VSS University of Technology

---------- BURLA ----------

DEPARTMENT OF COMPUTER APPLICATIONS (MCA)

MCA-204 COMPUTER GRAPHICS AND MULTIMEDIA 3rd Semester

Veer Surendra Sai University of Technology, Burla

(A UGC & AICTE affiliated Unitary Technical University)

Sambalpur-768018, Odisha

INDIA

www.vssut.ac.in

MCA-204 Computer Graphics & Multimedia L-T-P: 3-1-0

Prerequisite: In addition to programming proficiency in C++, the student should have a basic understanding of linear algebra and calculus.

UNIT I: (5 Hours)

Graphics Primitives: Introduction – Raster & Random display concepts and devices – CRT – Primitive operations – The display file interpreter – Normalized device co-ordinates – Display file structure – Display file algorithms – Display control.

Output Primitives: Line-Drawing Algorithms: Simple DDA, Symmetrical DDA and,

Bresenham's Algorithm, Circle generating Algorithms: Properties of circle, Parametric, Trigonometric, Bresenham's and, Midpoint Circle algorithms, Ellipse Generation

Algorithms: Properties of ellipse, Midpoint Ellipse algorithm.

UNIT II: (10 Hours)

Two-Dimensional Geometric Transformations: Basic Transformations: Translation,

Rotation and, Scaling; Matrix representation and Homogeneous coordinates, Composite

Transformations: Translations, Rotations, Scalings, General Pivot-Point Rotation, General Fixed-Point Scaling, Concatenation Properties; Other Transformations: Reflections and shear

Polygons: Introduction-Polygons-An Inside-Outside Tests-Scan-Line Polygon Fill

Algorithm- Boundary Fill Algorithm- Flood Fill algorithm- Fill Area Functions-Character Generation- Ant aliasing.

UNIT III: (10 Hours)

Two Dimensional Viewing: The viewing Pipeline-Viewing Coordinate Reference

Frame- Window to View port transformation – Two Dimensional Viewing Functions-

Line Clipping: The Cohen-Sutherland Outcode algorithm-Liang Barsky Line clipping-Nicholl-Lee-Nicholl, Polygon Clipping: The Sutherlan Hodgman Algorithm –Weiler Atherton Polygon Clipping - Character and Text Clipping .

Three Dimensional Geometric and Modeling Transformations: Translation-

Rotation-Coordinate-Axes Rotations- General Three Dimensional Rotations-scaling-Other

Transformations: Reflections and Shears-Composite Transformations -3D Transformation Functions.

UNIT IV: (10 Hours)

Three Dimensional Viewing: Viewing Pipeline- Viewing Coordinates- Projections:

Parallel Projection and Perspective projection - General Parallel Projection Transformations - General Perspective Projection Transformations – Clipping.

Three Dimensional Object representations: Polygon Surfaces- Curved Lines and

Surfaces- Quadratic Surfaces- Spline Representations - Cubic Spline methods-Bezier Curves and Surfaces- B Spline Curves and Surfaces.

UNIT V: (5 Hours)

Multimedia Fundamentals:

Introduction, Multimedia and Hypermedia, WWW, Multimedia software tools, Multimedia Authoring and Tools, Graphics and image data Represention. Color models in images and video, Fundamental concepts in video, Basics of digital Audio, Raster scanning Principle, MPEG(MPEg-1 and 2), DVI Technology, Multimedia application Toolkit and Hyper application.

Text Book:

1. Donald Hearn and M.Pauline Baker, *Computer Graphics*. 2 ed, PHI.

Reference Books:

1. Steven Harrington, *Computer Graphics – A Programming Approach.* 2 ed, Tata McGraw Hill Co.

2. Zhigang Xiang and Roy A Plastock, *Computer Graphics.* TMH

3. W.M.Newman & RF Sproull, *Principles of Interactive Computer Graphics*. 2 ed, Tata McGraw Hill Co.

4. Foley, Vandam, Feiner and Hughes*, Computer Graphics. 2 ed,* Pearson Education.

Course Outcomes:

1. Understand contemporary graphics principles and graphics hardware.

2. Understand and demonstrate geometrical transformations.

3. Understand and demonstrate 2D image processing techniques.

4. Understand and demonstrate 3D image processing techniques.

5. Understand and demonstrate computer graphics animation.

6. Create interactive graphics applications in C++ using one or more graphics application programming interfaces.

DISCLAIMER

This document does not claim any originally and cannot be used as a substitute for prescribed textbooks. The information presented here is merely a collection of knowledge base by the committee members for their respective teaching assignments. Various online/offline sources as mentioned at the end of the document as well as freely available material from internet were helpful for preparing this document. The ownership of the information lies with the respective authors/institution/publisher. Further, this study material is not intended to be used for commercial purpose and the committee members make no representations or warranties with respect to the accuracy or completeness of the information contents of this document and specially disclaim any implied warranties of merchantability or fitness for a particular purpose. The committee members shall not be liable for any loss or profit or any other commercial damages, including but not limited to special, incidental, consequential, or other damages.

UNIT I

Introduction

Computer graphics concerns the pictorial synthesis of real or imaginary objects from their computer based models. Computer graphics comprises the creation and representation of simple graphical elements and images, as well as modern techniques for rendering a virtual reality.

- Computer graphics generally means creation, storage and manipulation of models and images.

- Such models come from diverse and expanding set of fields including physical, mathematical, artistic, biological, and even conceptual (abstract) structures
- William Fetter coined term “computer graphics” in 1960 to describe new design methods he was pursuing at Boeing. He created a series of widely reproduced images on pen plotter exploring cockpit design, using 3D model of human body.
- Refers to any computer device or program that makes a computer capable of displaying and manipulating pictures.
- Allows communication through pictures, charts and diagrams.
- Computer Graphics is a study of technique to improve communication between human and machine.

Application areas of Computer Graphics

Computer Graphics is used in diverse areas as advertising, entertainment, medicine, education, science, engineering, navigation, etc.

Some computer applications of Computer Graphics are:

Computer Aided Design (CAD)
1. Used to design the buildings, automobiles, aircraft, watercraft, spacecraft, computers, textiles and many other products. 2. CG is a useful tool for generating the architects, drawing and visualizing structures 3. A computer takes the data about building and makes various images of the building from different angles 4. Animations are used in CAD applications 5. Real time animations using wire frame displays on a video monitor are useful for testing performance of vehicles and also to see the interior of the vehicle and to watch the behavior of inner components during motion.
Presentation Graphics
1. Used to produce illustrations for reports 2. Commonly used to summarize financial, statistical, mathematical, scientific and economic data for research reports, managerial reports, consumer information bulletin and other reports. 3. Typical examples are bar charts, line graphs, surface graphs, pie charts and other display showing relationship between multiple parameters. 4. Time charts and task networks layouts are used in project management to schedule and monitor the progress of projects.
Computer Art
1. CG is used in both fine arts and commercial arts applications 2. Artists use various computer methods such as special-purpose hardware, artist’s paintbrush program(Lumena), other paint packages(Pixelpaint, SuperPaint), symbolic mathematics packages(Mathematica), CAD packages, desktop publishing software’s, animation packages that provide facilities for designing object shapes and specifying object motions. 3. Fine artists use other computer technologies to produce images. He uses a combination of three-dimensional modeling packages, texture mapping, drawing programs and CAD software. 4. These methods are also applied in commercial art for logos and other designs, page layouts combining text and graphics, TV advertising spots and other areas. 5. A common graphics method employed in many commercials is morphing, where one object is transformed into another. This method is used in TV commercials to turn oil can into automobile engine, etc.
Entertainment

1. Computer Graphics methods are commonly used in making motion pictures, music videos, television shows and cartoon animation films.
2. Many TV series regularly employ computer graphics methods.
3. Music videos use graphics in several ways. Graphics objects can be combined with live action, or graphics and image processing techniques can be used to produce a transformation of one person or object into another(morphing)

Education and Training

1. Computer generated models like physical systems, financial systems and economic systems used as education aids
2. Models of physical system, physiological system, population trends or equipment, can help trainees to understand the operation of the system.
3. Examples of some specialized systems are the simulators for practice sessions or training of ship captains, aircraft pilots, heavy-equipment operators, and air traffic-control personnel.
4. Various educational pictures with animations are used to present better understanding for learning with animations

Image processing

1. Image processing applies techniques to modify or interpret existing pictures, such as photographs and TV scans.
2. Two principle applications
    (1) Improving picture quality
    (2) Machine perception of visual information used in robotics
3. To digitize the shading and color, interesting, sharpen, improve the contrasting scanning image and to transfer them to monitor or screen or visual display unit.
4. These techniques are used in commercial art applications and to analyze satellite photos of the earth and photos of galaxies.
5. Medical applications also make extensive use of image-processing techniques
    (1) Tomography is a technique of X-ray photograph that allows cross-sectional views of physiological system to be displayed.
    (2) Computerized Axial Tomography (CAT) is used to compose the 3D model of the brain by taking x-ray of it, which can be used to detect problems like brain tumor etc
    (3) Ultrasonic are used to generate digital data.
    (4) Nuclear medicine scanners collect digital data from radiation emitted from ingested radionuclide and plot color-coded images.
    (5) Computer-aided surgery

Graphical User Interface

1. A major component of a graphical interface is a window manager that allows a user a display multiple-window area.
2. Each window can contain a different process that can contain graphical or nongraphical displays.
3. Interfaces also display menus and icons for selection of processing options or parameter values.
    (1) An icon is a graphical symbol that is designed to look like the processing option it represents.
    (2) Menus contain lists of textual descriptions and icons.

The challenge to computer graphics is to make that virtual world look real, sound real, move and respond to interaction in real time, and even feels real.

Overview of Graphics Systems

- Computer graphics is a complex and diversified technology. This can be achieved by considering the end product of Computer Graphics i.e., a picture.
- The picture may be used for a large variety of purposes such as engineering drawing.

- The picture is a fundamental cohesive concept in Computer Graphics. Therefore the following are considered.
    (1) Representing pictures in Computer Graphics
    (2) Preparing pictures for presentation
    (3) Presenting previously prepared pictures
    (4) Interacting with pictures
- Here the picture means any collection of lines, points, text, etc displayed on a graphics device.

Representing picture

- Although many algorithms accept pictures data as polygons or edges, each polygon can in turn be represented by points.
- Points are fundamental building blocks of picture representation.
- For example, the unit square can be represented by its four corner points.

    P_1 (0, 0)

    P_2 (1, 0)

    P_3 (1, 1)

    P_4 (0, 1)

- An associated algorithm description is as

    Connect $P_1P_2P_3P_4P_1$

- The unit square can also be described by its four edges

    $E_1 \equiv P_1P_2$

    $E_2 \equiv P_2P_3$

    $E_3 \equiv P_3P_4$

    $E_4 \equiv P_4P_1$

- Here the algorithm description is as

    Connect $E_1E_2E_3E_4$ in sequence

The fundamental building blocks i.e., points can be represented as either pairs or triplets of numbers, depending on whether the data are two-or-three dimensional space.

- Thus (x_1,y_1) or (x_1,y_1,z_1) represent a point in either two-or three dimensional space.
- Two points represent a line or edge and a collection of three or more points a polygon.
- The representation of curved lines is usually taken by approximating them by connected short straight line segments.
- The representation of textual material is quite complex, involving in many cases curved lines or dot matrices.
- Fundamentally, textual material is again represented by collection of lines and points and an organizing algorithm.

Preparing Pictures

- Pictures ultimately consist of points and a drawing algorithm to display them.
- This information is generally stored in a file before it is used to present the picture; this file is called a database.
- Very complex pictures require very complex databases, which require a complex algorithm to access them.

- These complex databases contain data organized in various ways. E.g.: ring structures, B-tree structure, quad tree structures, etc.
- The database itself contains pointers, substructures and other nongraphic data.
- Many computer graphics application involve much simpler pictures, for which the user can readily invert simple data structure which can be easily accessed, such as linear list. There are quite adequate for many reasonably complex pictures.
- The three fundamental operations for manipulating these points are:
    - Move the beam, pen, cursor, plotting head invisibly to the point.
    - Draw a visible line to a point from an initial point.
    - Display a dot at that point.

  Hence there are two ways to specify the position of a point: absolute coordinates or relative (incremental) coordinates. In relative or incremental coordinates, the position of a point is defined by giving the displacement of the point with respect to the previous point. All computer graphics software is based on these fundamental operations.

Presenting previous Prepared Pictures

- The data used to present the picture is frequently called a displaying file.
- The display file represents some position, view or scene of the picture represented by the total database.
- The displayed picture is usually formed by rotating, translating, scaling and performing various projections on data.
- These basic orientation or viewing preparations are generally performed using a 4x4 transformation matrix, operating on the data represented in homogenous coordinates.
- Hidden line or hidden surface removal, shading, transparency, texture or color effects may be added before final presentation of the picture.
- If the picture represented by the entire database is not required to be presented, the appropriate portion must be selected. This is called clipping, which may be two or three dimensional.
- Two important concepts associated with presenting a picture are windows and viewports.
    - Windowing is the process of extracting a portion of a database by clipping the database to the boundaries of the window.
    - A viewport is an area of the display device on which the window data is presented.

Interacting with Pictures

- Interacting with pictures permits higher user-computer interaction. This significantly enhances the ability to understand data, to perceive trends and to visualize real or imaginary objects.

Video-Display devices

1. Cathode Ray Tubes
2. Raster scan Display
3. Vector scan/Random scan Display
4. Color CRT monitors
5. Direct View Storage Tubes
6. Flat panel Display
7. Three-Dimensional Viewing Devices

Cathode Ray Tube

Typically 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 design. Basic operation is as shown on Cathode Ray Tubes

- A beam of electrons emitted by an electron gun, pass through focusing and deflection systems that direct the beam towards specified positions on the phosphor coated screen

- The phosphor then emits a small spot of light at each position contacted by the electron beam
- The light emitted by phosphorous fades very rapidly to keep glowing is done by redraw the picture repeatedly by quickly directing the electron beam back over the same points is called as “refresh CRT”
- Heated metal cathode and Control grid are the main components of electron gun (figure 1)

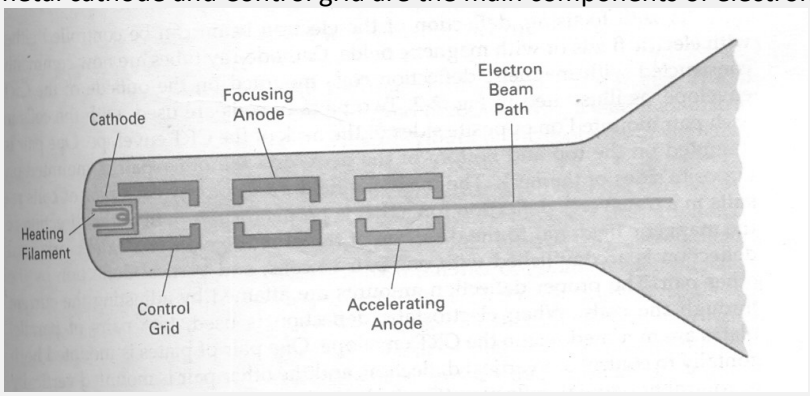


Figure 1: Operation of an electron gun with an accelerating anode.

- The heat is supplied to the cathode through current passing in coil of wire called filament
- This makes electrons to be “boiled off” the hot cathode surface
- The free negatively charged electrons inside CRT are accelerated towards the Phosphorous coated by high positive voltage generated by positively charged metal coating on the inside of CRT
- Intensity of electron beam is controlled by setting voltage levels in the control grid, which is fit over the cathode
- A high negative voltage applied to the control grid will shut off the beam by repelling electrons and stopping them from passing through the small hole at the end of the control grid structure. A smaller negative voltage on the control grid simply decreases the number of electrons passing through.
- Since the amount of light emitted by the phosphor coating depends on the number of electrons striking the screen, the brightness of a display is controlled by varying the voltage on the control grid. The intensity level is specified for individual screen positions with graphics software commands.
- A beam of electrons emitted by an electron gun, pass through focusing and deflection systems that direct the beam towards specified positions on the phosphor-coated screen. The phosphor then emits a small spot of light at each position contacted by the electron beam. The light emitted by phosphorous fades very rapidly to keep glowing is done by redraw the picture repeatedly by quickly directing the electron beam back over the same points is called as “refresh CRT”.
- The focusing system in a CRT is needed to force the electron beam to converge into a small spot as it strikes the phosphor. Otherwise, the electrons would repel each other, and the beam would spread out as it approaches the screen. Focusing is performed with either electric or magnetic fields.

