayed if at least two coordinate positions are not listed.

Code to display the following figure:

 	
glBegin(GL_LINES);
glVertex2iv (p1);
glVertex2iv (p2);
glVertex2iv (p3);
glVertex2iv (p4);
glVertex2iv (p5);
glEnd ();
	Code to display the following figure:
	
	
	glBegin(GL_LINE_STRIP);
	glVertex2iv (p1);
	glVertex2iv (p2);
	glVertex2iv (p3);
	glVertex2iv (p4);
	glVertex2iv (p5);
	glEnd ();
		Code to display the following figure:
		
		
		glBegin(GL_LINE_LOOP);
		glVertex2iv (p1);
		glVertex2iv (p2);
		glVertex2iv (p3);
		glVertex2iv (p4);
		glVertex2iv (p5);
		glEnd ();

14. OpenGL Point Functions

- A coordinate position in the world reference frame is given to specify the geometry of a point. Then this coordinate position, along with other geometric descriptions we may have in our scene, is passed to the viewing routines.
- OpenGL primitives are displayed with a default size and colour if other attribute values are not specified.
- We use the following OpenGL function to state the coordinate values for a single position:
  glVertex* ();
  The asterisk (*) indicates that suffix codes are required for this function.
- These suffix codes are used to identify the spatial dimension, the numerical data type to be used for the coordinate values, and a possible vector form for the coordinate specification.
- Calls to **glVertex** functions must be placed between a **glBegin** function and a **glEnd** function. The argument of the **glBegin** function is used to identify the kind of output primitive that is to be displayed, and **glEnd** takes no arguments.
- For point plotting, the argument of the **glBegin** function is the symbolic constant **GL_POINTS**.
  ```
  glBegin (GL_POINTS);
  glVertex* ();
  glEnd ();
  ```
- The **glVertex** function is used in OpenGL to specify coordinates for any point position.
- Coordinate positions in OpenGL can be given in two, three, or four dimensions. We use a suffix value of 2, 3, or 4 on the **glVertex** function to indicate the dimensionality of a coordinate position.
- The second suffix code on the **glVertex** function is used to specify the numerical values of the coordinates.
- Suffix codes for specifying a numerical data type are i(integer), s(short), f(float), and d(double).



Display of three-point positions generated with **glBegin**(**GL_POINTS**).

- Coordinate values can be listed explicitly in the **glVertex** function:
  glBegin (GL_POINTS);
  glVertex2i (50, 100);
  glVertex2i (75, 150);
  glVertex2i (100, 200);
  glEnd ();
- Or a single argument can be used that references a coordinate position as an array. If we use an array specification for a coordinate position, we need to append v("vector") as a third suffix code:
  int point1 [ ] = {50, 100};
  int point2 [ ] = {75, 150};
  int point3 [ ] = {100, 200};
- Call the OpenGL functions:
  glBegin (GL_POINTS);
  glVertex2iv (point1);
  glVertex2iv (point2);
  glVertex2iv (point3);
  glEnd ();

15. Point Attributes

Basically, we can set two attributes for points: color and size.\

OpenGL Point-Attribute Functions

The displayed color of a designated point position is controlled by the current color values in the state list. Also, a color is specified with either the **glColor** function.

We set the size for an OpenGL point with

glPointSize (size);

and the point is then displayed as a square block of pixels. Parameter **size** is assigned a positive floating-point value, which is rounded to an integer.

16. Line Attributes:

A straight-line segment can be displayed with three basic attributes: **color, width,** and **style**.
Line color is typically set with the same function for all graphics primitives, while line width and line style are selected with separate line functions.

OpenGL Line-Attribute Functions:

We can control the appearance of a straight-line segment in OpenGL with three attribute settings: line color, line width, and line style.

We have already seen how to make a color selection, and OpenGL provides a function for setting the width of a line and another function for specifying a line style, such as a dashed or dotted line.

OpenGL Line-Width Function

Line width is set in OpenGL with the function

glLineWidth (width);

We assign a floating-point value to parameter **width**, and this value is rounded to the nearest nonnegative integer.

OpenGL Line-Style Function

By default, a straight-line segment is displayed as a solid line.

However, we can also display dashed lines, dotted lines, or a line with a combination of dashes and dots, and we can vary the length of the dashes and the spacing between dashes or dots.

We set a current display style for lines with the OpenGL function

glLineStipple (repeatFactor, pattern);

Parameter **pattern** is used to reference a 16-bit integer that describes how the line should be displayed.

A 1 bit in the pattern denotes an "on" pixel position, and a 0 bit indicates an "off" pixel position.

The pattern is applied to the pixels along the line path starting with the low-order bits in the pattern.

The default pattern is 0xFFFF (each bit position has a value of 1), which produces a solid line.

Integer parameter **repeatFactor** specifies how many times each bit in the pattern is to be repeated before the next bit in the pattern is applied. The default repeat value is 1.

17. Line Drawing Algorithm

A straight-line segment in a scene is defined by coordinate positions for the endpoints of the segment.

To display the line on a raster monitor, the graphics system must first project the endpoints to integer screen coordinates and determine the nearest pixel positions along the line path between the two endpoints then the line color is loaded into the frame buffer at the corresponding pixel coordinates.

