COMPUTER GRAPHICS & IMAGE PROCESSING (21CS63)

VARALAKSHMI B D
ASSISTANT PROFESSOR
Department of Computer Science & Engineering

VI Semester

2023-24

What is Computer Graphics?

- "**Computer graphics** is a sub-field of computer science and is concerned with digitally synthesizing and manipulating visual content."
- **Computer graphics** deals with all aspects of creating images with a computer- Imaging, Rendering, Modeling, Animation.
  - Hardware: PC with graphics card for modeling and rendering.
  - Software: OpenGL
  - Applications: Any

Imaging (Representing 2D images)

- Image Representation
  - Sampling
  - Reconstruction
  - Quantization & Aliasing
- Image Processing
  - Filtering
  - Warping
  - Morphing
- Raster Graphics
  - Display devices
  - Color models

Image composition

Image morphing

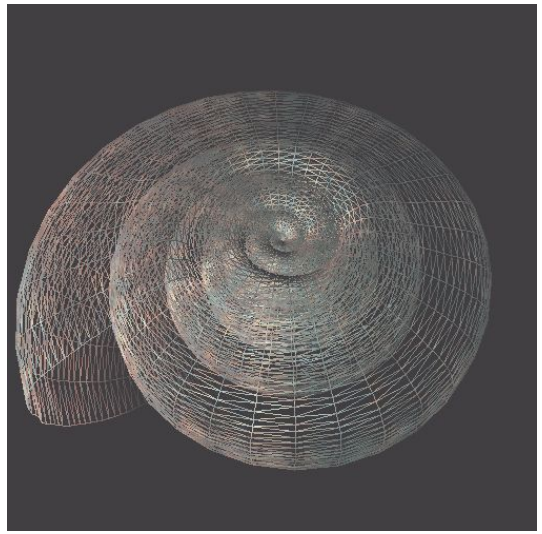
Rendering (Constructing 2D images from 3D models)

- 3D Rendering Pipeline
  - Modeling transformations
  - Viewing transformations
  - Hidden surface removal
  - Illumination, shading, and textures
  - Scan conversion, clipping
  - Hierarchical scene graphics
  - OpenGL
- Global illumination
  - Ray tracing
  - Radiosity

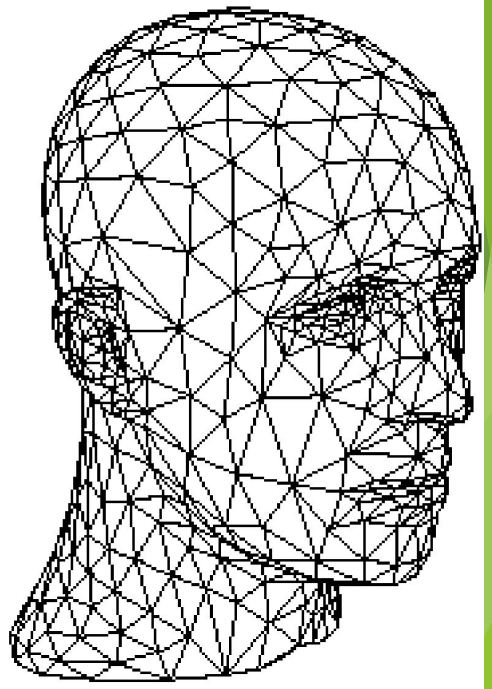
Modeling (Representing 3D objects)

- Representations of geometry
  - Curves
  - Surfaces
  - Solids
- Procedural modeling
  - Fractals mean

(relating to or of the nature of a fractal)



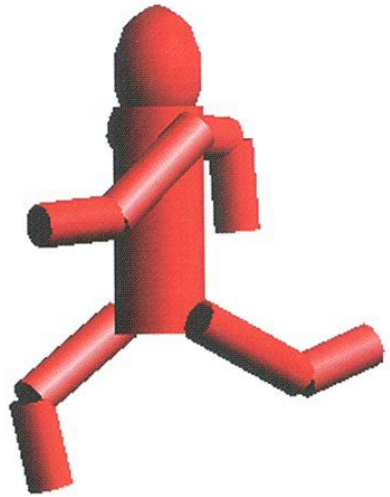
Shell



A curve or geometrical figure, each part of which has the same statistical character as the whole.

Animation (Simulating changes over time)

- Key framing
  - Kinematics
  - Articulated figures
- Motion capture
- Dynamics
  - Physically-based simulations



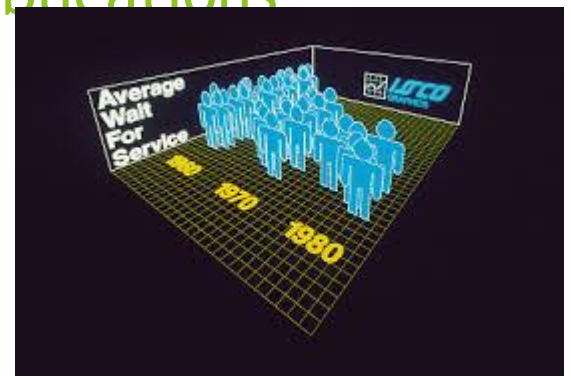
Ice Queen

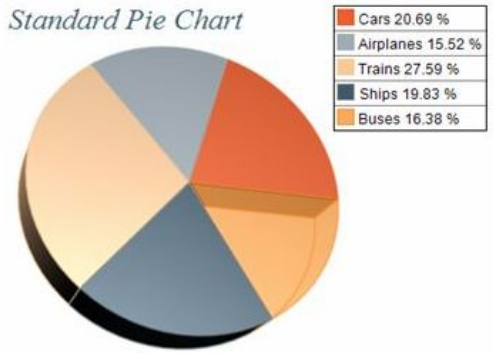
Graphics Applications

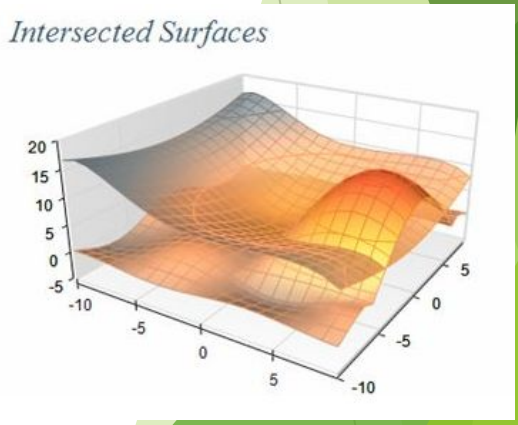
- Graphs and charts
- Computer-aided design
- Virtual-Reality Environments
- Scientific visualization
- Education and Training
- Computer art
- Entertainment
- Image Processing
- Graphical User Interfaces

Data plotting with dramatic effect, commonly used to summarize financial, statistical, mathematical, scientific, engineering, and economic data for research reports, managerial summaries, consumer information bulletins and other types of publications

- **Graphs and charts**
- Computer-aided design
- Virtual-Reality Environments
- Scientific visualization
- Education and Training
- Computer art
- Entertainment
- Image Processing
- Graphical User Interfaces

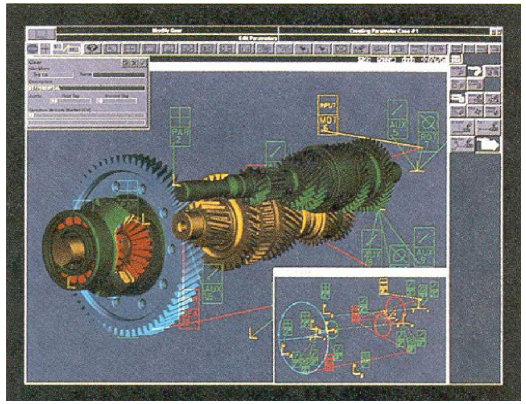






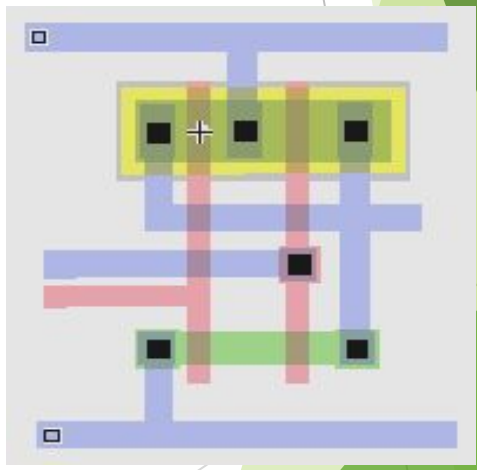
In design of building, automobiles, aircrafts, watercrafts, spacecrafts, computers, textiles, home appliances and a multitude of other products

- Graphs and charts
- **Computer-aided design**
- Virtual-Reality Environments
- Scientific visualization
- Education and Training
- Computer art
- Entertainment
- Image Processing
- Graphical User Interfaces



Gear Shaft Design

Los Angeles Airport



CMOS Circuit Design

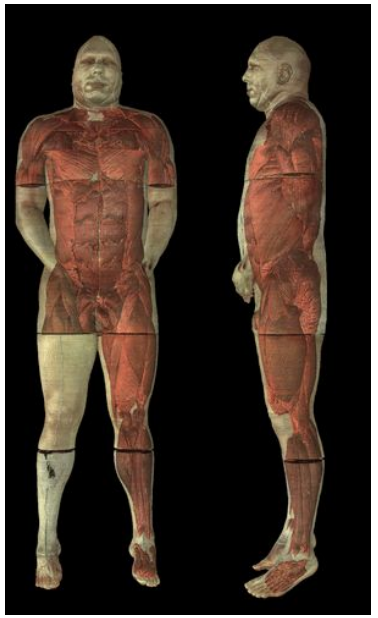
Contd…

Such animations are often used to train heavy-equipment operators, analyze the effectiveness of various cabin configuration and control placements, users can interact with the objects in a 3D scene.

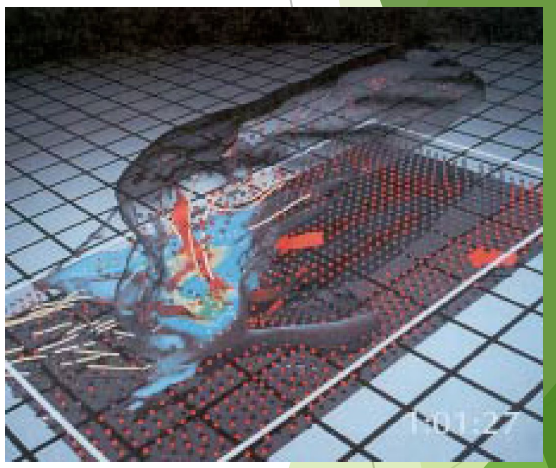
- Graphs and charts
- Computer-aided design
- **Virtual-Reality Environments**
- Scientific visualization
- Education and Training
- Computer art
- Entertainment
- Image Processing
- Graphical User Interfaces

Also called data visualization: produces graphical representations for scientific, engineering and medical data sets and processes it. Business visualization is used in connection with data sets related to commerce, industry, and other nonscientific areas. Also used in understanding and analyzing complex processes and mathematical functions

- Graphs and charts
- Computer-aided design
- Virtual-Reality Environments
- **Scientific visualization**
- Education and Training
- Computer art
- Entertainment
- Image Processing
- Graphical User Interfaces



Visible human

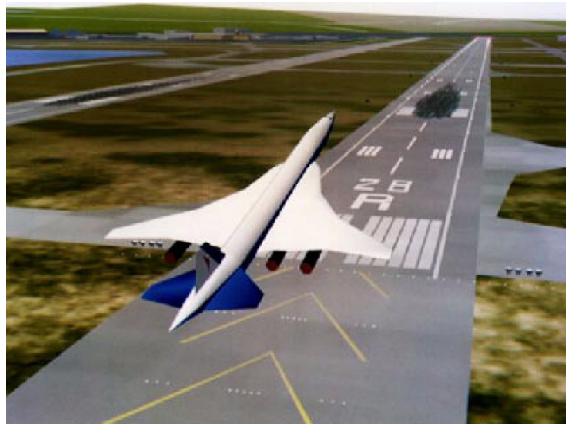


Airflow inside a thunderstorm

Contd…

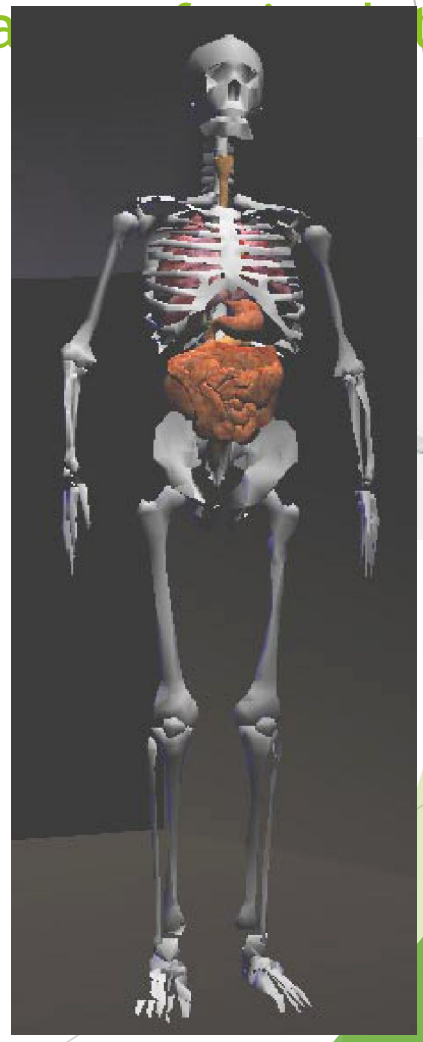
Computer generated models of physical, financial, political, social, economic, and other systems are often used as educational aids. Eg: color-coded diagram in operation of a nuclear reactor, aircraft, naval, spa tors

- Graphs and charts
- Computer-aided design
- Virtual-Reality Environments
- Scientific visualization
- **Education and Training**
- Computer art
- Entertainment
- Image Processing
- Graphical User Interfaces



Flight Simulation

Driving Simulation



Flight Simulation

Graphics are used in ***Fine Art***, ***Commercial Art*** applications and ***Mathematical Art***. The artist uses a combination of 3D modeling packages, texture mapping, drawing programs and CAD software. Mathematical Art is produced using mathematical functions and fractal procedures. Pen plotters create ***Automatic Art***

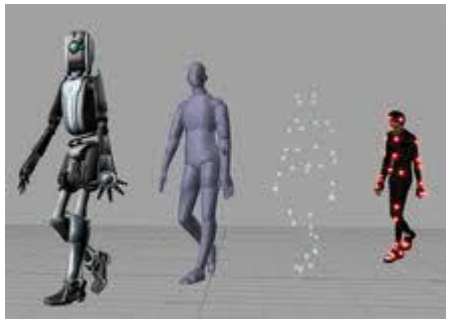
- Graphs and charts
- Computer-aided design
- Virtual-Reality Environments
- Scientific visualization
- Education and Training
- **Computer art**
- Entertainment
- Image Processing
- Graphical User Interfaces

Graphics are commonly used in making motion pictures, music videos and television shows. Graphics objects can be combined with live actions. Eg. Avtar movie

- Graphs and charts
- Computer-aided design
- Virtual-Reality Environments
- Scientific visualization
- Education and Training
- Computer art
- **Entertainment**
- Image Processing
- Graphical User Interfaces

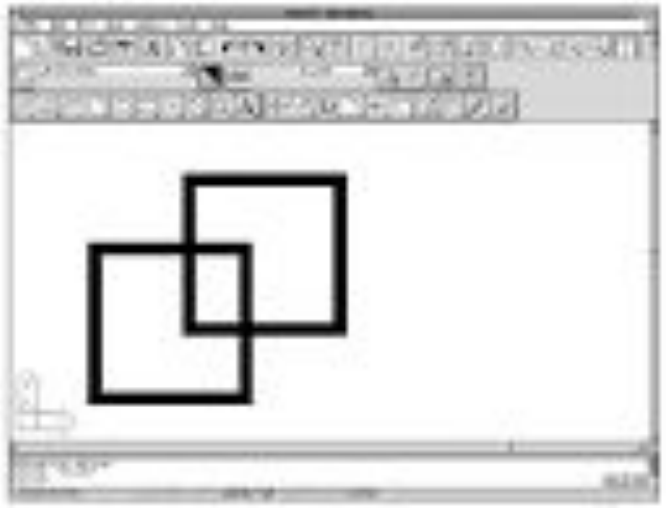
Image processing applies techniques to modify or interpret existing pictures (photographs, TV scans).to apply image processing methods, the image must be digitized first. Medical applications also make extensive use of image processing techniques for picture enhancement, simulations of surgical operations. It used to improve picture quality, analyze or recognize visual patterns.

- Graphs and charts
- Computer-aided design
- Virtual-Reality Environments
- Scientific visualization
- Education and Training
- Computer art
- Entertainment
- **Image Processing**
- Graphical User Interfaces

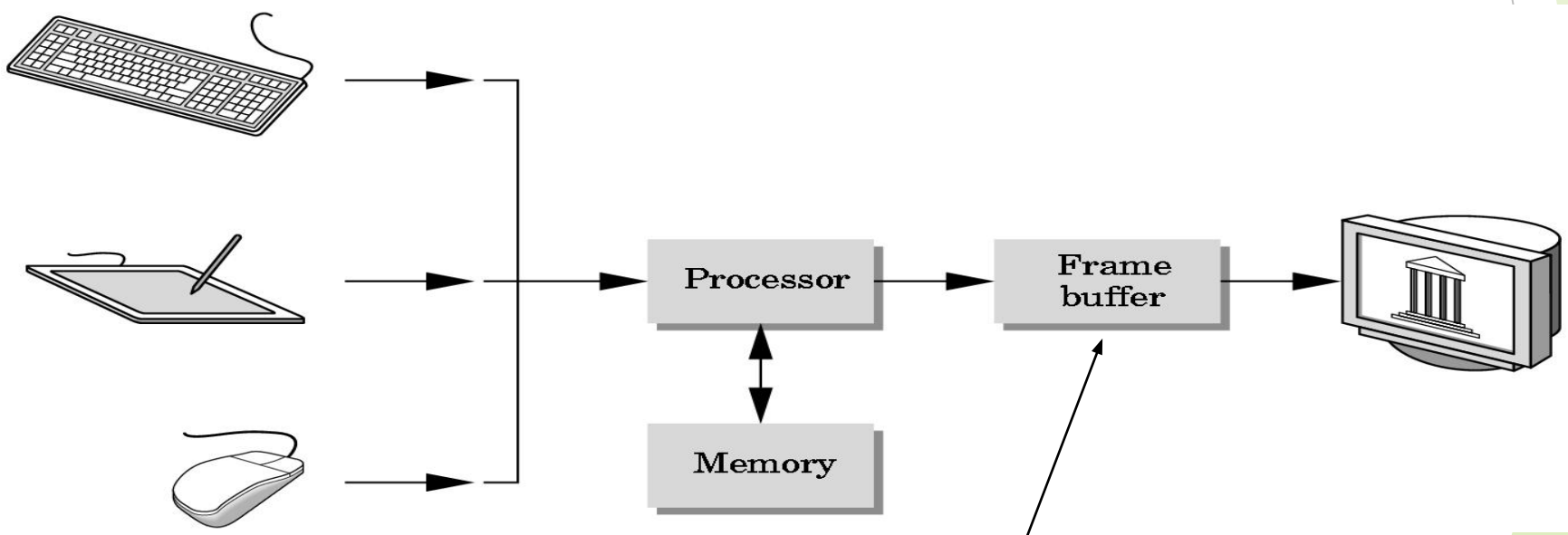


User is provided with point and click facility to select menu items, icons and objects on the screen. User can manage multiple windows simultaneously. Word-processing spreadsheets and desktop-publishing programs are typical applications that take advantage of such user interface techniques.