Focusing	Meaning
Electro static focusing	With electro static focusing, the beam pass through positively charged metal cylinder that forms electrostatic and it focus the beam at the centre of the screen. (as shown in figure 2)
Magnetic focusing	With magnetic field set up by a coil mounted around the outside of the CRT envelope. (as shown in figure 3)

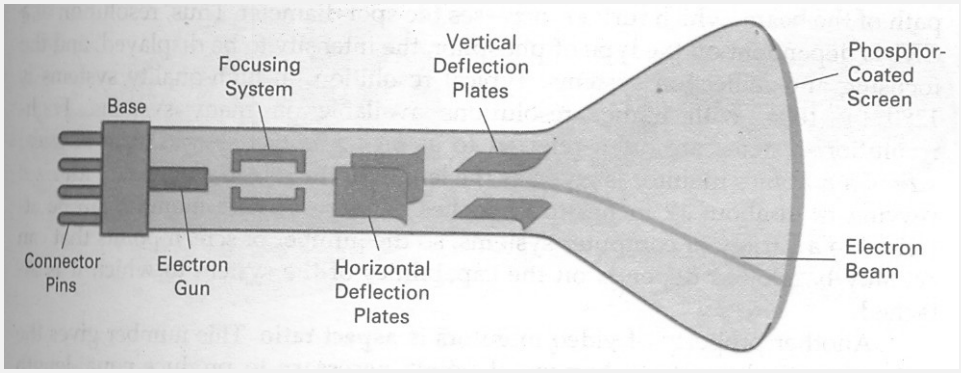


Figure 2: Electrostatic deflection of the electron beam in a CRT.

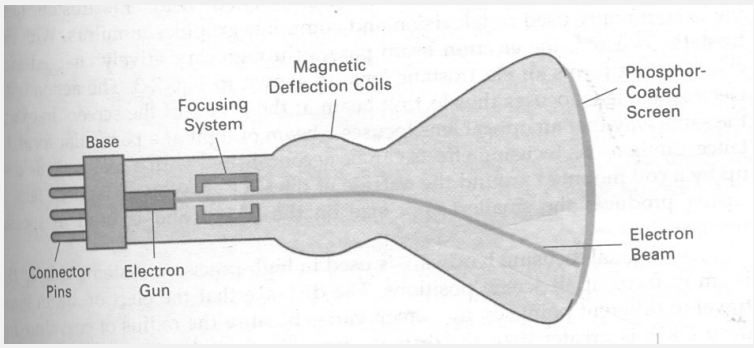


Figure 3: Basic design of a magnetic deflection CRT

Different kinds of phosphors are available for use in a CRT. The properties with which the phosphors vary are:

Term	Description
Persistence	Persistence is defined as the time it takes the emitted light from the screen to decay to one tenth of its original intensity.
Resolution	The maximum number of points displayed on the CRT screen or the number of points/centimeter that can be plotted horizontally and vertically.
Aspect ratio	The ratio of vertical points to horizontal points necessary to produce equal length line in both directions on the screen.

Raster-Scan Display

- The most common type of graphics monitor employing a CRT is the Raster-scan displays, based on television technology
- JPG images are raster based
- Light occurs when an electron beam stimulates a phosphor.

- In Raster scan, the electron beam from electron gun is swept horizontally across the phosphor one row at time from top to bottom.(figure 4)

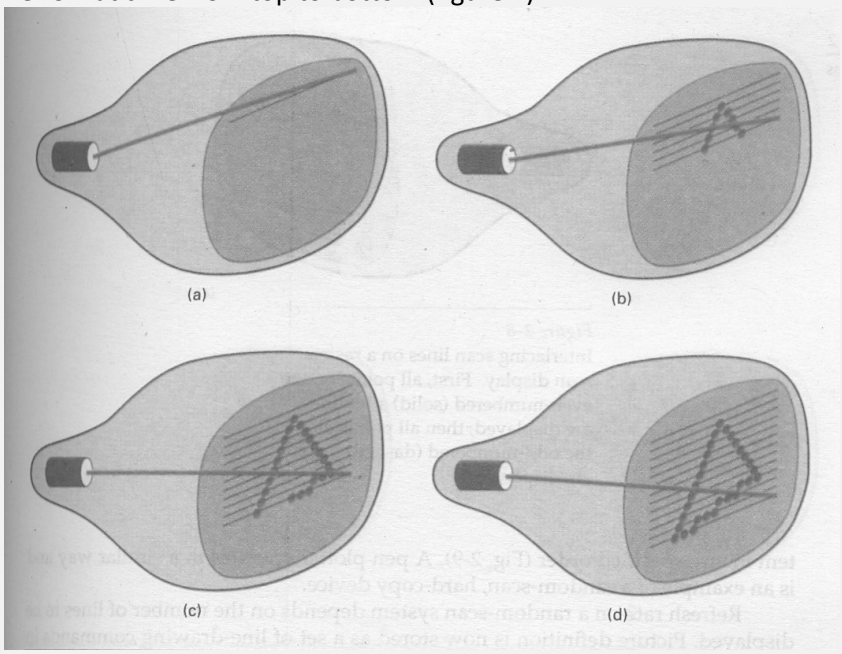


Figure 4: A raster-scan system displays an object as a set of discrete points across

Each scan line.

- After each horizontal sweep the beam is moved.
- After the bottom line is swept, the beam returns to the top and the sweep process begins again.
- As the 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 area called the refresh buffer or frame buffer
- Each screen point is called as “pixel”
- This memory area holds the set of intensity values for all the screen points
- This is part of the system memory
- The stored intensity values are then retrieved from frame buffer and painted on the screen one row at a time
- Intensity range for pixel position depends on capability of the raster system
- In black and white system, the point on screen is either on or off
- Only one bit is needed to control the intensity of the screen
- In case of color systems, 2 bits are required
- One to represent ON (1), another one is OFF (0).
- Refreshing on raster scan is carried out at the rate of 60 to 80 frames per seconds

Term	Description
Bit map	On a black and white system with one bit per pixel, the frame buffer is called bit map.
Pix map	The frame buffer with multiple bits per pixel (for color display). Refreshing of raster scan displays done by rate of 60 to 80 frames per second. The units used sometimes are Hertz or cycles/second.
Horizontal retrace	Horizontal retrace of the electron beam means the return to the left of the screen. After refreshing each Sean line.
Vertical retrace	If it returns to the top left corner of the screen to begin in the next frame called “vertical retrace”.

- On some raster-scan systems (and in 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. Then after the vertical retrace, the beam sweeps out the remaining scan lines (Figure 5).

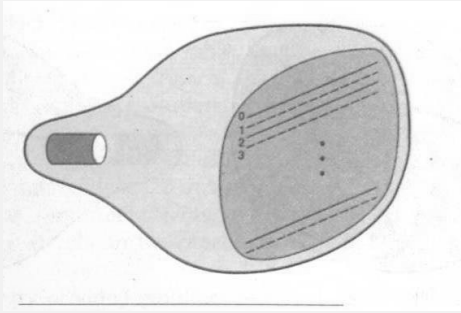


Figure 5: Interlacing scan lines on a raster scan display. First, all points on the even-numbered (solid) scan lines are displayed; and then all points along the odd-numbered (dashed) lines are displayed.

- 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.
- Interlacing is primarily used with slower refreshing rates. On an older, 30 frame per-second, noninterlaced display, for instance, some flicker is noticeable. But with interlacing, each of the two passes can be accomplished in 1/60th of a second, which brings the refresh rate nearer to 60 frames per second. This is an effective technique for avoiding flicker, providing that adjacent scan lines contain similar display information.
- Frame buffer size = to store bits per pixel* resolution

Advantages

- High degree realism is achieved in picture with the aid of advanced shading and hidden surface technique.
- Decreasing memory costs have made raster systems popular.
- Computer monitors and TVs use this method

Disadvantages

- Raster displays have less resolution.
- The lines produced are ziz-zag as the plotted values are discrete.

Random-Scan Display

In a Random scan system, also called vector, stoke writing, or calligraphic the electron beam directly draws the picture.

- A pen plotter operates in a similar way and is an example of a random-scan, hard-copy device.
- When operated as a random-scan display unit, a CRT has the electron beam directed only to the parts of the screen where a picture is to be drawn.
- Random scan monitors draw a picture one line at a time and for this reason are also referred to as vector displays (or stroke-writing or calligraphic displays).
- Here the electron gun of a CRT illuminates points and / or straight lines in any order.
- Refresh rate on a random-scan system depends on the number of lines to be displayed.
- Picture definition stored as a set of line drawing commands in an area of memory called “refresh display file” or also called as display list or display program or refresh buffer
- This displays to draw all the component lines of picture 30 to 60 frames/second
- This system is designed for line drawing applications
- Vector displays produces smooth line drawings but raster produces jagged lines that are plotted points

- To display a given 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 drawing command in the list
- Random scan suitable for applications like engineering and scientific drawings
- Graphics patterns are displayed by directing the electron beam along the component lines of the picture
- A scene is then drawn one line at a time by positioning the beam to fill in the line between specified end points

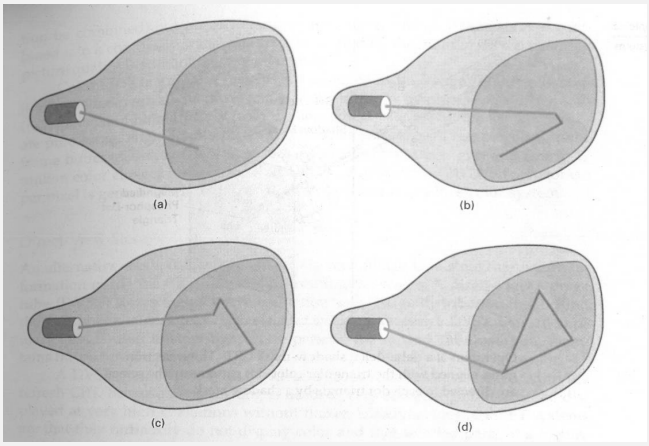


Figure 6: A random-scan system draws the component lines of an object in any order specified.

Advantages

- Very high resolution, limited only by monitor.
- Easy animation, just draw at different position.
- Requires little memory (just enough to hold the display program).

Disadvantages

- Requires intelligent electron beam, i.e., processor controlled.
- Limited screen density before have flicker, can't draw a complex image.
- Limited color capability (very expensive).

Color CRT Monitors

- Colored pictures can be displayed by using a combination of phosphorous that emit different colored light
- By combining the emitted light from the different phosphors, a range of colors can be generated.
- The two basic techniques for producing color displays with a CRT are the beam-penetration method and the shadow-mask method.

Beam-penetration method

- The beam-penetration method for displaying color pictures has been used with random-scan monitors.
- Two layers of phosphor, usually red and green, are coated onto the inside of the CRT screen, and the displayed color depends on how far the electron beam penetrates into the phosphor layers.
- A beam of slow electrons excites only the outer red layer.

- A beam of very fast electrons penetrates through the red layer and excites the inner green layer. At intermediate beam speeds, combinations of red and green light are emitted to show two additional colors, orange and yellow.
- The speed of the electrons, and hence the screen color at any point, is controlled by the beam-acceleration voltage.
- Beam penetration has been an inexpensive way to produce color in random-scan monitors, but only four colors are possible, and the quality of pictures is not as good as with other methods.

Shadow-mask methods

- Shadow-mask methods are commonly used in raster scan systems (including color TV) because they produce a much wider range of colors than the beam penetration method.
- A shadow-mask CRT has 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.

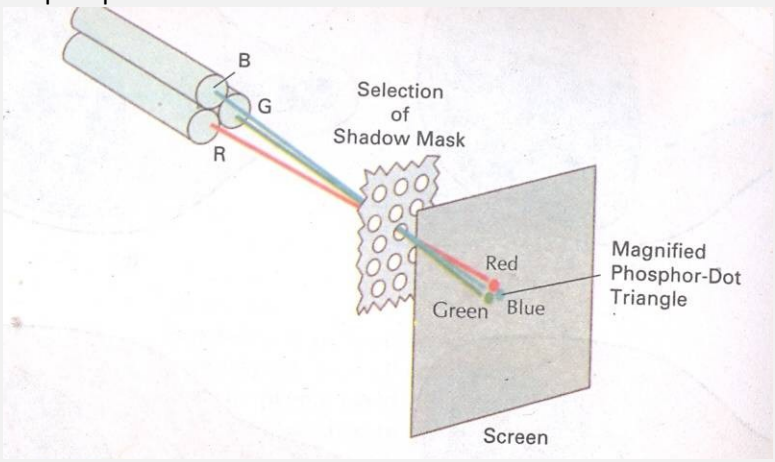


Figure 7: Operation of a delta-delta, shadow-mask CRT. Three electrons

Guns, aligned with the triangular color dot patterns on the screen,

Are directed to each dot triangle by a shadow mask.

- Figure 7 illustrates the delta-delta shadow-mask method, commonly used in color CRT systems.
    - The three electron beams are deflected and focused as a group onto the shadow mask, which contains a series of holes 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.
- This in-line arrangement of electron guns is easier to keep in alignment and is commonly used in high-resolution color CRTs.
- Color variations are obtained in a shadow-mask CRT by varying the intensity levels of the three electron beams. By turning off the red and green guns, we get only the color coming from the blue phosphor.
- Other combinations of beam intensities produce a small light spot for each pixel position, since our eyes tend to merge the three colors into one composite.
- The color we see depends on the amount of excitation of the red, green, and blue phosphors. A white (or gray) area is the result of activating all three dots with equal

intensity. Yellow is produced with the green and red dots only, magenta is produced with the blue and red dots, and cyan shows up when blue and green are activated equally.
- In some low-cost systems, the electron beam can only be set to on or off, limiting displays to eight colors. More sophisticated systems can set intermediate intensity levels for the electron beams, allowing several million different colors to be generated.

Color graphics systems can be designed to be used with several types of CRT display devices. Some inexpensive home-computer systems and video games are designed for use with a color TV set and an RF (radio-frequency) modulator.

Composite monitors are adaptations of TV sets that allow bypass of the broadcast circuitry. These display devices still require that the picture information be combined, but no carrier signal is needed. Picture information is combined into a composite signal and then separated by the monitor, so the resulting Video Display Devices picture quality is still not the best attainable.

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 (red, green, and blue) directly from the computer system without any intermediate processing. High-quality raster-graphics systems have 24 bits per pixel in the frame buffer, allowing 256 voltage settings for each electron gun and nearly 17 million color choices for each pixel. 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.

Direct-View Storage Tubes

- A direct-view Storage Tube (DVST) stores the picture information as a charge distribution just behind the phosphor-coated screen.
- Two electron guns are used in a DVST.
  - One, the primary gun, is used to store the picture pattern;
  - the second, the flood gun, maintains the picture display.
- An alternative method for maintaining a screen image is to store the picture information inside the CRT instead of refreshing the screen.

A DVST monitor has both disadvantages and advantages compared to the refresh CRT.

Advantages

- It has a flat screen.
- Refreshing of screen is not required.
- Because no refreshing is needed, very complex pictures can be displayed at very high resolutions without flicker.

Disadvantages

- This has poor contrast.
- Performance is inferior to the refresh CRT.
- Selective or part erasing of screen is not possible.
- They ordinarily do not display color and that selected parts of a picture cannot he erased. To eliminate a picture section, the entire screen must be erased and the modified picture redrawn. The erasing and redrawing process can take several seconds for a complex picture. For these reasons, storage displays have been largely replaced by raster systems.

Flat-Panel Displays

- The term Flat-panel display refers to a class of video devices that have reduced volume, weight, and power requirements compared to a CRT.
- 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.

- Current uses for flat-panel displays include small TV monitors, calculators, pocket video games, laptop computers, armrest viewing of movies on airlines, as advertisement boards in elevators, and as graphics displays in applications requiring rugged, portable monitors.
- Flat-panel displays are of two categories:
    - Emissive displays e.g. Plasma panel, LEDs etc
    - Nonemmissive displays. e.g. LCD

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 arts accelerated parallel to the screen, then deflected 90' to the screen. But flat CRTs have not proved to be as successful as other emissive devices.

Nonemmissive displays (or nonemitters) use optical effects to convert sunlight or light from some other source into graphics patterns. The most important example of a Nonemmissive 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 ribbons is built into the other glass panel (Figure 8).