A computed line positions of (10.48,20.51) is converted to pixel position (10,21). This rounding of coordinates values to integers causes all but horizontal and vertical lines to be displayed with a stair-step appearance ("the jaggies").

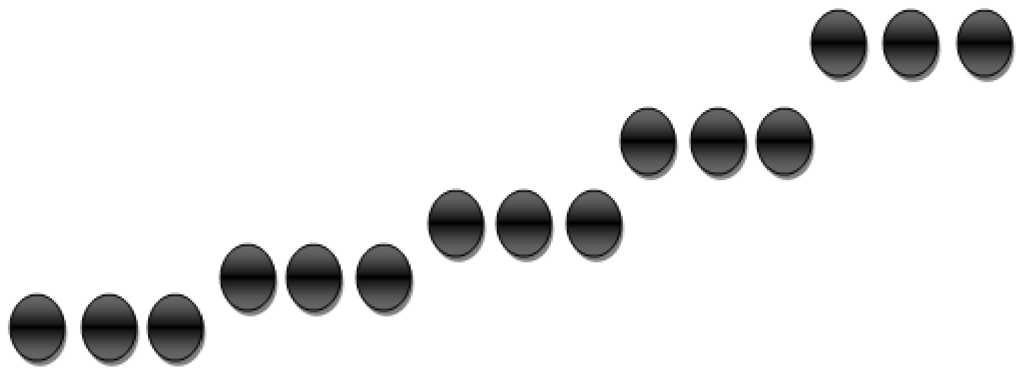


Fig: stair-step effect(jaggies) produced when a line is generated as a series of pixel positions.

17.1 Line Equations

The Cartesian slope-intercept equation for a straight line is

$y=m * x +b$------------>(1)

with m as the slope of the line and b as the y intercept.

Given that the two endpoints of a line segment are specified at positions (x_0,y_0) and (x_{end}, y_{end}) ,as shown in fig.

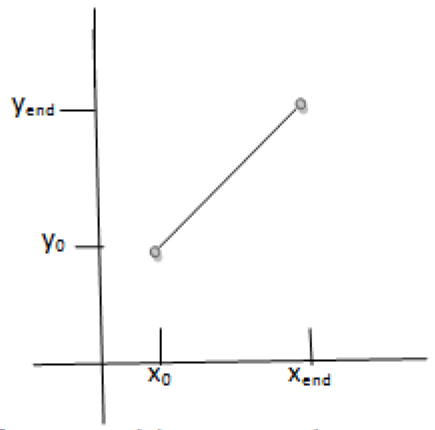


fig. Line path between endpoint positions (x_0,y_0) and (x_{end}, y_{end}).

we determine values for the slope m and y intercept b with the following equations:

$m=(y_{end} - y_0)/(x_{end} - x_0)$----------------->(2)

$b=y_0 - m.x_0$-------------->(3)

Algorithms for displaying straight line are based on the line equation (1) and calculations given in eq(2) and (3).

for given x interval δx along a line, we can compute the corresponding y interval δy from eq.(2) as

δy=m. δx----------------->(4)

similarly, we can obtain the x interval δx corresponding to a specified δy as

δx=δy/m------------------>(5)

These equations form the basis for determining deflection voltages in analog displays, such as vector-scan system, where arbitrarily small changes in deflection voltage are possible.

For lines with slope magnitudes

- |m|<1, δx can be set proportional to a small horizontal deflection voltage with the corresponding vertical deflection voltage set proportional to δy from eq.(4)
- |m|>1, δy can be set proportional to a small vertical deflection voltage with the corresponding horizontal deflection voltage set proportional to δx from eq.(5)
- |m|=1, δx=δy and the horizontal and vertical deflections voltages are equal

On raster systems, lines are plotted with pixels ,and the step sizes in the horizontal and vertical directions are constrained by pixel separations. That is we must "sample" a line at discrete positions and determine the nearest pixel to the line at each sample position. The scan-conversion process for straight lines is illustrated in fig. with discrete sample positions along the x -axis.

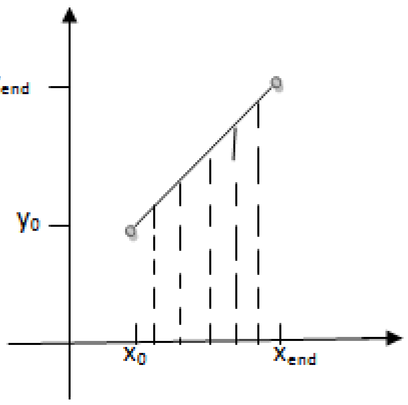


Fig. straight line segment with five sampling positions along x-axis between x_0 and x_{end}

<u>17.2 DDA Algorithm (DIGITAL DIFFERENTIAL ANALYZER)</u>

The DDA is a scan-conversion line algorithm based on calculating either δy or δx.