- Graphs and charts
- Computer-aided design
- Virtual-Reality Environments
- Scientific visualization
- Education and Training
- Computer art
- Entertainment
- Image Processing
- **Graphical User Interfaces**



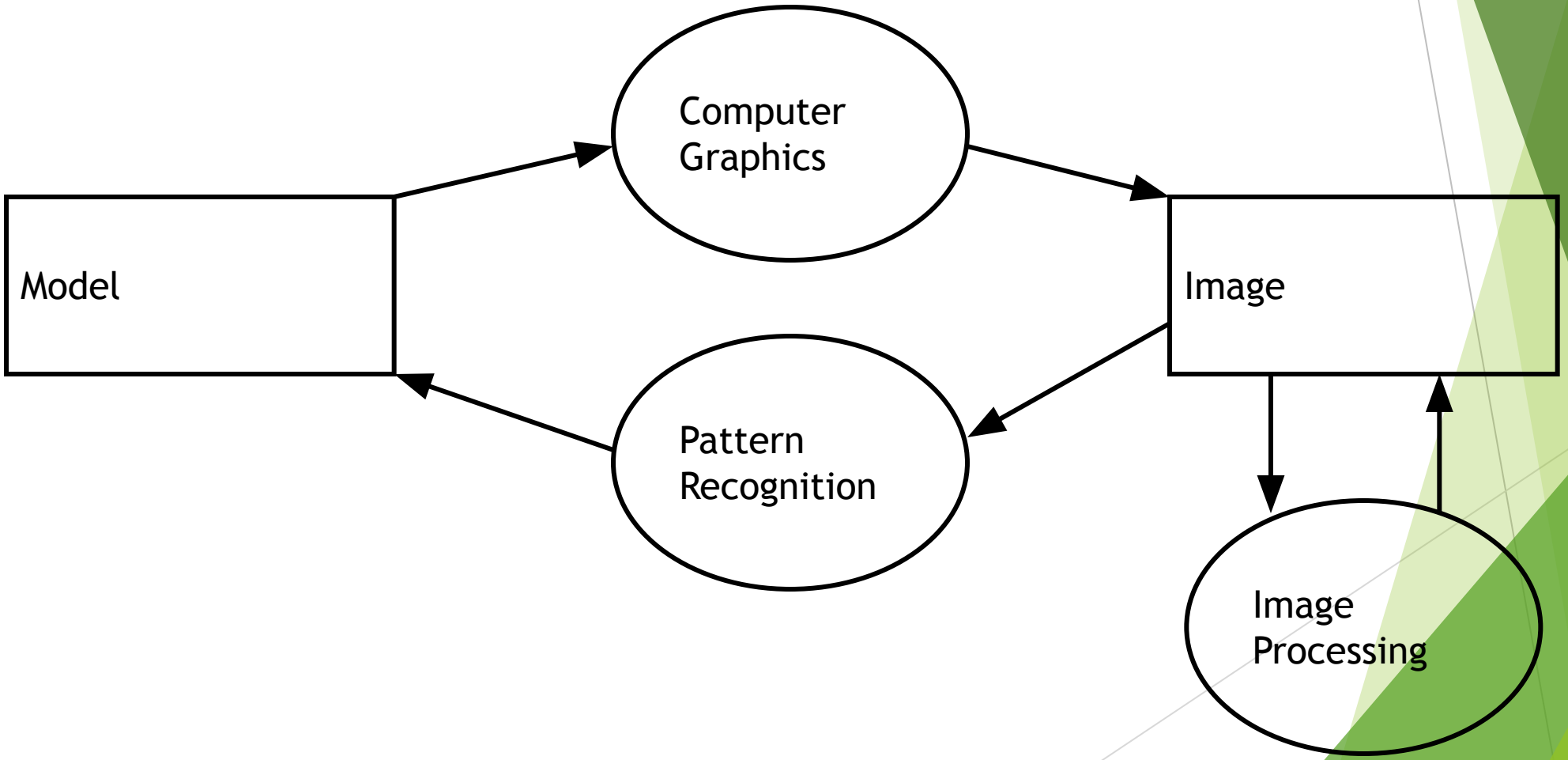
Basic Graphics System



Input devices

Image formed in FB

Also…

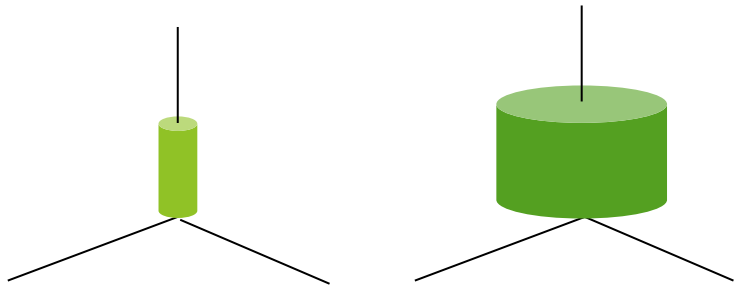

Computer
Graphics
Model
Image
Pattern
Recognition
Image
Processing

From model to image

Graphics pipeline

Coordinates and transformations

From model to image



Geometric modeling

Cylinde

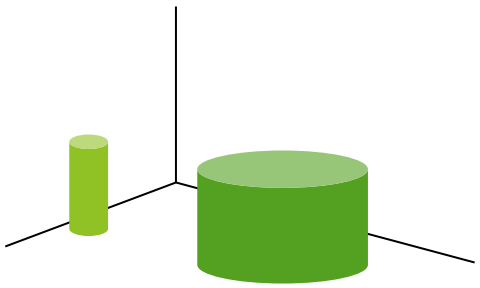
r:

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

$$0 \leq z \leq h$$

Local or modeling coordinates

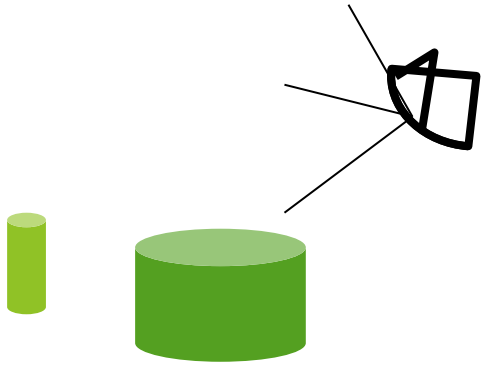
From model to image



Position cylinders in scene:

World coordinates

From model to image

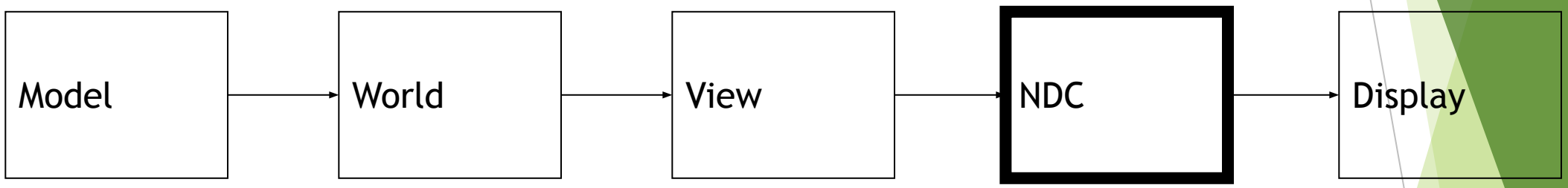


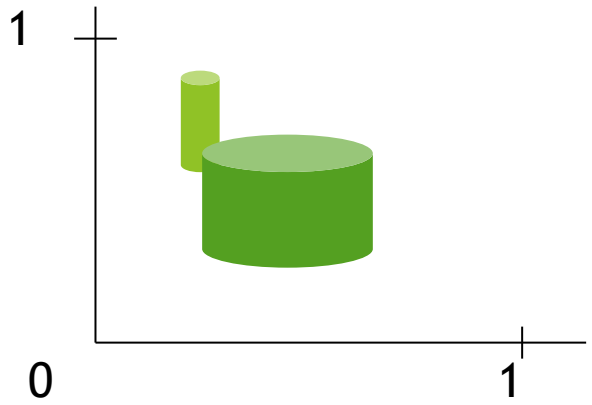
Visible surfaces, shading

Look at cylinders, project on virtual screen

Viewing coordinates

From model to image

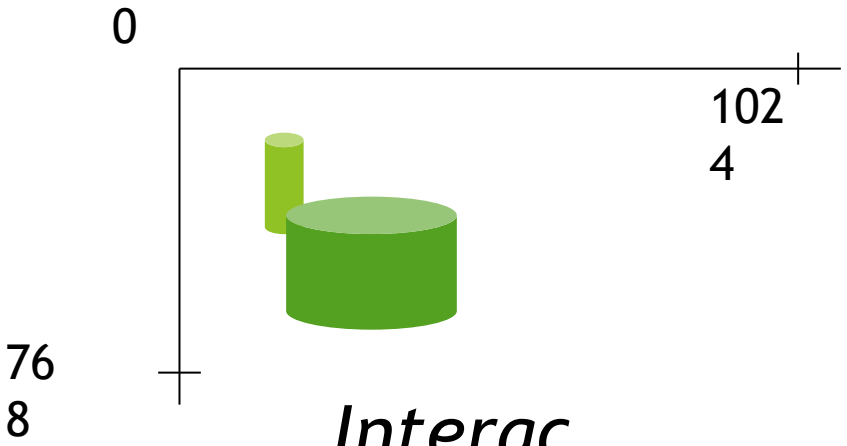




Display:

Normalized Device Coordinates

From model to image



Interac tion

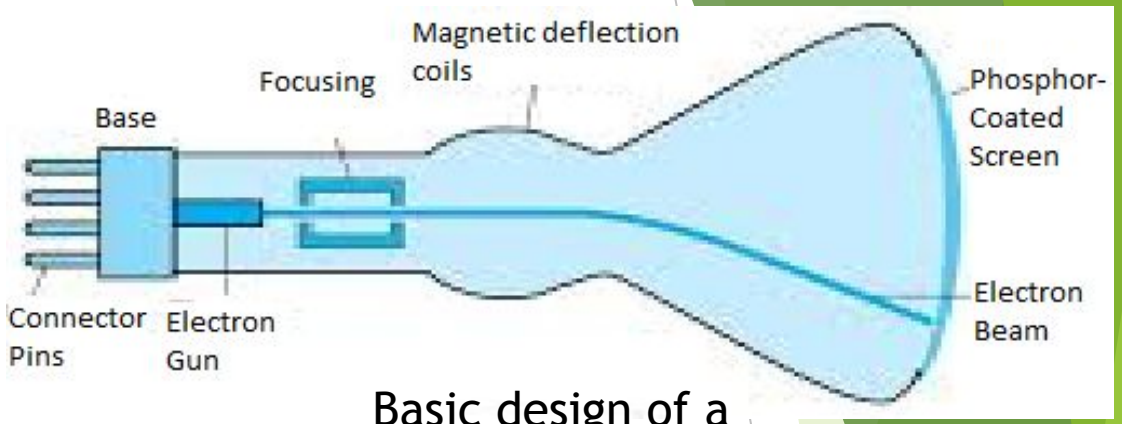
Display on screen:

Device Coordinates

Video Display Devices

- Emissive display - Convert electrical energy into light
  - Cathode ray tube (CRT)
  - Flat panel CRT
  - Plasma panels (gas-discharge display)
  - Thin-film electroluminescent (EL) display
  - Light-emitting diodes
- Non-Emissive display - Optical effect: convert sunlight or light from other source into graphic patterns.
  - Liquid-crystal device (LCD) – flat panel
  - Passive-matrix LCD
  - Active-matrix LCD

Cathode Ray Tube (CRT)



Basic design of a magnetic-deflection CRT

Cathode Ray – beam of electrons

- emitted by an electron gun
- accelerated by a high positive voltage near the face of the tube
- forced into a narrow stream by a focusing system
- directed toward a point on the screen by the magnetic field generated by the deflection coils
- hit onto the phosphor-coated screen
- phosphor emits visible light, whose intensity depends on the number of electrons striking on the screen

Cathode Ray Tube

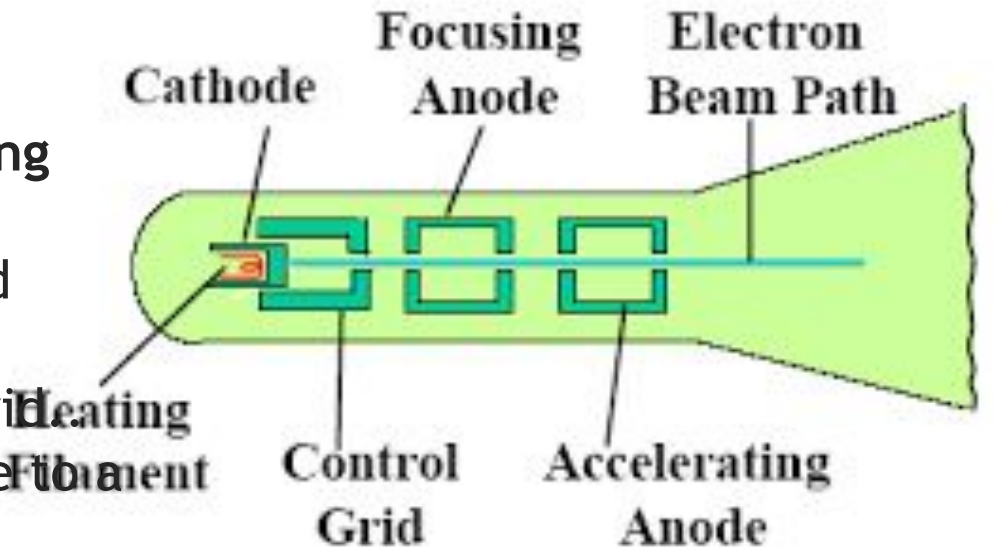
Operation of an electron gun with an accelerating anode.

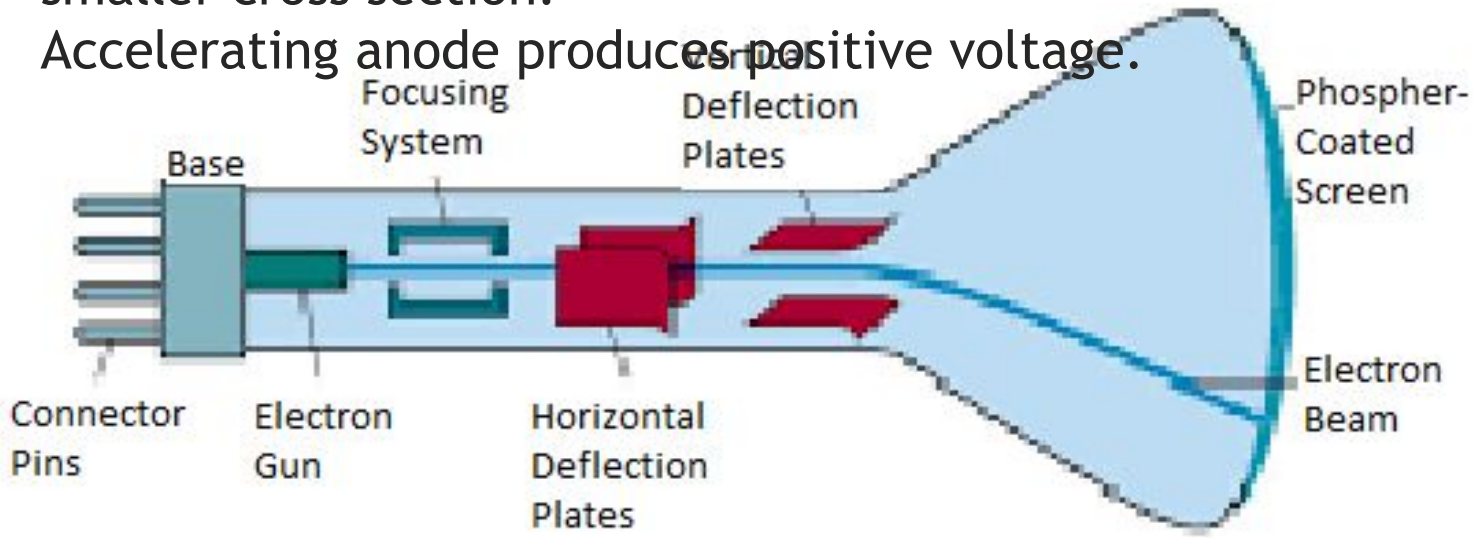
2 primary components: Heated metal cathode and control grid.

Intensity is controlled by voltage at the control grid.

Focusing system forces electron beam to converge to a smaller cross section.

Accelerating anode produces positive voltage.





Electrostatic deflection of the electron beam in a CRT

Cathode Ray Tube (conti…)

- **Phosphor Persistence (PP)**
  - the light output decays (fades) exponentially with time.
  - a phosphor's persistence is defined as the time from the removal of excitation to the moment of decaying the light to one-tenth of its original intensity.
  - one way: store the picture information as charge distribution, that can be used to keep the phosphor activated.
  - now: redraw the picture repeatedly by quickly directing the electron beam back over the same screen points.
  - low persistence -> good for animation
  - high persistence -> good for static picture with high complexity
  - typical range: 10ms – 60ms
- **Refresh rate (RR)** - number of times per second the image is redrawn (e.g., 60 or higher)
- **Critical fusion frequency (CFF)** - the refresh rate above which a picture stops flickering and becomes steady
- **Longer PP -> lower CFF required**

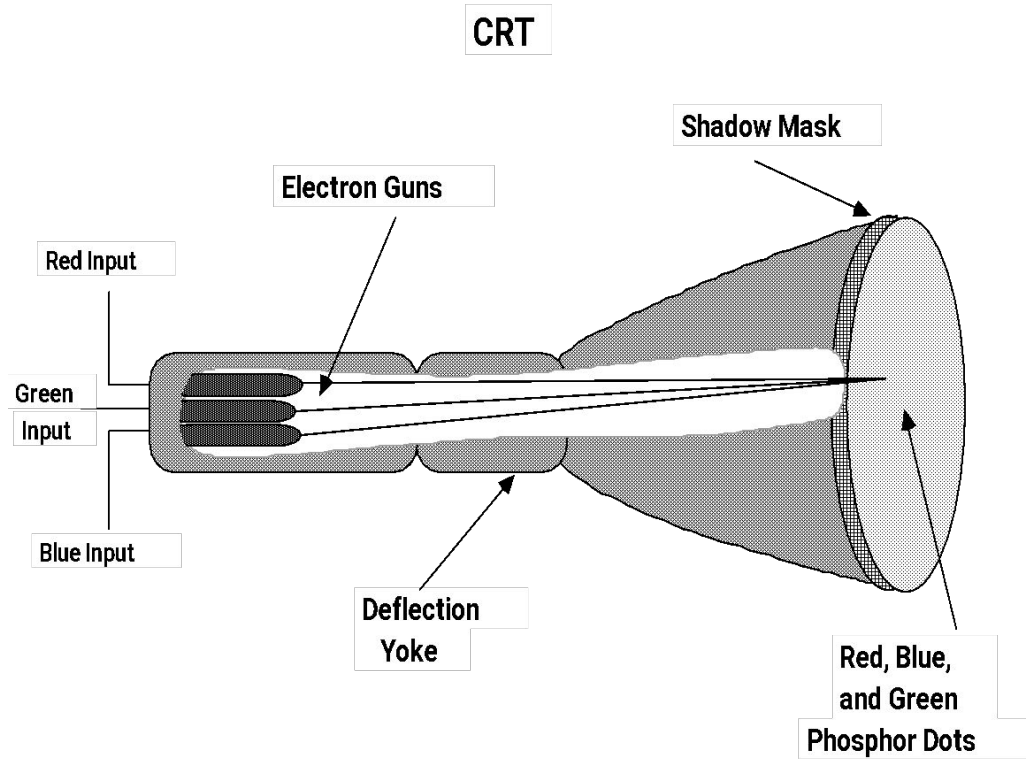
Properties of the CRT

- Resolution
  - the maximum number of points that can be displayed without overlap on a CRT
  - high-definition system, e.g. 1280 * 1024 pixels
  - resolution depends on the type of phosphor, the intensity to be displayed, focusing and deflection systems, size of video memory

- Horizontal scan rate
  - the number of scan lines per second that the CRT is able to display
  - refresh rate * number of scan lines per frame

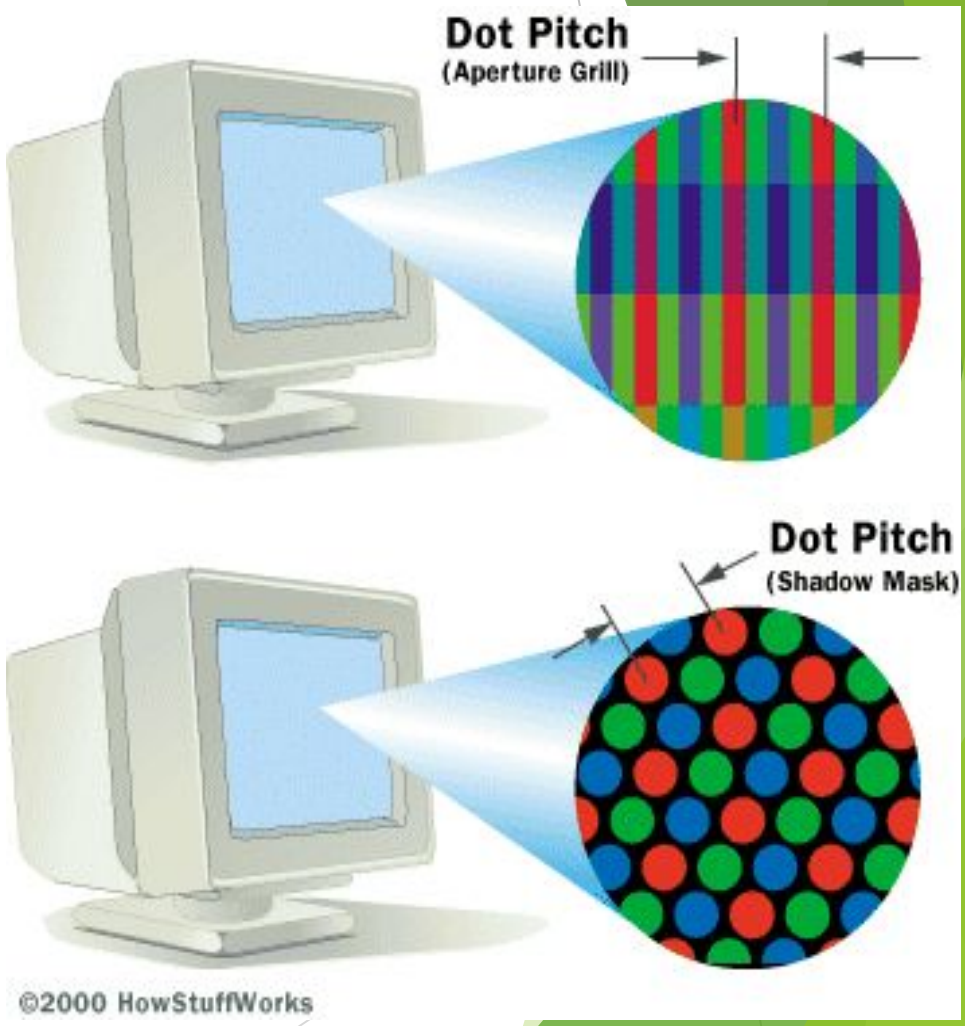
Color CRT Monitor

- ✔ Displays color pictures by using a combination of phosphors that emit different-colored light and merges to form a single perceived color.



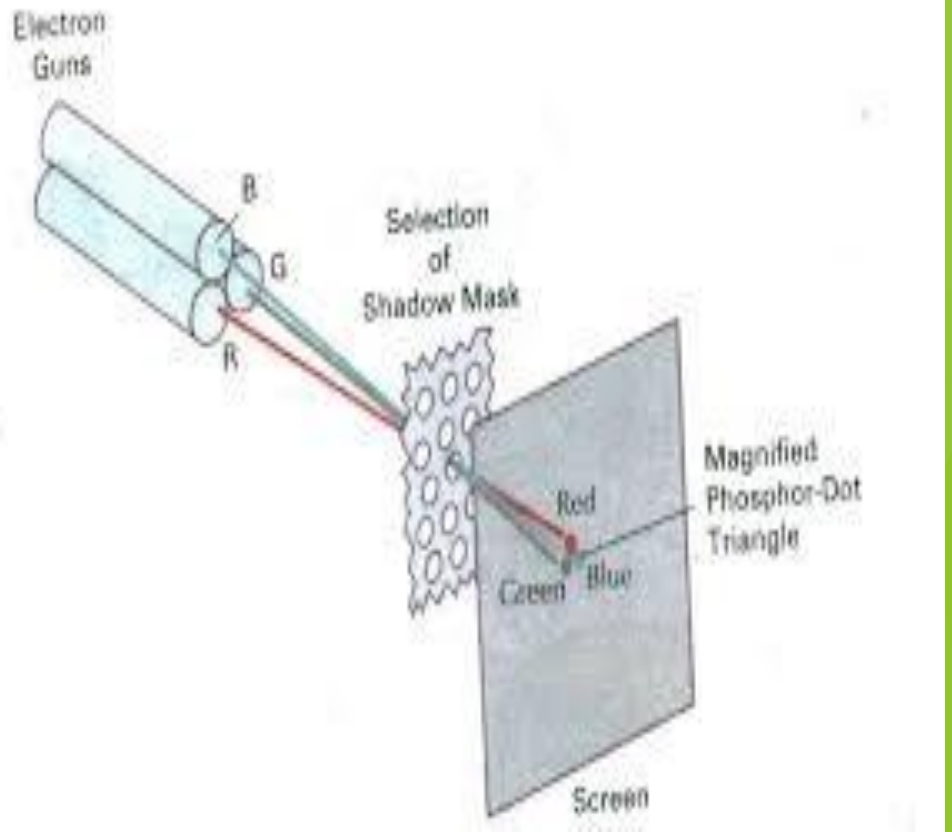
Beam Penetration Method

- Coat the screen with layers of different colored phosphors.
  - ✔ Beam of slow electrons excite outer red layer, beam of very fast electrons penetrates through red layer and excites the inner green layer.
  - ✔ (Drawback)Limited to number of colors since only two colors used.
  - ✔ (Drawback)Picture quality is not as good as with the other method.
- Dot Pitch –the spacing between pixels on a CRT, measured in millimeters. Generally, the lower the number, the more detailed the image.