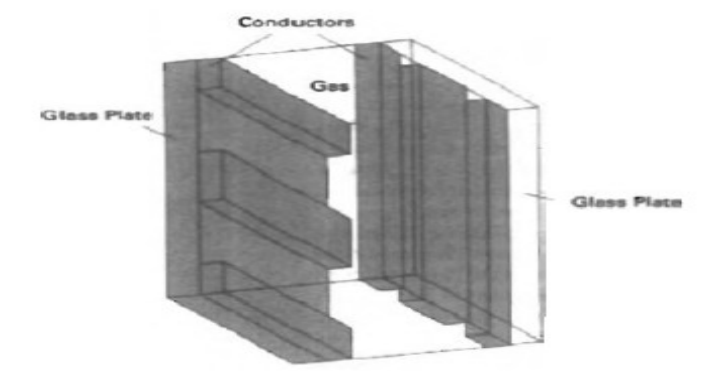


Figure 8: Basic design of a plasma-panel display device.

- Firing voltages applied to a pair of horizontal and vertical conductors cause the gas at the intersection of the two conductors to break down into 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 have been developed that are now capable of displaying color and grayscale.
- Thin-film electroluminescent displays are similar in construction to a plasma panel.
- 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 (Figure 9).

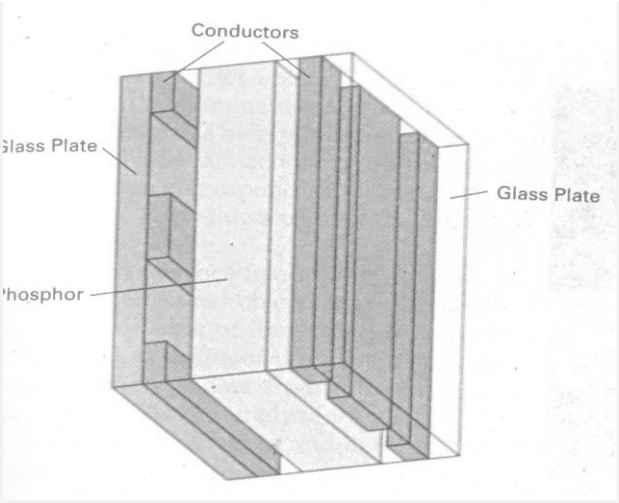


Figure 9: Basic design of a thin-film electroluminescent display device.

- 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 then 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 and gray scale displays are hard to achieve.
- 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.
- As in 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.
- LCDs are commonly used in small systems, such as calculators and portable, laptop computers.
- These Nonemmissive 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.

Three-Dimensional Viewing Devices

- Graphics monitors for the display of three-dimensional scenes have been devised using a technique that reflects a CRT image from a vibrating, flexible mirror.
- As the varifocal mirror vibrates, it changes focal length. These vibrations are synchronized with the display of an object on a CRT so that each point on the object is reflected from the mirror into a spatial position corresponding to the distance of that point from a specified viewing position. This allows us to walk around an object or scene and view it from different sides.
- Such systems have been used in medical applications to analyze data from ultrasonography and CAT scan devices, in geological applications to analyze topological and seismic data, in design applications involving solid objects, and in three-dimensional simulations of systems, such as molecules and terrain.
- Another technique for representing three-dimensional objects is displaying stereoscopic views. This method does not produce true three-dimensional images, but it does provide a three-dimensional effect by presenting a different view to each eye of an observer so that scenes do appear to have depth.

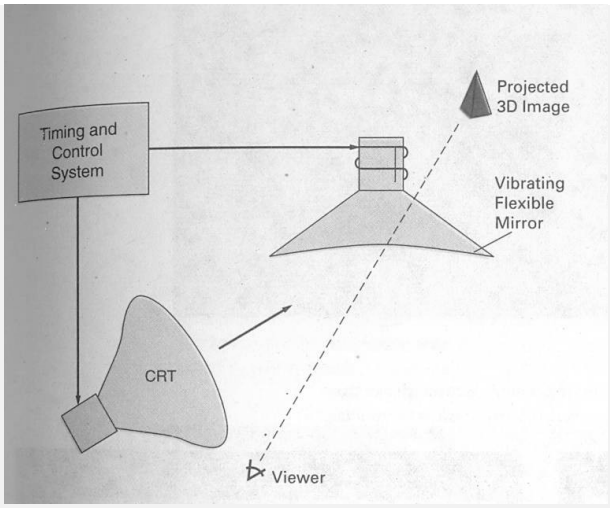


Figure 10: Operation of a three-dimensional display system using a vibrating mirror that changes focal length to match the depth of points in a scene.

- To obtain a stereoscopic projection, we first need to obtain two views of a scene generated from a viewing direction corresponding to each eye (left and right). We can construct the two views as computer-generated scenes with different viewing positions, or we can use a stem camera pair to photograph some object or scene. When we simultaneous look at the left view with the left eye and the right view with the right eye, the two views merge into a single image and we perceive a scene with depth.
- Stereoscopic viewing is also a component in virtual-reality systems, where users can step into a scene and interact with the environment. A headset containing an optical system to generate the stereoscopic views is commonly used in conjunction with interactive input devices to locate and manipulate objects in the scene. A sensing system in the headset keeps track of the viewer's position, so that the front and back of objects can be seen as the viewer "walks through" and interacts with the display.
- An interactive virtual-reality environment can also be viewed with stereoscopic glasses and a video monitor, instead of a headset. This provides a means for obtaining a lower cost virtual-reality system.

Raster-Scan Systems

- Raster scan or raster scanning: is the pattern of image detection and reconstruction in television, and is the pattern of image storage and transmission used in most computer image systems.
- Interactive raster graphics systems 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 in Figure 11

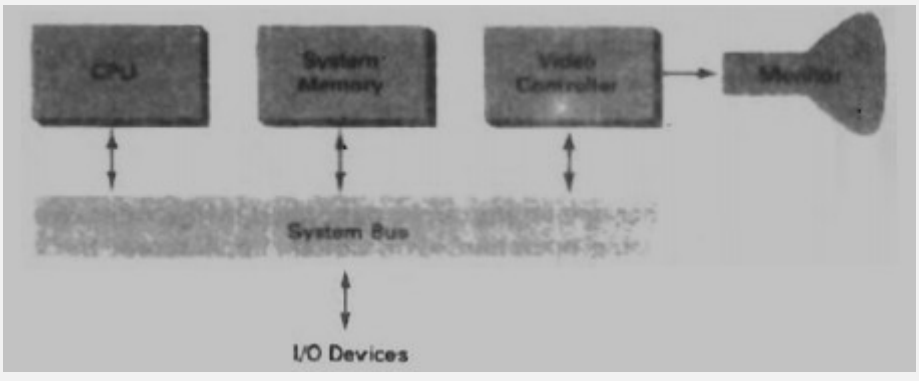


Figure 11: Architecture of a simple raster graphics system.

- 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 co- processors and accelerators to implement various graphics operations.

Video Controller

Figure 12 shows a commonly used organization for raster systems.

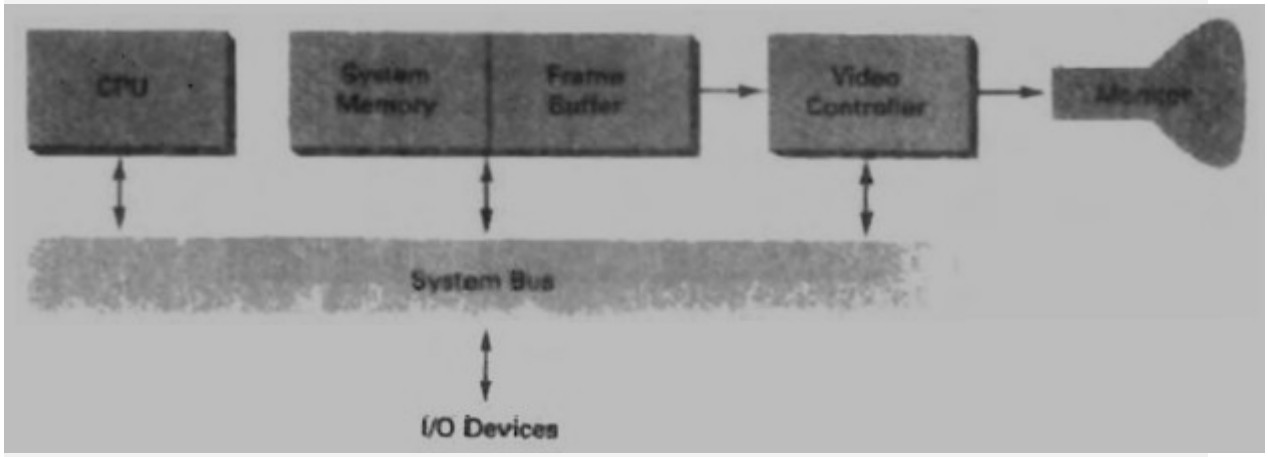


Figure 12: Architecture of a raster system with a fixed portion of the system memory reserved for the frame buffer.

- 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 Cartesian coordinates.
- For many graphics monitors, the coordinate origin is defined at the lower left screen corner (Figure13).

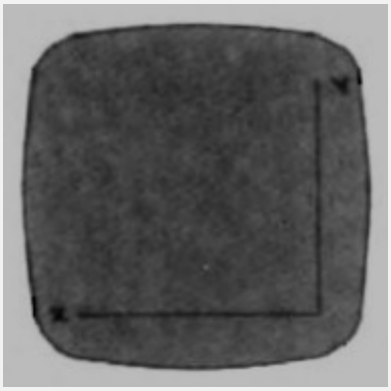


Figure 13: The origin of the coordinate system for identifying screen positions is usually specified in the lower-left corner.

- The screen surface is then represented as the first quadrant of a two-dimensional system, with positive x values increasing to the right and positive y values increasing from bottom to top. (On some personal computers, the coordinate origin is referenced at the upper left corner of the screen, so the y values are inverted.)
- Scan lines are then labeled from y_{max}, at the top of the screen to 0 at the bottom. Along each scan line, screen pixel positions are labeled from 0 to x_{max}.
- In Figure 14, the basic refresh operations of the video controller are diagrammed. Two registers are used to store the coordinates of the screen pixels.

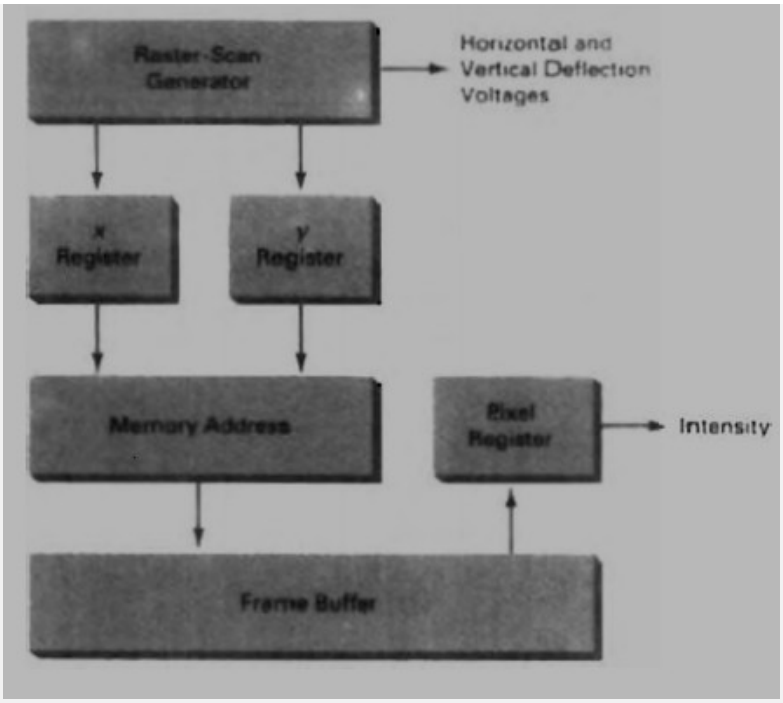


Figure 14: Basic video-controller refresh operations.

- Initially, the x register is set to 0 and the y register is set to y,.
- The value stored in the frame buffer for this pixel position is then retrieved and used to set the intensity of the CRT beam.
- Then the x register is incremented by 1, and the process repeated for the next pixel on the top scan line.
- This procedure is repeated for each pixel along the scan line.
- After the last pixel on the top scan line has been processed, the x register is reset to 0 and the y register is decremented by 1.
- Pixels along this scan line are then processed in turn, and the procedure is repeated for each successive scan line.
- After cycling through all pixels along the bottom scan line (y = O), the video controller resets the registers to the first pixel position on the top scan line and the refresh process starts over.
- Since the screen must be refreshed at the rate of 60 frames per second, the simple procedure illustrated in above figure cannot be accommodated by typical RAM chips. The cycle time is too slow. To speed up pixel processing, video controllers can retrieve multiple pixel values from the refresh buffer on each pass. The multiple pixel intensities are then stored in a separate register and used to control the CRT beam intensity for a group of adjacent pixels. When that group of pixels has been processed, the next block of pixel values is retrieved from the frame buffer.
- A number of other operations can be performed by the video controller, besides the basic refreshing operations.
- For various applications, the video controller can retrieve pixel intensities from different memory areas on different refresh cycles.
- In high-quality systems, for example, two frame buffers are often provided so that one buffer can be used for refreshing while the other is being filled with intensity values. Then the two buffers can switch roles. This provides a fast mechanism-for generating real-time animations, since different views of moving objects can be successively loaded into the refresh buffers.
- Also, some transformations can be accomplished by the video controller. Areas of the screen can be enlarged, reduced, or moved from one location to another during the refresh cycles.
- In addition, the video controller often contains a lookup table, so that pixel values in the frame buffer are used to access the lookup table instead of controlling the CRT beam intensity directly. This provides a fast method for changing screen intensity values.

- Finally, some systems are designed to allow the video controller to mix the frame-buffer image with an input image from a television camera or other input device.

Raster-Scan Display Processor

- Figure 15 shows one way to set up the organization of a raster system containing a separate display processor, sometimes referred to as a graphics controller or a display coprocessor.

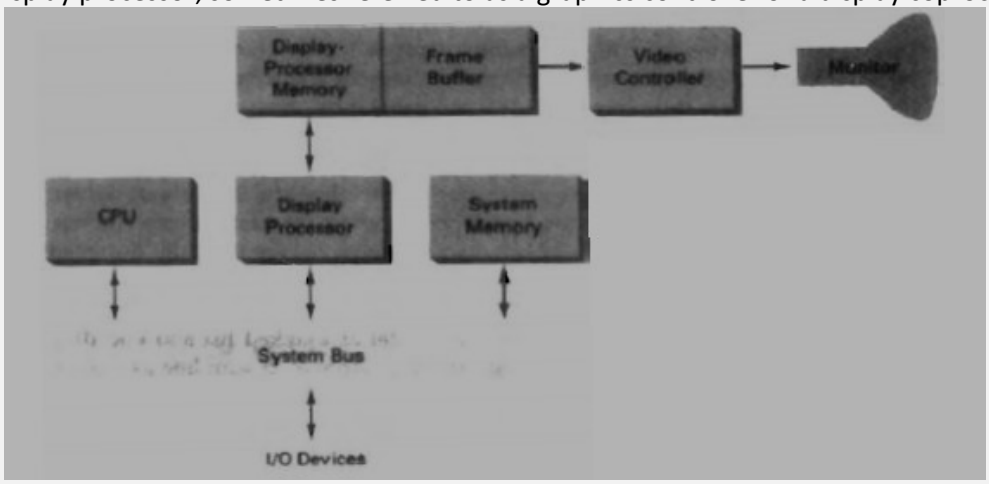


Figure 15: Architecture of a raster-graphics system with a display processor.

- 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 also be provided.
- A major task of the display processor is digitizing a picture definition given in an application program into a set of pixel-intensity values for storage in the frame buffer.
- This digitization process is called scan conversion.
- Graphics commands specifying straight lines and other geometric objects are scan converted into a set of discrete intensity points.
- Scan converting a straight-line segment, for example, means that we have to locate the pixel positions closest to the line path and store the intensity for each position in the frame buffer.
- Similar methods are used for scan converting curved lines and polygon outlines.
- Characters can be defined with rectangular grids, as in Fig. 2-30, or they can be defined with curved outlines, as in Fig. 2-31.

