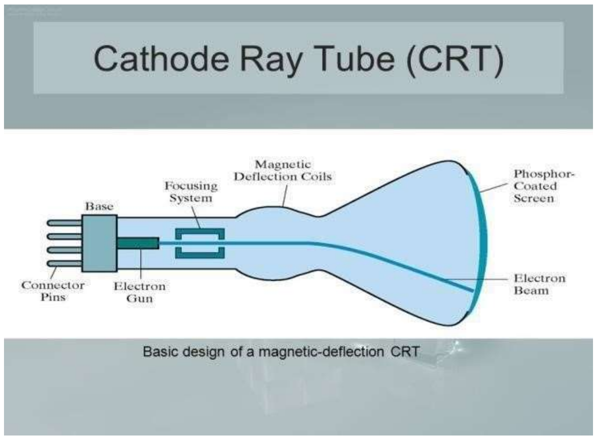
Computer Graphics hardware and software and OpenGL: Computer Graphics: Video Display Devices, Raster-Scan Systems Basics of computer graphics, Application of Computer Graphics. OpenGL: Introduction to OpenGL, coordinate reference frames, specifying two-dimensional world coordinate reference frames in OpenGL, OpenGL point functions, OpenGL line functions, point attributes, line attributes, curve attributes, OpenGL point attribute functions, OpenGL line attribute functions, Line drawing algorithms(DDA, Bresenham's)

1) **Write Basics and Application of computer graphics?**
2) **Explain video displays : Raster scan display , color CRT Monitors?**
3) **Short Note On the video Controller and Display Processor May ask for brief also?**
4) **Explain Graphic Work Stations And Viewing Sysytems?**
5) **Write note on introduction To OpenGL?**
6) **What are Coordinate reference Frames?**
7) **Explain Line drawing algorithms mainly Bresenham's and DDA?**

Video Display Devices

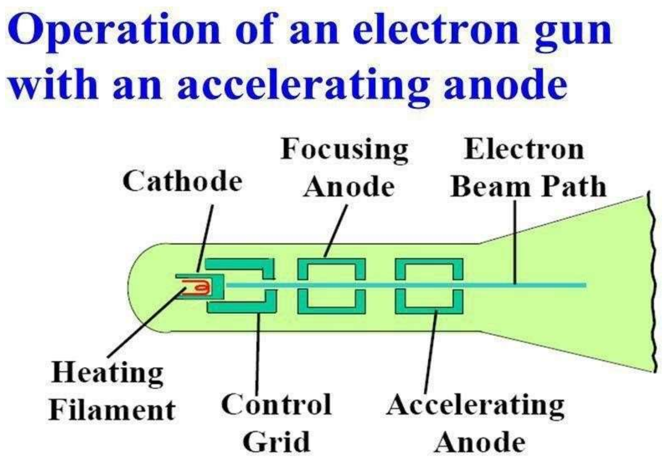
- The primary output device in a graphics system is a video monitor.
- Historically, the operation of most video monitors was based on the standard cathode ray tube (CRT) design, but several other technologies exist.
- In recent years, flat-panel displays have become significantly more popular due to their reduced power consumption and thinner designs.