A line is sampled at unit intervals in one coordinate and the corresponding integer values nearest the line path are determined for the other coordinate

DDA Algorithm has three cases so from equation i.e.., $m=(y_{k+1} - y_k)/(x_{k+1} - x_k)$

Case1:if m<1,x increment in unit intervals

i.e..,$x_{k+1}=x_k+1$

then, $m=(y_{k+1} - y_k)/(x_k+1 - x_k)$

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

$y_{k+1} = y_k + m$------------>(1)

where k takes integer values starting from 0,for the first point and increases by 1 until final endpoint is reached. Since m can be any real number between 0.0 and 1.0, Case2:if m>1, y increment in unit intervals

i.e.., $y_{k+1} = y_k + 1$

then, m= $(y_k + 1- y_k)/(x_{k+1} - x_k)$

$m(x_{k+1} - x_k)=1$

$x_{k+1} =(1/m)+ x_k$-----------------(2)

Case3:if m=1,both x and y increment in unit intervals

i.e..,$x_{k+1}=x_k + 1$ and y= $y_{k+1} + 1$

Equations (1) and (2) are based on the assumption that lines are to be processed from the left endpoint to the right endpoint. If this processing is reversed, so that the starting endpoint is at the right, then either we have δx=-1 and

$y_{k+1} = y_k - m$-----------------(3)

or(when the slope is greater than 1)we have δy=-1 with

$x_{k+1} = x_k - (1/m)$----------------(4)

Similar calculations are carried out using equations (1) through (4) to determine the pixel positions along a line with negative slope. thus, if the absolute value of the slope is less than 1 and the starting endpoint is at left ,we set δx==1 and calculate y values with eq(1).

when starting endpoint is at the right(for the same slope),we set δx=-1 and obtain y positions using eq(3).

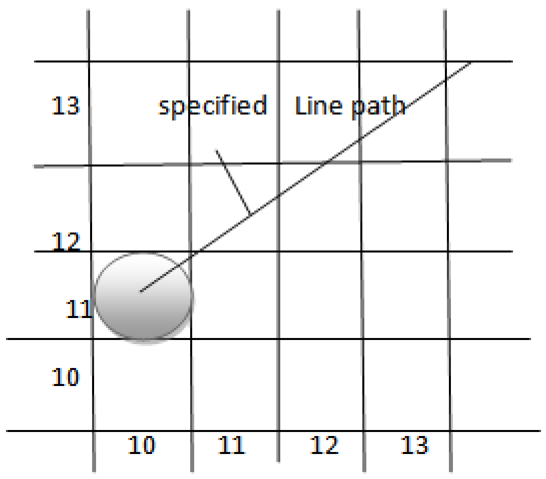


fig. A section of a display screen where a straight line segment is plotted

For negative slope with absolute value greater than 1,we use δy=-1 and eq (4) or we use δy=1 and eq(2).

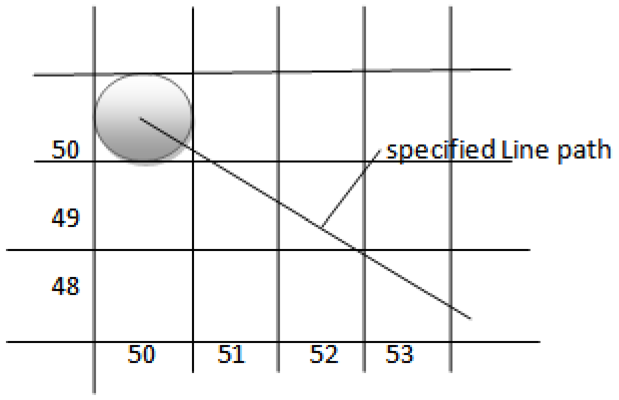


fig. A section of a display screen where a negative slope line segment is plotted

This algorithm is summarized in the following procedure, which accepts as input two integer screen positions for the endpoints of a line segment.

if m<1,where x is incrementing by 1

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

- So initially x=0,Assuming (x_0,y_0)as initial point assigning x= x_0,y=y_0 which is the starting point .

Illuminate pixel(x, round(y))

- $x_1 = x + 1$, $y_1 = y + 1$

  Illuminate pixel(x_1,round(y_1))

- $x_2 = x_1 + 1$, $y_2 = y_1 + 1$

  Illuminate pixel(x_2,round(y_2))

.............Till it reaches final point.

if m>1,where y is incrementing by 1

$x_{k+1} = (1/m) + x_k$

- So initially y=0,Assuming (x_0,y_0)as initial point assigning x= x_0,y=y_0 which is the starting point .

Illuminate pixel(round(x),y)

- $x_1 = x + (1/m)$,$y_1 = y$

  Illuminate pixel(round(x_1),y_1)

- $x_2 = x_1 + (1/m)$, $y_2 = y_1$

  Illuminate pixel(round(x_2),y_2)

.............Till it reaches final point.

The DDA algorithm is faster method for calculating pixel position than one that directly implements .

It eliminates the multiplication by making use of raster characteristics, so that appropriate increments are applied in the x or y directions to step from one pixel position to another along the line path.

The accumulation of round off error in successive additions of the floating point increment, however can cause the calculated pixel positions to drift away from the true line path for long line segments. Furthermore ,the rounding operations and floating point arithmetic in this procedure are still time consuming.

we improve the performance of DDA algorithm by separating the increments m and 1/m into integer and fractional parts so that all calculations are reduced to integer operations.