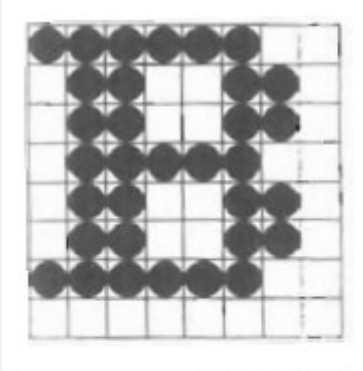


Figure 2-30
A character defined as a rectangular grid of pixel positions.

Figure 2-31
A character defined as a curve outline.

- The array size for character grids can vary from about 5 by 7 to 9 by 12 or more for higher quality displays.
- A character grid is displayed by superimposing the rectangular grid pattern into the frame buffer at a specified coordinate position.
- With characters that are defined as curve outlines, character shapes are scan converted into the frame buffer.
- Display processors are also designed to perform a number of additional operations.
- These functions include generating various line styles (dashed, dotted, or solid), displaying colour areas, and performing certain transformations and manipulations on displayed objects.
- 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 intensity information.
- One way to do this is to store each scan line as a set of integer pairs.
- One number of each pair indicates an intensity value, and the second number specifies the number of adjacent pixels on the scan line that are to have that intensity.
- This technique, called run-length encoding, can result in a considerable saving in storage space if a picture is to be constructed mostly with long runs of a single colour each.
- A similar approach can be taken when pixel intensities change linearly.
- Another approach is to encode the raster as a set of rectangular areas (cell encoding).
- The disadvantages of encoding runs are that intensity changes are difficult to make and storage requirements actually increase as the length of the runs decreases.
- In addition, it is difficult for the display controller to process the raster when many short runs are involved.

Random-Scan Systems

- The organization of a simple random-scan (vector) system is shown in Figure 18

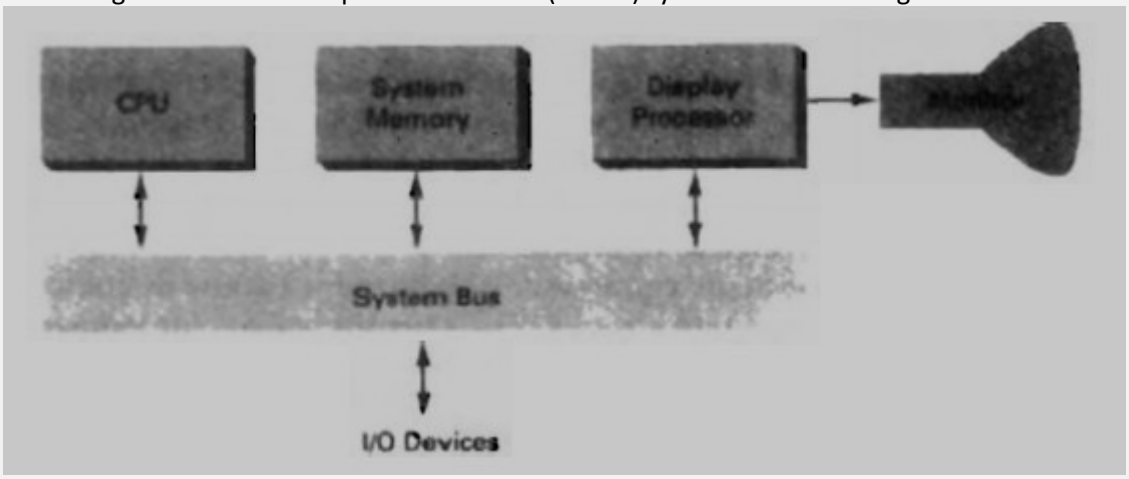


Figure 2-18: Architecture of a simple random scan system

- An application program is input and stored in the system memory along with a graphics package.
- Graphics commands in the application program are translated by the graphics package into a display file stored in the system memory.
- This display file is then accessed by the display processor to refresh the screen.
- The display processor cycles through each command in the display file program once during every refresh cycle.
- Sometimes the display processor in a random-scan system is referred to as a display processing unit or a graphics controller.
- Graphics patterns are drawn on a random-scan system by directing the electron beam along the component lines of the picture.

- Lines are defined by the values for their coordinate endpoints, and these input coordinate values are converted to x and y deflection voltages.
- A scene is then drawn one line at a time by positioning the beam to fill in the line between specified endpoints.

Graphics monitors and work stations

- Most graphics monitors operate as raster-scan displays.
- Graphics systems range from small general-purpose computer systems with graphics capabilities to sophisticated full-color systems that are designed specifically for graphics applications.
- A typical screen resolution for personal computer systems, such as the Apple Quadra is 640 by 480, although screen resolution and other system capabilities vary depending on the size and cost of the system.
- Diagonal screen dimensions for general-purpose personal computer systems can range from 12 to 21 inches, and allowable color selections range from 16 to over 32,000.
- For workstations specifically designed for graphics applications, typical screen resolution is 1280 by 1024, with a screen diagonal of 16 inches or more.
- Graphics workstations can be configured with from 8 to 24 bits per pixel (full-color systems), with higher screen resolutions, faster processors, and other options available in high-end systems.
- A high-definition graphics monitor used in applications such as air traffic control, simulation, medical imaging, and CAD.
- This system has a diagonal screen size of 27 inches, resolutions ranging from 2048 by 1536 to 2560 by 2048, with refresh rates of 80 Hz or 60 Hz noninterlaced.
- A multiscreen system called the Media Wall, provides a large "wall-sized display area. This system is designed for applications that require large area displays in brightly lighted environments, such as at trade shows, conventions, retail stores, museums, or passenger terminals.
- Media Wall operates by splitting images into a number of Sections and distributing the sections over an array of monitors or projectors using a graphics adapter and satellite control units.
- An array of up to 5 by 5 monitors, each with a resolution of 640 by 480, can be used in the Media Wall to provide an overall resolution of 3200 by 2400 for either static scenes or animations.
- Scenes can be displayed behind mullions, or the mullions can be eliminated to display a continuous picture with no breaks between the various sections.
- Many graphics workstations are configured with two monitors. One monitor can be used to show all features of an object or scene, while the second monitor displays the detail in some part of the picture. Another use for dual-monitor systems is to view a picture on one monitor and display graphics options (menus) for manipulating the picture components on the other monitor.

Input devices

An input device is that is used to interact with or provide data to the computer. Various devices are available for data input on graphics workstations. The most common input devices are the mouse and keyboard. However, additional devices like trackball, spaceball, joysticks, digitizers, button boxes are specially designed for interactive input. Some other input devices used in particular applications are data gloves, touch panels, image scanners and voice systems.

KEYBOARD

1. An alphanumeric keyboard on a graphics system is used primarily as a device for entering text strings
2. Keyboard is an efficient device for inputting such as non graphic data as picture labels associated with a graphic display

3. Cursor keys and function keys are common features on general purpose keyboards
4. Function keys allow users to enter frequently used operations in a single keystroke
5. Cursor control keys can be used to select displayed objects or coordinate positions by positioning the screen cursor
6. Other types of cursor pointing devices such as track ball or joystick are included on some keyboards
7. A numeric keypad is often included on the keyboard for fast entry of numeric data
8. For specialized applications, input to a graphics application may come from a set of buttons, dials or switches that select data values or customized graphics operations
9. Buttons and switches are often used to input predefined functions and dials are common devices for entering scalar values
10. Real numbers within some defined range are selected for input with dial rotations
11. Potentiometers are used to measure dial rotations, which are then converted to deflection voltages for cursor movement

MOUSE

1. A mouse is small hand-held box used to position the screen cursor wheels or rollers on the bottom of the mouse can be used to record the amount and direction of movement
2. Another method for detecting mouse motion is with an optical sensor
3. For these systems, the mouse is moved over a special mouse pad that has a grid of horizontal and vertical lines
4. The optical sensor detects movement across the lines in the grid
5. Since a mouse can be picked up and put down at another position without change in cursor movement, it is used for making relative changes in the position of the screen cursor
6. One, two or three buttons are usually included on the top of the mouse for signaling the execution of some operation such as recording cursor position or invoking a function
7. Additional devices can be included in the basic mouse design to increase the number of allowable input parameters
8. The z mouse includes 3 buttons, a thumb wheel on the side, a track ball on the top and a standard mouse ball underneath
9. With z mouse, we can pick up an object, rotate it and move it in any direction or we can navigate our viewing position and orientation through a 3-D scene
10. Applications of z-mouse include virtual reality, CAD and animation

Track ball and Space ball

1. A track ball is a ball that can be rotated with fingers or palm of the hand to produce screen cursor movement
2. Potentiometers, attached to the ball measure the amount and direction of rotation
3. Track balls are often mounted on keyboards or other devices such as the z-mouse
4. While a track ball is a 2-D positioning device, a space ball provides six degrees of freedom
5. Unlike the track ball, space ball does not actually move strain gauges measure the amount of pressure applied to the space ball to provide input for spatial positioning and orientation as the ball is pushed or pulled in various directions
6. Space balls are used for 3-D positioning and selection operations in virtual reality systems, modeling, animation, CAD and other applications.

Joystick

1. A Joystick has a small, vertical lever (called the stick) mounted on the base and used to steer the screen cursor around
2. It consists of two potentiometers attached to a single lever
3. Moving the lever changes the settings on the potentiometers
4. The left or right movement is indicated by one potentiometer and forward or back movement is indicated by other potentiometer

Data Glove

1. The data glove is used to grasp a virtual object
2. It is constructed with a series of sensors that detect hand and finger motions
3. Each sensor is a short length of f fiber optic cable, with a light-emitting diode at one end and a phototransistor at the other end
4. The surface of a cable is roughened in the area where it is to be sensitive to bending
5. When the cable is fixed, some of the LED's light is lost, so less light is received by the phototransistor

Digitizers

1. A common device for drawing, painting, or interactively selecting coordinate positions on an object is a digitizer.
2. These devices can be used to input coordinate values in either a two-dimensional or a three-dimensional space.
3. Typically, a digitizer is used to scan over a drawing or object and to input a set of discrete coordinate positions, which can be joined with straight-line segments to approximate the curve or surface shapes.
4. One type of digitizer is the graphics tablet (also referred to as a data tablet), which is used to input two-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, while a stylus is a pencil-shaped device that is pointed at positions on the tablet.
5. The artist's digitizing system uses electromagnetic resonance to detect the three-dimensional position of the stylus. This allows an artist to produce different brush strokes with different pressures on the tablet surface. Tablet size varies from 12 by 12 inches for desktop models to 44 by 60 inches or larger for floor models.
6. Graphics tablets provide a highly accurate method for selecting coordinate positions, with an accuracy that varies from about 0.2 mm on desktop models to about 0.05 mm or less on larger models.
7. Many graphics tablets are constructed with a rectangular grid of wires embedded in the tablet surface.
8. Electromagnetic pulses are generated in sequence along the wires, and an electric signal is induced in a wire coil in an activated stylus or hand cursor to record a tablet position.
9. Depending on the technology, signal strength, coded pulses, or phase shifts can be used to determine the position on the tablet.
10. Acoustic (or sonic) tablets use sound waves to detect a stylus position. Either strip microphones or point microphones can be used to detect the sound emitted by an electrical spark from a stylus tip.

Image Scanners

1. Drawings, graphs, color and black-and-white photos, or text can be stored for computer processing with an image scanner by passing an optical scanning mechanism over the information to be stored.
2. The gradations of gray scale or color are then recorded and stored in an array.
3. Once we have the internal representation of a picture, we can apply transformations to rotate, scale, or crop the picture to a particular screen area. We can also apply various image-processing methods to modify the array representation of the picture. For scanned text input, various editing operations can be performed on the stored documents.
4. Some scanners are able to scan either graphical representations or text, and they come in a variety of sizes and capabilities.

Touch Panels

1. Touch panels allow displayed objects or screen positions to be selected with the touch of a finger.
2. A typical application of touch panels is for the selection of processing options that are represented with graphical icons.
3. Some systems, such as the plasma panels, are designed with touch screens.
4. Other systems can be adapted for touch input by fitting a transparent device with a touch

sensing mechanism over the video monitor screen.

5. Touch input can be recorded using optical, electrical, or acoustical methods.
6. Optical touch panels employ a line of infrared light-emitting diodes (LEDs) along one vertical edge and along one horizontal edge of the frame.
7. The opposite vertical and horizontal edges contain light detectors. These detectors are used to record which beams are interrupted when the panel is touched.
8. The two crossing beams that are interrupted identify the horizontal and vertical coordinates of the screen position selected. Positions can be selected with an accuracy of about ¼ inch
9. With closely spaced LEDs, it is possible to break two horizontal or two vertical beams simultaneously. In this case, an average position between the two interrupted beams is recorded.
10. The LEDs operate at infrared frequencies, so that the light is not visible to a user.
11. An electrical touch panel is constructed with two transparent plates separated by a small distance. One of the plates is coated with a conducting material, and the other plate is coated with a resistive material. When the outer plate is touched, it is forced into contact with the inner plate. This contact creates a voltage drop across the resistive plate that is converted to the coordinate values of the selected screen position.
12. In acoustical touch panels, high-frequency sound waves are generated in the horizontal and vertical directions across a glass plate.
13. Touching the screen causes part of each wave to be reflected from the finger to the emitters.
14. The screen position at the point of contact is calculated from a measurement of the time interval between the transmission of each wave and its reflection to the emitter.

Light Pens

1. Such pencil-shaped devices are used to select screen positions by detecting the light coming from points on the CRT screen.
2. They are sensitive to the short burst of light emitted from the phosphor coating at the instant the electron beam strikes a particular point.
3. Other Light sources, such as the background light in the room, are usually not detected by a light pen.
4. An activated light pen, pointed at a spot on the screen as the electron beam lights up that spot, generates an electrical pulse that causes the coordinate position of the electron beam to be recorded.
5. As with cursor-positioning devices, recorded Light-pen coordinates can be used to position an object or to select a processing option.
6. Disadvantages:
    - When a light pen is pointed at the screen, part of the screen image is obscured by the hand and pen.
    - Prolonged use of the light pen can cause arm fatigue.
    - Light pens require special implementations for some applications because they cannot detect positions within black areas.
    - To be able to select positions in any screen area with a light pen, we must have some nonzero intensity assigned to each screen pixel.
    - Light pens sometimes give false readings due to background lighting in a room.

Voice Systems

1. Speech recognizers are used in some graphics workstations as input devices to accept voice commands
2. The voice-system input can be used to initiate graphics operations or to enter data.
3. These systems operate by matching an input against a predefined dictionary of words and phrases.
4. A dictionary is set up for a particular operator by having, the operator speak the command words to be used into the system.
5. Each word is spoken several times, and the system analyzes the word and establishes a frequency pattern for that word in the dictionary along with the corresponding function to be performed.

6. Later, when a voice command is given, the system searches the dictionary for a frequency-pattern match.
7. Voice input is typically spoken into a microphone mounted on a headset
8. The microphone is designed to minimize input of other background sounds.
9. If a different operator is to use the system, the dictionary must be reestablished with that operator's voice patterns.
10. Voice systems have some advantage over other input devices, since the attention of the operator does not have to be switched from one device to another to enter a command.

Output primitives

- A picture can be described in several ways.
  - In a raster display, a picture is completely specified by the set of intensities for the pixel positions in the display.
  - At the other extreme, a picture can be described as a set of complex objects, such as trees and terrain or furniture and walls, positioned at specified coordinate locations within the scene.
- Shapes and colors of the objects can be described internally with pixel arrays or with sets of basic geometric structures, such as straight line segments and polygon color areas.
- The scene is then displayed either by loading the pixel arrays into the frame buffer or by scan converting the basic geometric-structure specifications into pixel patterns.
- Graphics programming packages provide functions to describe a scene in terms of these basic geometric structures, referred to as output primitives, and to group sets of output primitives into more complex structures.
- Each output primitive is specified with input coordinate data and other information about the way that object is to be displayed.
- Points and straight line segments are the simplest geometric components of pictures.
- Additional output primitives that can be used to construct a picture include circles and other conic sections, quadric surfaces, spline curves and surfaces, polygon color areas, and character strings.