Refresh Cathode-Ray Tubes



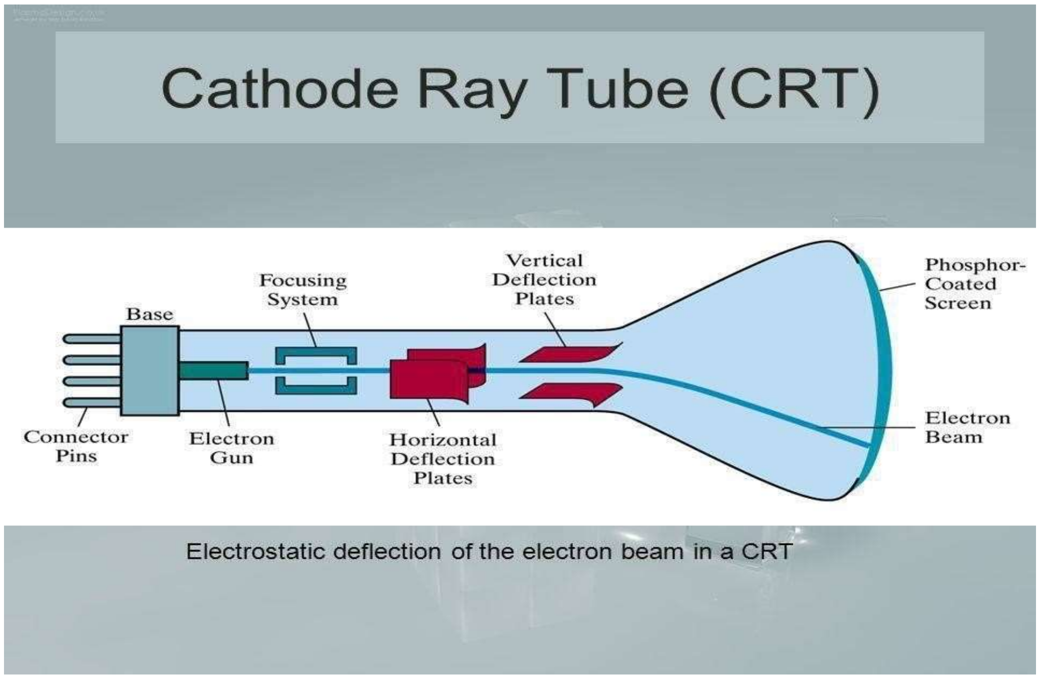
- A beam of electrons, emitted by an electron gun, passes through focusing and deflection systems that direct the beam toward specified positions on the phosphor-coated screen.
- The phosphor then emits a small spot of light at each position contacted by the electron beam and the light emitted by the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

<u>Operation of an electron gun with an accelarating anode</u>



- The primary components of an electron gun in a CRT are the heated metal cathode and a control grid.
- The heat is supplied to the cathode by directing a current through a coil of wire, called the filament, inside the cylindrical cathode structure.
- This causes electrons to be “boiled off” the hot cathode surface.
- Inside the CRT envelope, the free, negatively charged electrons are then accelerated toward the phosphor coating by a high positive voltage.
- Intensity of the electron beam is controlled by the voltage at the control grid.

- Since the amount of light emitted by the phosphor coating depends on the number of electrons striking the screen, the brightness of a display point is controlled by varying the voltage on the control grid.
- The focusing system in a CRT forces the electron beam to converge to a small cross section as it strikes the phosphor and it is accomplished with either electric or magnetic fields.
- With electrostatic focusing, the electron beam is passed through a positively charged metal cylinder so that electrons along the center line of the cylinder are in equilibrium position.
- Deflection of the electron beam can be controlled with either electric or magnetic fields.
- Cathode-ray tubes are commonly constructed with two pairs of magnetic-deflection coils
- One pair is mounted on the top and bottom of the CRT neck, and the other pair is mounted on opposite sides of the neck.
- The magnetic field produced by each pair of coils results in a traverse deflection force that is perpendicular to both the direction of the magnetic field and the direction of travel of the electron beam.
- Horizontal and vertical deflections are accomplished with these pair of coils