Shadow Mask Method

- Uses 3 phosphor color dots at each pixel position, each emitting one of red, blue and green lights.
- It has 3 electron guns, one for each color dot and shadow mask grid just behind the phosphor-coated screen.
- Shadow mask contains series of holes aligned with phosphor dot patterns, one hole for each phosphor triad
- When the 3 beams pass through a hole in the shadow mask, they activate a dot triangle, which appears as 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.
- The number of electrons in each beam controls the amount of red, blue and green light generated by the triad.
- Another configuration is the in-line arrangement, in which the electron guns and corresponding color dots are aligned along one scan line.



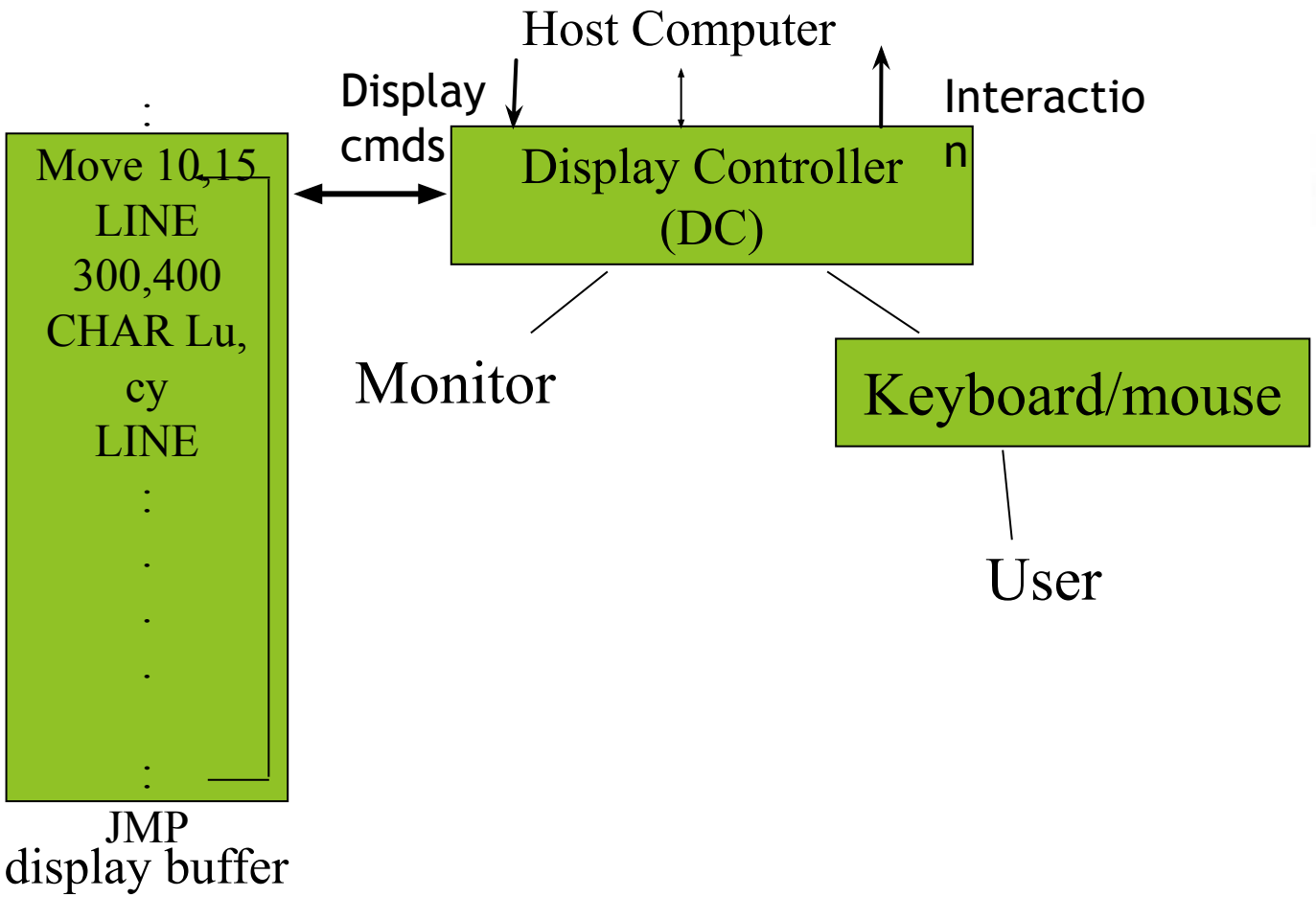
Architectures for displaying objects

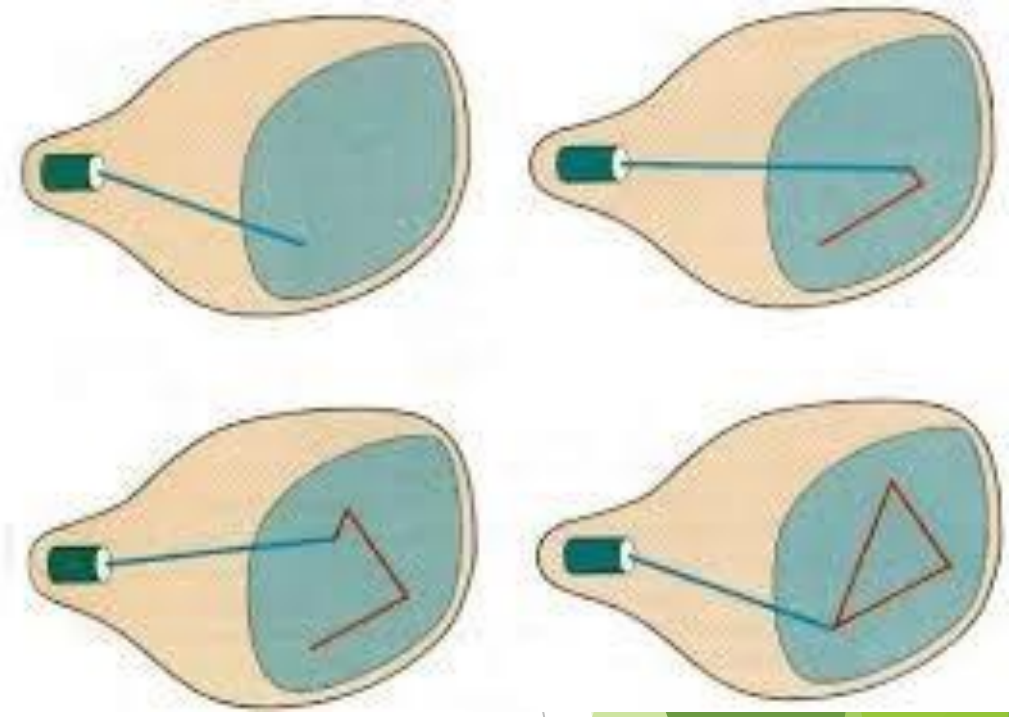
- Vector system Architecture (Random-Scan)
  - ✔ Line drawing and stroke drawing in a random order
- Raster Refresh Architecture(Raster-Scan)
  - ✔ Horizontal scan line order

Vector System Architecture(Random-Scan), consists of:

- Vector system consists of: display processor (controller), display buffer memory, CRT
  - ✔ In addition to the CPU, a special processor, called the **Video controller** or **Display controller**, is used to control the operations of the display device - Connected as an I/O peripheral to the CPU.
  - ✔ **Display buffer memory**: stores computer-produced **display list** or **display program**- contains point and line plotting commands
  - ✔ The commands interpreted by the display processor sends digital and point co-ordinates to vector generator that converts digital co-ordinate values to analog voltages for beam-deflection circuits that displays an electron beam writing on the CRT's phosphor coating.
  - ✔ Display commands dictate the beam deflection from endpoint to endpoint. This technique is called Random-Scan - Line drawing and stroke drawing in a random order.
  - ✔ Display list needed to be refreshed(Display processor must cycle through the display list- e.g 30Hz)

Vector Display

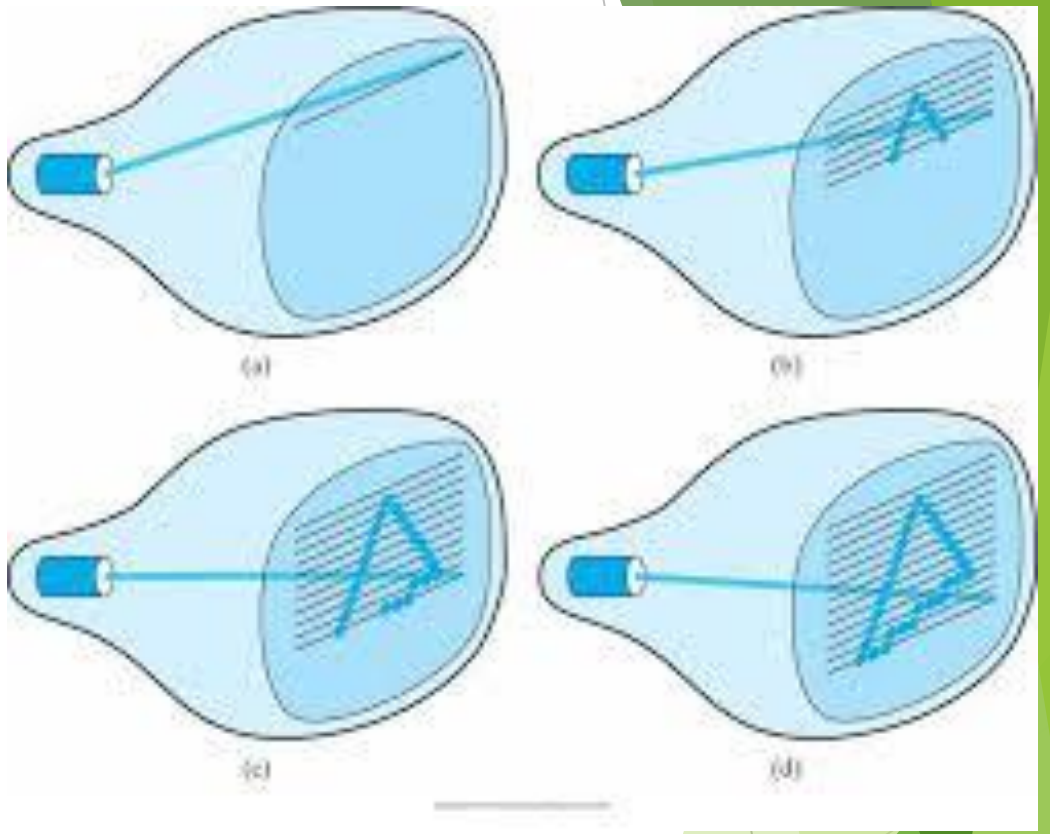


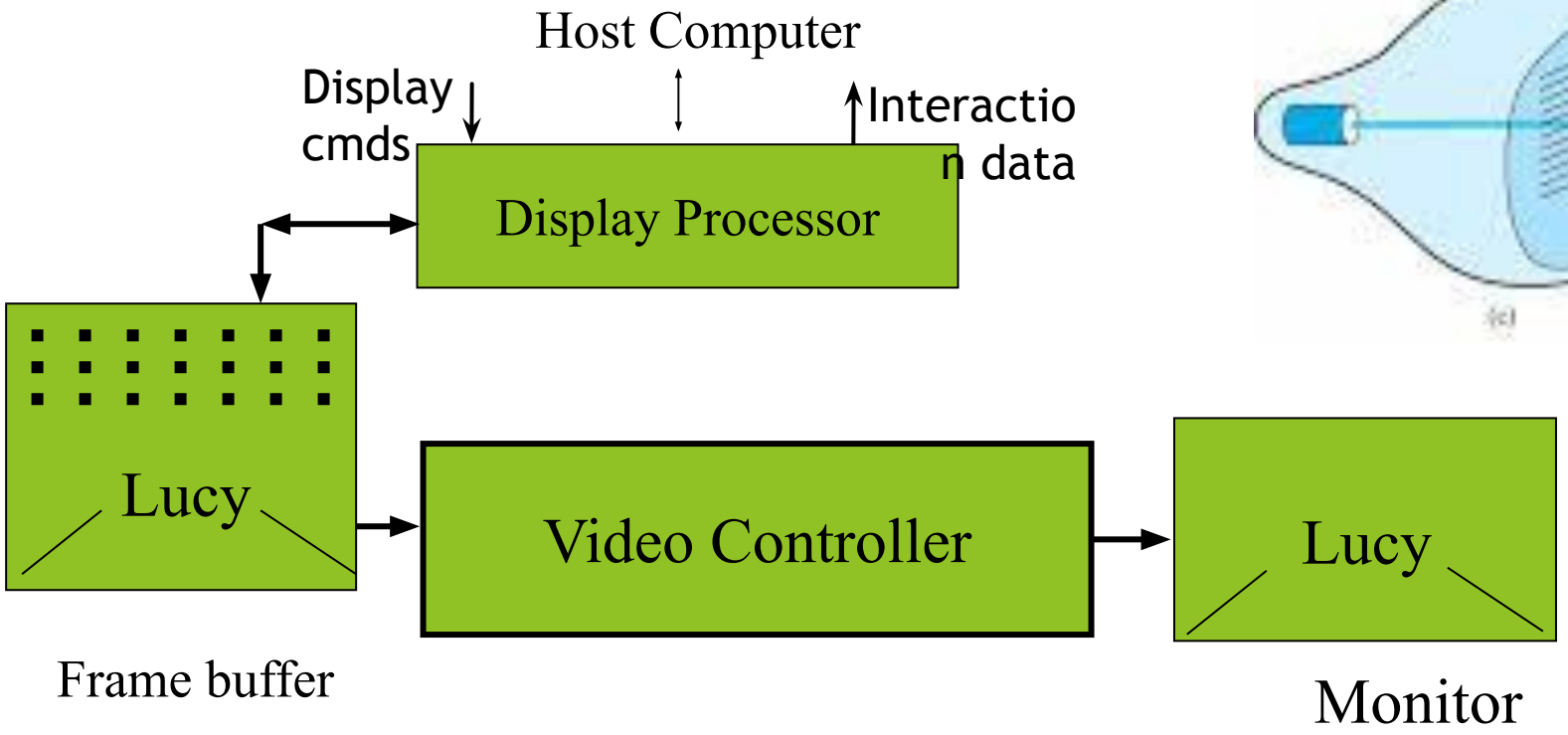


Raster Display (since 1970s)

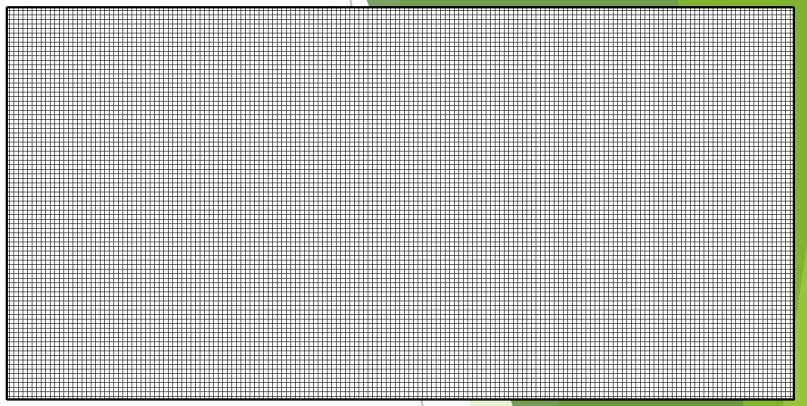
- Raster system consists of display processor (input, refreshing, scan converting), video controller, buffer memory (frame buffer), CRT.
- The refresh buffer stores the *display primitives*(lines char and solid shaded or patterned areas), rather than display list or display program.
- Raster: set of horizontal raster lines each a row of individual pixels and thus stored as a matrix of pixels representing entire screen.
- Video controller reads these pixel contents to produce the actual image on the screen, one raster line at a time, from top to bottom and then back to top.
- need refresh the raster display (e.g., 60Hz)

Raster Display

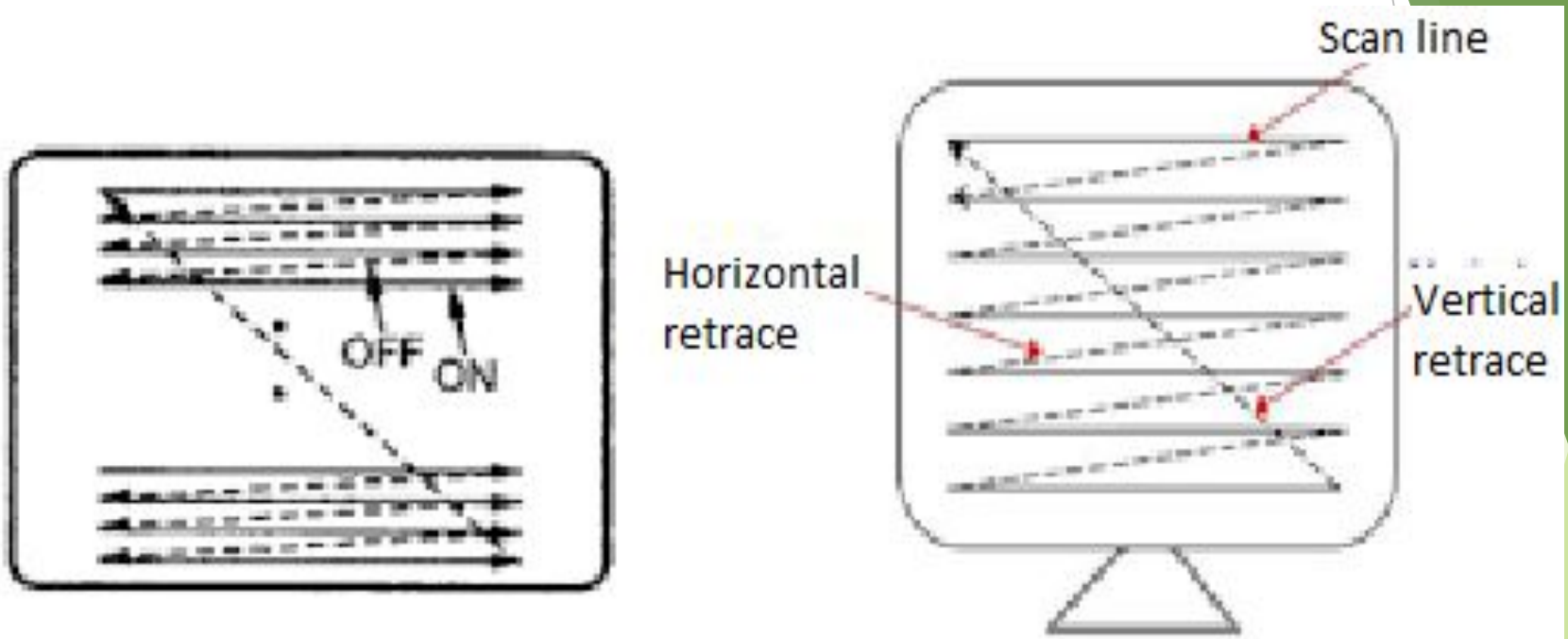




Basic definitions



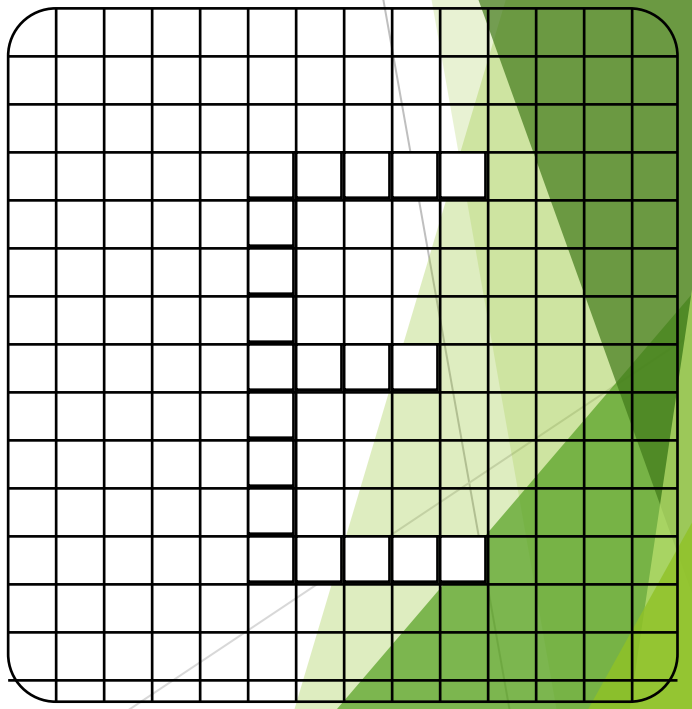
- Video raster devices display an image by sequentially drawing out the pixels of the scan lines that form the raster.
  - **Raster**: A rectangular array of points or dots.
  - **Pixel** (Pel, picture elements): One dot or picture element of the raster, defined as smallest addressable area on screen.
  - **Scan line**: A row of pixels.
  - **Resolution**: # of pixel positions that can be plotted.
  - **Aspect Ratio**: # of horizontal points to vertical points(or vice versa).
  - **Depth**: # of bits per pixel in a frame buffer.
  - **Bitmap**: a frame buffer with one bit per pixel
  - **Pixmap**: a frame buffer with multiple bits per pixel.



- **Horizontal retrace**: return to the left of the screen, after refreshing each scan line.
- **Vertical retrace**: electron beam returns to the top left corner of the screen to begin the next frame.
- **Refresh rates** are described in units of cycles per seconds, or Hertz(Hz).