EXAMPLE:

1) Consider two points (2,3) and (12,8) solve by using DDA Algorithm.

solution :: Assign $(x_1,y_1)=(2,3)$ and $(x_2,y_2)=(12,8)$

$m=(y_2 - y_1)/(x_2 - x_1) = (8-3)/(12-2) = 0.5$

It is in case1 :m<1

$(x_1,y_1)=(2,3)$

$x_2=x_1 + 1$ $y_2 = y_1 + m$

$x_2 = 2+1 = 3$ $y_2 = 3+0.5 = 3.5$

illuminate pixel(3,round(3.5)) = (3,3.5)

$(x_2,y_2)=(3,4)$

$x_3 = x_2 + 1$ $y_3 = y_2 + m$

$x_3 = 3+1 = 4$ $y_3 = 3.5+0.5 = 4$

illuminate pixel(4,round(4)) = (4,4)

$(x_3,y_3) = (4,4)$

$x_4 = x_3 + 1$ $y_4 = y_3 + m$

$x_4 = 4+1 = 5$ $y_4 = 4+0.5 =4.5$

X	Y	x-plot	y-plot
2	3	2	3
3	3.5	3	4
4	4	4	4
5	4.5	5	5
6	5	6	5

7	5.5	7	6
8	6	8	6
9	6.5	9	7
10	7	10	7
11	7.5	11	8
12	8	12	8

17.3 Bresenham's Algorithm:

It is an efficient raster scan generating algorithm that uses incremental integral calculations.

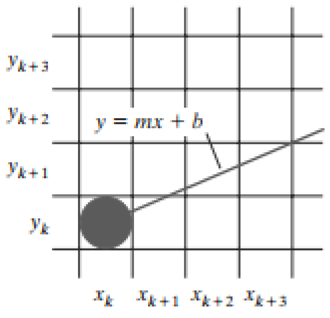


FIGURE 6
A section of the screen showing a pixel in column x_k on scan line y_k that is to be plotted along the path of a line segment with slope $0 < m < 1$.

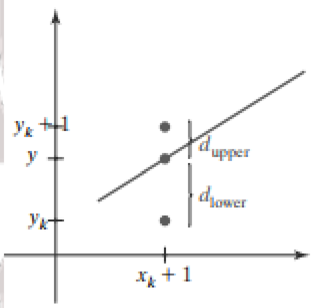


FIGURE 7

Assumptions:

1) Consider two points with coordinates (x_1, y_1) and (x_2, y_2) of a line.

2) The slope of the line i.e., m<=45

3) $x_1 < x_2$--

4) Consider the equation of a straight line y=mx+c where m=dy/dx

ALGORITHM:

- Here we have to decide upon which pixel the line has to be drawn.
- The first pixel (x_i, y_i) is selected without any confusion but in the next line is passed on the two pixels i.e., (x_i, y_i)and (x_{i+1}, y_{i+1}).Hence through this algorithm we need to select which pixel has to be selected.
- This decision has to be made by taking the distances d1 and d2.
- Since the equation of line is y=mx+c and next point is x_{i+1}

'y' can be calculated as,

$Y=m(x_i +1)+b$ --->(1)

where $m=\Delta y/\Delta x$

Consider $d1=y-y_i$

Replace y with eqn(1),

$d1=m(x_i +1)+b-y_i$

Consider $d2=y_{i+1} -y$

$d2=y_i + 1-y$

replace eqn(1) to y,

$d2=y_i +1-m(x_i +1)-b$

Lets subtract,

d1-d2

$=>m(x_i +1)+b-y_i-y_i-1+m(x_i +1)+b$

$=>2m(x_i +1)+2b-2y_i-1$

$=>2[m(x_i +1)-2y_i+2b-1$--->(2)

We subtracted d1-d2,because to know which distance is near to the line.

- If (d1-d2)<0 means d1<d2 where d1 is nearer to the line[closer to y_i].Hence select the point(x_{i+1},y_i)
  $=>y_{i+1}=y_i$

- If (d1-d2)>0 means d1>d2 where d2 is nearer to the line[closer to y_{i+1}].Hence select the point (x_{i+1},y_{i+1})
  $\Rightarrow y_{i+1}=y_i + 1$

As WKT,$m=\Delta y/\Delta x$ which always gives fraction value and Bresenham's algorithm avoids this fraction values,so eqn(2) will be multiplied by Δx,

i.e,$(d1-d2=2m(x_i + 1)-2y_i+2b-1)*\Delta x$

Let us consider the new equation variable P_i

i.e.,$P_i=(d1-d2)\Delta x$

Here the sign of Pi=sign of (d1-d2)

If $P_i<0$ then $d1-d2<0$ and so on.

$P_i=2.\Delta y.x_i+2.\Delta y-2.y_i.\Delta x+2.\Delta x.b-\Delta x$---(3)

$=2.\Delta y.x_i-2.y_i.\Delta x+(2.\Delta y+2.\Delta x.b-\Delta x)$

From the above eqn $(2.\Delta y+2.\Delta x.b-\Delta x)$ is considered constant.

$C=2.\Delta y+2.\Delta x.b-\Delta x$

Hence eqn becomes,

$P_i=2.\Delta y.x_i-2.y_i.\Delta x+C$---(4)

As the sign of P_i is same as (d1-d2),P_i can be used to decide which pixel needs to be selected.