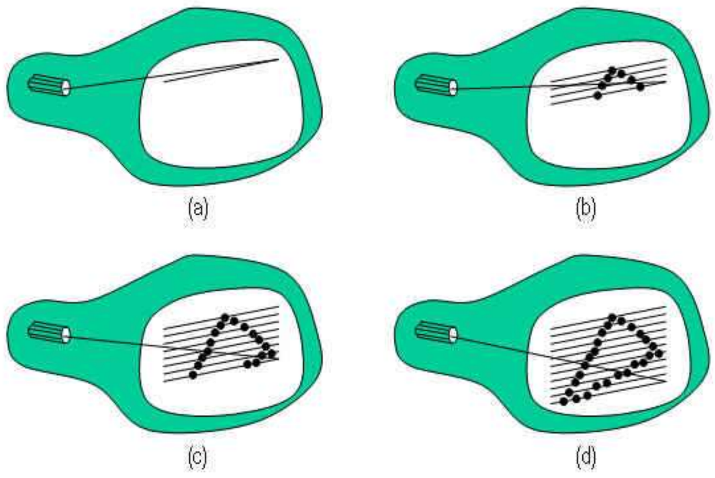

Cathode Ray Tube (CRT)
Base
Focusing System
Vertical Deflection Plates
Phosphor-Coated Screen
Connector Pins
Electron Gun
Horizontal Deflection Plates
Electron Beam
Electrostatic deflection of the electron beam in a CRT

- What we see on the screen is the combined effect of all the electrons light emissions: a glowing spot that quickly fades after all the excited phosphor electrons have returned to their ground energy level.
- The frequency of the light emitted by the phosphor is proportional to the energy difference between the excited quantum state and the ground state.
- Lower persistence phosphors required higher refresh rates to maintain a picture on the screen without flicker.

- The maximum number of points that can be displayed without overlap on a CRT is referred to as a resolution.
- Resolution of a CRT is dependent on the type of phosphor, the intensity to be displayed, and the focusing and deflection systems.
- High-resolution systems are often referred to as high-definition systems.

Raster-Scan Systems Basics of computer graphics:

In a raster-scan system, the electron beam is swept across the screen, one row at a time from top to bottom. 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 used for redrawn. Each screen point is referred to as a pixel or pel (picture element). Refreshing on raster-scan displays is carried out at the rate of 60 to 80 frames per second. The unit for refreshing rate is Hertz (Hz).



Case 1: In case of black and white systems

On black and white systems, the frame buffer storing the values of the pixels is called a bitmap. Each entry in the bitmap is a 1-bit data which determine the on (1) and off (0) of the intensity of the pixel.

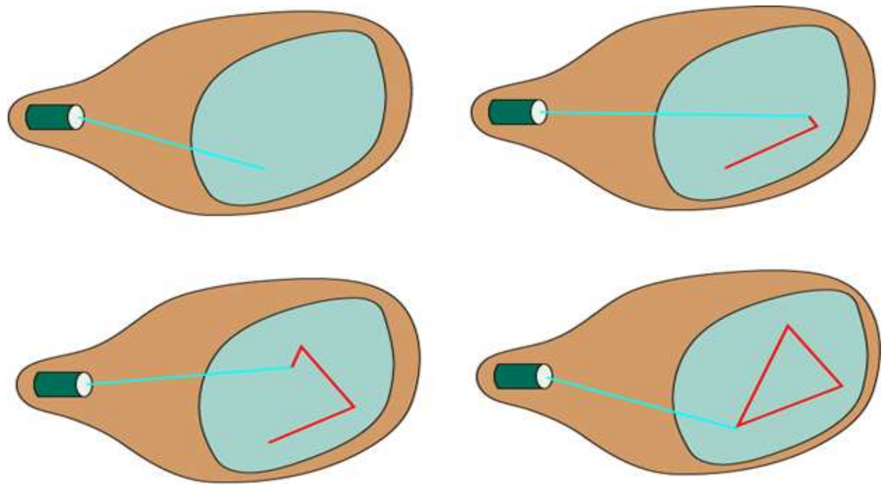
Case 2: In case of color systems

On color systems, the frame buffer storing the values of the pixels is called a pixmap (Though now a days many graphics libraries name it as bitmap too). ç Each entry in the pixmap occupies a number of bits to represent the color of the pixel. For a true color display, the number of bits for each entry is 24

(8 bits per red/green/blue channel, each channel 28=256 levels of intensity value, ie. 256 voltage settings for each of the red/green/blue electron guns).