Scanning An Image

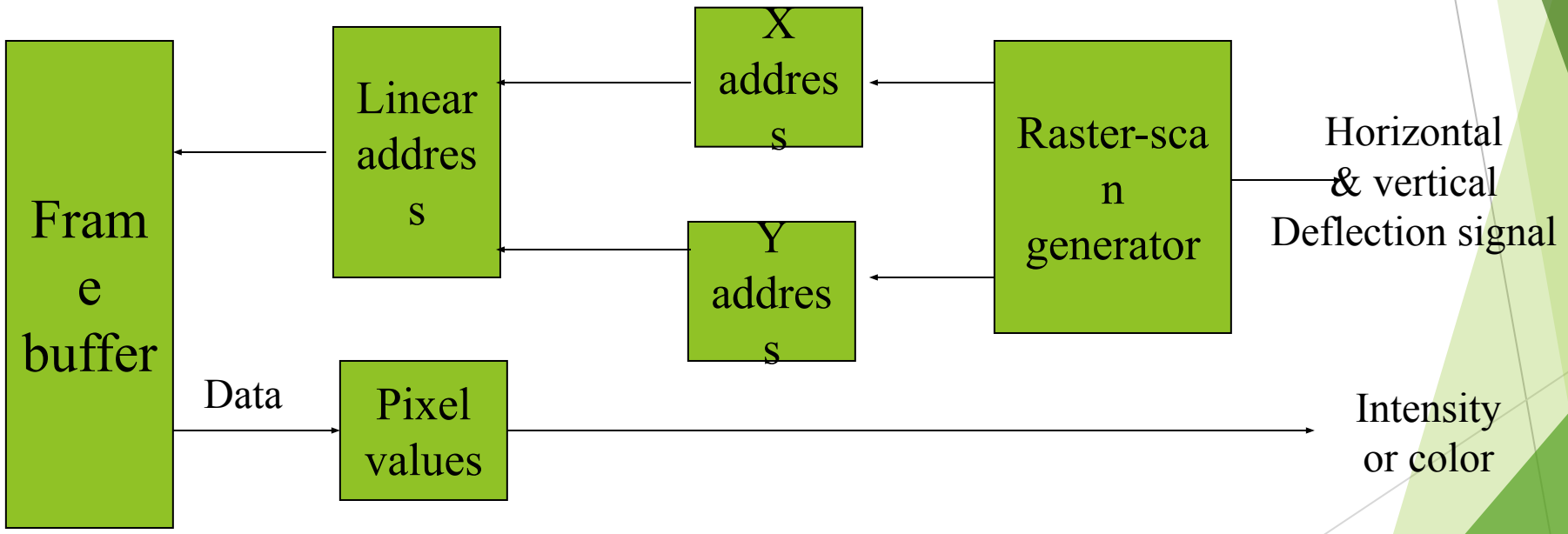
- Frame: The image to be scanned out on the CRT.
- Some minimum number of frames must be redisplayed (or refreshed) each second to eliminate flicker in the image.



- Critical Fusion Frequency --The refresh rate above which a picture stops flickering and fuses into a steady image is called the critical fusion frequency.
- Typically 60 times per second for raster displays.
- Varies with intensity, individuals, phosphor persistence, room lighting.

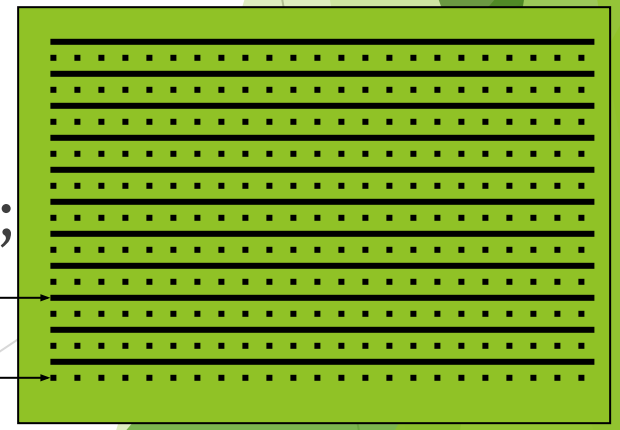
Video Controller

- Access the frame buffer to refresh the screen
- Control the operation for display
- Color look-up table



Types of refresh

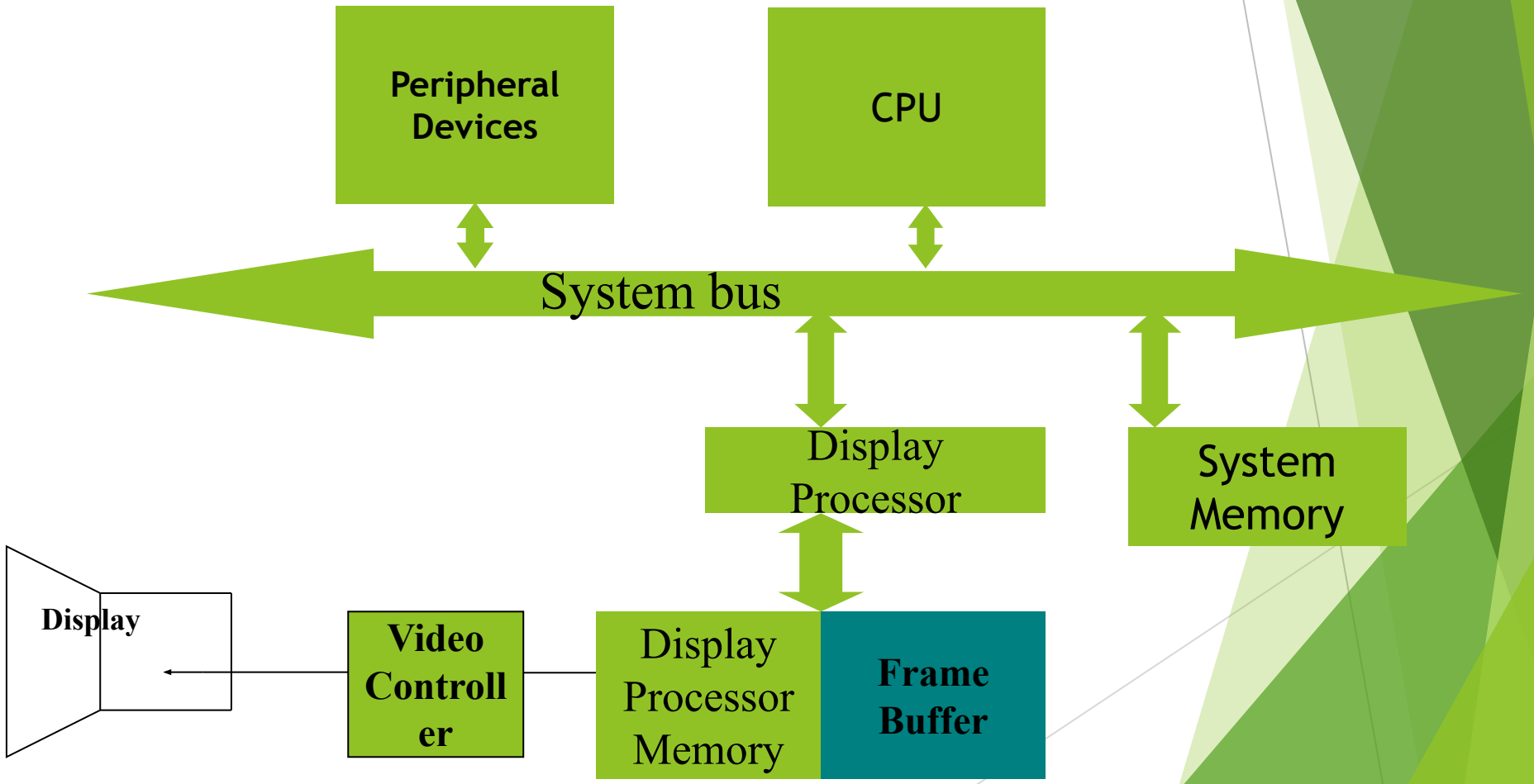
- Interlaced (mostly for TV for reduced flickering effect -**National Television System Committee** NTSC) each frame is displayed in two passes or two fields
  - ✔ **First pass**: The beam sweeps across every other scan line form top to bottom. odd-field: odd-numbered scan lines
  - ✔ **Second pass**: After vertical retrace, the beam then sweeps out the remaining scan lines. even-field: even-numbered scan lines
  - ✔ **Refresh rate**: e.g., NTSC: 60Hz (60 fields per second); 30 frame/s. PAL(Phase Alternating Line) : 50Hz
- Non-interlaced (mostly for monitor)
  - ✔ refresh rate: e.g., 60Hz or more



Display Processor

- Also called either a Graphics Controller or Display Co-Processor
- Specialized hardware to assist in scan converting output primitives into the frame buffer.
- Fundamental difference among display systems is how much the display processor does versus how much must be done by the graphics subroutine package executing on the general-purpose CPU.

Architecture of a raster-graphics system with a display processor



Frame Buffer

A frame buffer may be thought of as computer memory organized as a two-dimensional array with each (x, y) addressable location corresponding to one pixel.

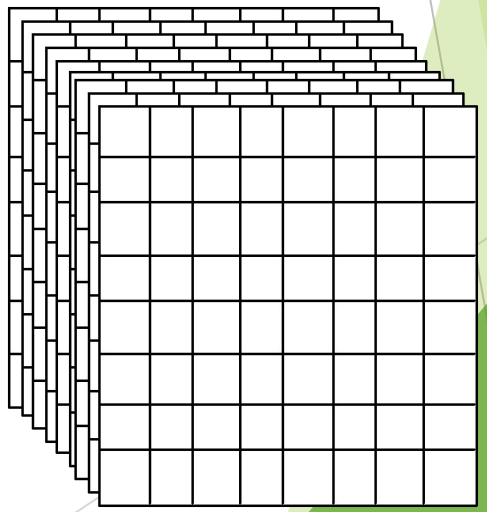
Bit Planes or *Bit Depth* is the number of bits corresponding to each pixel.

A typical frame buffer resolution might be

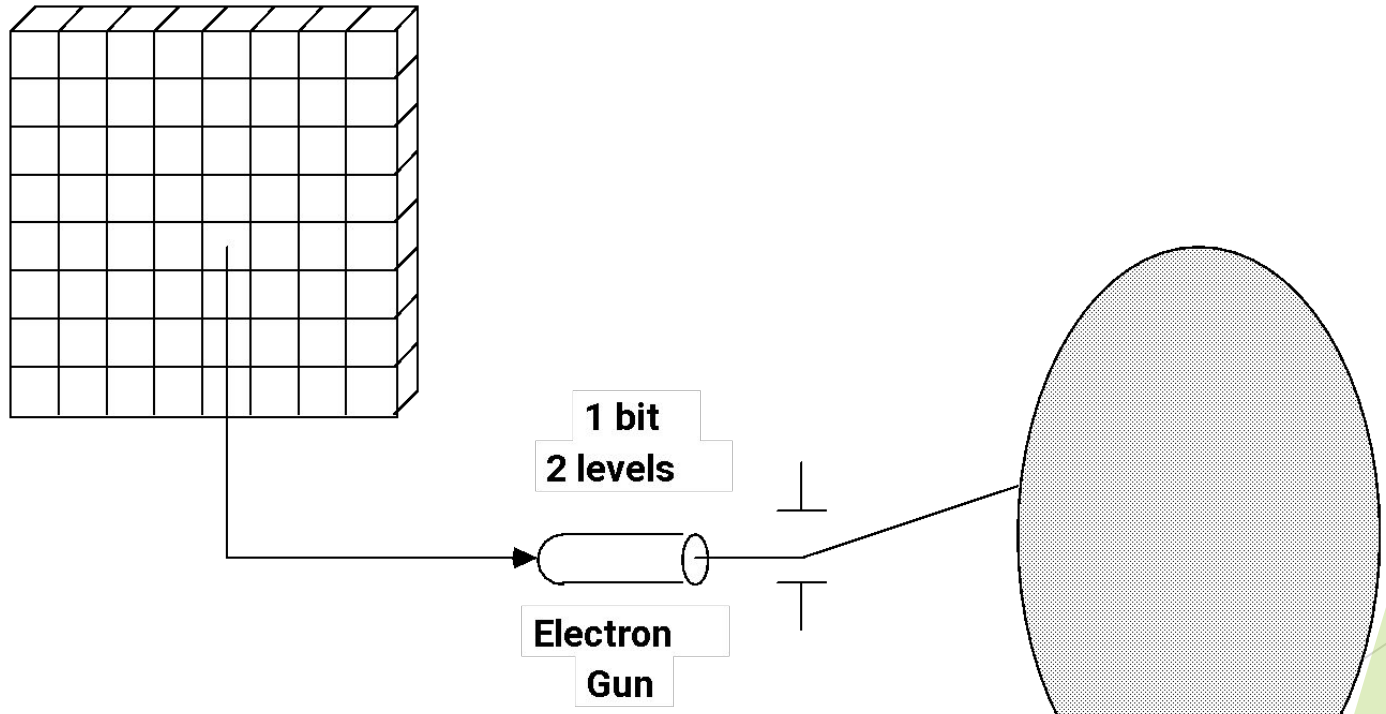
640 x 480 x 8

1280 x 1024 x 8

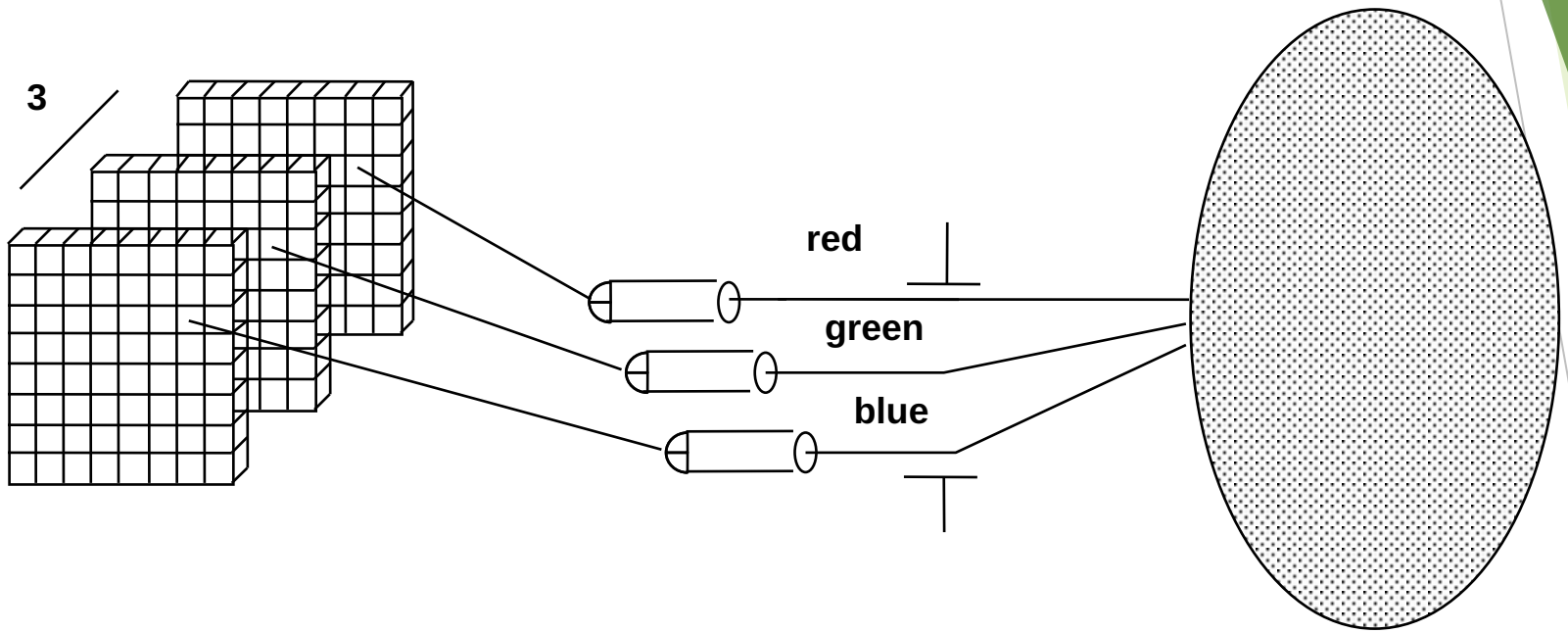
1280 x 1024 x 24



1-Bit Memory. Monochrome Display (Bit-map Display)

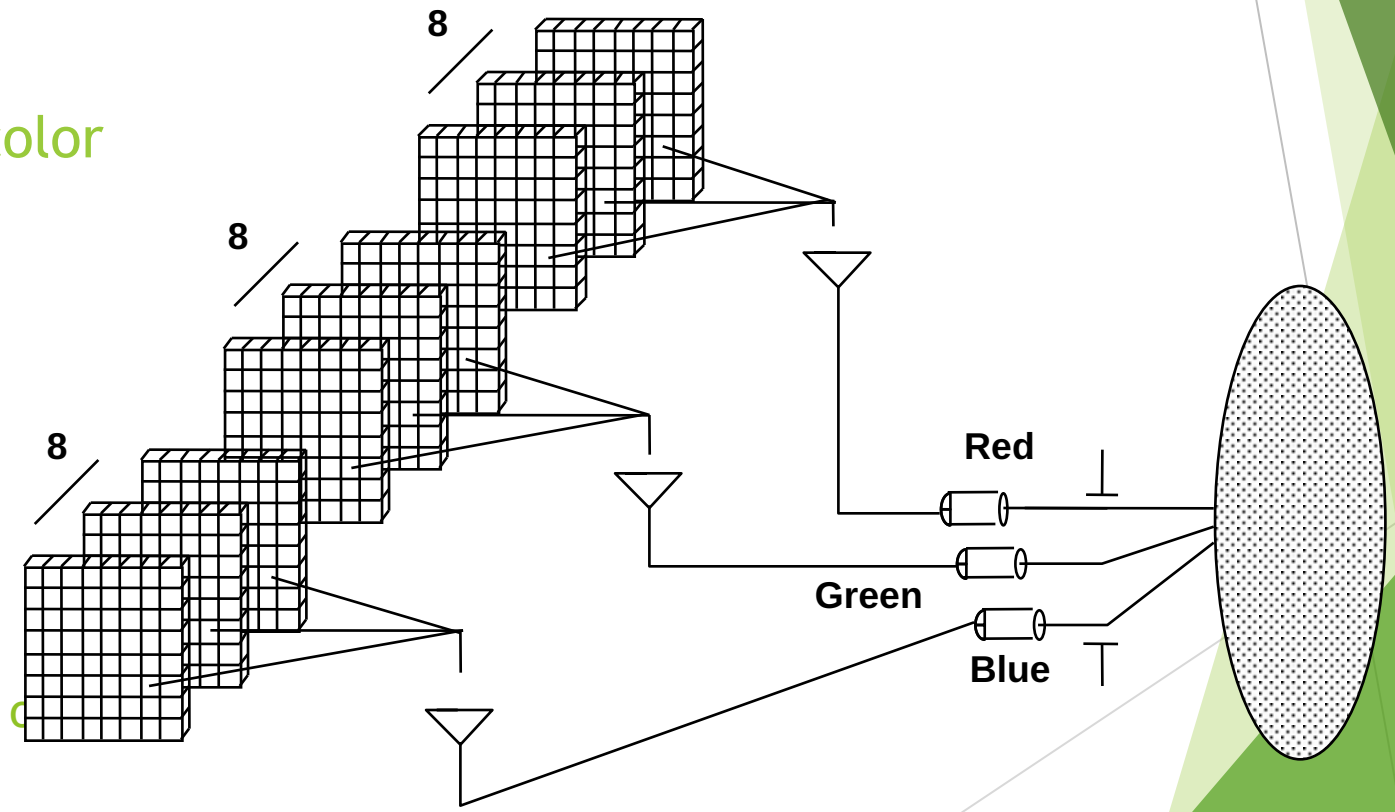


3-Bit Color Display



COLOR:	black	red	green	blue	yellow	cyan	magenta	white
R	0	1	0	0	1	0	1	1
G	0	0	1	0	1	1	0	1
B	0	0	0	1	0	1	1	1

True Color Display
24 bit planes, 8 bits per color gun.
2^{24} = 16,777,216 colors
32 bit true *Color current system*
2^{32} = 4.3 billion colors



(1280 X 1024 X 24) = 3,840 KB c
3,932,160 bytes

Color Look-up Table

- **LUT (Look-Up Table)**
- LUT has as many entries as there are pixel values, values in the bit planes are used as indices in one or more LUT.
- A pixel value is used not to control the beam directly, but rather as an index into the look-up table.
- The table entry's value is used to control the intensity or color of the CRT.
  for example:
  If each pixel consists of 8 bits in the frame buffer the LUT requires a table with 256 entries.
  Pixel value 67 ▫ access the content in the entry 67 of the table ▫ use the color
  content to control the CRT beam
- The total number of bits in each table entry is called the *width of the LUT,*
  which is the capability for providing all possible colors
- The look-up operation is done for each pixel on each display cycle, fast access of the table is required

Color Map Look-Up Tables

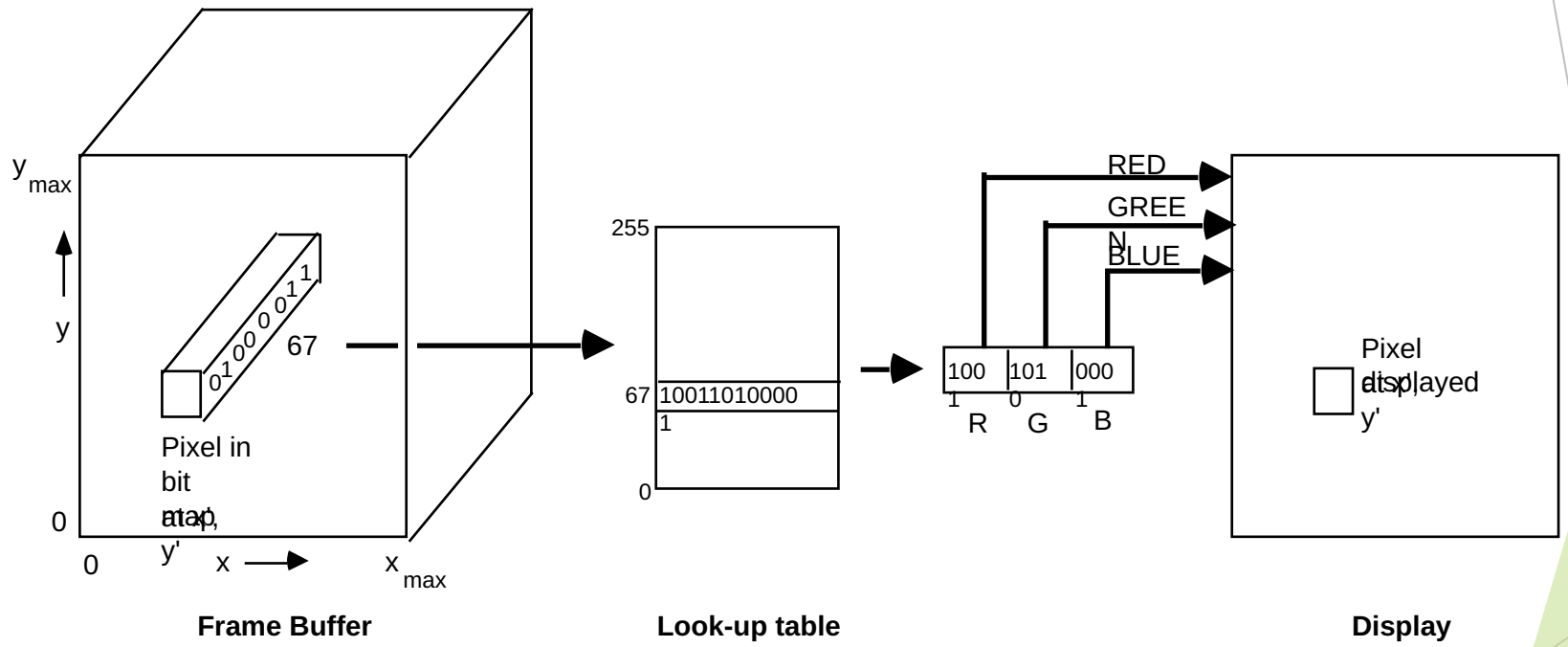
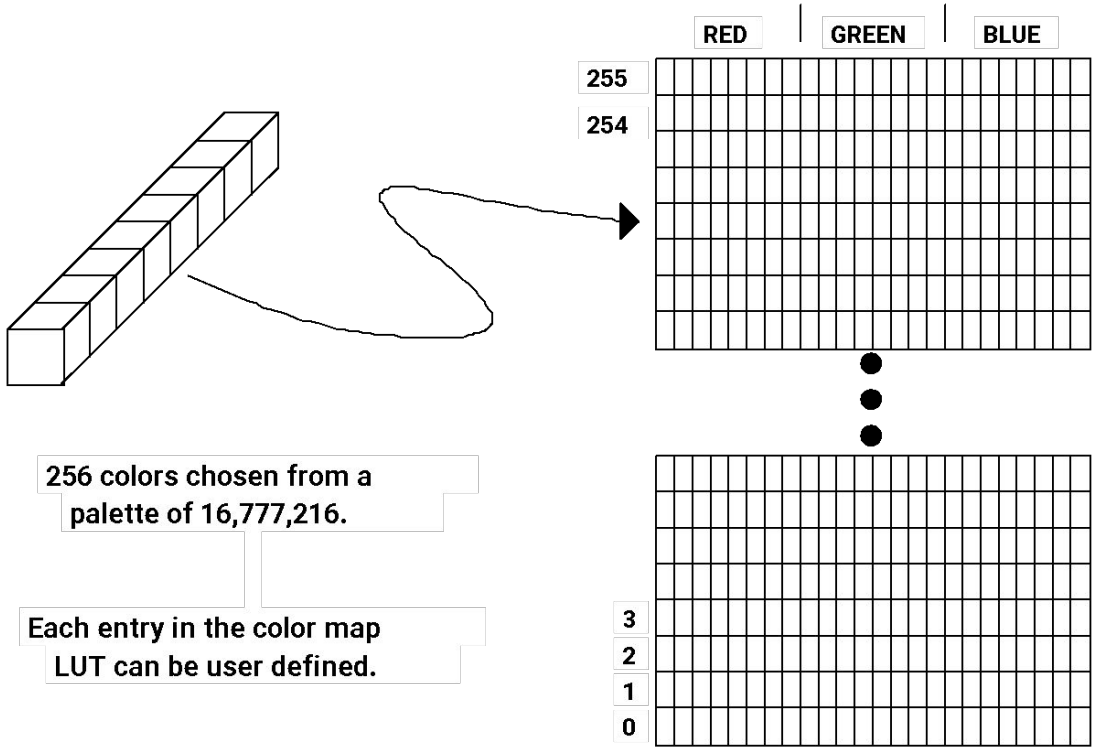


Fig. LUT Video look-up table organization. A pixel with value 67 (binary 01000011) is displayed on the screen with the red electron gun at 9/15 of maximum, green at 10/15, and blue at 1/15. This look-up table is shown with 12 bits per entry. Up to 24 bits per entry are common.