Points and lines

POINT	It is the position in a plane and can be specified with an ordered pair of coordinates (x, y) where x is Horizontal distance from the origin and y is vertical distance from origin.
LINE	Two points will specify a line. A line is specified by equations such that if a point (x, y) satisfy the equation. Then the point is said to be lying on the line. Example: If two points are used to specify a line are (x1, y1) and (x2, y2), then equation of the line is $(y_2-y_1)\ (x-x_1) = (x_2-x_1)(y-y_1)$ $y = y_1 + (y_2-y_1)\ (x-x_1)/(x_2-x_1)$ This is in the form of $y = mx + c$ where $m = (y_2-y_1)/(x_2-x_1)$ and $c = y_1-mx_1$

Point Plotting

- Point plotting is attained by converting a single coordinate position furnished by an application program into appropriate operations for output device in use.
- For example, with a CRT monitor, the electron beam is turned on to illuminate the screen phosphor at the selected location. The electron beam is positioned based on the display technology.
- For a random-scan(vector) system
  - This stores point-plotting instructions in the display list.
  - Coordinate values in these instructions are converted to deflection voltages that position the electron beam at the screen locations to be plotted during each refresh cycle.
- For a black-and-white raster system,
  - a point is plotted by setting the bit value corresponding to a specified screen position within the frame buffer to 1.
  - Then, as the electron beam sweeps across each horizontal scan line, it emits a burst of electrons (plots a point) whenever a value of 1 is encountered in the frame buffer.
- With an RGB system raster system
  - The frame buffer is loaded with the color codes for the intensities that are to be displayed at the screen pixel positions.

Line Drawing

- The line is the most fundamental drawing primitive with many uses - charts, engineering drawings, illustrations, 2D pencil-based animation and curve approximation.
- Desired properties of line drawing algorithms
  - Lines should appear sharp and straight
  - Line should terminate accurately
  - Line should have constant density
  - Line density should be independent of line, length and angle.
  - Efficient
- Line drawing is accomplished by calculating intermediate positions along directed to fill in these positions between the endpoint positions. An output device is then directed to fill in these positions between the endpoints.
- For analog devices, such as a vector pen plotter or a random-scan display, a straight line can be drawn smoothly from one endpoint to the other. Linearly varying horizontal and vertical deflection voltages are generated that are proportional to the required changes in the x and y directions to produce smooth lines.
- Digital devices display a straight line segment by plotting discrete points between two endpoints. Discrete coordinate positions along the line path are calculated from the equation of the line.
- For a raster video display, the line color (intensity) is then loaded into the frame buffer at the corresponding pixel coordinates. Reading from the frame buffer, the video controller then "plots" the screen pixels. Screen locations are referenced with integer values, so plotted positions may only approximate actual Line positions between two specified endpoints. A computed line position of (10.48,20.51), for example, would be converted to pixel position (10,211. Thus rounding of coordinate values to integers causes lines to be displayed with a stairstep appearance ("the jaggies"), as represented in Figure 2-1.

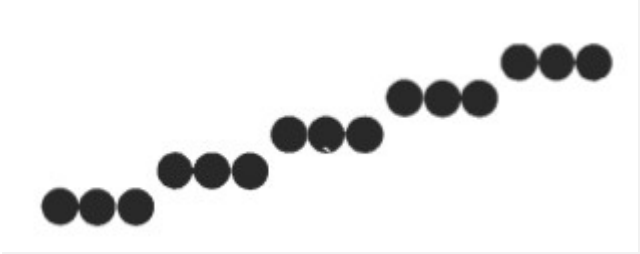


Figure 2-1 Stairstep effect (jaggies) produced when a line is generated as a series of pixel positions.

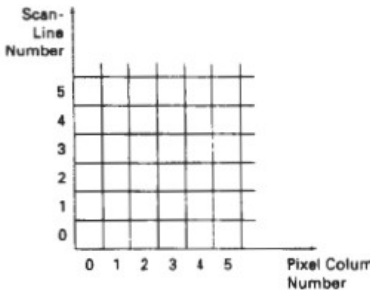


Figure 2-2: Pixe1 positions referenced by scan line number and column number.

- The characteristic stairstep shape of raster lines is particularly noticeable on systems with low resolution, their appearance is improved by displaying them on high-resolution systems.
- More effective techniques for smoothing raster lines are based on adjusting pixel intensities along the line paths.
- For the raster-graphics device-level algorithms, object positions are specified directly in integer device coordinates. The pixel positions are referenced according to scan-line number and column number (pixel position across a scan line). This addressing scheme is shown in Figure 2-2.
- Scan lines are numbered consecutively from 0, starting at the bottom of the screen; and pixel columns are numbered from 0, left to right across each scan line.
- To load a specified color into the frame buffer at a position corresponding to column x along scan line y, a low-level procedure set Pixel of the form is used

set Pixel(x,y)

- To retrieve the current frame buffer intensity setting for a specified location the low-level function get Pixel is used

get Pixel(x,y)

Line Drawing algorithms

- Straight line segments are used greatly in computer generated pictures. They occur in block diagrams, bar charts and graphics, civil and mechanical engineering drawings, logical schematics, and architectural plans are created and used in computer graphics.

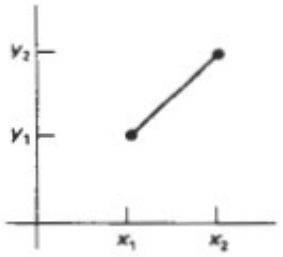


Figure 2.3 Line path between endpoint positions $(x_1,y_1)$and (x_2,y_2).

The Cartesian slope-intercept equation for a straight line is

$$y = m \, . \, x + b \qquad -(1)$$

with *m* representing 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_1,y_1) and (x_2, y_2), as shown in Figure. 2-3, values for the slope m and y intercept b are determined with the following calculations:

$$m = \frac{y_2 - y_1}{x_2 - x_1} \qquad -(2)$$

$$b = y_1 - m.x_1 \qquad -(3)$$

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

For any given x interval Δx along a line, we can compute the corresponding y interval from (2) as

$$\Delta y = m\Delta x \qquad -(4)$$

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

$$\Delta x = \frac{\Delta y}{m} \qquad -(5)$$

These equations form the basis for determining deflection voltages in analog devices.

- For lines with slope magnitudes $|m| < 1$, Δx can be set proportional to a small horizontal deflection voltage and the corresponding vertical deflection is then set proportional to Δy as calculated from Eq. (4).

- For lines whose slopes have magnitudes |m| > 1, Δy can be set proportional to a small vertical deflection voltage with the corresponding horizontal deflection voltage set proportional to Δx, calculated from Eq. (5).
- For lines with m = 1, Δx = Δy and the horizontal and vertical deflections voltages are equal. In each case, a smooth line with slope m is generated between the specified endpoints.

On raster systems, lines are plotted with pixels, and step sizes in the horizontal and vertical directions are constrained by pixel separations. That is, a line is sampled at discrete positions and determines the nearest pixel to the line at each sampled position. This scan-conversion process for straight lines is illustrated in Fig. 2-Fig 2-4, for a near horizontal line with discrete sample positions along the x axis.

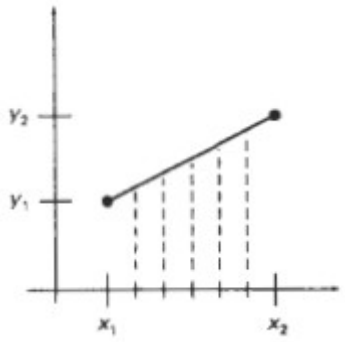


Figure 2-4: Straight line segment with five sampling positions along the x axis between x_1 and x_2.

There are three types of line drawing algorithms:

- Digital differential Algorithm (DDA)
- Bresenham's Line drawing Algorithm.
- Parallel Line Algorithms

DDA Algorithm

- The digital differential analyzer (DDA) is a scan-conversion line algorithm based on calculating either Δy or Δx, using Eq. (4) or Eq. (5).
- The line is sampled at unit intervals in one coordinate and corresponding integer values nearest the line path for the other coordinate are determined.
- For line with positive slope(slope is less than or equal to 1), we sample at unit x intervals (Δx = 1) and compute each successive y value as

$$y_{k+1} = y_k + m \qquad \text{- (6)}$$

- Subscript k takes integer values starting from 1, for the first point, and increases by 1 until the final endpoint is reached.
- Since m can be any real number between 0 and 1, the calculated y values must be rounded to the nearest integer.

- For lines with a positive slope greater than 1, we reverse the roles of x and y.
- That is, we sample at unit y intervals (Δy = 1) and calculate each succeeding x value as

$$x_{k+1} = x_k + \frac{1}{m} \qquad \text{-(7)}$$

- Equations (6) and (7) are based on the assumption that lines are to be processed from the left endpoint to the right endpoint (Fig. 2-3).
- If this processing is reversed, so that the starting endpoint is at the right, then either we have Δx = -1 and

$$y_{k+1} = y_k - m \qquad \text{-(8)}$$

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

$$x_{k+1} = x_k - \frac{1}{m} \qquad \text{-(9)}$$

- Equations (6) through (9) can also be used to calculate pixel positions along a line with negative slope.
- If the absolute value of the slope is less than 1 and the start endpoint is at the left, we set Δx = 1 and calculate y values with Eq. (6).
- When the start endpoint is at the right (for the same slope), we set Δx = -1 and Output Primitives obtain y positions from Eq. (8).

- Similarly, when the absolute value of a negative slope is greater than 1, we use $\Delta y = -1$ and Eq. (9) or we use $\Delta y = 1$ and Eq. (7).

Algorithm:

Step1

Input two end point pixel positions (x_1, y_1) and (x_2, y_2)

Step2

Find horizontal and vertical difference between the end points

$dx = x_2 - x_1$

$dy = y_2 - y_1$

Step3

The difference with the greater magnitude determines the value of parameter steps

```
If abs (dx)>abs (dy) then
        steps = abs (dx)
Else
        steps = abs (dy)
```

Step4

Starting with pixel position (x_1, y_1) be determined offset needed at each step to generate next pixel along the line path.

```
xIncrement = dx/steps
ylincrement = dy/steps
```

Step5

Assign the values of x_1, y_1 to x, y

$x = x_1$

$y = y_1$

Step6

Plot the pixel at (x, y) position on screen set pixel (round(x), round(y), 1). Here '1' is the intensity of pixel i.e., intensity with which picture is illuminated.

step7

Calculate the values of x and y for the next pixel position.

```
x = x + xIncrement
y = y + yIncrement
```

Step8

Plot the pixel at (x, y) position, set pixel (round(x), round(y), 1)

Step9

Repeat the steps 7 and 8 until the value of I i.e., start from 1 to steps.

EXAMPLE: Draw A line between the points (20,10) and (30,18)

Step 1: x1 = 20, x2 =30, y1 = 10, y2 = 18.
Step 2: dx = 30 -20 = 10, dy = 18 – 10 = 8.
Step 3: dx > dy => 10 >8, steps =10.
Step 4: x increment = 10/10 = dx/steps = 1.
y increment = 8/10 = dy/steps = 0.8.
Step 5: x = 20, y = 10.
Step 6: pixel (20,10) .
Step 7: x = x + x increment.
y = y + y increment.

I	x	y	(round(x),round(y))
1	20+1=21	10+0.8=10.8	(21,11)
2	21+1=22	10.8+0.8=11.6	(22,12)
3	22+1=23	11.6+0.8=12.4	(23, 12)
4	23+1=24	12.4+0.8=13.2	(24,13)
5	24+1=25	13.2+0.8=14	(25,14)
6	25+1=26	14+0.8=14.8	(26,15)
7	26+1=27	14.8+0.8=15.6	(27,16)
8	27+1=28	15.6+0.8=16.4	(28,16)
9	28+1=29	16.4+0.8=17.2	(29,17)
10	29+1=30	17.2+0.8=18	(30,18)

Advantages:

- The DDA algorithm is a faster method for calculating pixel positions than the direct use of Eqs.-(1).
- It eliminates the multiplication in Eq. (1) by making use of raster characteristics, so that appropriate increments are applied in the x or y direction to step to pixel positions along the line path.
- Does not calculate coordinates based on the complete equation (uses offset method)

Disadvantages:

- Round-off errors are accumulated, thus line diverges more and more from straight line
- Round-off operations take time
- Perform integer arithmetic by storing float as integers in numerator and denominator and performing integer arithmetic

Bresenham's Line Algorithm

- An accurate and efficient raster line-generating algorithm, developed by Bresenham, scans converts lines using only incrementa1 integer calculations that can be adapted to display circles and other curves.
- Figures 2-5 and 2-6 illustrate sections of a display screen where straight line segments are to be drawn.
- The vertical axes show-scan-line positions, and the horizontal axes identify pixel columns.
- Sampling at unit x intervals in these examples, it is decided which of two possible pixel positions is closer to the line path at each sample step.
- Starting from the left endpoint shown in Fig. 2-5, the next sample position whether to plot the pixel at position (11, 11) or the one at (11, 12) is determined
- Similarly, Fig. 2-6 shows-a negative slope-line path starting from the left endpoint at pixel position (50, 50). In this, the next pixel position as (51,50) or as (51,49) is decided
- Bresenham's line algorithm is used here, by testing the sign of an integer parameter, whose value is proportional to the difference between the separations of the two pixel positions from the actual line path.
- To illustrate Bresenham's approach, first the scan-conversion process is considered for lines with positive slope less than 1.
- Pixel positions along a line path are then determined by sampling at unit x intervals.
- Starting from the left endpoint (x_0, y_0) of a given line, we step to each successive column (x position) and plot the pixel whose scan-line y value is closest to the line path.
- Figure 2-7 demonstrates the kth step in this process. Assuming that the pixel at (x_k, y_k) is to be displayed are determined, it is to needed to decide which pixel to plot in column x_{k+1}. The choices are the pixels at positions (x_{k+1}, y_k) and (x_{k+1}, y_{k+1}).
- At sampling position x_{k+l}, vertical pixel separations from the mathematical line path are labeled as d_1, and d_2 (Fig. 2-8). The y coordinate on the mathematical line at pixel column position x_{k+l} is calculated as

$$y = m(x_k + 1) + b \qquad \text{-(10)}$$

- Then

$$d_1 = y - y_k$$

$$= m(x_k + 1) + b - y_k$$

- And

$$d_2 = (y_k + 1) - y$$

$$= y_k + 1 - m(x_k + 1) - b$$

- The difference between these two separations is

$$d_1 - d_2 = 2m(x_k + 1) - 2y_k + 2b - 1 \qquad \text{-(11)}$$

- A decision parameter p_k for the k_{th} step in the line algorithm can be obtained by rearranging Eq. (11) so that it involves only integer calculations.