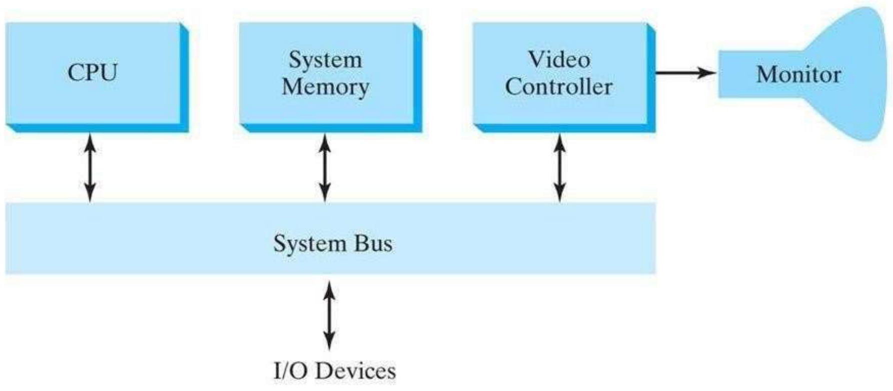
Random Scan Display:

Random Scan System uses an electron beam which operates like a pencil to create a line image on the CRT screen. The picture is constructed out of a sequence of straight-line segments. Each line segment is drawn on the screen by directing the beam to move from one point on the screen to the next, where its x & y coordinates define each point. After drawing the picture. The system cycles back to the first line and design all the lines of the image 30 to 60 time each second. The process is shown in fig:



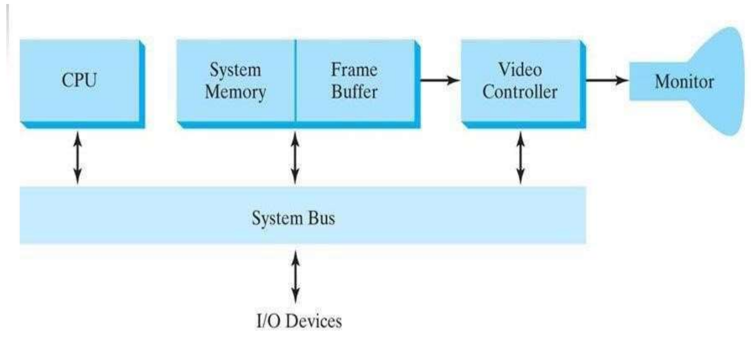
Raster-Scan Systems

Interactive raster-graphics systems typically employ several processing units.
In addition to the central processing unit (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 below Figure.



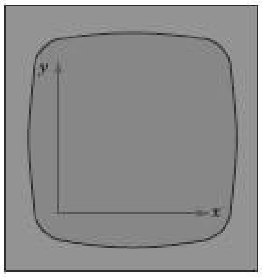
1 Video controller:

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

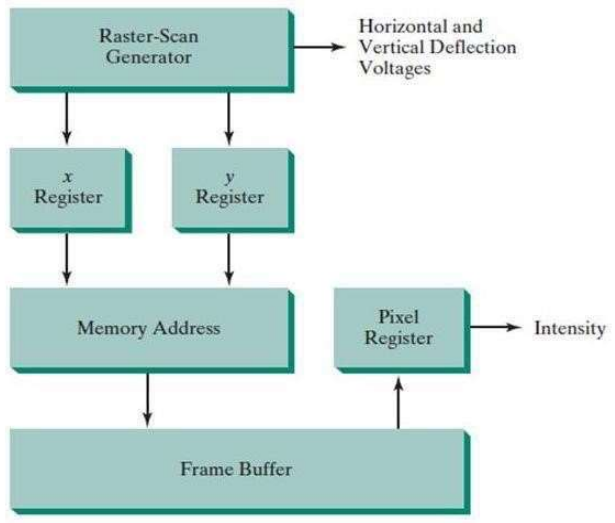


Working:

- Figure shows a two-dimensional Cartesian reference frame with the origin at the lowerleft screen corner.