Pseudo Color: 2^8 x 24 Color Map LUT



256 colors chosen from a palette of 16,777,216.

Each entry in the color map LUT can be user defined.

Color Look-up Table

- The number of the bit planes in the frame buffer▫ determines the number of colors displayable on the screen *simultaneously*
- The width of the LUT determines the number of *possible* colors that we can choose from (also called the **color palette**)
- Example:
  8 bit planes ▫ 2^8 or 256 colors can be displayed simultaneously
  A LUT width of 12 bits ▫ color palette consists of 2^{12} colors in all

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

Flat Panel Display

- The term flat panel display refers to a class of video device that have reduced volume, weight, power requirement and are thinner than CRTs (that could be hung on walls or worn on wrists).
- We can separate flat panel display in two categories:

1. **Emissive displays (Emitters): -** convert electrical energy into light.

   Eg. Plasma panel,

   Thin film electroluminescent displays

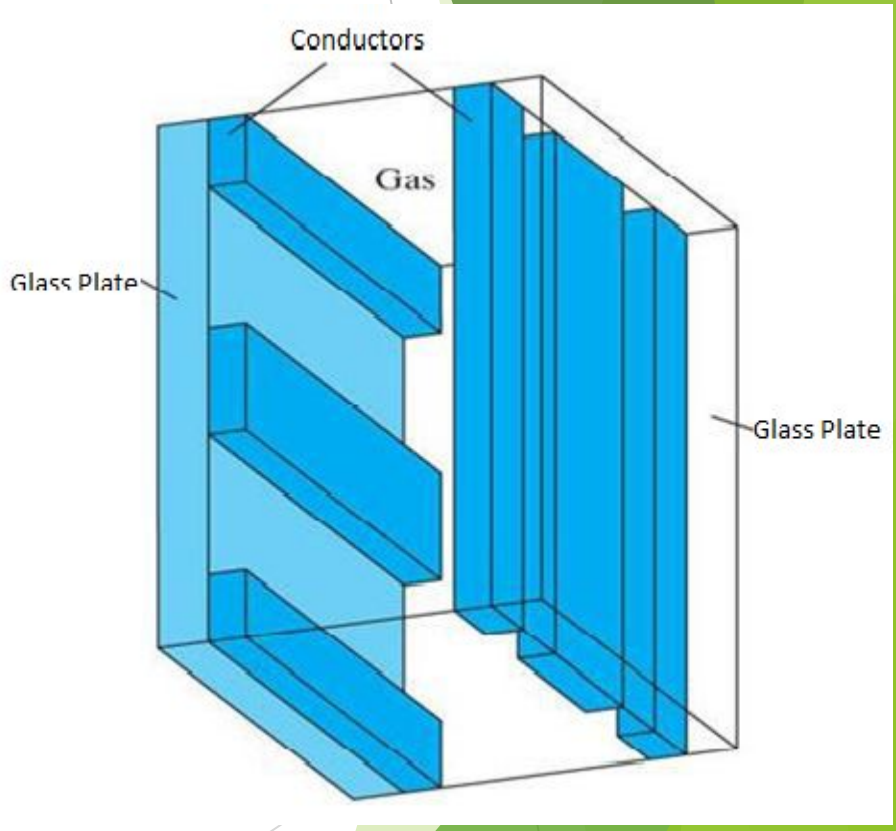
   Light emitting diodes(LED).

2. **Non emissive displays (Non Emitters): -** use optical effects to convert sunlight or light from some other source into graphics patterns.

   Eg. Liquid Crystal Display(LCD).

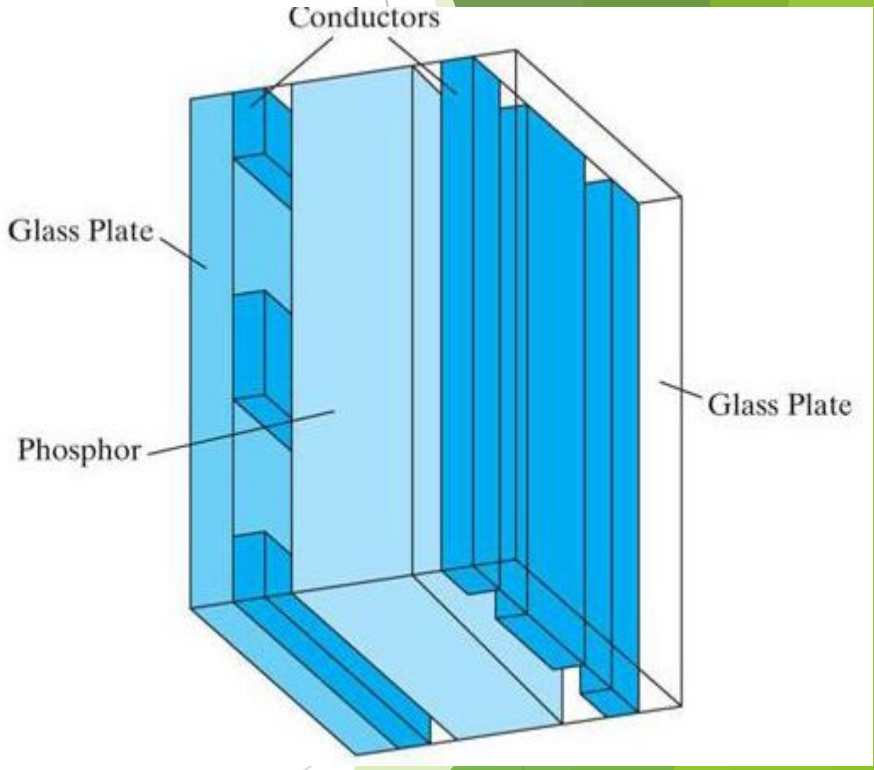
Plasma Panels displays

- This is also called gas discharge displays.
- It is 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 ribbon is built into the other glass panel.
- Firing voltage is 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.
- Refresh rate: 60 times per second.
- Separation between pixels is provided by the electric field of conductor.
- Disadvantage: strictly monochromatic device



Thin Film Electroluminescent Displays

- It is similar to plasma panel display but region between the glass plates is filled with phosphors such as Zinc sulfide doped with magnesium instead of gas.
- When sufficient voltage is applied the phosphors becomes a conductor in area of intersection of the two electrodes.
- Electrical energy is then absorbed by the manganese atoms whi then release the energy as a spot of light similar to the glowing plasma effect in plasma panel.
- It requires more power than plasma panel.
- In this good color displays are difficult to achieve.



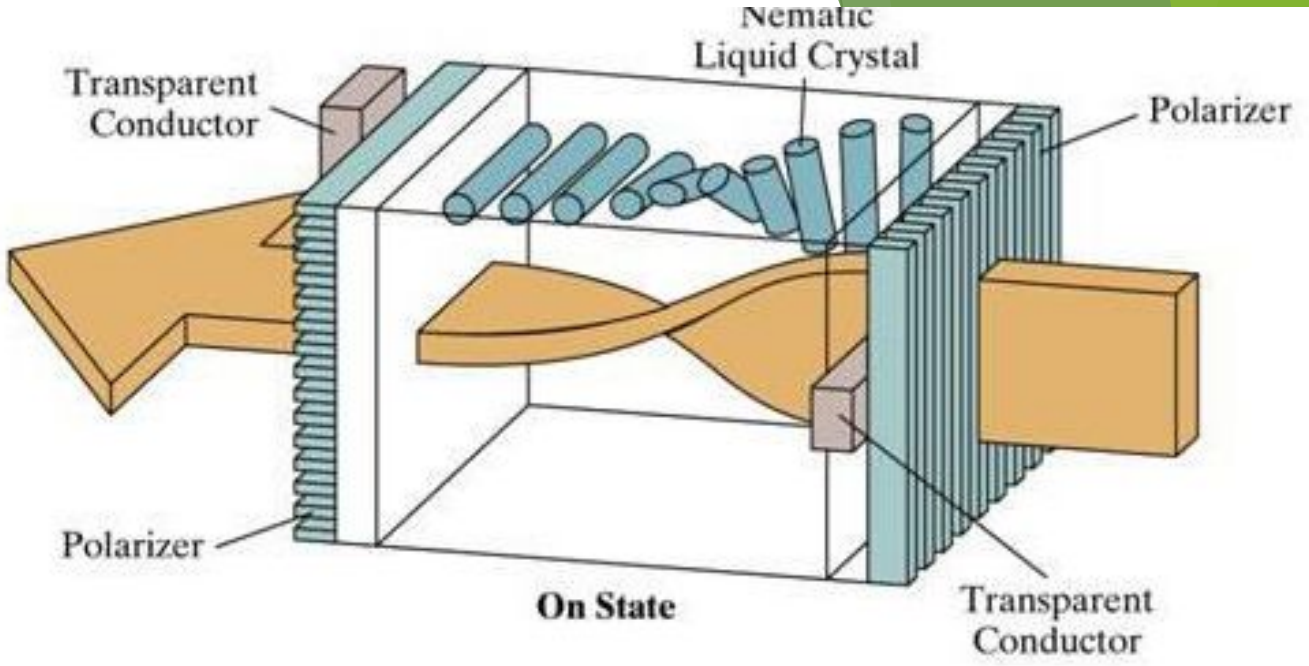
Light Emitting Diode (LED)

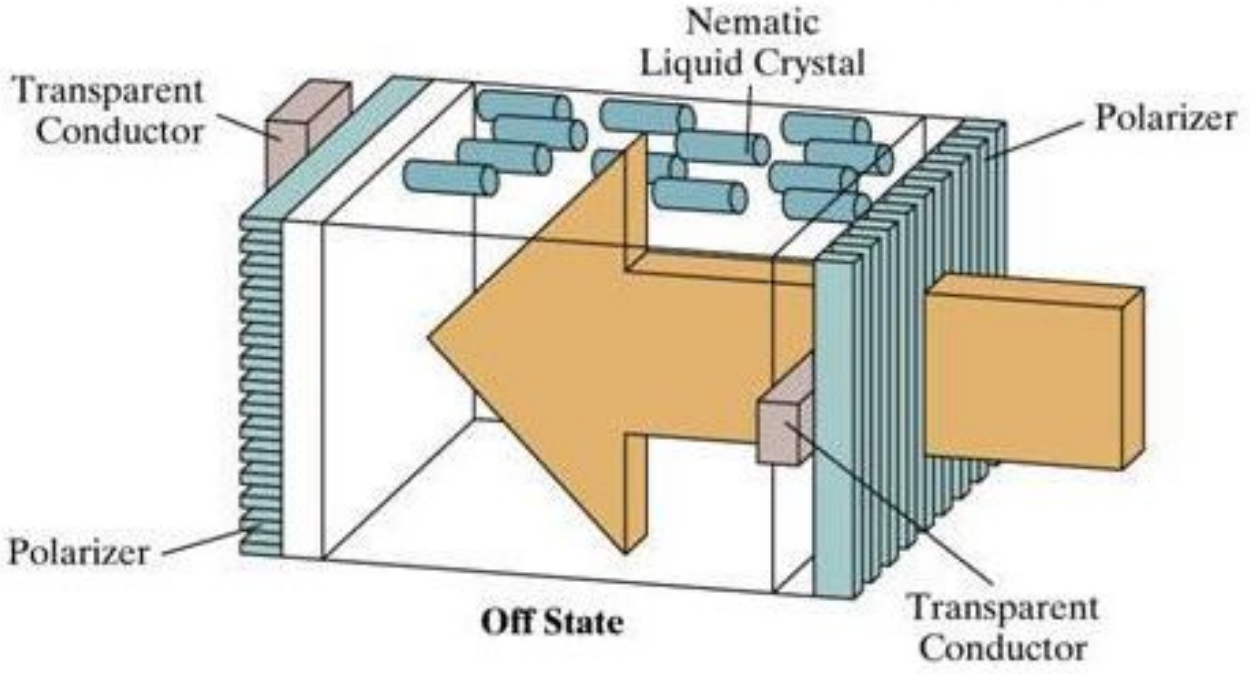
- In this display a matrix of multi-color light emitting diode is arranged to form the pixel position in the display and the picture definition is stored in refresh buffer.
- Similar to scan line refreshing of CRT information is read from the refresh buffer and converted to voltage levels that are applied to the diodes to produce the light pattern on the display.

Liquid Crystal Display (LCD)

- This non emissive device produce picture by passing polarized light from the surrounding or from an internal light source through liquid crystal material that can be aligned to either block or transmit the light.
- The liquid crystal refreshes to fact that these compounds have crystalline arrangement of molecules then also flows like liquid.
- It consists of two glass plates each with light polarizer at right angles to each other sandwich the liquid crystal material between the plates.
- Rows of horizontal transparent conductors are built into one glass plate, and column of vertical conductors are put into the other plates.
- The intersection of two conductors defines a pixel position.

- Passive-matrix LCD:
  - In the ON state polarized light passing through material is twisted so that it will pass through the opposite polarizer, the light is then reflected back to the viewer.
  - In the OFF state, voltage applied to the two intersecting conductors align the molecules so that the light is not twisted.
- Active-matrix LCD:
  - A transistor is placed at each pixel location, using thin-film transistor technology, that control the voltage at pixel locations and prevent charge from gradually leaking out of the liquid-crystal cells.



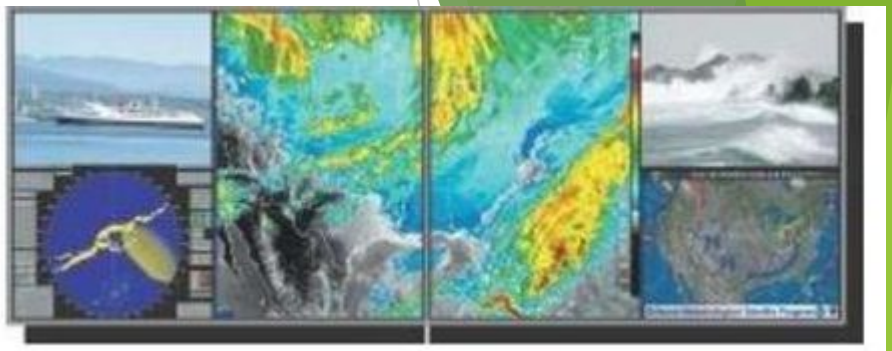


Graphics workstations and viewing systems

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

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

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



A multi-panel display system called the “Super Wall”.

Curved viewing screen

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

- A 360 degree paneled viewing system in the NASA control-tower simulator, which is used for training and for testing ways to solve air-traffic and runway problems at airports.
- A large screen stereoscopic view of pressure contours in a vascular blood-flow simulation.

Large-screen stereoscopic view

Input devices

- Graphics workstations make use of various devices for data input. Most systems have keyboards and mouses, while some other systems have trackball, spaceball, joystick, digitizers, dials, button boxes, data gloves, touch panels, image scanners and voice systems.
- Locator Devices
- Keyboard
- Scanner
  - Images
  - Laser
- Cameras (research)

- Graphical input is more varied than input to standard programs which is usually numbers, characters, or bits
- Two older APIs (GKS, PHIGS) defined six types of logical input for graphics
  - **Locator**: return a position
  - **Pick**: return ID of an object
  - **Keyboard**: return strings of characters
  - **Stroke**: return array of positions
  - **Valuator**: return floating point number
  - **Choice**: return one of n items
- Why necessary --- To make input devices independent of hardware implementations.

Keyboard

- Text input
  - List boxes, GUI
  - CAD/CAM
  - Modeling
- Hard coded
  - Vertex locations are inserted into code

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

Button Boxes and Dials

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

Locator Devices

When queried, locator devices return a position and/or orientation.

- Mouse (2D and 3D)
- Trackball
- Joystick (2D and 3D)

Mouse Devices

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

Trackballs and Spaceballs

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

Joysticks

- Joystick is used as a positioning device, which uses a small vertical lever(stick) mounded on a base. It is used to steer the screen cursor around and select screen position with the stick movement.
- A push or pull on the stick is measured with strain gauges and converted to movement of the screen cursor in the direction of the applied pressure.

Locator Devices

When queried, locator devices return a position and/or orientation.

- Tablet
- Virtual Reality Trackers
  - Data Gloves
  - Digitizers

Data Gloves

- Data glove can be used to grasp a virtual object. The glove is constructed with a series of sensors that detect hand and finger motions.
- Input from the glove is used to position or manipulate objects in a virtual scene.

Digitizers

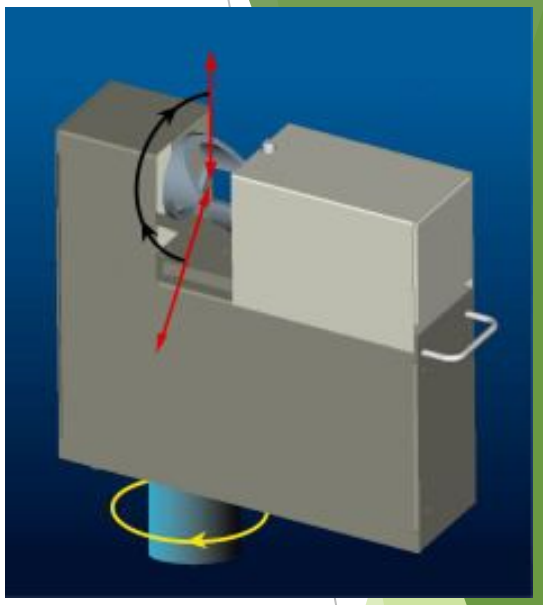
- Digitizer is a common device for drawing, painting or selecting positions.
- Graphics tablet is one type of digitizer, which is used to input 2-dimensional coordinates by activating a hand cursor or stylus at selected positions on a flat surface.
- A hand cursor contains cross hairs for sighting positions and stylus is a pencil-shaped device that is pointed at positions on the tablet.

Image Scanners

- Drawings, graphs, photographs or text can be stored for computer processing with an image scanner by passing an optical scanning mechanism over the information to be stored.
- Once we have the representation of the picture, then we can apply various image processing method to modify the representation of the picture and various editing operations can be performed on the stored documents.

Scanners

- Image Scanners - Flatbed, etc.
  - What type of data is returned? Bitmap
- Laser Scanners - Deltasphere
  - Emits a laser and does time of flight. Returns 3D point
- Camera based - research
  - Examine camera image(s) and try to figure out vertices from them.



Touch Panels

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

Light Pens

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

Voice Systems

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

Graphics Networks

- So far, we have mainly considered graphics applications on an isolated system with a single user.
- Multiuser environments & computer networks are now common elements in many graphics applications.
- Various resources, such as processors, printers, plotters and data files can be distributed on a network & shared by multiple users.
- A graphics monitor on a network is generally referred to as a graphics server.
- The computer on a network that is executing a graphics application is called the client.
- A workstation that includes processors, as well as a monitor and input devices can function as both a server and a client.

Graphics on Internet

- A great deal of graphics development is now done on the Internet.
- Computers on the Internet communicate using TCP/IP.
- Resources such as graphics files are identified by URL (Uniform resource locator).
- The World Wide Web provides a hypertext system that allows users to locate and view documents, audio and graphics.
- Each URL sometimes also called as universal resource locator.
- The URL contains two parts Protocol- for transferring the document, and Server contains the document.