Lets consider the next iteration for P_i i.e, P_{i+1} and eqn is

$P_{i+1}=2.\Delta y.x_{i+1} - 2.y_{i+1}.\Delta x+C$---(5)

Subtract $P_{i+1}-P_i$,

$=2.\Delta y(x_{i+1}-x_i)-2.\Delta x(y_{i+1}-y_i)$

x_{i+1} is nothing but moving to next pixel where each pixel is one unit difference.

So,x_{i+1} can also be written as

$x_{i+1}=x_i + 1$ which can be replaced in the above eqn.

$P_{i+1} - P_i=2.\Delta y-2.\Delta x(y_{i+1}-y_i)$---(6)

while y_{i+1} is not equal to $y_i + 1$,because the value of y will always not be incremented,it can either stay in y_i or y_{i+1}.

Move P_i to RHS,

$P_{i+1}=P_i+2.\Delta y-2.\Delta x(y_{i+1}-y_i)$

Simplify the above eqns as,

If $P_i<0$,

Then replace y_{i+1}->y_i

$P_{i+1}=P_i+2.\Delta y-2.\Delta x(y_i-y_i)$

$P_{i+1}=P_i+2.\Delta y$---(7)

If $P_i>0$,

Then replace y_{i+1}->y_i +1

$P_{i+1}=P_i+2.\Delta y-2.\Delta x$---(8)

Hence the equation for algorithm depends on eqn(7) and eqn(8),

$P_{i+1}=P_i+2.\Delta y$

$P_{i+1}=P_i+2.\Delta y-2.\Delta x$

Now we have to find the intial parameters by considering the eqn(3),

$P_i=2.\Delta y.x_i-2.\Delta x.y_i+2.\Delta y+2.\Delta x.b-\Delta x$

Assigning i=0,

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

Now from y=mx+b

$b=y_0-mx_0$

$b=y_0-(\Delta y/\Delta x).x_0$

Put $b=y_0-(\Delta y/\Delta x).x_0$ in the eqn above,

$P_0=2.\Delta y.x_0-2.\Delta x.y_0+2.\Delta y+2.\Delta.x[y_0- (\Delta y/\Delta x)x_0]-\Delta.x$

$P_0=2\Delta.y-\Delta.x$-----Initial Parameter

Bresenham's Line-Drawing Algorithm for |m| < 1.0

1. Input the two line endpoints and store the left endpoint in (x_0, y_0).

2. Set the color for frame-buffer position (x_0, y_0); i.e., plot the first point.

3. Calculate the constants x, y, 2y, and 2y − 2x, and obtain the starting value for the decision parameter as $p_0 = 2y - x$

4. At each xk along the line, starting at k = 0, perform the following test: If pk < 0, the next point to plot is (xk + 1, yk) and pk+1 = pk + 2y Otherwise, the next point to plot is (xk + 1, yk + 1) and pk+1 = pk + 2y − 2x

5. Repeat step 4 x − 1 more times.

EXAMPLE 1 Bresenham Line Drawing To illustrate the algorithm, we digitize the line with endpoints (20, 10) and (30, 18). This line has a slope of 0.8, with x = 10, y = 8 The initial decision parameter has the value $p_0 = 2y - x = 6$ and the increments for calculating successive decision parameters are 2y = 16, 2y − 2x = −4 We plot the initial point $(x_0, y_0) = (20, 10)$, and determine successive pixel positions along the line path from the decision parameter as follows:

k	p_k	(x_{k+1}, y_{k+1})
0	6	(21, 11)
1	2	(22, 12)
2	−2	(23, 12)
3	14	(24, 13)
4	10	(25, 14)

k	p_k	(x_{k+1}, y_{k+1})
5	6	(26, 15)
6	2	(27, 16)
7	−2	(28, 16)
8	14	(29, 17)
9	10	(30, 18)

A plot of the pixels generated along this line path is shown in Figure 8.

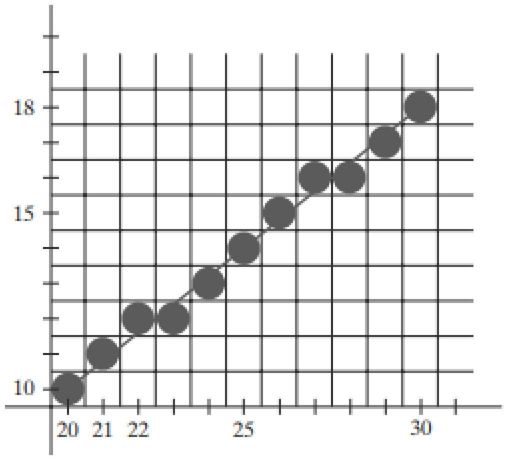


FIGURE 8
Pixel positions along the line path between endpoints (20, 10) and (30, 18), plotted with Bresenham's line algorithm.