- The screen surface is then represented as the first quadrant of a two -dimensional system with positive x and y values increasing from left to right and bottom of the screen to the top respectively.
- Pixel positions are then assigned integer x values that range from 0 to xmax across the screen, left to right, and integer y values that vary from 0 to ymax, bottom to top.

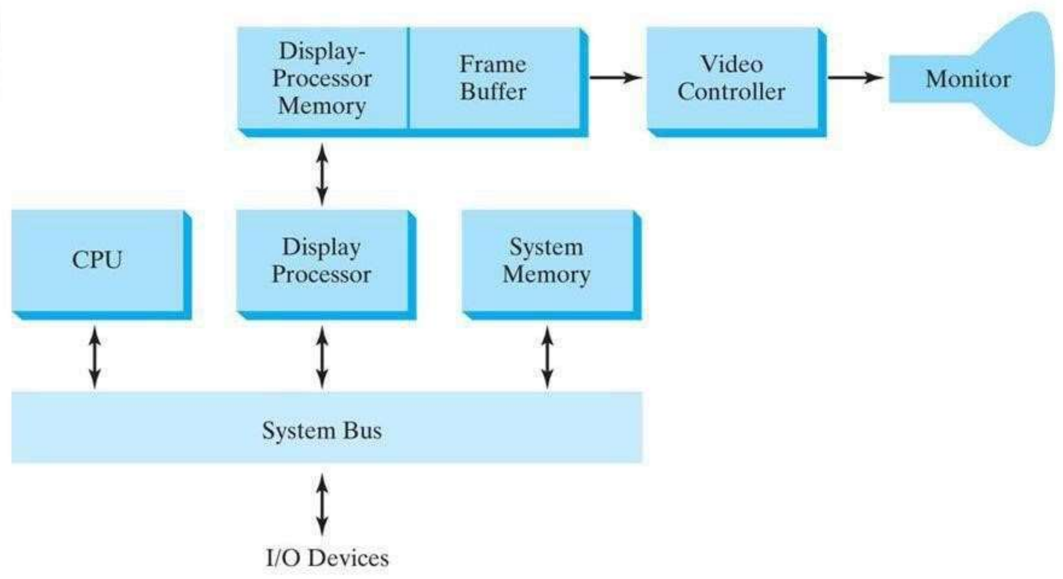


The basic refresh operations of the video controller are diagrammed

Basic Video Controller Refresh Operations

Two registers are used to store the coordinate values for the screen pixels.

- Initially, the x register is set to 0 and the y register is set to the value for the top scan line.
- The contents of the frame buffer at this pixel position are then retrieved and used to set the intensity of the CRT beam.
- Then the x register is incremented by 1, and the process is repeated for the next pixel on the top scan line.
- This procedure continues for each pixel along the top 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 set to the value for the next scan line down from the top of the screen.
- The procedure is repeated for each successive scan line.
- After cycling through all pixels along the bottom scan line, the video controller resets the registers to the first pixel position on the top scan line and the refresh process starts over



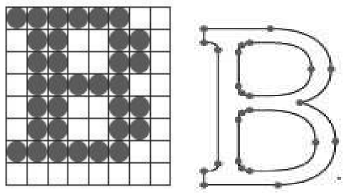
Raster-Scan Display Processor

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

Scan conversion:

- A major task of the display processor is digitizing a picture definition given in an application program into a set of pixel values for storage in the frame buffer.
- This digitization process is called scan conversion. **Example 1: displaying a line**
- Graphics commands specifying straight lines and other geometric objects are scan converted into a set of discrete points, corresponding to screen pixel positions.
- Scan converting a straight-line segment. **Example 2: displaying a character**
- Characters can be defined with rectangular pixel grids

- A character grid is displayed by superimposing the rectangular grid pattern into the frame buffer at a specified coordinate position.