Graphics Software

- There are two broad classifications for computer-graphics software
  - Special-purpose packages: Special-purpose packages are designed for nonprogrammers
    - Example: generate pictures, graphs, charts, painting programs or CAD systems in some application area without worrying about the graphics procedure
  - General programming packages: general programming package provides a library of graphics functions that can be used in a programming language such as C, C++, Java, or FORTRAN.
    - Example: GL (Graphics Library), OpenGL, VRML (Virtual-Reality Modeling Language), Java 2D And Java 3D

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

Graphics Programming/ Software

- Development of the OpenGL API (Open Graphics Library Application Program Interface)
- OpenGL Architecture
  - OpenGL as a state machine
- Functions
  - Types
  - Formats
- Simple program

Early History of APIs

- IFIPS (1973) formed two committees to come up with a standard graphics API
  - Graphical Kernel System (GKS) Two versions 2D & 3D
    - 2D but contained good workstation model
  - Core
    - Both 2D and 3D
  - GKS adopted as IS0 and later ANSI standard (1980s)
- GKS not easily extended to 3D (GKS-3D)
- Far behind hardware development

PHIGS

- Programmers Hierarchical Graphics System (PHIGS)
  - Arose from CAD community
  - Database model with retained graphics (structures)
- X Window System
  - DEC/MIT effort
  - Client-server architecture with graphics
- PEX combined the two
  - Not easy to use (all the defects of each)

SGI

- Silicon Graphics (SGI) revolutionized the graphics workstation by implementing the pipeline in hardware (1982)
- To use the system, application programmers used a library called GL
- With GL, it was relatively simple to program three dimensional interactive applications

- **Low-level 3D API**
  - These APIs for 3D computer graphics are particularly popular:
  - Direct3D (a subset of DirectX)
  - Glide
  - Mantle developed by AMD.
  - Metal developed by Apple.
  - MonoGame
  - OpenGL and the OpenGL Shading Language
  - OpenGL ES 3D API for embedded devices
  - QuickDraw 3D developed by Apple Computer starting in 1995, abandoned in 1998
  - RenderMan, RenderWare
  - LibGCM
- **Web-based API**
  - WebGL is a JavaScript interface for OpenGL-ES-2.x API, promoted by Khronos. This is gaining more interest recently, as this enables applications to use native graphics.
- **High-level 3D API**
  - There are also higher-level 3D scene-graph APIs which provide additional functionality on top of the

OpenGL

- The success of GL lead to OpenGL (1992), a platform-independent API that was
  - Easy to use
  - Close enough to the hardware to get excellent performance
  - Focus on rendering
  - Omitted windowing and input to avoid window system dependencies

OpenGL Evolution

- Controlled by an Architectural Review Board (ARB)
  - Members include SGI, Microsoft, Nvidia, HP, 3DLabs,IBM,…….
  - Relatively stable (present version 1.4)
    - Evolution reflects new hardware capabilities
      - 3D texture mapping and texture objects
      - Vertex programs
  - Allows for platform specific features through extensions

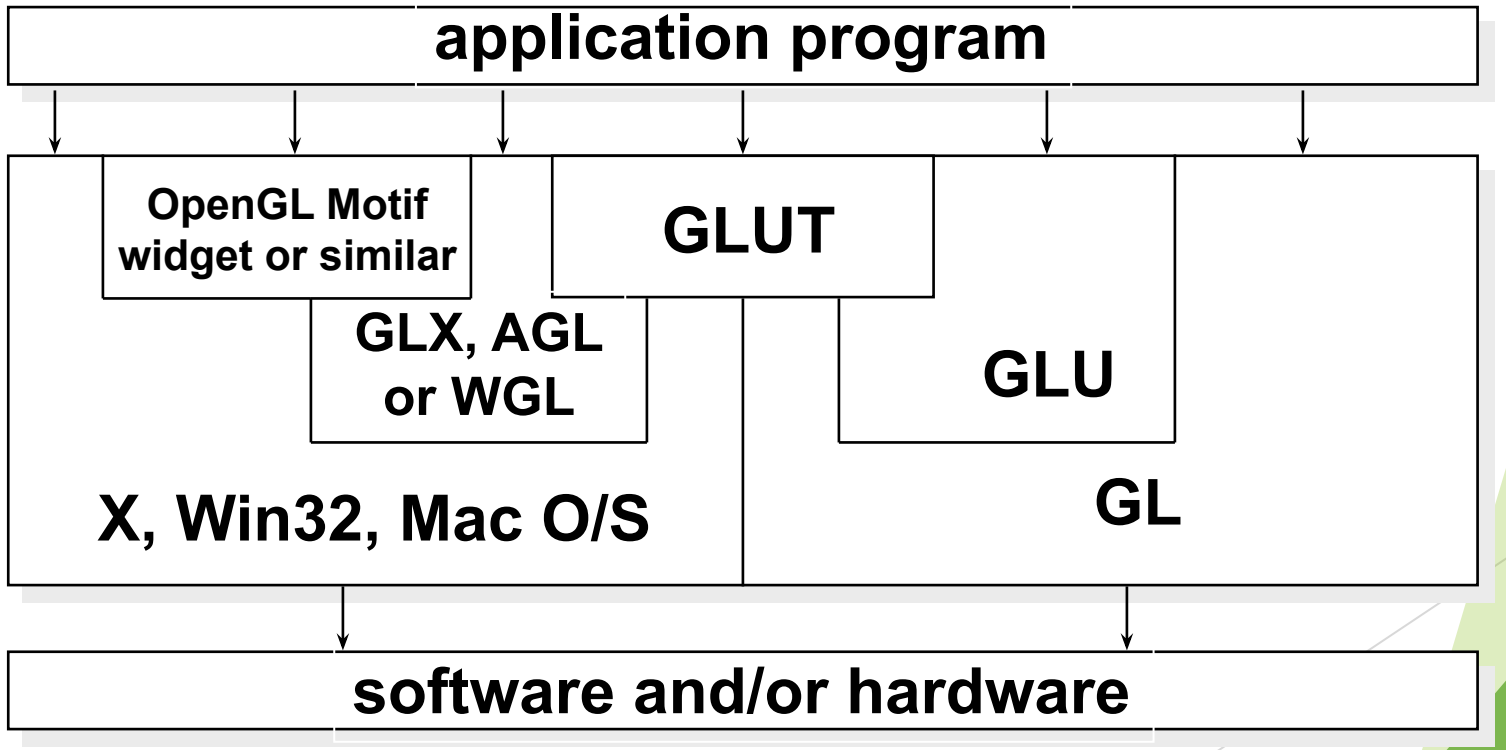
OpenGL Libraries

- OpenGL core library
  - OpenGL32 on Windows
  - GL on most unix/linux systems
- OpenGL Utility Library (GLU)
  - Provides functionality in OpenGL core but avoids having to rewrite code
- Links with window system
  - GLX for X window systems
  - WGL for Widows
  - AGL for Macintosh

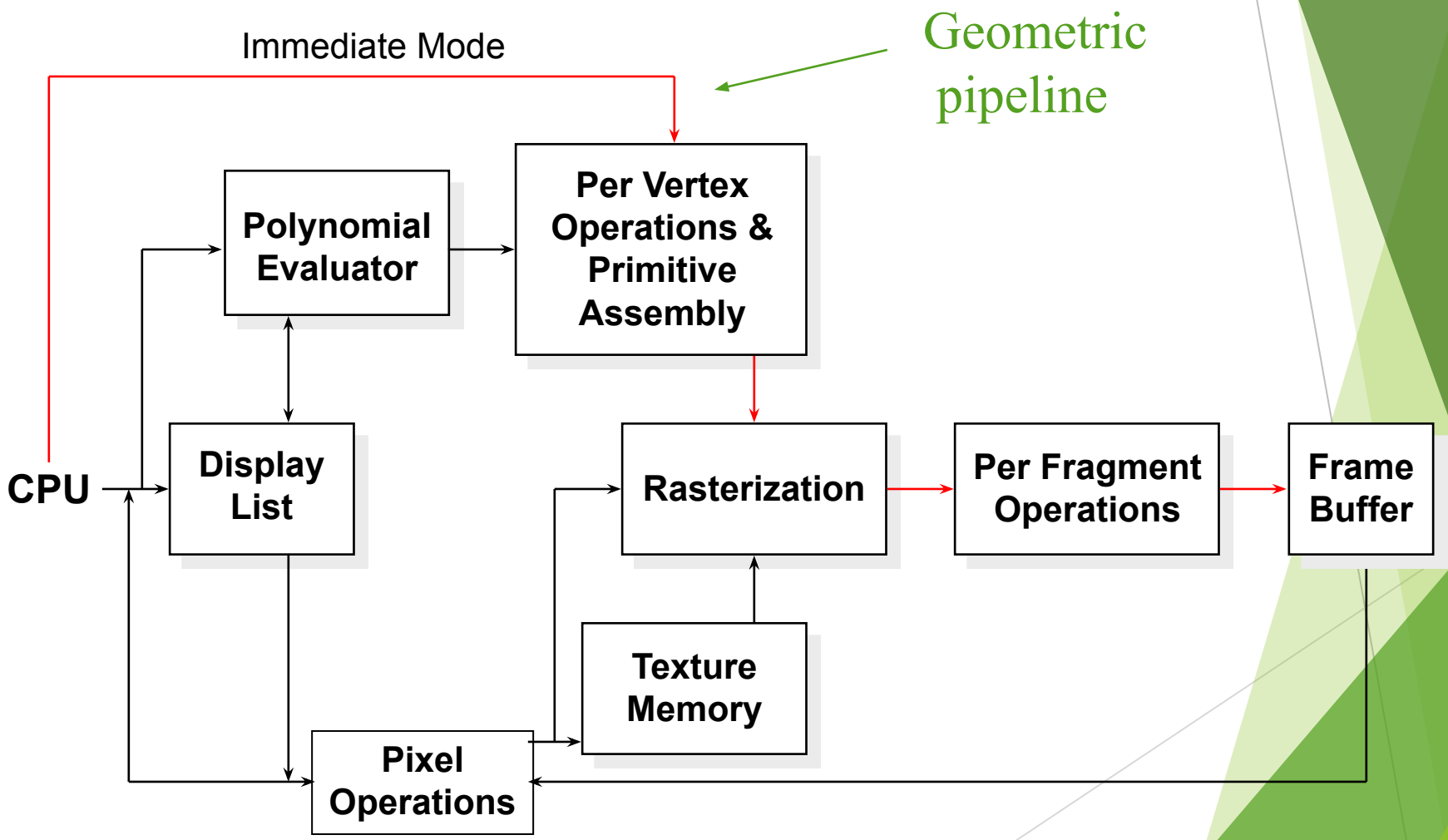
GLUT

- OpenGL Utility Library (GLUT)
  - Provides functionality common to all window systems
    - Open a window
    - Get input from mouse and keyboard
    - Menus
    - Event-driven
  - Code is portable but GLUT lacks the functionality of a good toolkit for a specific platform
    - Slide bars

Software Organization



OpenGL Architecture



OpenGL Functions-API's

- Primitives
  - Points
  - Line Segments
  - Polygons
- Attributes
- Transformations
  - Viewing
  - Modeling
- Control
- Input (GLUT)

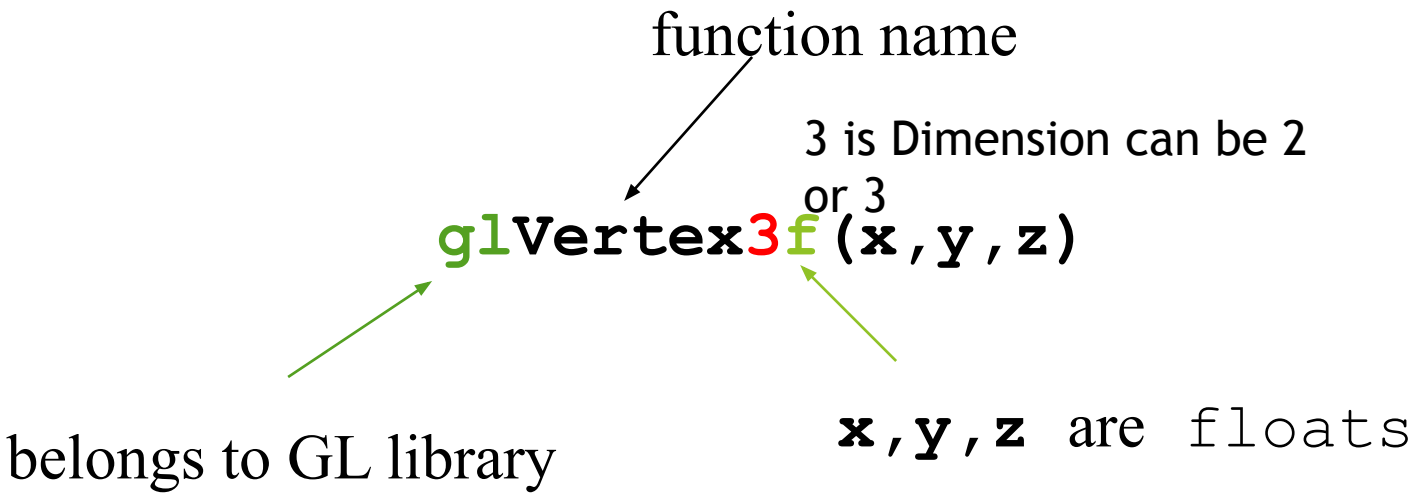
OpenGL State

- OpenGL is a state machine
- OpenGL functions are of two types
  - Primitive generating
    - Can cause output if primitive is visible
    - How vertices are processes and appearance of primitive are controlled by the state
  - State changing
    - Transformation functions
    - Attribute functions

Lack of Object Orientation

- OpenGL is not object oriented so that there are multiple functions for a given logical function, e.g. **`glVertex3f`**, **`glVertex2i`**, **`glVertex3dv`**,…..
- Underlying storage mode is the same
- Easy to create overloaded functions in C++ but issue is efficiency

OpenGL Function Format




```
glVertex3fv(p)
```

p is a pointer to an array

OpenGL #defines

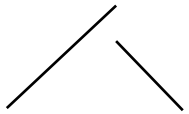
- Most constants are defined in the include files **`gl.h`**, **`glu.h`** and **`glut.h`**
  - Note **`#include <GL/glut.h>`** should automatically include the others
  - Examples
  - **`glBegin(GL_POLYGON)`**
  - **`glClear(GL_COLOR_BUFFER_BIT)`**
- Include files also define OpenGL data types: **`Glfloat`**, **`Gldouble`**,….

Program Structure

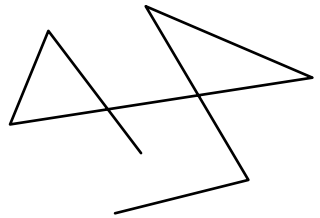
- Most OpenGL programs have a similar structure that consists of the following functions
  - **`main()`:**
    - defines the callback functions
    - opens one or more windows with the required properties
    - enters event loop (last executable statement)
  - **`init()`**: sets the state variables
    - viewing
    - Attributes
  - **callbacks**
    - Display function
    - Input and window functions

OpenGL Primitives

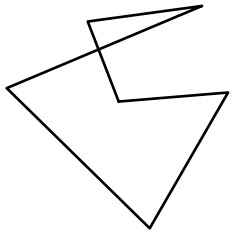
GL_POINTS



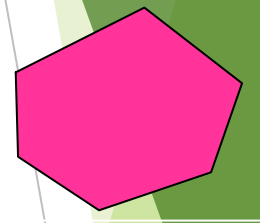
GL_LINES



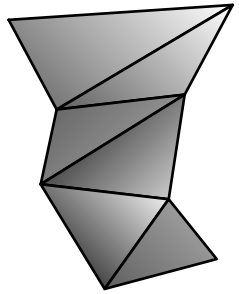
GL_LINE_STRIP



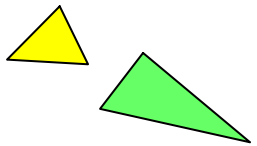
GL_LINE_LOOP



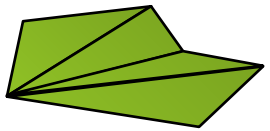
GL_POLYGON



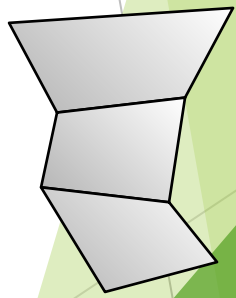
GL_TRIANGLE_STRIP



GL_TRIANGLES



GL_TRIANGLE_FAN



GL_QUAD_STRIP

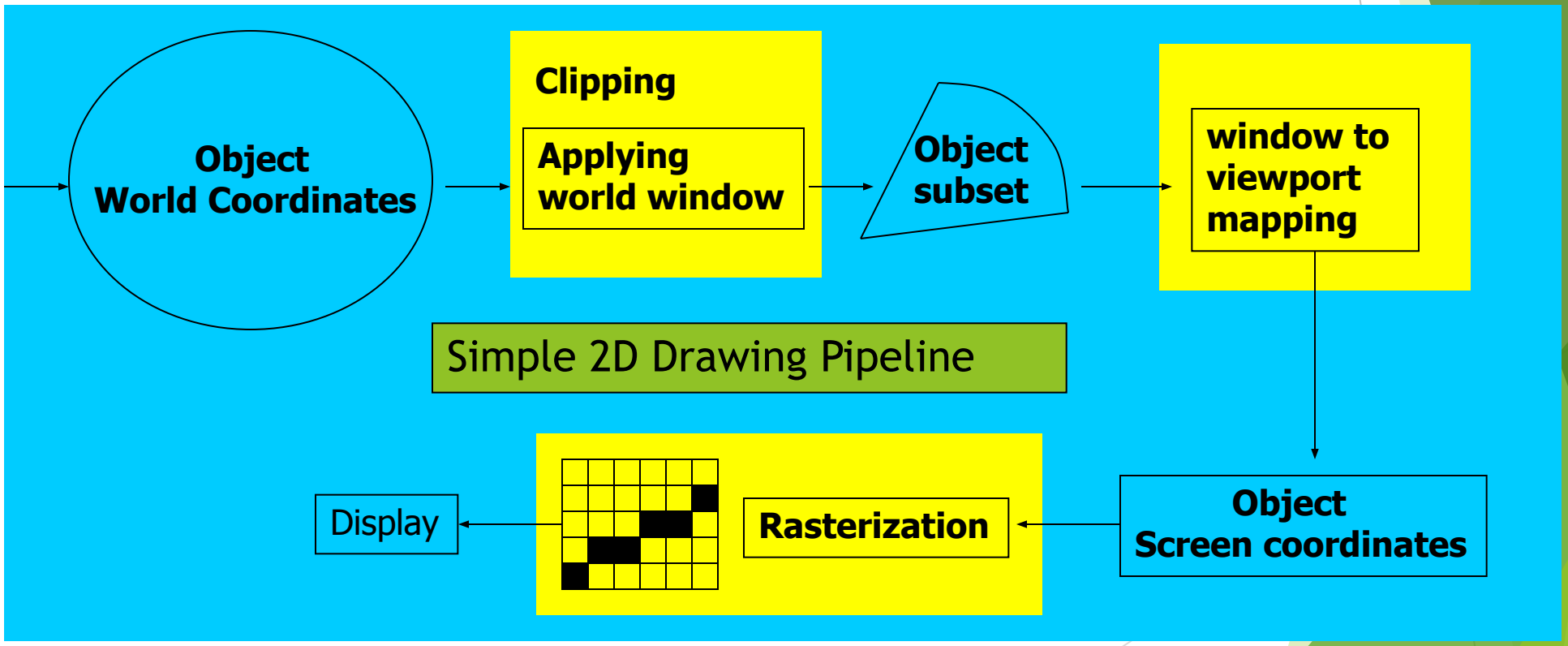
Attributes

- Attributes are part of the OpenGL state and determine the appearance of objects
  - Color (points, lines, polygons)
  - Size and width (points, lines)
  - Stipple pattern (lines, polygons)
  - Polygon mode
    - Display as filled: solid color or stipple pattern
    - Display edges
    - Display vertices

Graphics Pipelines

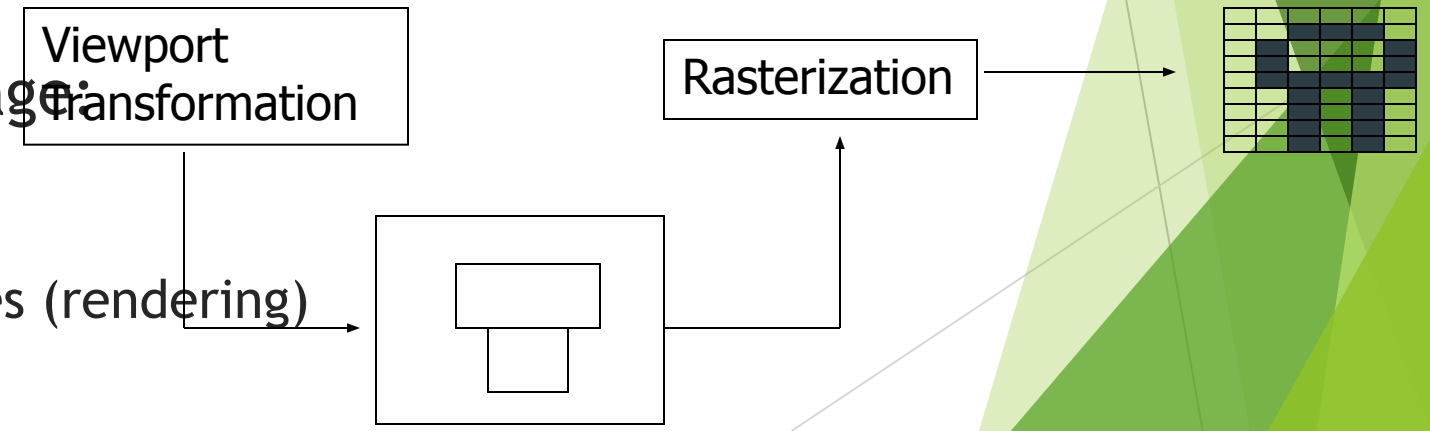
- Graphics processes generally execute sequentially
- Typical ‘pipeline’ model
- There are two ‘graphics’ pipelines
  - The Geometry or 3D pipeline
  - The Imaging or 2D pipeline

2D Graphics Pipeline



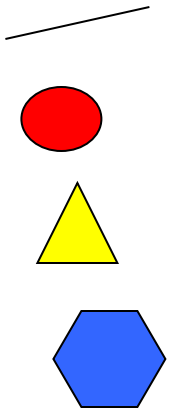
Rasterization (Scan Conversion)

- Convert high-level geometry description to pixel colors in the frame buffer
- Example: given vertex x, y coordinates determine pixel colors to draw line
- Two ways to create an image:
  - Scan existing photograph
  - Procedurally compute values (rendering)



Rasterization

- A fundamental computer graphics function
- Determine the pixels' colors, illuminations, textures, etc.
- Implemented by graphics hardware
- Rasterization algorithms
  - Lines
  - Circles
  - Triangles
  - Polygons