17.4 <u>Midpoint Circle Algorithm</u>

- Midpoint circle algorithm generates all points on a circle centered at the origin by incrementing all the way around circle.
- The strategy is to select which of 2 pixels is closer to the circle by evaluating a function at the midpoint between the 2 pixels

Eight way symmetry

The shape of the circle is similar in each quadrant.Therefore ,if we determine the curve positions in the first quadrant ,we can generate the circle positions in the second quadrant of xy plane.The circle sections in the third and fourth quadrant can be obtained from sections in the first and second quadrant by considering the symmetry along X axis

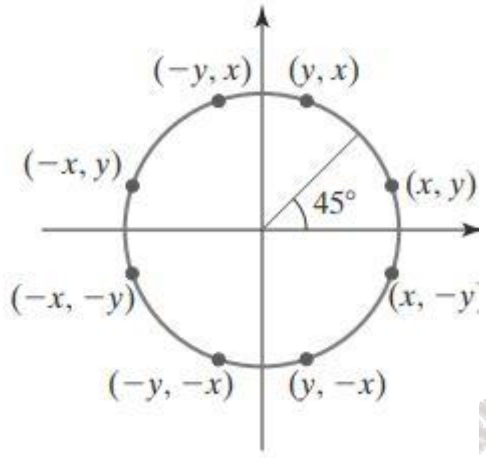
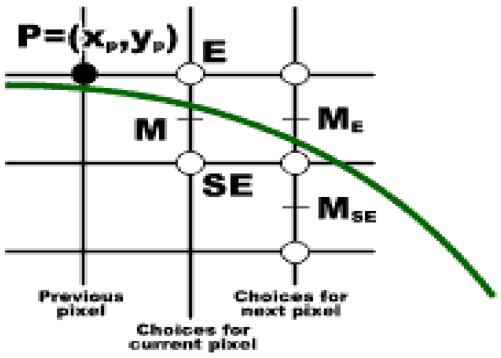


FIGURE 13
Symmetry of a circle. Calculation of a circle point (x, y) in one octant yields the circle points shown for the other seven octants.

- Conside the circle centered at the origin,if the point (x, y) is on the circle,then we can compute 7 other points on the circle as shown in the above figure.
- So we need to compute only one 45 degree segment to determine the circle completely.

- In **Midpoint Circle Algorithm** by evaluating a function at the midpoint between the 2 pixel,we can decide which pixel is closer to the circle.

- If pixel P at (x_p,y_p) has been chosen as the starting pixel,then the next pixel can be E or SE as shown in figure below.
- Now the choice is between E or SE



where E= (x_{p+1}, y_p)

SE= (x_{p+1}, y_{p-1})

M=(x_{p+1} , y_p - 1/2)

- According to implicit formula:

  $F(x,y) = x^2 + y^2 - R^2$, we use implicit formula for this algorithm.

- The relative position of any point (x ,y) can be determined by checking the sign of the circle function

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

- These tests are performed for the midpositions between pixels near the circle path at each sampling step.Thus the circle function is the decision parameter in the midpoint algorithm.
- If the midpoint **between** the pixel E and SE is outside the circle,then SE is closer to the circle
- If the midpoint is **inside** the circle,then pixel E is closer to the circle.
- As for lines,we choose on the basis of the decision variable d,which is the value of the function at the midpoint,

$$p_k = F(x_p+1 , y_p - 1/2)$$

$$= (x_p + 1)^2 + (y_p - 1/2)^2 - R^2$$

➢ If $p_k >= 0$

- SE is chosen ,where x is incremented by 1 and y is decremented by 1.

$$P_{k+1} = F(x_p + 2, y_p - 3/2)$$

$$\Delta d_{se} = p_{k+1} - p_k$$

$= F(x_p+2, y_p-3/2) – F(x_p+1, y_p – ½)$

$=(x_p + 2)^2 + (y_p – 3/2)^2 – R^2 – \{(x_p +1)^2 +$
$(y_p – ½)^2 - R^2$

Expand this equation using the formula $(a + b)^2$ and $(a - b)^2$

$=\{(x_p^2 + 4 x_p+ 4) + (y_p^2 – 2y_p .(3/2) + 9/4) – R^2\} – \{ (x_p^2 + 2x_p+ 1) +$
$(y_p^2 – 2y_p (1/2) + (1/4) – R^2 \}$

$=x_p^2 + 4x_p+ 4+ y_p^2- 3y_p+ 9/4 - R^2 – x_p^2 - 2x_p - 1 – y_p^2 + y_p - 1/4 + R^2$

$= 2x_p + 3 – 2y_p + 8/4$

$\Delta d_{se} = 2x_p – 2y_p + 5$

- If $p_k < 0$
  - E is chosen, and the next midpoint will be one increment over in x.That is x= x+1

$\Delta d_E = p_{k+1} – p_k$

$= F(x_p + 1 + 1 ,y_p – 1/2) – F(x_p + 1, y_p – 1/2)$

$= F (x_p +2 , y_p - 1/2) – F(x_p + 1, y_p – 1/2)$

$= (x_p +2)^2 + (y_p - ½)^2 – R^2 – \{ (x_p + 1)^2 + (y_p – 1/2)^2 – R^2)\}$

$= (x_p +2)^2 + (y_p - ½)^2 – R^2 – (x_p + 1)^2 - (y_p – 1/2)^2 + R^2)$

$= (x_p +2)^2 – (x_p + 1)^2$

$= x_p^2 +4xp + 4 – (x_p^2 + 2xp + 1)$

$= x_p^2 +4xp + 4 – x_p^2 - 2xp - 1$

$\Delta d_E = 2x_p + 3$

- The initial decision parameter(P_0) is obtained by evaluating the circle function at the start position $(x_0 , y_0) = (0, r)$