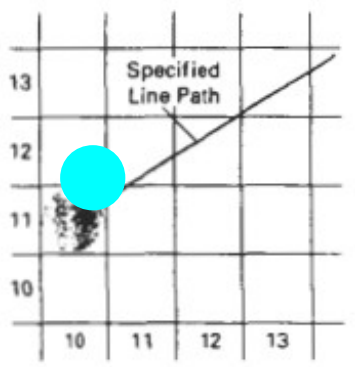


Figure 2-5: Section of a display screen where a straight line segment is to be

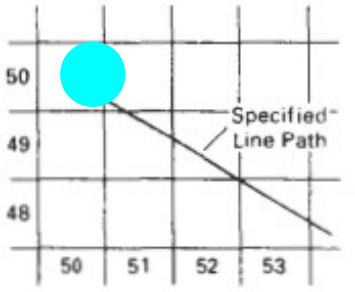
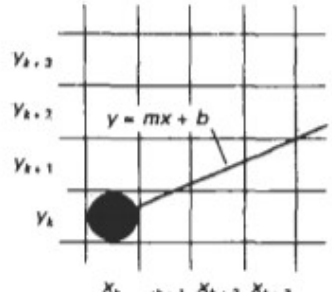


Figure 2-6:Section of a display screen where a negative slope line segment is to be plotted, starting from the



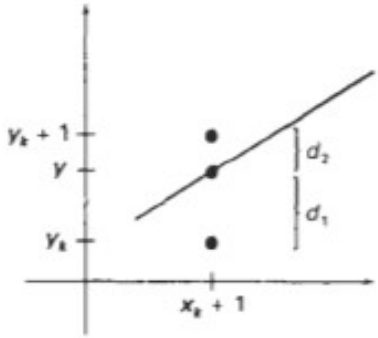


Figure 2-8: Distances between pixel positions and the line y coordinate at sampling position $x_k + 1$

- This is achieved by substituting m = Δy/Δx, where Δy and Δx are the vertical and horizontal separations of the endpoint positions, and defining:

$$p_k = \Delta x(d_1 + d_2)$$

$$= 2\Delta y.x_k - 2\Delta x.y_k + c \qquad \text{-(12)}$$

- The sign of p_k, is the same as the sign of $d_l - d_2$, since $\Delta x > 0$.
- For example here, Parameter c is constant and has the value $2\Delta y + \Delta x$ (2b- l), which is independent of pixel position and will be eliminated in the recursive calculations for p_k.
- If the pixel at y_k is closer to the line path than the pixel at y_k+l (that is, $d_1 < d_2$), then decision parameter p_k is negative. In that case, the lower pixel is plotted; otherwise, the upper pixel is plotted.
- Coordinate changes along the line occur in unit steps in either the x or y directions.
- Therefore, we can obtain the values of successive decision parameters using incremental integer calculations.
- At step k + 1, the decision parameter is evaluated from Eq. 3-12 as

$$p_{k+1} = 2\Delta y x_{k+1} - 2\Delta x y_{k+1} + c$$

- Subtracting Eq. 3-12 from the preceding equation,

$$p_{k+1} - p_k = 2\Delta y(x_{k+1} - x_k) - 2\Delta x(y_{k+1} - y_{k+1})$$

- But $x_{k+1} = x_k + 1$, so that

$$p_{k+1} = p_k + 2\Delta y - 2\Delta x(y_{k+1} - y_k) \qquad \text{-(13)}$$

- where the term y_{k+1}, y_k is either 0 or 1, depending on the sign of parameter p_k.
- This recursive calculation of decision parameters is performed at each integer x position, starting at the left coordinate endpoint of the line.
- The first parameter, p_0 is evaluated from Eq. 3-12 at the starting pixel position (x_o, y_o) and with
  m evaluated as Δy/Δx:

$$p_0 = 2\Delta y - \Delta x \qquad \text{-(14)}$$

- Bresenham's line drawing for a line with a positive slope less than 1 in the following listed steps. The constants $2\Delta y$ and $2\Delta y - 2\Delta x$ are calculated once for each line to be scan converted, so the arithmetic involves only integer addition and subtraction of these two constants.

Algorithm:

Step1
Input the two line end points and store the left endpoints in (x_0, y_0)
Step2
Load (x_0, y_0) into the frame buffer; plot the first point.
Step3
Calculate constants Δx, Δy, 2Δy, and 2Δy - 2Δx, and obtain the starting value for the decision parameter as $\Delta x = x_n - x_0$ and $\Delta y = y_n - y_0$ $p_0 = 2\Delta y - \Delta x$

Step4

At each x_k along the line, starting at k = 0, perform the following test:

If $p_k < 0$, the next point to plot is (x_k+1, y_k) and

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

Otherwise, the next point to plot is (x_k+1, y_k+1) and

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

Step5

Repeat step 4 Δx times.

- Bresenham's algorithm is generalized to lines with arbitrary slope by considering the symmetry between the various octants and quadrants of the xy plane.
  - For a line with positive slope greater than 1, the roles of the x and y directions are interchanged. That is, we step along the y direction in unit steps and calculate successive x values nearest the line path.
  - Also, the algorithm can be revised to plot pixels starting from either endpoint. If the initial position for a line with positive slope is the right endpoint, both x and y decrease as stepping from right to left. To ensure that the same pixels are plotted regardless of the starting endpoint, we always choose the upper (or the lower) of the two candidate pixels whenever the two vertical separations from the line path are equal ($d_1 = d_2$).
  - For negative slopes, the procedures are similar, except that now one coordinate decreases as the other increases.
  - Finally, specia1 cases can be handled separately:
    - Horizontal lines($\Delta y = 0$), Vertical lines ($\Delta x = 0$), and diagonal lines with $|\Delta x|=|\Delta y|$ each can be loaded directly into the frame buffer without processing them through the line-plotting algorithm.

EXAMPLE: Draw A line between the points (20,10) and (30,18)

This line has a slope of 0.8, with

$$\Delta x = 10 , \Delta y = 8$$

The initial decision parameter has the value

$$p_0 = 2\Delta y - \Delta x$$

$$= 6$$

and the increments for calculating successive decision parameters are

$$2\Delta y = 16$$

$$2\Delta y - 2\Delta x = -4$$

We plot the initial point $(x_o,y_o) = (20, 10)$, and determine successive pixel positions along the line path from the decision parameter as

k		
0		
1		
2		
3		
4		
5		
6		
7	-2	(28,16)
8	14	(29,17)
9	10	(30,18)

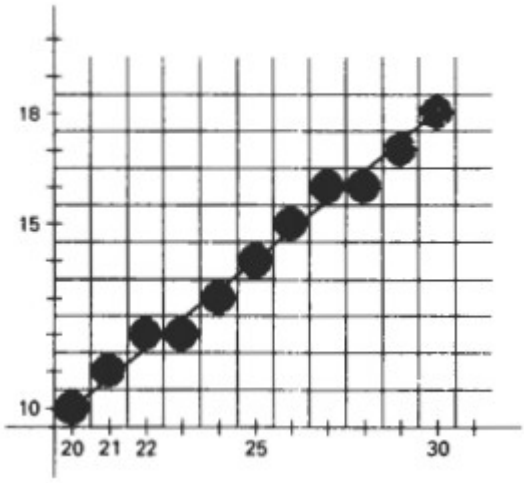


Figure 2-9: Pixel positions along the line path between endpoints (20.10) and (30,18), plotted with Bresenham's line algorithm

A plot of the pixels generated along this line path is shown in Fig. 2-9.

Advantages:

- Bresenham's algorithm generally faster than DDA
- Bresenham's algorithm uses integer arithmetic
- Constants need to be computed only once

Parallel Line Algorithms

- The line-generating algorithms such as DDA or Bresenham's determine pixel positions sequentially.
- With a parallel computer, pixel positions are along a line path simultaneously by partitioning the computations among the various processors available.
- Approaches to the partitioning problem
  - to adapt an existing sequential algorithm to take advantage of multiple processors.
  - look for other ways to set up the processing so that pixel positions can be calculated efficiently in parallel.
- Devising a parallel algorithm is to balance the processing load among the available processors.

Parallel Bresenham's algorithm

- Given n, processors, a parallel Bresenham's line algorithm is by subdividing the line path into n, partitions and simultaneously generating line segments in each of the subintervals.
- For a line with slope 0<m<1 and left endpoint coordinate position (x_o, y_o), the line is partitioned along the positive x direction.
- The distance between beginning x positions of adjacent partitions can be calculated as

  -(15)

- where Δx is the width of the line, and the value for partition width Δx is computed using integer division.
- Numbering the partitions, and the processors as 0,1,2, up to $n_p - 1$, the starting x coordinate for the k_{th} partition is calculated as

  $$x_k = x_0 + k\Delta x_p \qquad \text{-(16)}$$

- **Example:** Suppose $\Delta x = 15$ and we have $n_p = 4$ processors.

- Then the width of the partitions is 4 and the starting x values for the partitions are x_o, x_o+4, x_o+8 and x_o+12.

- To apply Bresenham's algorithm over the partitions, the initial value for the y coordinate is required and the initial value for the decision parameter in each partition. The change Δy, in the y direction over each partition is calculated from the line slope m and partition width Δx_p.

$$\Delta y_p = m\Delta x_p \qquad -(17)$$

- At the k_{th} partition, the starting y coordinate is then

$$y_k = y_0 + \text{round}(k\Delta y_p) \qquad -(18)$$

- The initial decision parameter for Bresenham's algorithm at the start of the k_{th} subinterval is obtained from Eq. 3-12:

$$p_k = (k\Delta x_p)(2\Delta x_p) - \text{round}(k\Delta x)(2\Delta x) + 2\Delta y - \Delta x \qquad -(18)$$

- Each processor then calculates pixel positions over its assigned subinterval using the starting decision parameter value for that subinterval and the starting coordinates (x_k, y_k).
- The floating-point calculations to integer arithmetic in the computations are reduced for starting values y_k and p_k by substituting $m = \Delta y/\Delta x$ and rearranging terms.
- The extension of the parallel Bresenham's algorithm to a line with slope greater than 1 is achieved by partitioning the line in the y direction and calculating beginning x values for the partitions.
- For negative slopes, we increment coordinate values in one direction and decrement in the other.

Advantages:

- With this partitioning scheme, the width of the last(rightmost) subinterval will be smaller than the others in some cases.
- If the line endpoints are not integers, truncation errors can result in variable width partitions along the length of the line.

Other Approach

- Other Approach to set up Parallel algorithms on raster systems is to assign each processor to a particular group of screen pixels.
- With a sufficient number of processors (such as a Connection Machine CM-2 with over 65,000 processors), we can assign each processor to one pixel within some screen region.
- This approach can be adapted to line display by assigning one processor to each of the pixels within the limits of the line coordinate extents (bounding rectangle) and calculating pixel distances from the line path.
- The number of pixels within the bounding box of a line is Δx. Δy (Fig. 2-10).
- Perpendicular distance d from the line in Fig. 2-10 to a pixel with coordinates (x, y) is obtained with the calculation

$$d = Ax + By + C \qquad -(20)$$

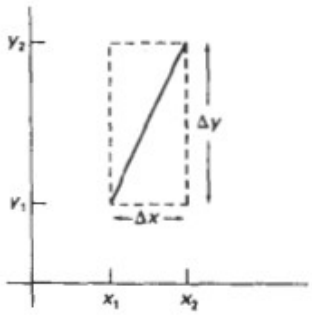


Figure 2-10: Bounding box for a line with coordinate extents

- here

$$A = \frac{-\Delta y}{linelength}$$

$$B = \frac{\Delta x}{linelength}$$

$$C = \frac{x_0 \Delta y - y_0 \Delta x}{linelength}$$

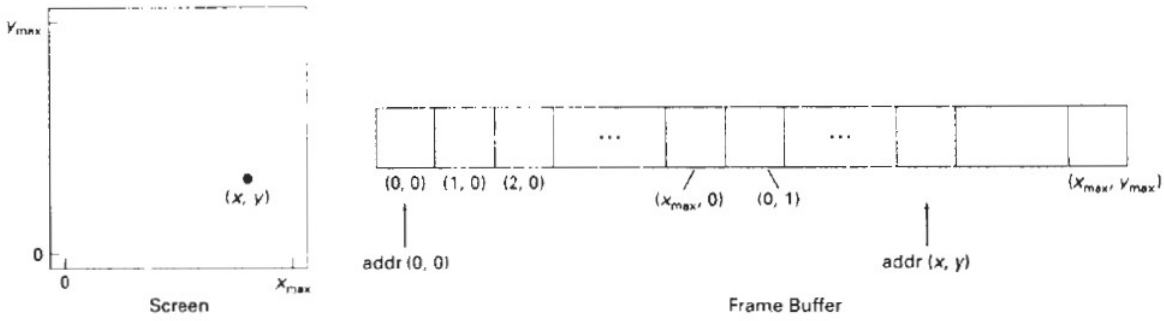
- with

$$linelength = \sqrt{\Delta x^2 - \Delta y^2}$$

- Once the constants A, B, and C have been evaluated for the line, each processor needs to perform two multiplications and two additions to compute the pixel distance d.
- A pixel is plotted if d is less than a specified line-thickness parameter.
- Instead of partitioning the screen into single pixels; we can assign to each processor either a scan line or a column of pixels depending on the line slope.
- Each processor then calculates the intersection of the line with the horizontal row or vertical column of pixels assigned that processor.
- For a line with slope |m| < 1, each processor simply solves the line equation for y, given an x column value.
- For a line with slope magnitude greater than 1, the line equation is solved for x by each processor, given a scan-line y value.
- Such direct methods, although slow on sequential machines, can be performed very efficiently using multiple processors

Loading the Frame Buffer

- When straight line segments and other objects are scan converted for display with a raster system, frame-buffer positions must be calculated.
- This is accomplished with the set pixel procedure, which stores intensity values for the pixels at corresponding addresses within the frame-buffer array.
- Scan-conversion algorithms generate pixel positions at successive unit intervals.
- Hence incremental methods are used to calculate frame-buffer addresses.
- As a specific example, suppose the frame-buffer array is addressed in row major order and that pixel positions vary from (0,0) at the lower left screen corner to (x_{max}, y_{max}) at the top right corner (Fig. 2-11).



- For a bi-level system (1 bit per pixel), the frame-buffer bit address for pixel position (x, y) is calculated as

$$addr(x,y) = addr(0,0) + y(x_{max}+1) + x \quad -(21)$$

- Moving across a scan line, the frame-buffer address for the pixel at (x + 1, y) is calculated as the following offset from the address for position (x, y):

$$addr(x+1,y) = addr(x,y) + 1 \quad -(22)$$

- Stepping diagonally up to the next scan line from (x, y), frame buffer address of (x +1,y+1) is calculated as

$$addr(x+1,y+1) = addr(x,y) + x_{max} + 2 \quad -(23)$$

- Similar incremental calculations can be obtained from Eq. 21 for unit steps in the negative x and y screen directions. Each of these address calculations involves only a single integer addition.
- Methods for implementing the set pixel procedure to store pixel intensity values depend on the capabilities of a particular system and the design requirements of the software package.
- With systems that can display a range of intensity values for each pixel, frame-buffer address calculations would include pixel width (number of bits), as well as the pixel screen location.

Mid-Point circle and Ellipse algorithms

Mid-Point circle Algorithm

- As in the raster line algorithm, unit intervals are taken and the closest pixel position is determined to the specified circle path at each step.
- For a given radius r and screen center position (x_c, y_c), the algorithm is first setup to calculate pixel positions around a circle path centered at the coordinate origin (0,0).
- Then each calculated position (x, y) is moved to its proper screen position by adding x_c to x and y_c to y.
- Along the circle section from x = 0 to x = y in the first quadrant, the slope of the curve varies from 0 to -1.
- Therefore, unit steps are taken in the positive x direction over this octant and a decision parameter is used to determine which of the two possible y positions is closer to the circle path at each step.
- Positions in the other seven octants are then obtained by symmetry.
- To apply the midpoint method, a circle function is defined:

$$f_{circle}(x,y) = x^2 + y^2 - r^2 \qquad -(24)$$

- Any point (x, y) on the boundary of the circle with radius r satisfies the equation $f_{circle}(x,y)= 0$.
    - If the point is in the interior of the circle, the circle function is negative.
    - And if the point is outside the circle, the circle function is positive.
- Hence, the relative position of any point(x,y) can be determined by checking the sign of the circle function:

$$f_{circle}(x,y) \begin{cases} < 0, & \text{if } (x,y) \text{ is inside the circle boundary} \\ = 0, & \text{if } (x,y) \text{ is on the circle boundary} \\ > 0, & \text{if } (x,y) \text{ is outside the circle boundary} \end{cases} \qquad -(25)$$

- The circle-function tests in Eqs. (25) 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, and incremental calculations for this function are similar to the line algorithm.
- Figure 2-12 shows the midpoint between the two candidate pixels at sampling position x_k+1.
- Assuming the pixel at (x_k,y_k) is plotted, next the pixel at position (x_k+1,yk) or the one at position $(x_k+1, y_k - 1)$ is closer to the circle is determined.
- Decision parameter is the circle function (25) is evaluated at the midpoint between these two pixels:

$$p_k = f_{circle}\left(x_k + 1, y_k - \frac{1}{2}\right)$$

$$= (x_k + 1)^2 + \left(y_k - \frac{1}{2}\right)^2 - r^2 \qquad -(26)$$

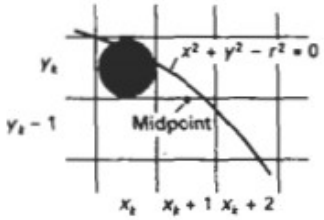


Figure 2-12: Midpoint between candidate pixels at

- If $p_k < 0$, this midpoint is inside the circle and the pixel on scan line y_k is closer to the circle boundary.
- Otherwise, the midpositions is outside or on the circle boundary, and the pixel on scan line $y_k - 1$ is selected.
- Successive decision parameters are obtained using incremental calculations.
- A recursive expression for the next decision parameter is determined by evaluating the circle function at sampling position $x_{k+1}+1 = x_k+2$:

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

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

or

$$p_{k+1} = p_k + 2(x_k + 1) + (y_{k+1}^2 - y_k^2) - (y_{k+1} - y_k) + 1 \quad \text{-(27)}$$

- where y_{k+1} is either y_k or y_{k-1}, depending on the sign of p_k.
- Increments for obtaining p_{k+1} are either $2x_{k+1}+1$ (if pk is negative) or $2x_{k+1}+1-2y_{k+1}$.
- Evaluation of the terms $2x_{k+1}$ and $2y_{k+1}$, can also be done incrementally as

$$2x_{k+1} = 2x_k + 2$$

$$2y_{k+1} = 2y_k - 2$$

- At the start position (0, r), these two terms have the values 0 and 2r, respectively.
- Each successive value is obtained by adding 2 to the previous value of 2x and subtracting 2 from the previous value of 2y.
- The initial decision parameter is obtained by evaluating the circle function at the start position $(x_o, y_o) = (0, r)$:

$$p_0 = f_{circle}\left(1, r - \frac{1}{2}\right)$$

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

- or

$$p_0 = \frac{5}{4} - r \quad \text{-(28)}$$

- If the radius r is specified as an integer, p_o is rounded to

$$p_o = 1 - r \text{ (for } r \text{ an integer)}$$

- since all increments are integers.
- As in Bresenham's line algorithm, the midpoint method calculates pixel positions along the circumference of a circle using integer additions and subtractions, assuming that the circle parameters are specified in integer screen coordinates.