- For characters that are defined as outlines, the shapes are scan-converted into the frame buffer by locating the pixel positions closest to the outline.

Methods to reduce memory requirements in display processor:

- In an effort to reduce memory requirements in raster systems, methods have been devised for organizing the frame buffer as a linked list and encoding the color information.
- One organization scheme is to store each scan line as a set of number pairs.
- Encoding methods can be useful in the digital storage and transmission of picture information

i)Run-length encoding:

- ❋ The first number in each pair can be a reference to a color value, and the second number can specify the number of adjacent pixels on the scan line that are to be displayed in that color.
- ❋ 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 color each.
- ❋ A similar approach can be taken when pixel colors change linearly.

ii)Cell encoding:

Base of Difference	Raster Scan System	Random Scan System
Electron Beam	The electron beam is swept across one row at a time from top to bottom	The electron beam is directed only to the parts of screen where a picture e drawn .
Resolution	Its resolution is poor because raster system in contrast produces zigzag lines that are plotted as discrete point sets.	Its resolution is good because this system produces smooth lines drawings because CRT beam directly follows the line path.
Picture Definition	Picture definition is stored as a set of intensity values for all screen points,called pixels in a refresh buffer area.	Picture definition is stored as a set of line drawing instructions in a display file.
Realistic Display	The capability of this system to store intensity values for pixel makes it well suited for the realistic display of scenes	These systems are designed for linedrawing and can't display realistic shaded scenes.

Color CRT Monitors

A CRT monitor displays color pictures by using a combination of phosphors that emit different-colored light.

It produces range of colors by combining the light emitted by different phosphors.

There are two basic techniques for color display:

1.Beam-penetration technique

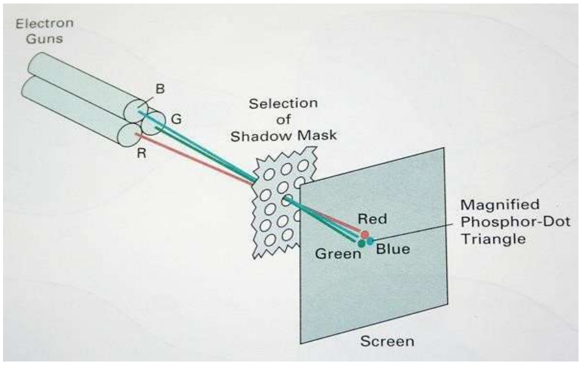
2.Shadow-mask technique

1) Beam-penetration technique:

- This technique is used with random scan monitors.
- In this technique inside of CRT coated with two phosphor layers usually red and green.
- The outer layer of red and inner layer of green phosphor.
- The color depends on how far the electron beam penetrates into the phosphor layer.
- A beam of fast electron penetrates more and excites inner green layer while slow eletron excites outer red layer.
- At intermediate beam speed we can produce combination of red and green lights which emit additional two colors orange and yellow.
- The beam acceleration voltage controls the speed of the electrons and hence color of pixel

2)Shadow-mask technique

- It produces wide range of colors as compared to beam-penetration technique.
- This technique is generally used in raster scan displays. Including color TV.
- In this technique CRT has three phosphor color dots at each pixel position.
- One dot for red, one for green and one for blue light. This is commonly known as Dot triangle.
- Here in CRT there are three electron guns present, one for each color dot. And a shadow mask grid just behind the phosphor coated screen.
- The shadow mask grid consists of series of holes aligned with the phosphor dot pattern.
- Three electron beams are deflected and focused as a group onto the shadow mask and when they pass through a hole they excite a dot triangle.
- In dot triangle three phosphor dots are arranged so that each electron beam can activate only its corresponding color dot when it passes through the shadow mask.
- A dot triangle when activated appears as a small dot on the screen which has color of combination of three small dots in the dot triangle.