  $P_0 = p_k$

  $= F(x_p + 1, y_p – 1/2)$

$= F(x_0 + 1, y_0 – 1/2)$

$=F(0+1, R – 1/2)$

$= 1^2 + (R – ½)^2 – R^2$

$=1 + R^2 – R + 1/4 – R^2$

$P_0 = 5/4 – R$

- If the radius r is specified as an integer,we can simply round it to:

  $P_0=1 – R$

Midpoint Circle Algorithm

- Midpoint circle algorithm generates all points on a circle centered at the origin by incrementing all the way around circle.
- The strategy is to select which of 2 pixels is closer to the circle by evaluating a function at the midpoint between the 2 pixels

Eight way symmetry

The shape of the circle is similar in each quadrant. Therefore, if we determine the curve positions in the first quadrant, we can generate the circle positions in the second quadrant of xy plane. The circle sections in the third and fourth quadrant can be obtained from sections in the first and second quadrant by considering the symmetry along X axis

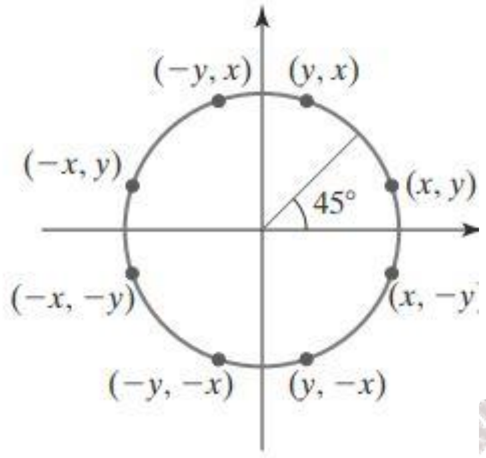
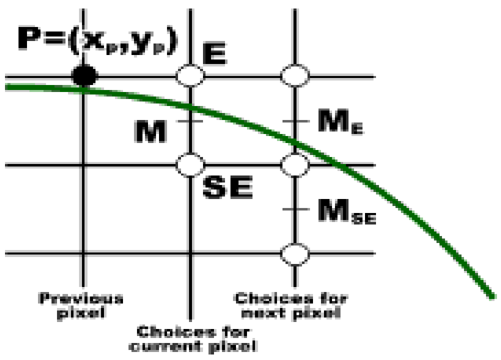


FIGURE 13
Symmetry of a circle. Calculation of a circle point (x, y) in one octant yields the circle points shown for the other seven octants.

- Conside the circle centered at the origin,if the point (x, y) is on the circle,then we can compute 7 other points on the circle as shown in the above figure.
- So we need to compute only one 45 degree segment to determine the circle completely.

- In **Midpoint Circle Algorithm** by evaluating a function at the midpoint between the 2 pixel,we can decide which pixel is closer to the circle.

- If pixel P at (x_p,y_p) has been chosen as the starting pixel,then the next pixel can be E or SE as shown in figure below.
- Now the choice is between E or SE



where E= (x_{p+1}, y_p)

SE= (x_{p+1} , y_{p-1})

M=(x_{p+1} , y_p - 1/2)

- According to implicit formula:

  F(x,y) = $x^2 + y^2 - R^2$, we use implicit formula for this algorithm.

- The relative position of any point (x ,y) can be determined by checking the sign of the circle function

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

- These tests are performed for the midpositions between pixels near the circle path at each sampling step.Thus the circle function is the decision parameter in the midpoint algorithm.
- If the midpoint **between** the pixel E and SE is outside the circle,then SE is closer to the circle
- If the midpoint is **inside** the circle, then pixel E is closer to the circle.
- As for lines,we choose on the basis of the decision variable d,which is the value of the function at the midpoint,

  p_k =F(x_p+1 , y_p - 1/2)

  = $(x_p + 1)^2 + (y_p - 1/2)^2 - R^2$

➢ If p_k>= 0

- SE is chosen, where x is incremented by 1 and y is decremented by 1.

  P_{k+1}=F(x_p + 2, y_p – 3/2)

  $\Delta d_{se} = p_{k+1} - p_k$

$= F(x_p+2, y_p-3/2) – F(x_p+1, y_p – ½)$

$=(x_p + 2)^2 + (y_p – 3/2)^2 – R^2 – \{(x_p +1)^2 +$
$(y_p – ½)^2 - R^2$

Expand this equation using the formula $(a + b)^2$ and $(a - b)^2$

$=\{(x_p^2 + 4x_p + 4) + (y_p^2 – 2y_p .(3/2) + 9/4) – R^2\} – \{(x_p^2 + 2x_p + 1) +$
$(y_p^2 – 2y_p (1/2) + (1/4) – R^2\}$

$=x_p^2 + 4x_p + 4 + y_p^2 - 3y_p + 9/4 - R^2 – x_p^2 - 2x_p - 1 – y_p^2 + y_p - 1/4 + R^2$

$= 2x_p + 3 – 2y_p + 8/4$

$\Delta d_{se} = 2x_p – 2y_p + 5$

- If $p_k < 0$
  - E is chosen, and the next midpoint will be one increment over in x.That is x= x+1