Algorithm:

Step 1

Input radius r and circle center (x_c, y_c) and obtain the first point on the circumference of a circle

centered on the origin as

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

Step 2

Calculate the initial value of the decision parameter as

$$p_0 = 5/4 - r$$

Step 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 $2y_{k+1} = 2y_k - 2$.

Step 4

Determine symmetry points in the other seven octants.

Step 5

Move each calculated pixel position (x, y) onto the circular path centered on (x_c, y_c) and plot the coordinate values:

$$x = x + x_c, y = y + y_c.$$

Step 6

Repeat steps 3 through 5 until x >= y.

Example:

- Given a circle radius r= 10, midpoint circle algorithm is demonstrated 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

  $p_0 = 1 - r = -9$

- For the circle centered on the coordinate origin, the initial point is $(x_0, y_0)=(0,10)$ and initial increment terms for calculating the decision parameters are

  $2x_0=0$ and $2y_0=20$

- Successive decision parameter values and positions along the circle path are calculated using the midpoint method as

k	p
0	-9
1	-6
2	-1
3	6
4	-3
5	8
6	5

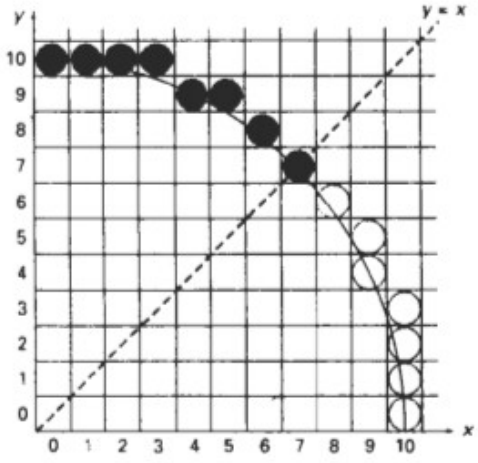


Figure 2-13: Selected pixel positions(solid circles) along a circle path with radius r=10 centered on the origin, using the midpoint circle algorithm. Open circles show the symmetry positions in the first quadrant

- A plot of the generated pixel positions in the first quadrant is shown in Fig. 2-13.

Mid-point Ellipse algorithms

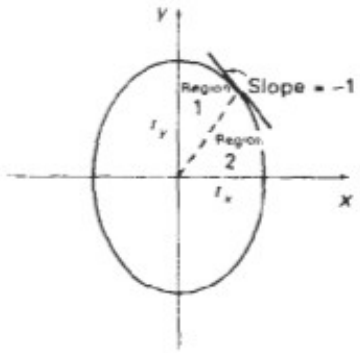


Figure 2-14: Ellipse processing regions, Over region 1, the magnitude of the ellipse slope is less than 1; over region 2, the magnitude of the slope is

- Approach is similar to that used in displaying a raster circle.
- Given parameters d_x, r_y, and (x_c,y_c), the points (x, y) for an ellipse in standard position centered on the origin are determined, and then shifted to the points so the ellipse is centered at (x_c,y_c).
- The midpoint ellipse method is applied throughout the first quadrant in two parts.
- Figure 2-14 shows the division of the first quadrant according to the slope of an ellipse with $d_x < r_y$
- This quadrant by is processed taking unit steps in the x direction where the slope of the curve has a magnitude less than 1, and taking unit steps in the direction where the slope has a magnitude greater than 1.
- Regions 1 and 2 (Fig. 2-14), can be processed in various ways.
- One can start at position $(0, r_y)$ and step clockwise along the elliptical path in the first quadrant, shifting from unit steps in x to unit steps in y when the slope becomes less than -1.
- Alternatively, one can start at $(r_y,0)$ and select points in a counterclockwise order, shifting from unit steps in y to unit steps in x when the slope becomes greater than -1.
- With parallel processors, one could calculate pixel positions in the two regions simultaneously.
- As an example of a sequential implementation of the midpoint algorithm, we take the start position at $(0, r_y)$ and step along the ellipse path in clockwise order throughout the first quadrant.
- an ellipse function is defined with $(x_c,y_c)=(0,0)$ as

$$f_{ellipse}(x,y) = r_y^2x^2 + r_x^2y^2 - r_x^2r_y^2 \quad -(29)$$

- which has the following properties

$$f_{ellipse}(x,y)\begin{cases}<0, & \text{if } (x,y) \text{ is inside the ellipse boundary}\\ =0, & \text{if } (x,y) \text{ is on the ellipse boundary}\\ >0, & \text{if } (x,y) \text{ is outside the ellipse boundary}\end{cases}$$ -(30)

- Thus, the ellipse function $f_{ellipse}(x,y)$ serves as the decision parameter in the midpoint algorithm.
- At each sampling position, the next pixel is selected along the ellipse path according to the sign of the ellipse function evaluated at the midpoint between the two candidate pixels.
- Starting at $(0, r_y)$ unit steps are taken in the x direction until the boundary between region 1 and region 2 is reached.
- Then switched to unit steps in the y direction over the remainder of the curve in the first quadrant.
- At each step, the value of the slope of the curve is tested.
- The ellipse slope is calculated from Eq. (29) as

$$\frac{dy}{dx} = -\frac{2r_y^2 x}{2r_x^2 y}$$ -(31)

- At the boundary between region 1 and region 2, $dy/dx = -1$ and
  $2r^2_y x = 2r^2_y y$
- Therefore, we move out of region 1 whenever
  $2r^2_y x \geq 2r^2_y y$ - (32)

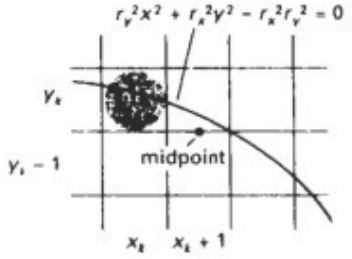


Figure 2-15: Midpoint between candidate pixels at sampling position $x_l + 1$ along an elliptical

- Figure 2-15 shows the midpoint between the two candidate pixels at sampling position x_k+1 in the first region.
- Assuming position (x_k,y_k) has been selected at the previous step, the next position along the ellipse path is determined by evaluating the decision parameter (that is, the ellipse function (29)) at this midpoint:

$$p1_k = f_{ellipse}\left(x_k + 1, y_k - \frac{1}{2}\right)$$
$$= r_y^2(x_k + 1)^2 + r_x^2\left(y_k - \frac{1}{2}\right)^2 - r_x^2 r_y^2$$ -(33)

- If $pl_k<0$, the midpoint is inside the ellipse and the pixel on scan line y_k, is closer to the ellipse boundary.
- Otherwise, the midpositions is outside or on the ellipse boundary, and the pixel on scan line y_k-1 is selected.
- At the next sampling position $(x_{k+1} +1 = x_k + 2)$, the decision parameter for region 1 is evaluated as

$$p1_{k+1} = f_{ellipse}\left(x_{k+1} + 1, y_{k+1} - \frac{1}{2}\right)$$
$$= r_y^2[(x_k + 1) + 1]^2 + r_x^2\left(y_{k+1} - \frac{1}{2}\right)^2 - r_x^2 r_y^2$$

or

$$p1_{k+1} = p1_k + 2r_y^2(x_k + 1) + r_y^2 + r_x^2\left[\left(y_{k+1} - \frac{1}{2}\right)^2 - \left(y_k - \frac{1}{2}\right)^2\right]$$ -(34)

- where y_{k+1}, is either y_k or y_k-1, depending on the sign of $p1_k$.
- Decision parameters are incremented by the following amounts:

$$\text{increment} = \begin{cases}2r_y^2 x_{k+1} + r_y^2, & \text{if } p1_k < 0\\ 2r_y^2 x_{k+1} + r_y^2 - 2r_x^2 y_{k+1}, & \text{if } p1_k \geq 0\end{cases}$$

- As in the circle algorithm, increments for the decision parameters can be calculated using only addition and subtraction, since values for the terms $2r^2_y x$ and $2r^2_x y$ can also be obtained incrementally.
- At the initial position $(0, r_y)$, the two terms evaluate to

$$2r^2_y x = 0 \qquad -(35)$$

$$2r^2_x y = 2r^2_x r_y \qquad -(36)$$

- As x and y are incremented, updated values are obtained by adding $2r^2_y$ to (35) and subtracting $2r^2_x$ from (36).
- The updated values are compared at each step, and we move from region 1 to region 2 when condition (32) is satisfied.
- In region 1, the initial value of the decision parameter is obtained by evaluating the ellipse function at the start position $(x_o, y_o) = (0, r_y)$:

$$p1_0 = f_{\text{ellipse}}\left(1, r_y - \frac{1}{2}\right)$$

$$= r_y^2 + r_x^2\left(r_y - \frac{1}{2}\right)^2 - r_x^2 r_y^2$$

or

$$p1_0 = r_y^2 - r_x^2 r_y + \frac{1}{4} r_x^2 \qquad -(37)$$

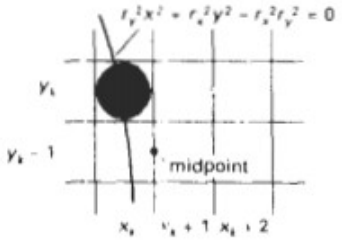


Figure 2-16 Midpoint between candidate pixels at sampling position $y_k - 1$ along an elliptical

- Over region 2, we sample at unit steps in the negative y direction, and the midpoint is now taken between horizontal pixels at each step (Fig. 2-16.
- For this region, the decision parameter is evaluated as

$$p2_k = f_{\text{ellipse}}\left(x_k + \frac{1}{2}, y_k - 1\right)$$

$$= r_y^2\left(x_k + \frac{1}{2}\right)^2 + r_x^2(y_k - 1)^2 - r_x^2 r_y^2 \qquad -(38)$$

- If $p2_k > 0$, the midpositions is outside the ellipse boundary, and the pixel at x_k is selected.
- If $p2_k \le 0$, the midpoint is inside or on the ellipse boundary, and pixel position at x_{k+1} is selected.
- To determine the relationship between successive decision parameters in region 2, the ellipse function at the next sampling step $y_{k+1}-1 = y_k-2$ is evaluated as:

$$p2_{k+1} = f_{\text{ellipse}}\left(x_{k+1} + \frac{1}{2}, y_{k+1} - 1\right)$$

$$= r_y^2\left(x_{k+1} + \frac{1}{2}\right)^2 + r_x^2[(y_k - 1) - 1]^2 - r_x^2 r_y^2 \qquad -(39)$$

- or

$$p2_{k+1} = p2_k - 2r_x^2(y_k - 1) + r_x^2 + r_y^2\left[\left(x_{k+1} + \frac{1}{2}\right)^2 - \left(x_k + \frac{1}{2}\right)^2\right] \qquad -(40)$$

- with x_{k+1}, set either to x_k or to $x_k + 1$, depending on the sign of $p2_k$.
- When we enter region 2, ;he initial position (x_o, y_o) is taken as the last position selected in region 1 and the initial decision parameter in region 2 is then

$$p2_0 = f_{\text{ellipse}}\left(x_0 + \frac{1}{2}, y_0 - 1\right)$$

$$= r_y^2\left(x_0 + \frac{1}{2}\right)^2 + r_x^2(y_0 - 1)^2 - r_x^2 r_y^2 \qquad -(41)$$

- To simplify the calculation of $p2_0$, pixel positions are selected in counterclockwise order starting at $(r_k,0)$.
- Unit steps would then be taken in the positive y direction up to the last position selected in region 1.
- The midpoint algorithm can be adapted to generate an ellipse in nonstandard position using the ellipse function Eq. (42) and calculating pixel positions over the entire elliptical path.

$$Ax^2 + By^2 + Cxy + Dx + Ey + F = 0 \qquad \text{- (42)}$$

- Assuming (x_k,y_k) and the ellipse center are given in integer screen coordinates, incremental integer calculations to values for the decision parameters are determined in the midpoint ellipse algorithm.
- The increments r^2_x, r^2_y, $2r^2_x$ and $2r^2_y$ are evaluated once at the beginning of the procedure.

Algorithm:

Step 1

Input radius r_x,r_y and circle center (x_c,y_c) and obtain the first point on an ellipse centered on the origin as

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

Step 2

Calculate the initial value of the decision parameter in region 1 as

$$p1_0 = r_y^2 - r_x^2 r_y + \frac{1}{4} r_x^2$$

Step 3

At each x_k position in region 1, starting at k=0, perform the following test:

If $p1_k<0$, the next point along the ellipse centered on (0, 0) is (x_{k+1}, y_k) and

$$p1_{k+1} = p1_k + 2r^2_y x_{k+1} + r^2_y$$

Otherwise, the next point along the circle is $(x_k + 1, y_k - 1)$ and

$$p1_{k+1} = p1_k + 2r^2_y x_{k+1} - 2r^2_x y_{k+1} + r^2_y$$

where $2r^2_y x_{k+1} = 2r^2_y x_k + 2r^2_y$ and $2r^2_x y_{k+1} = 2r^2_x y_k - 2r^2_x$

and continue until $2r^2_y x \geq 2r^2_y x$

Step 4

Calculate the initial value of the decision parameter in region 2 using the last point (x_o,y_o) calculated in region 1 as

$$p2_0 = r_y^2 \left(x_0 + \frac{1}{2}\right)^2 + r_x^2 (y_0 - 1)^2 - r_x^2 r_y^2$$

Step 5

At each y_k position in region 2, starting at k = 0, perform the following test: If pZk> 0, the next

point along the ellipse centered on (0, 0) is

If $p2_k<0$, the next point along the ellipse centered on (0, 0) is (x_k, y_k-1) and

$$p2_{k+1} = p2_k - 2r^2_x y_{k+1} + r^2_x$$

Otherwise, the next point along the circle is (x_k+1, y_k-1) and

$$p2_{k+1} = p2_k + 2r^2_y x_{k+1} - 2r^2_x y_{k+1} + r^2_x$$

using the same incremental calculations for x and y as in region 1.

Step 6

Determine symmetry points in the other three quadrants.