Rasterization Operations

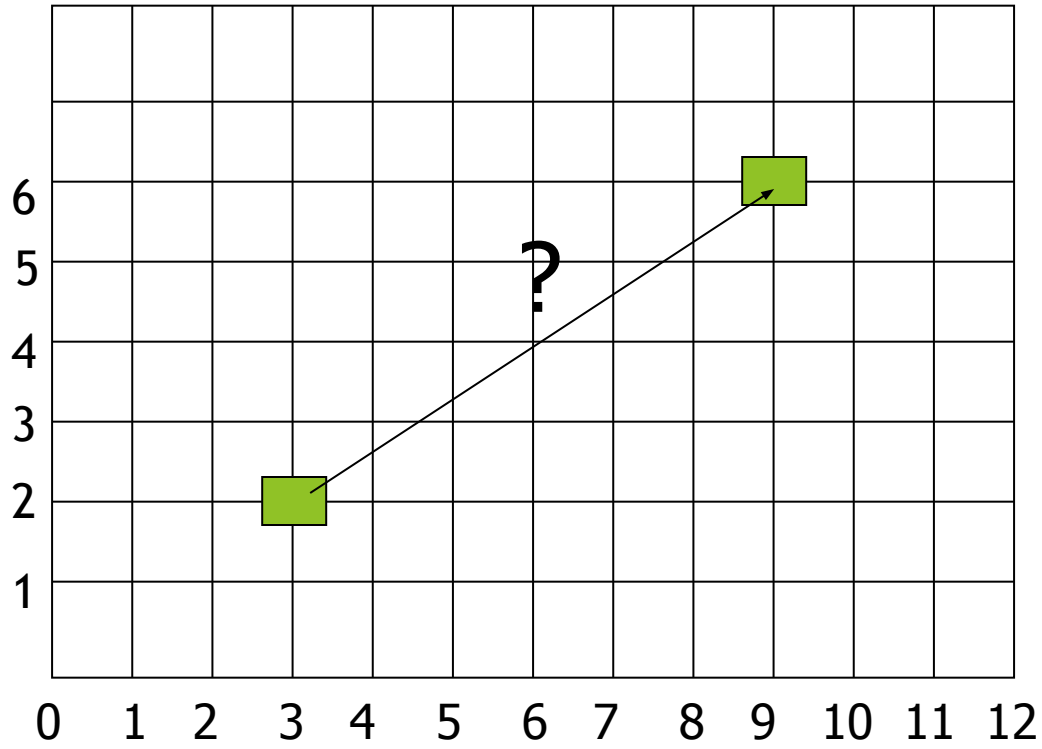
- Drawing lines on the screen
- Manipulating pixel maps (pixmaps): copying, scaling, rotating, etc
- Compositing images, defining and modifying regions
- Drawing and filling polygons
  - Previously glBegin(GL_POLYGON), etc
- Aliasing and anti-aliasing methods

Line drawing algorithm

- Programmer specifies (x,y) values of end pixels
- Need algorithm to figure out which intermediate pixels are on line path
- Pixel (x,y) values constrained to integer values
- Actual computed intermediate line values may be floats
- Rounding may be required. E.g. computed point

  (10.48, 20.51) rounded to (10, 21)
- Rounded pixel value is off actual line path (jaggy!!)
- Sloped lines end up having jaggies
- Vertical, horizontal lines, no jaggies

Line Drawing Algorithm



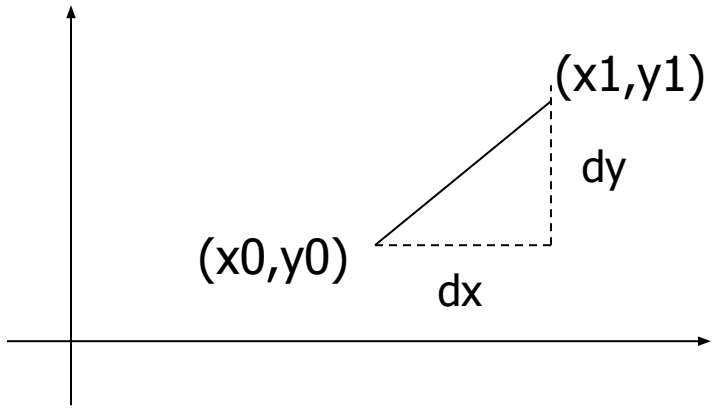
Line: (3,2) -> (9,6)

Which intermediate pixels to turn on?

Line Drawing Algorithm

- Slope-intercept line equation
  - y = mx + b Equation of Line where m is slope & b is intercept
  - Given two end points (x0,y0), (x1, y1), how to compute m and b?

$$m = \frac{dy}{dx} = \frac{y1 - y0}{x1 - x0} \qquad b = y0 - m * x0$$



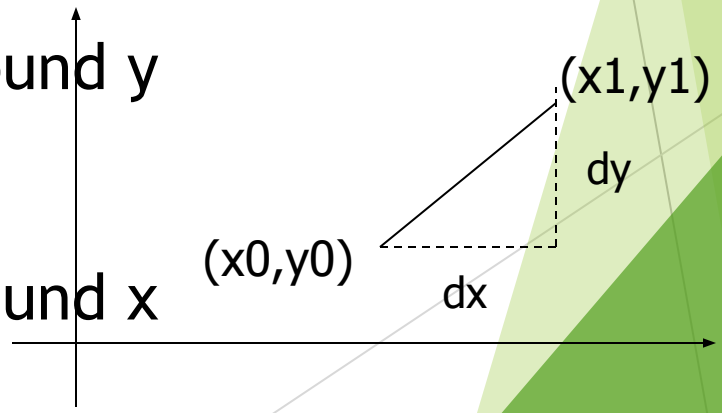
Line Drawing Algorithm

- Numerical example of finding slope m:
- (Ax, Ay) = (23, 41), (Bx, By) = (125, 96)

$$m = \frac{By - Ay}{Bx - Ax} = \frac{96 - 41}{125 - 23} = \frac{55}{102} = 0.5392$$

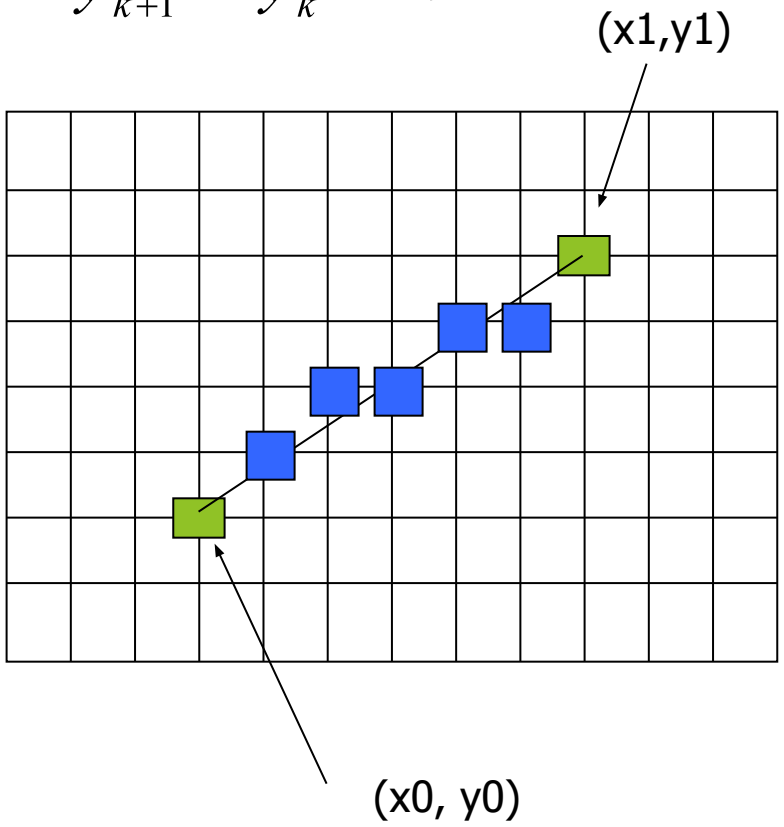
Digital Differential Analyzer (DDA): Line Drawing Algorithm

- Walk through the line, starting at (x0,y0)
- Constrain x, y increments to values in [0,1] range
- Case a: x is incrementing faster (m < 1)
  - Step in x=1 increments, compute and round y
- Case b: y is incrementing faster (m > 1)
  - Step in y=1 increments, compute and round x



DDA Line Drawing Algorithm (Case a: m < 1)

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



x = x0 y = y0

Illuminate pixel (x, round(y))

x = x0 + 1 y = y0 + 1 * m

Illuminate pixel (x, round(y))

x = x + 1 y = y + 1 * m

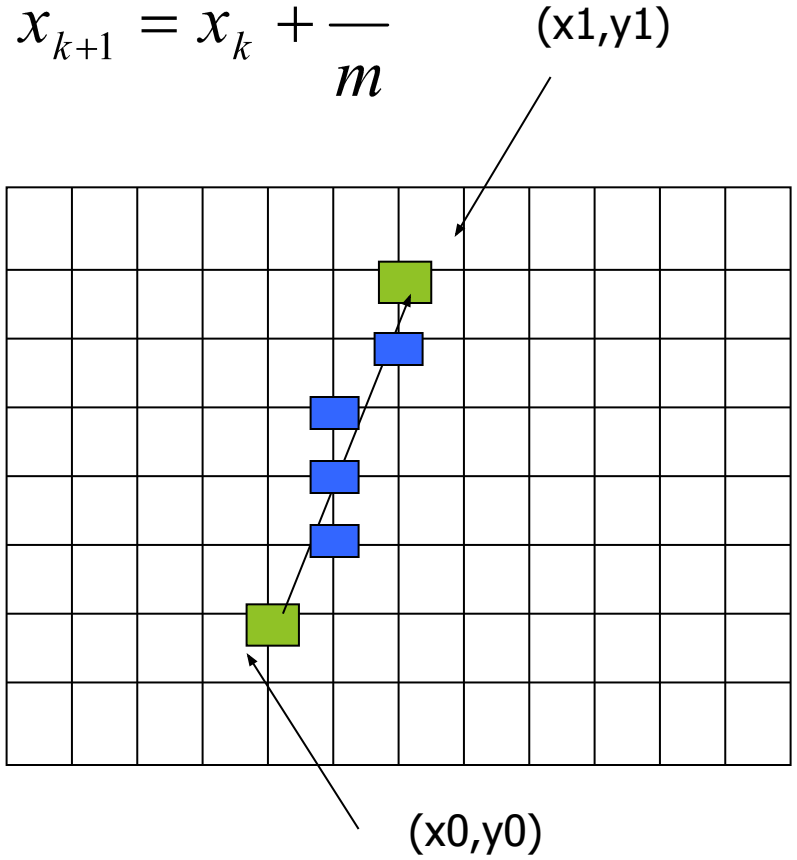
Illuminate pixel (x, round(y))

…

Until x == x1

DDA Line Drawing Algorithm (Case b: m > 1)

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



x = x0 y = y0

Illuminate pixel (round(x), y)

y = y0 + 1 x = x0 + 1 * 1/m

Illuminate pixel (round(x), y)

y = y + 1 x = x + 1 /m

Illuminate pixel (round(x), y)

…

Until y == y1

DDA Line Drawing Algorithm Pseudocode

```
compute m;
if m < 1:
{
  float y = y0;         // initial value
  for(int x = x0;x <= x1; x++, y += m)
              setPixel(x, round(y));
}
else // m > 1
{
  float x = x0;         // initial value
  for(int y = y0;y <= y1; y++, x += 1/m)
              setPixel(round(x), y);
}
```

Note: `setPixel(x, y)` writes current color into pixel in column x and row y in frame buffer

DDA Line Drawing Algorithm Drawbacks

- **DDA is the simplest line drawing algorithm**
  - Not very efficient
  - Round operation is expensive
- **Optimized algorithms typically used.**
  - Integer DDA
  - Eg. Bresenham algorithm (Hill, 10.4.1)
- **Bresenham algorithm**
  - Incremental algorithm: current value uses previous value
  - Integers only: avoid floating point arithmetic
  - Several versions of algorithm: we'll describe midpoint version of

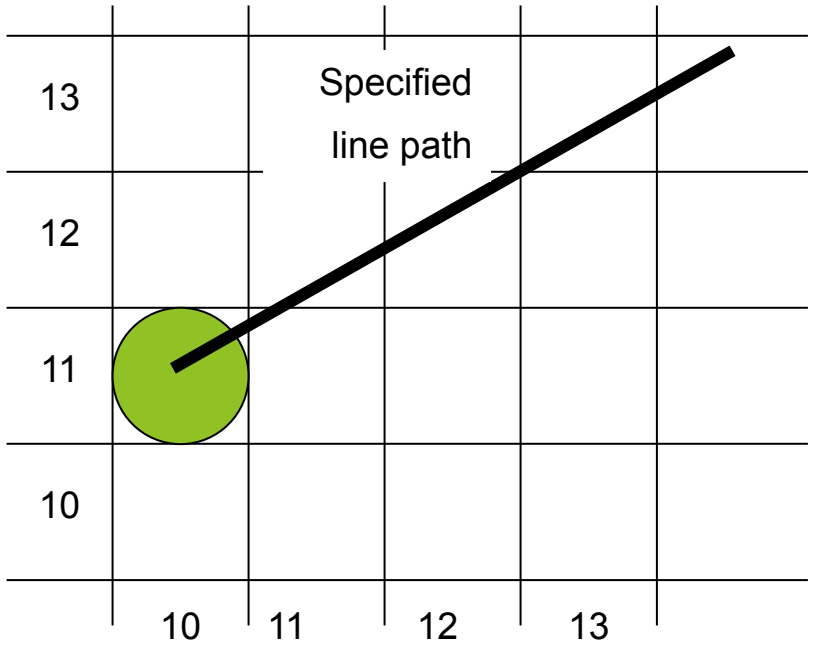
Bresenham's Line-Drawing Algorithm

- Accurate and efficient
- Uses only incremental integer calculations

✔ The method is described for a line segment with a positive slope less than one

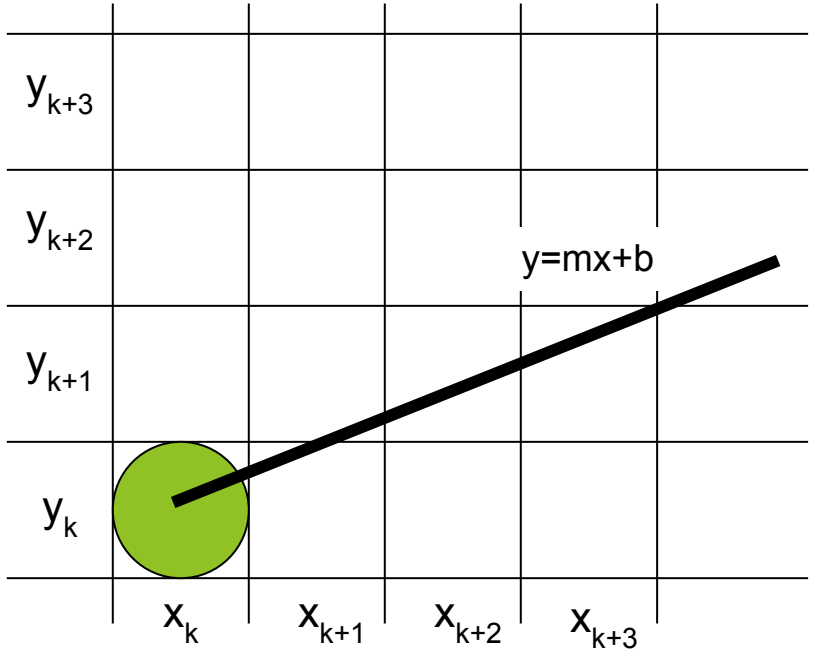
✔ The method generalizes to line segments of other slopes by considering the symmetry between the various octants and quadrants of the xy plane

Bresenham's Line-Drawing Algorithm



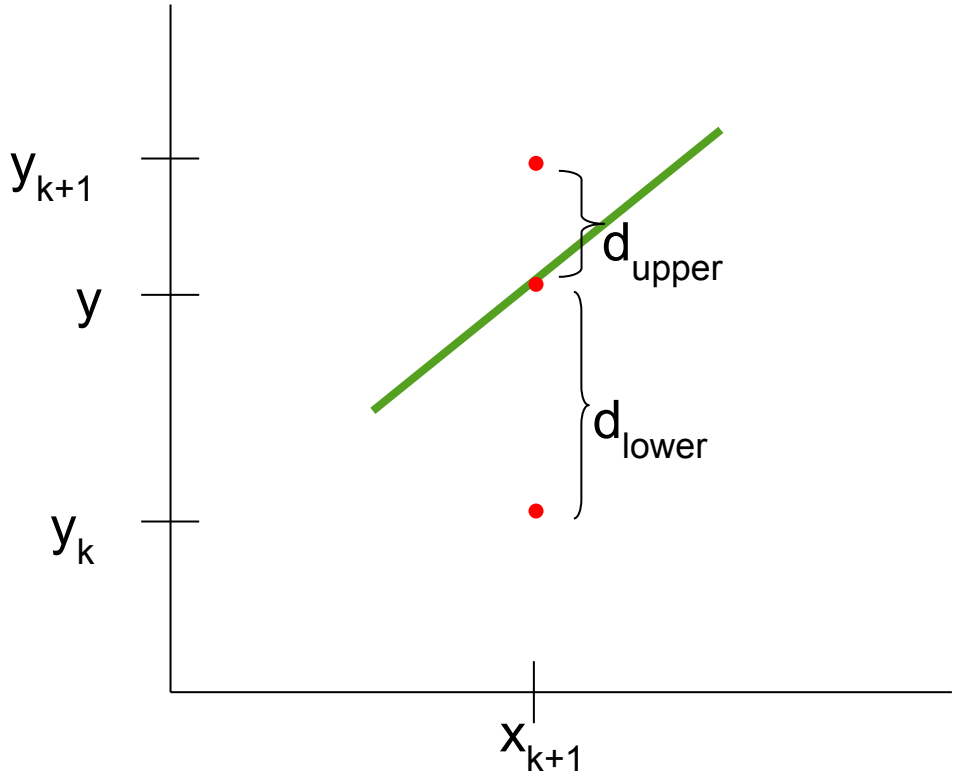
- Decide what is the next pixel position
  - (11,11) or (11,12)

Bresenham's Line-Drawing Algorithm



- For the pixel position $x_{k+1}=x_k+1$, which one we should choose:
- (x_{k+1},y_k) or (x_{k+1}, y_{k+1})

Bresenham's Approach



- $y=m(x_k + 1)+b$
- $d_{lower}=y-y_k$

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

- $d_{upper}=(y_k+1)-y$

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

- $d_{lower}- d_{upper}= 2m(x_k + 1)-2y_k+2b-1$
- Rearrange it to have integer calculations:

  $m=\Delta y/\Delta x$

  Decision parameter: $p_k= \Delta x(d_{lower}- d_{upper})=2\Delta y.x_k - 2\Delta x. y_k + c$

The Decision Parameter

Decision parameter: $p_k = \Delta x(d_{lower} - d_{upper}) = 2\Delta y.x_k - 2\Delta x.\ y_k + c$

- p_k has the same sign with $d_{lower} - d_{upper}$ since $\Delta x > 0$.
- c is constant and has the value $c = 2\Delta y + \Delta x(2b-1)$
  - c is independent of the pixel positions and is eliminated from decision parameter p_k.
- If $d_{lower} < d_{upper}$ then p_k is negative.
  - Plot the lower pixel (East)
- Otherwise
  - Plot the upper pixel (North East)

Successive decision parameter

- At step k+1

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

- Subtracting two subsequent decision parameters yields:

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

- $x_{k+1}=x_k+1$ so

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

  - $y_{k+1}-y_k$ is either 0 or 1 depending on the sign of p_k

- First parameter p_0

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

Bresenham's Line-Drawing Algorithm for |m| < 1

1. Input the twoline endpoints and store the left endpoint in (x_0, y_0).
2. Load (x_0, y_0) into the frame buffer; that is, plot the first point.
3. Calculate constants Δx, Δy, $2\Delta y$, and $2\Delta y - 2\Delta x$, and obtain the starting value for the decision parameter as

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

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

   If $p_k < 0$, 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

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

5. Repeat step 4 Δx -1 times.

Trivial Situations: Do not need Bresenham

- $m = 0 \quad \Rightarrow \quad \text{horizontal line}$
- $m = \pm 1 \quad \Rightarrow \quad \text{line } y = \pm x$
- $m = \infty \quad \Rightarrow \quad \text{vertical line}$

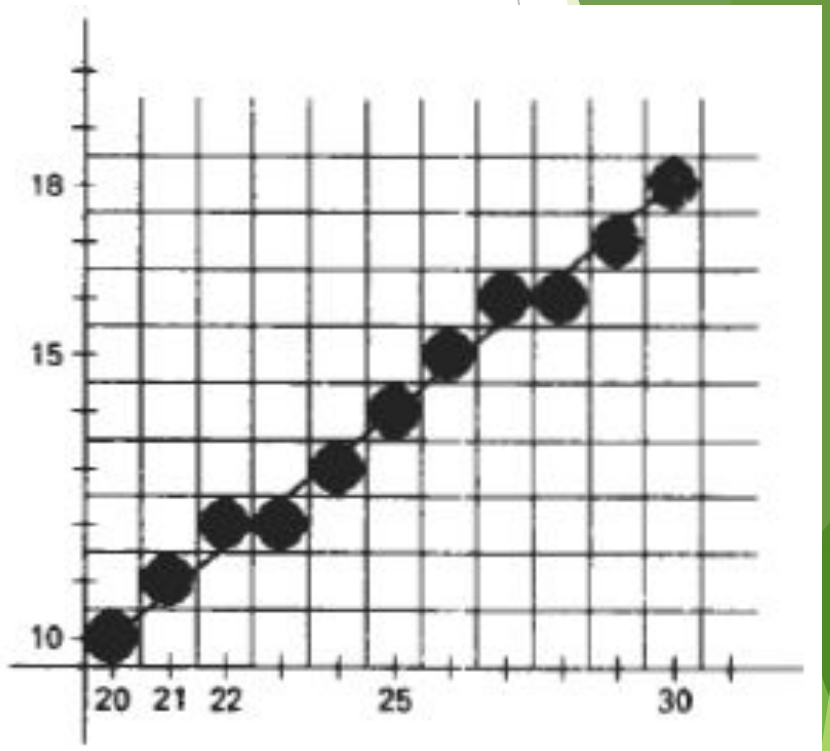
Example

- Draw the line with endpoints (20,10) and (30, 18).
  - **Δx=30-20=10, Δy=*18-10*=8,**
  - **p_0 = *2*Δy – Δx=16-10=6**
  - ***2*Δy=*16*, and *2*Δy - 2Δx=-4**
- Plot the initial position at (20,10), then

k	p_k	(x_{k+1}, y_{k+1})	k	p_k	(x_{k+1}, y_{k+1})
0	6	(21, 11)	5	6	(26, 15)
1	2	(22, 12)	6	2	(27, 16)
2	−2	(23, 12)	7	−2	(28, 16)
3	14	(24, 13)	8	14	(29, 17)
4	10	(25, 14)	9	10	(30, 18)