$\Delta d_E = p_{k+1} – p_k$

$= F(x_p + 1 + 1, y_p – 1/2) – F(x_p + 1, y_p – 1/2)$

$= F(x_p +2, y_p - 1/2) – F(x_p + 1, y_p – 1/2)$

$= (x_p +2)^2 + (y_p - ½)^2 – R^2 – \{(x_p + 1)^2 + (y_p – 1/2)^2 – R^2)\}$

$= (x_p +2)^2 + (y_p - ½)^2 – R^2 – (x_p + 1)^2 - (y_p – 1/2)^2 + R^2)$

$= (x_p +2)^2 – (x_p + 1)^2$

$= x_p^2 +4xp + 4 – (x_p^2 + 2xp + 1)$

$= x_p^2 +4xp + 4 – x_p^2 - 2xp - 1$

$\Delta d_E = 2x_p + 3$

- The initial decision parameter(P_0) is obtained by evaluating the circle function at the start position $(x_0, y_0) = (0, r)$

  $P_0 = p_k$
  $= F(x_p + 1, y_p – 1/2)$

$= F(x_0 + 1, y_0 – 1/2)$

$= F(0 + 1, R – 1/2)$

$= 1^2 + (R – ½)^2 – R^2$

$= 1 + R^2 – R + 1/4 – R^2$

$P_0 = 5/4 – R$

- If the radius r is specified as an integer,we can simply round it to:

$P_0 = 1 – R$

<u>Midpoint Circle Algorithm</u>

1. Input radius r and circle center (x_c , y_c), then set the coordinates for the first point on the circumference of a circle centered on the origin as

   $(x_0, y_0) = (0, r)$
2. Calculate the initial value of the decision parameter as

   $p_0 = 5/4 − r$
3. At each x_k position, starting at k = 0, perform the following test: If $p_k < 0$, the next point along the circle centered on (0, 0) is (x_{k+1}, y_k) and

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

   Otherwise, the next point along the circle is (x_{k+1}, y_{k-1}) and

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

   where $2x_{k+1} = 2x_k + 2$ and $2y_{k+1} = 2y_{k-2}$.

4. Determine symmetry points in the other seven octants.

5. Move each calculated pixel position (x, y) onto the circular path centered at (xc , yc) and plot the coordinate values as follows

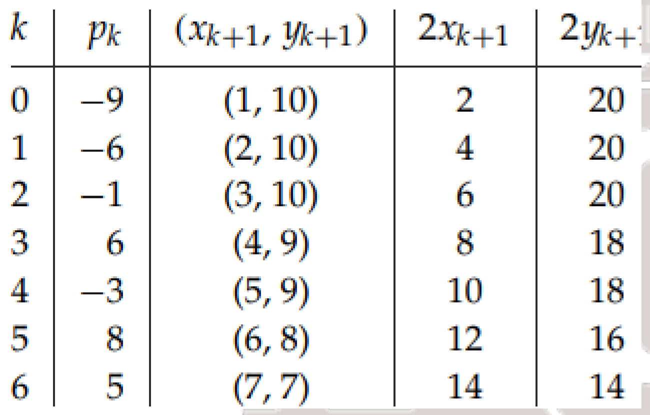
Problem

Given a circle radius r = 10, we demonstrate the midpoint 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
p0 = 1 − r = −9

For the circle centered on the coordinate origin, the initial point is (x0, y0) = (0, 10), and initial increment terms for calculating the decision parameters are

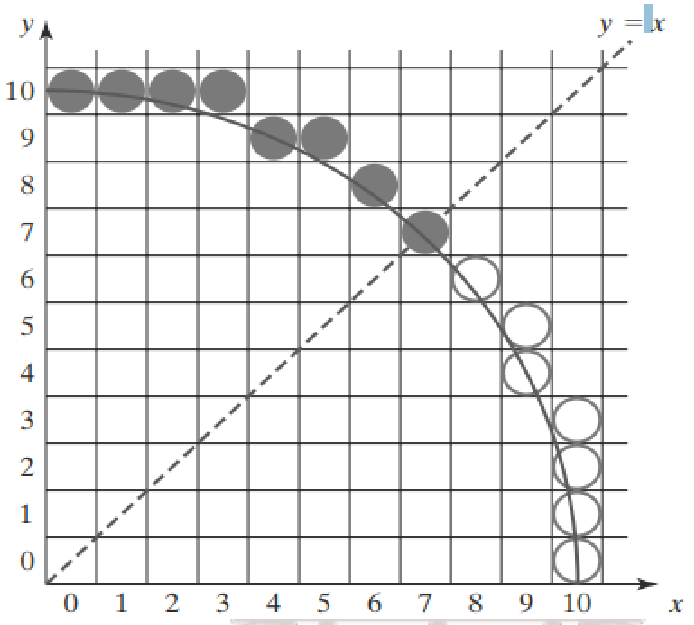
2x0 = 0, 2y0 = 20

Successive midpoint decision parameter values and the corresponding coordinate positions along the circle path are listed in the following table:



k	p_k	(x_{k+1}, y_{k+1})	$2x_{k+1}$	$2y_{k+}$
0	−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

A plot of generated pixel positions in the first quadrant is shown


y
y = x
10
9
8
7
6
5
4
3
2
1
0
0 1 2 3 4 5 6 7 8 9 10
x