Step 7

Move each calculated pixel position (x, y) onto the elliptical path centered on (x_c, y_c) and plot the coordinate values:

$$x = x + x_c \quad , \quad y = y + y_c$$

Step 8

Repeat the steps for region 1 until $2r^2_y x \geq 2r^2_y x$

Example:

- Given input ellipse parameters $d_x = 8$ and $r_y = 6$, the steps in the midpoint ellipse algorithm are illustrated by determining raster positions along the ellipse path in the first quadrant.
- Initial values and increments for the decision parameter calculations are

  $2r^2_y x = 0$ (with increment $2r^2_y = 72$)

  $2r^2_x y = 2r^2_x r_y$ (with increment $-2r^2_x = -128$)

- For region 1: The initial point for the ellipse centered on the origin is $(x_o, y_o)=(0,6)$, and the initial decision parameter value is 1

$$p1_0 = r_y^2 - r_x^2 r_y + \frac{1}{4} r_x^2 = -332$$

- Successive decision parameter values and positions along the ellipse path are calculated using the midpoint method as

k	$p1_k$	(x_{k+1}, y_{k+1})	$2r^2_y x_{k+1}$	$2r^2_x y_{k+1}$
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 $2r^2_y x > 2r^2_x y$
- For region 2, the initial point is $(x_o, y_o) = (7,3)$ and the initial decision parameter is

$$p2_0 = f\left(7 + \frac{1}{2}, 2\right) = -151$$

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

k	$p1_k$	(x_{k+1}, y_{k+1})	$2r^2_y x_{k+1}$	$2r^2_x y_{k+1}$
0	-151	(8,2)	576	256
1	-233	(8,1)	576	128
2	745	(8,0)	-	-

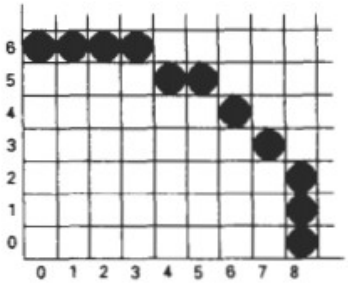


Figure 2-17:Positions along an elliptical path centered on the origin with d_x=8and r_y=6 using the midpoint

- A plot of the selected positions around the ellipse boundary within the first quadrant is shown in Fig. 2-17

Filled area primitives

- Area is filled with a certain color
- Most often the shape is that of a polygon
- Boundaries are linear as they are polygons
- Standard graphics objects are objects made of a set of polygon surface patches.
- There are two basic approaches to area filling on raster systems.
  - To fill an area is to determine the overlap intervals for scan lines that cross the area.
  - Area filling is to start from a given interior position and paint outward from this point until we encounter the specified boundary conditions.
- The scan-line approach is typically used in general graphics packages to fill polygons, circles, ellipses, and other simple curves.
- All methods starting from an interior point are useful with more complex boundaries and in interactive painting systems.

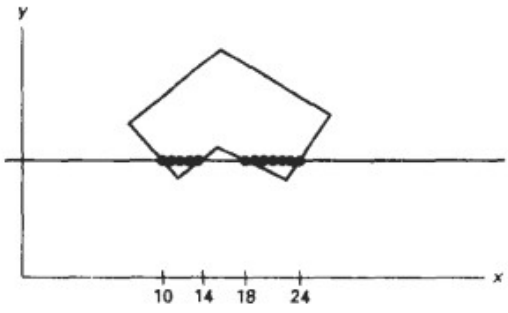


Figure 2-18: Interior pixels along a scan line

Scan line polygon fill algorithm

- Figure 2-18 illustrates the scan-line procedure for solid tilling of polygon areas.
- For each scan line crossing a polygon, the area-fill algorithm locates the intersection points of the scan line with the polygon edges.
- These intersection points are then sorted from left to right, and the corresponding frame-buffer positions between each intersection pair are set to the specified fill color.
- In the example of Figure 2-18, the four pixel intersection positions with the polygon boundaries define two stretches of interior pixels from x =10 to x =14 and from x =18 to x =24.
- Some scan-line intersections at polygon vertices require special handling. A scan line passing through a vertex intersects two edges at that position, adding two points to the list of intersections for the scan line.
- Figure 2-19 shows two scan lines at positions y and y' that intersect edge endpoints.
- Scan line y intersects five polygon edges. Scan line y' intersects an even number of edges although it also passes through a vertex.
- Intersection points along scan line y' correctly identify the interior pixel spans. But with scan line y, some additional processing is required to determine the correct interior points.

- The topological difference between scan line y and scan line y' in Fig. 2-19 is identified by noting the position of the intersecting edges relative to the scan line.
- For scan line y, the two intersecting edges sharing a 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, the vertices that require additional processing are those that have connecting edges on opposite sides of the scan line.
- These vertices are identified by tracing around the polygon boundary either in clockwise or counterclockwise order and observing the relative changes in vertex y coordinates as we move from one edge to the next.

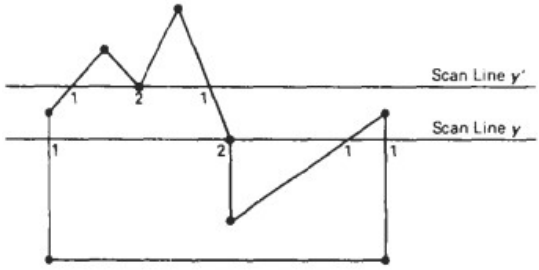


Figure 2-19: Intersection points along scan lines that intersect polygon vertices. Scan line y generates an odd number of intersections, but scan line y' generals an even number of intersections that can be paired to identify

- If the endpoint y values of two consecutive edges monotonically increase or decrease, we need to count the middle vertex as a single intersection point for any 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 way to resolve the question as to whether we should count a vertex as one intersection or two 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 counter-clockwise.
- 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.
- Figure 2-20 illustrates shortening of an edge.

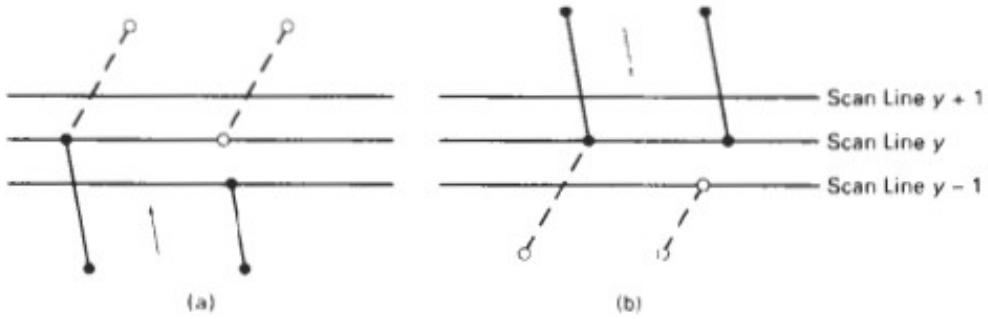


Figure 2-20: Adjusting endpoint y values for a polygon, as we process edges in order around the polygon perimeter. The edge currently being processed is indicated as a solid line. In (a), they coordinate of the upper endpoint of the current edge is decreased by 1. In (b), they coordinate of

- When the endpoint y coordinates of the two edges are increasing, the y value of the upper endpoint for the current edge 1s decreased by 1, as in Fig. 2-20(a).
- When the endpoint y values are monotonically decreasing, as in Fig. 2-20(b), we decrease they coordinate of the upper endpoint of the edge following the current edge.
- Calculations performed in scan-conversion and other graphics algorithms typically take advantage of various coherence properties of a scene that is to be displayed.
- Coherence means that the properties of one part of a scene are related in some way to other parts of the scene so that the relation-ship can be used to reduce processing.
- Coherence methods often involve incremental calculations applied along a single scan line or between successive scan lines.
- In determining 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.
- Figure 2-20 shows two successive scan lines crossing a left edge of a polygon.
- The slope of this polygon boundary line can be expressed in terms of the scan-line intersection coordinates:

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

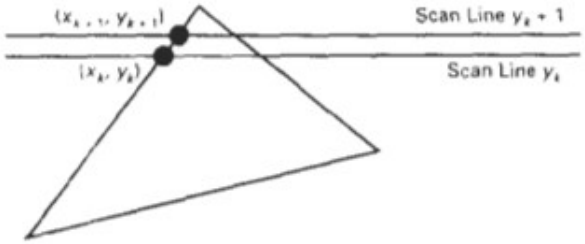


Figure 2-20: Two successive scan lines intersecting a

- 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.
- Parallel implementation of the fill algorithm is to assign each scan line crossing the polygon area 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 l/m along an edge can be accomplished with integer operations

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

- where Δx and Δy are the differences 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

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

- Using this equation, integer evaluation of the x intercepts is done 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, the current x intersection value is incremented by 1, and decrease the counter by the value Δy.

- To efficiently perform a polygon fill, the polygon boundary are stored in a sorted edge table that contains all the information necessary to process the scan lines efficiently.
- Proceeding around the edges in either a clockwise or a counterclockwise order, sorted on the smallest y value of each edge, in the correct scan-line positions.
- Only nonhorizontal edges are entered into the sorted edge table.
- Each entry in the table for a particular scan line contains the maximum y value for that edge, the x-intercept value (at the lower vertex) for the edge, and the inverse slope of the edge.
- For each scan line, the edges are in sorted order from left to right. Figure 2-21 shows a polygon and the associated sorted edge table.

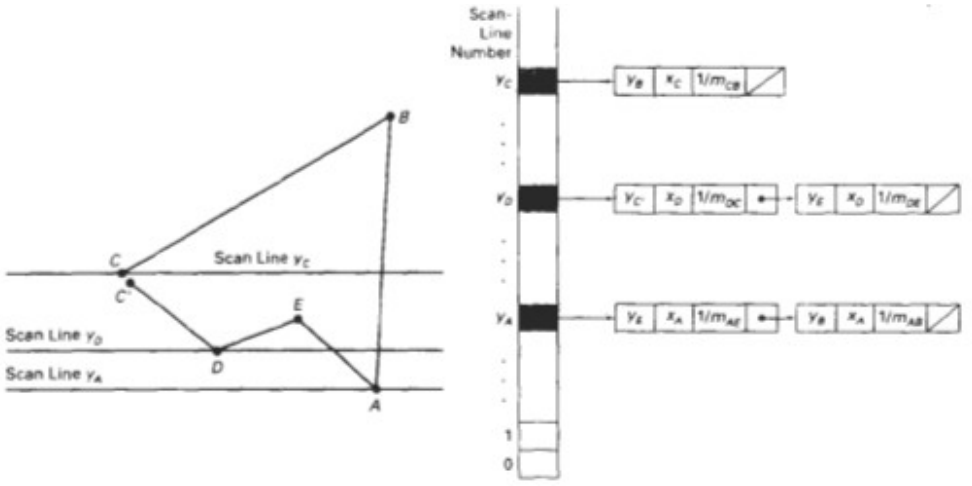


Figure 2-21: A polygon and its sorted edge table, with edge DC shortened by one unit in the y direction

- Next, the scan lines are processed from the bottom of the polygon to its top, producing an active edge list for each scan line crossing the polygon boundaries.
- The active edge list for a scan line contains all edges crossed by that scan line, with iterative coherence calculations used to obtain the edge intersections.
- Implementation of edge-intersection calculations is by storing Δx and Δy values in the sorted edge table.
- For each scan line, the pixel spans are filled for each pair of x-intercepts starting from the leftmost x-intercept value and ending at one position before the rightmost r intercept.
- And each polygon edge can be shortened by one unit in the y direction at the top endpoint.
- These measures also guarantee that pixels in adjacent polygons will not overlap each other.

Algorithm:

Step 1

Read n, the number of vertices of polygon

Step 2

Read x and y co-ordinates of all vertices in array x[n] and y[n]

Step 3

Find y_{min} and y_{max}

Step 4

Store the initial x value x_1 and x_2 and y value y_1 and y_2 for two endpoints and x increment Δx from scan line to scan line for each edge in the array edges[][]. [If $y_1 < y_2$, then interchange y_1 and y_2 and corresponding x_1 and x_2]

Step 5

Sort the rows of array, edges[][] in decending order of y_1, decending order of y_2, ascending order of x_2

Step 6

Set $y = y_{max}$

Step 7

Find the active edges and update active edge list:

if ($y>y_2$ and $y \leq y_1$)

[edge is active]

else

[edge is not active]

Step 8

Compute the x intersects for all active edges for current y value [initially x-intersect is x_1]

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

Step 9

If x intersect is vertex i.e., $x=x_1$ and $y=y_1$ then apply vertex test to check whether to consider one intersect or two intersects. Store all x intersects in the x-intersect[] array

Step 10

Sort x-intersect[] array in ascending order

Step 11

Extract pairs of intersects from the sorted x-intersect[] array

Step 12

Pass pairs of x values to line drawing routing to draw corresponding line segments
Step 13
Set y=y-1
Step 14
Repeat steps 7 through 13 until $y \geq y_{min}$

boundary-fill and flood-fill algorithms

Boundary-fill Algorithm

- In Boundary fill algorithm, area-filling is started at a point inside a region and paint the interior towards the boundary
- If the boundary is specified in a single color, the fill algorithm proceeds outward pixel by pixel until the boundary color is encountered
- This method is particularly useful in interactive painting packages, where interior points are easily selected
- Using a graphics tablet or other interactive device, an artist or designer can sketch a figure outline , select a fill color or pattern from a color menu, and pick an interior point
- The system then paints the figure interior
- To display a solid color region (with no border),the designer can choose the fill color to be the same as the boundary color
- A boundary fill procedure accepts as input the coordinates of an interior point(x, y), a fill color, and a boundary color
- Starting from (x, y), the procedure tests neighboring positions to determine whether they are of the boundary color. If not, they are painted with the fill color, and their neighbors are tested
- This process continues until all pixels up to the boundary color for the area has been tested
- Both inner and outer boundaries can be set up to specify an area, and some examples of defining regions for boundary fill are shown in Fig. 2-22.
- Figure 2-23 shows two methods for proceeding to neighboring pixels from the current test position.
- In Fig. 2-23(a), four neighboring points are tested. These are the pixel positions that are right, left, above, and below the current pixel. Areas filled by this method are called 4-onnected.
- The second method, shown in Fig. 2-23(b), is used to fill more complex figures. Here the set of neighboring positions to be tested includes the four diagonal pixels. Fill methods using this approach are called 8-connected.
-
- An 8-connected boundary-fill algorithm would correctly fill the interior of the area defined in Fig. 2-44, but a 4-connected boundary-fill algorithm produces the partial fill shown.

Figure 2-22: Exampl ary-fill

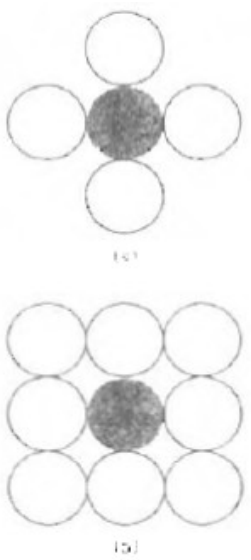


Figure 2-23: Fill methods applied to a 4-connected area (a) and to an 8-connected area **(b)** . Open circles represent pixels to be tested from the current test position, shown as a solid color

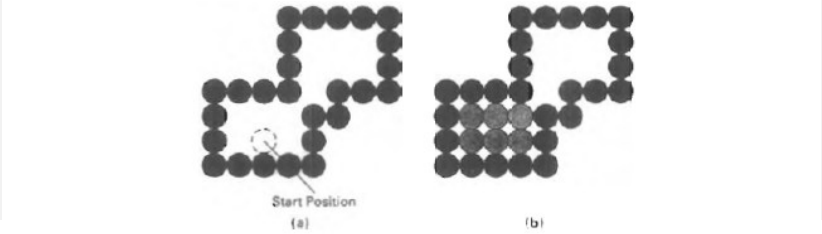


Figure 2-24: The area defined within the color boundary (a) is only partially

Procedure for boundary fill algorithm:

```
void boundaryFill4(int x, int y, int fill, int  boundary: integer)
{

    int current;

    current = get pixel(x, y) ;
    if (current != boundary) and (current != fill)

   {
        setColor (fill);

          set pixel(x, y);
          boundaryFill4 (x+1, y, fill, boundary);
          boundaryFill4 (x-1, y, fill, boundary);
          boundaryFill4 (x, y+1, fill, boundary);
          boundaryFill4 (x, y-1, fill, boundary);

   }

} // { end of procedure boundaryFill4}.
```

Flood-fill algorithm

- Flood-Fill Algorithm is used to fill in (or recolor) an area that is not defined within a single color boundary.
- Figure 2-25 shows an area bordered by several different color regions.
- We can paint such areas by replacing a specified interior color instead of searching for a boundary color value.
- 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.
- The process will be a recursive one and stopped when there is no neighboring pixels which can be colored.
- 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.

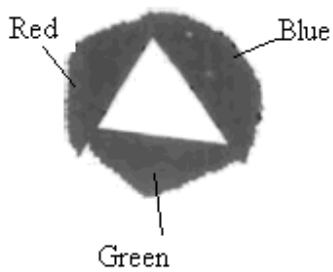


Figure 2-25: An area defined within

Procedure for flood fill algorithm:

```
void floodFill4 (int x, int y, int fillcolor, int oldcolor)
{
```

```
   if (getpixel(x, y) == oldcolor)

  {

          set pixel(x, y, fill color);
          floodFill4 (x+1, y, fill color, old color);
          floodFill4 (x-1, y, fill color, old color);
          floodFill4 (x, y+1, fill color, old color);
          floodFill4 (x, y-1, fill color, old color);

   }
} // {end of procedurefloodFill4}
```