Example

k	p_k	(x_{k+1}, y_{k+1})	k	p_k	(x_{k+1}, y_{k+1})
0	6	(21, 11)	5	6	(26, 15)
1	2	(22, 12)	6	2	(27, 16)
2	−2	(23, 12)	7	−2	(28, 16)
3	14	(24, 13)	8	14	(29, 17)
4	10	(25, 14)	9	10	(30, 18)



Example

- Line end points: $(x_0, y_0) = (5,8); \quad (x_1, y_1) = (9,11)$
- Deltas: $dx = 4; dy = 3$

initially $p(5,8) = 2(dy)-(dx)$

$= 6 - 4 = 2 > 0$

$p = 2 \quad \Rightarrow \quad NE$

Pseudocode

```
/*  Bresenham line-drawing procedure for |m| < 1.0.  */
  void lineBres (int x0, int y0, int xEnd, int yEnd)
  {
     int dx = fabs (xEnd - x0),  dy = fabs(yEnd - y0);
     int x, y, p = 2 * dy - dx;
     int twoDy = 2 * dy,  twoDyMinusDx = 2 * (dy - dx);

     /* Determine which endpoint to use as start position.  */
     if (x0 > xEnd) {
        x = xEnd;    y = yEnd;   xEnd = x0;
     }
     else {
        x = x0;    y = y0;
     }
     setPixel (x, y);
```

Pseudocode

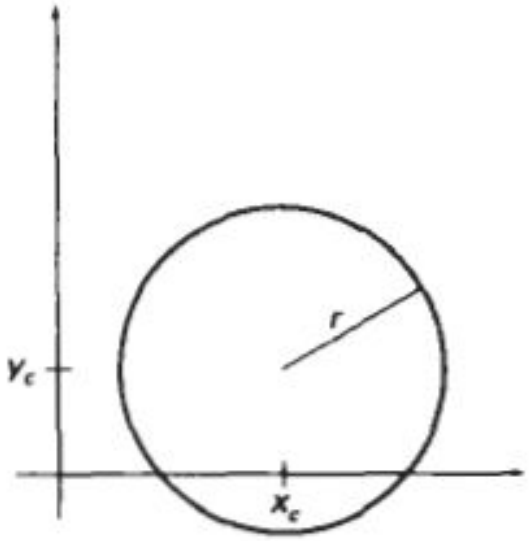
```
while (x < xEnd) {
        x++;
        if (p < 0)
           p += twoDy;
        else {
           y++;
           p += twoDyMinusDx;
        }
        setPixel (x, y);
     }
  }
```

Scan Converting Circles

3 Methods: Parametric, Trignometric & mid-point

1 Parametric method

$$(x - x_c)^2 + (y - y_c)^2 = R^2$$

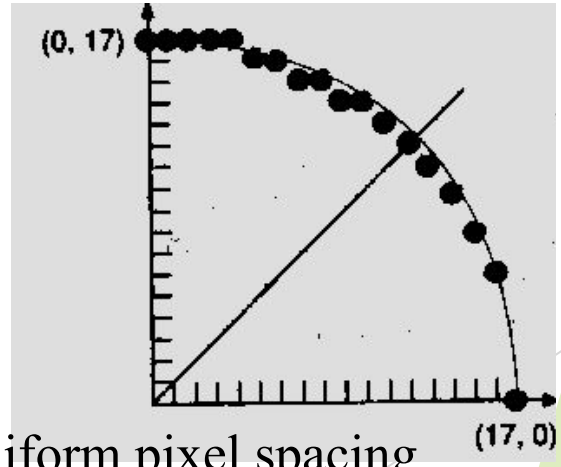


$$y = y_c \pm \sqrt{R^2 - (x - x_c)^2}$$

- Explicit: y = f(x)

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

We could draw a quarter circle by incrementing x from 0 to R in unit steps and solving for +y for each step.



Method needs lots of computation, and gives non-uniform pixel spacing

Scan converting Circles

- 2 Trignometric :

$$x = R\cos\theta$$

$$y = R\sin\theta$$

Draw quarter circle by stepping through the angle from 0 to 90
-avoids large gaps but still unsatisfactory
-How to set angular increment
Computationally expensive trigonometric calculations

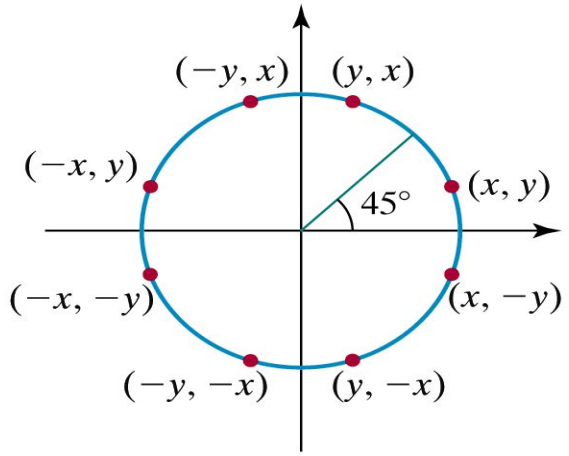
Circle Generation with mid-point

- Implicit: $f(x,y) = x^2+y^2-R^2$

If $f(x,y) = 0$ then it is on the circle.
$f(x,y) > 0$ then it is outside the circle.
$f(x,y) < 0$ then it is inside the circle.

Try to adapt the Bresenham midpoint approach

Again, exploit symmetries



Circle Symmetry

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

Generalising the Bresenham midpoint approach

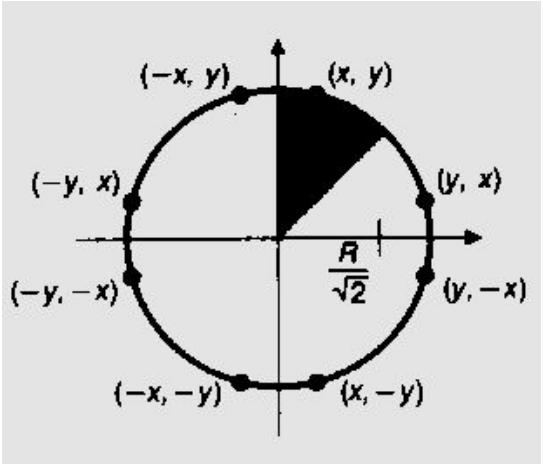
- Set up decision parameters for finding the closest pixel to the circumference at each sampling step
  - Avoid square root calculations by considering the squares of the pixel separation distances
- Use direct comparison without squaring.

  Adapt the midpoint test idea: test the halfway position between pixels to determine if this midpoint is inside or outside the curve

  This gives the **midpoint algorithm for circles**

  Can be adapted to other curves: conic sections

Eight-way Symmetry
only one octant's calculation needed




```
void CirclePoints (int x, int y, int value)
{
    WritePixel (x, y, value);
    WritePixel (y, x, value);
    WritePixel (y, −x, value);
    WritePixel (x, −y, value);
    WritePixel (−x, −y, value);
    WritePixel (−y, −x, value);
    WritePixel (−y, x, value);
    WritePixel (−x, y, value);
}   /* CirclePoints */
```

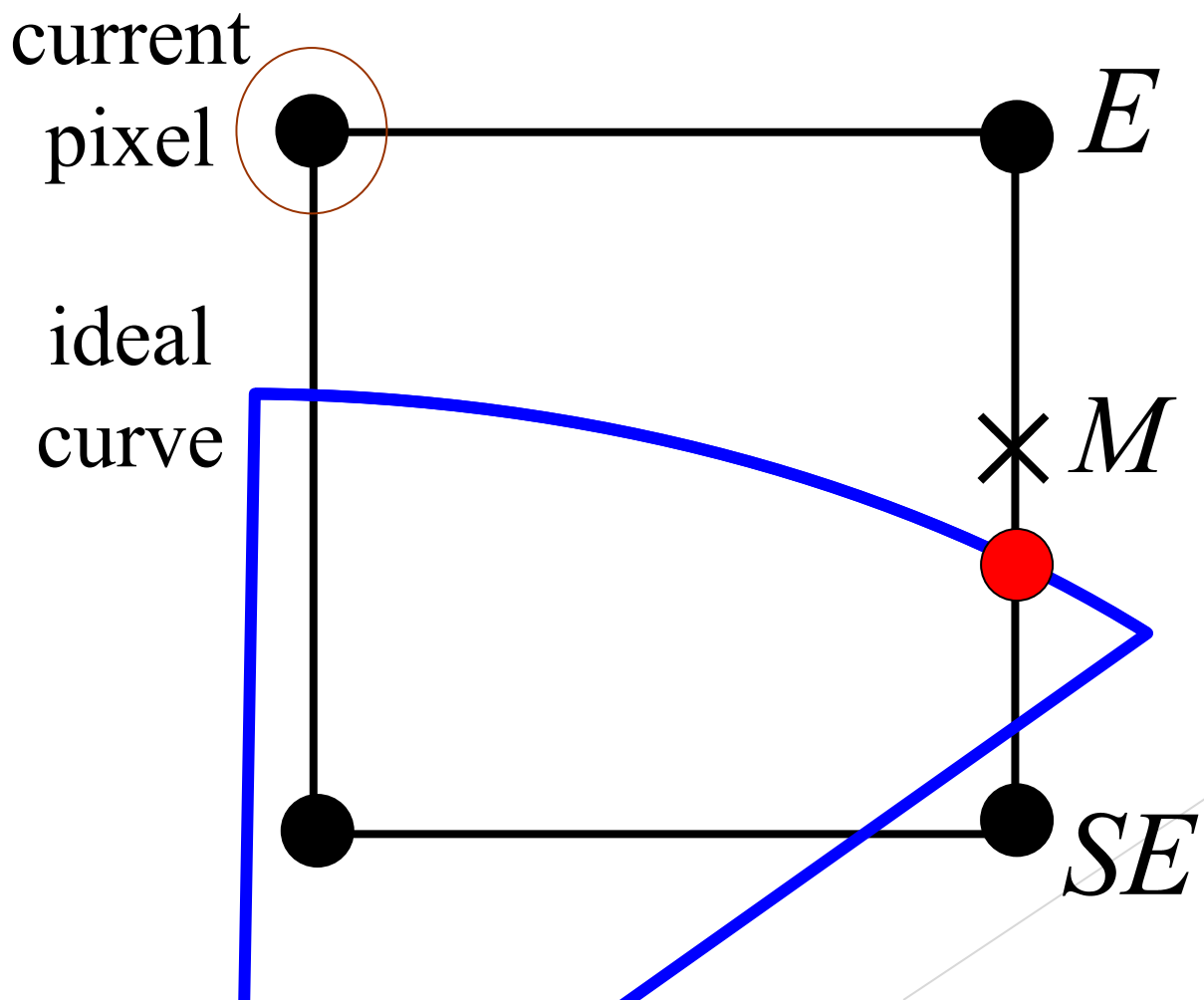
The 2nd Octant is a good arc to draw

- It is a well-defined function in this domain
  - single-valued
  - no vertical tangents: $|slope| \leq 1$
- Lends itself to the midpoint approach
  - only need consider *E* or *SE*
- Implicit formulation $F(x,y) = x^2 + y^2 - r^2$
  - For (x,y) on the circle, $F(x,y) = 0$
  - $F(x,y) > 0 \Rightarrow (x,y)$ *Outside*
  - $F(x,y) < 0 \Rightarrow (x,y)$ *Inside*

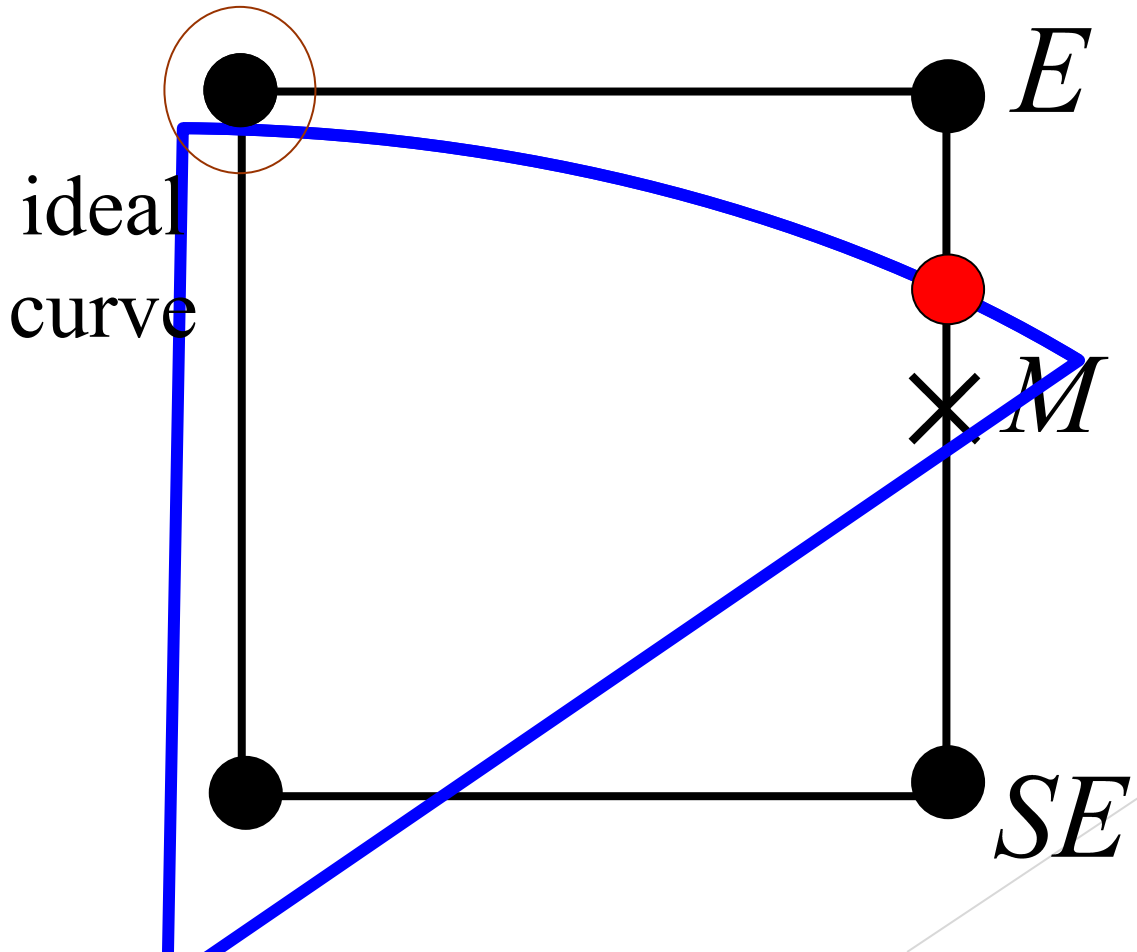
The 2nd Octant is a good arc to draw

- Decision variable d is $x^2 + y^2 - r^2$
- Then d = $F(M) \square 0 \Rightarrow$ SE
- Or d = $F(M) < 0 \Rightarrow E$

$$F(M) \square 0 \Rightarrow SE$$



$F(M) < 0 \Rightarrow E$



As in the Bresenham line algorithm we use a decision variable to direct the selection of pixels.

Use the implicit form of the circle equation

$$p = F(M) = x^2 + y^2 - r^2$$

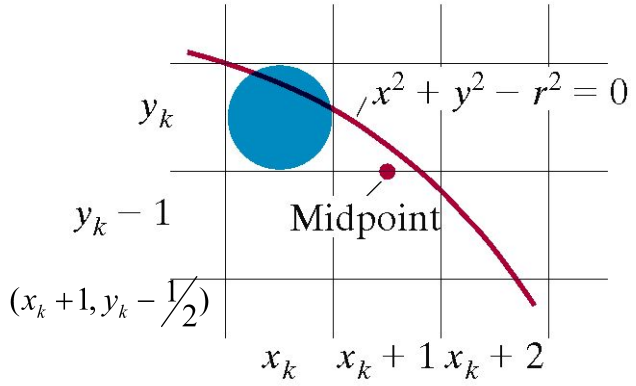


Figure 3-19

Midpoint between candidate pixels at sampling position $x_k + 1$ along a circular path.

Midpoint coordinates are

Assuming we have just plotted point at (x_k,y_k) we determine whether move E or SE by evaluating the circle function at the midpoint between the two candidate pixel positions

$$p_k = F_{circ}(x_k + 1, y_k - 1/2)$$
$$= (x_k + 1)^2 + (y_k - 1/2)^2 - r^2$$

pk is the decision variable

if *pk* <0 the midpoint is inside the circle

Thus the pixel above the midpoint is closer to the ideal circle, and we select pixel on scan line *yk*. i.e. Go E

If *pk* >0 the midpoint is outside the circle.
Thus the pixel below the midpoint is closer to the ideal circle, and we select pixel on scan line *yk*-1. i.e. Go SE

Calculate successive decision parameter values *p* by incremental calculations.

$$p_{k+1} = F_{circ}(x_{k+1}+1, y_{k+1}-\tfrac{1}{2})$$

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

recursive definition for successive decision parameter values *p*

$$p_{k+1} = F_{circ}(x_{k+1}+1, y_{k+1}-\tfrac{1}{2})$$

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

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

Where yk+1 = yk if p<0 (move E)

yk+1 = yk-1 if p>0 (move SE) yk+1 and xk+1 can also be defined recursively

Initialisation

$x_0 = 0, \quad y_0 = r$

Initial decision variable found by evaluating circle function at first midpoint test position

$$p_0 = F_{circ}(1, r - 1/2)$$
$$= 1 + (r - 1/2)^2 - r^2$$
$$= \frac{5}{4} - r$$

For integer radius r p_0 can be rounded to $p_0 = 1 - r$ since all increments are integer.

Midpoint Circle Algorithm

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

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

2. Calculate the initial value of the decision parameter as

$$p_0 = \frac{5}{4} - r$$

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

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

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

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

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

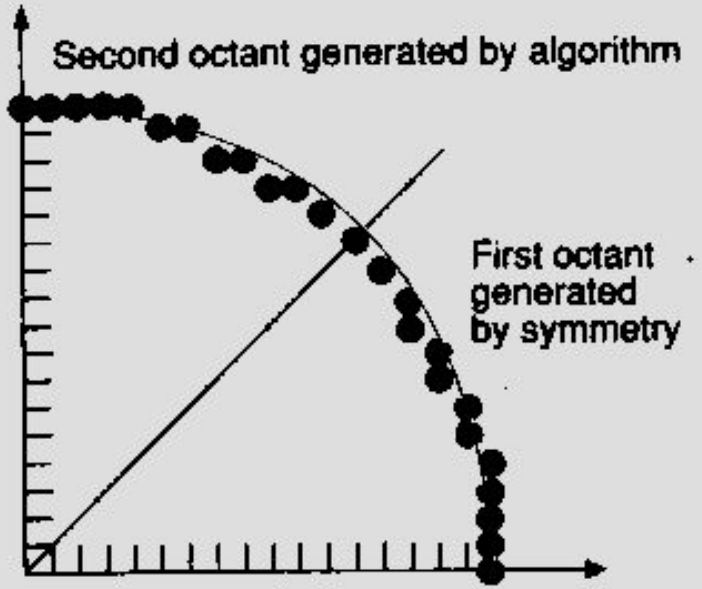
4. Determine symmetry points in the other seven octants.
5. Move each calculated pixel position (x, y) onto the circular path centered on (x_c, y_c) and plot the coordinate values:

$$x = x + x_c \qquad y = y + y_c$$

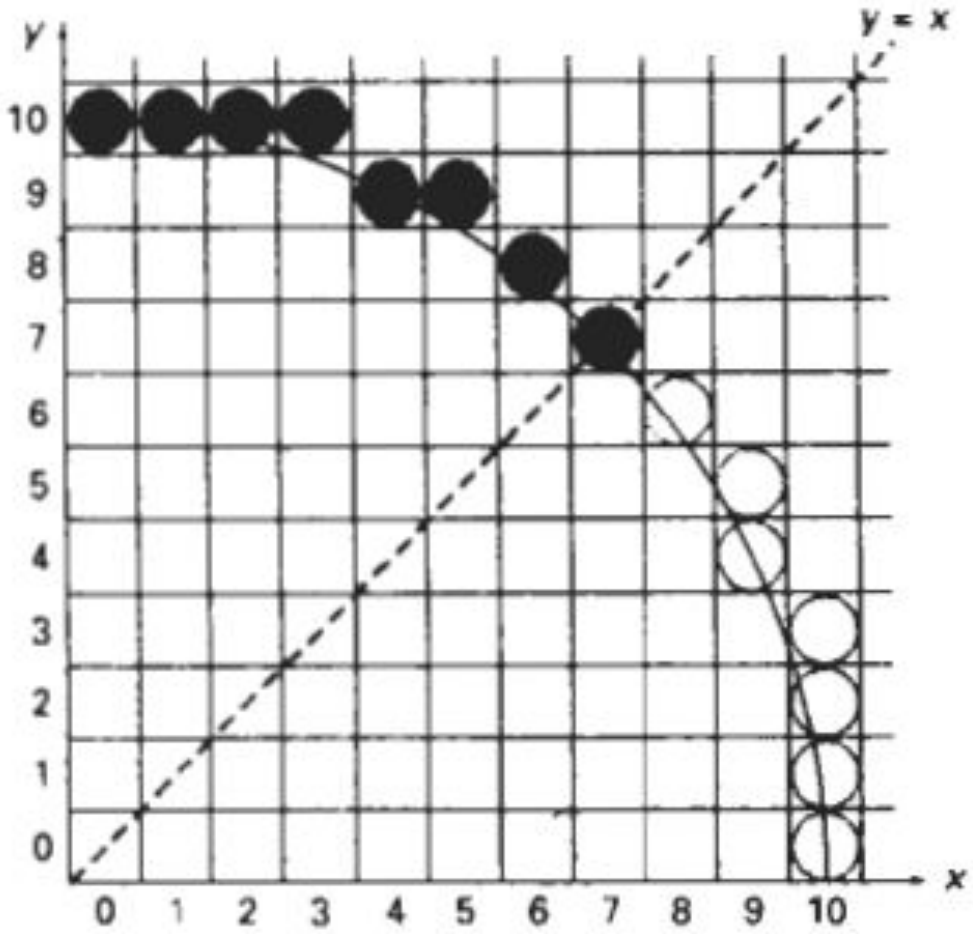
6. Repeat steps 3 through 5 until $x \geq y$.

```
void MidpointCircle (int radius, int value)
/* Assumes center of circle is at origin. Integer arithmetic only */
{
    int x = 0;
    int y = radius;
    int d = 1 - radius;
    CirclePoints (x, y, value);

    while (y > x) {
        if (d < 0)              /* Select E */
            d += 2 * x + 3;
        else {                  /* Select SE */
            d += 2 * (x - y) + 5;
            y--;
        }
        x++;
        CirclePoints (x, y, value);
    }  /* while */
}  /* MidpointCircle */
```



- r=10



Example

- End of Module 1