Some of the applications of computer graphics are:

1. **ComputerArt:**
   Using computer graphics we can create fine and commercial art which include animation packages, paint packages. These packages provide facilities for designing object shapes and specifying object motion.Cartoon drawing, paintings, logo design can also be done.

2. **ComputerAidedDrawing:**
   Designing of buildings, automobile, aircraft is done with the help of computer aided drawing, this helps in providing minute details to the drawing and producing more accurate and sharp drawings with better specifications.

3. **PresentationGraphics:**
   For the preparation of reports or summarising the financial, statistical, mathematical, scientific, economic data for research reports, managerial reports, moreover creation of bar graphs, pie charts, time chart, can be done using the tools present in computer graphics.

4. **Entertainment:**
   Computer graphics finds a major part of its utility in the movie industry and game industry. Used for creating motion pictures , music video, television shows, cartoon animation films. In the game industry where focus and interactivity are the key players, computer graphics helps in providing such features in the efficient way.

5. **Education:**
   Computer generated models are extremely useful for teaching huge number of concepts and fundamentals in an easy to understand and learn manner. Using computer graphics many educational models can be created through which more interest can be generated among the students regarding the subject.

6. **Training:**
   Specialised system for training like simulators can be used for training the candidates in a way that can be grasped in a short span of time with better understanding. Creation of training modules using computer graphics is simple and very useful.

7. **Visualisation:**
   Today the need of visualise things have increased drastically, the need of visualisation can be seen in many advance technologies , data visualisation helps in finding insights of the data , to check and study the behaviour of processes around us we need appropriate visualisation which can be achieved through proper usage of computer graphics.

8. **ImageProcessing:**
   Various kinds of photographs or images require editing in order to be used in different places. Processing of existing images into refined ones for better interpretation is one of the many applications of computer graphics.

9. **MachineDrawing:**
   Computer graphics is very frequently used for designing, modifying and creation of various parts of machine and the whole machine itself, the main reason behind using computer graphics for this purpose is the precision and clarity we get from such drawing is ultimate and extremely desired for the safe manufacturing of machine using these drawings.

10. **GraphicalUserInterface:**
    The use of pictures, images, icons, pop-up menus, graphical objects helps in creating a user friendly environment where working is easy and pleasant, using computer graphics we can create such an atmosphere where everything can be automated and anyone can get the desired action performed in an easy fashion

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.
- 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 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 airport

Introduction to OpenGL

Open Graphics Library (OpenGL) is a cross-language (language independent), cross-platform (platform-independent) API for rendering 2D and 3D Vector Graphics(use of polygons to represent image). OpenGL API is designed mostly in hardware.

Basic syntax

➔Function names in the OpenGL basic library (also called the OpenGL core library) are

prefixed with gl. The component word first letter is capitalized.

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

➔ Symbolic constants that are used with certain functions as parameters are all in capital

letters, preceded by "GL", and component are separated by underscore.

➔ For eg:- GL_2D, GL_RGB, GL_CCW, GL_POLYGON,

GL_AMBIENT_AND_DIFFUSE.

➔The OpenGL functions also expect specific data types. For example, an OpenGL function

parameter might expect a value that is specified as a 32-bit integer. But the size of an

integer specification can be different on different machines.

➔ To indicate a specific data type, OpenGL uses special built-in, data-type names, such as

GLbyte, GLshort, GLint, GLfloat, GLdouble, Glboolean

Header Files

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

library.

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

of OpenGL libraries

#include<GL/gl.h>

#include <GL/glu.h>

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

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

Display-Window Management Using GLUT

Step 1: initialization of GLUT

- We are using the OpenGL Utility Toolkit, our first step is to initialize GLUT.
- This initialization function could also process any command line arguments, but we will not need to use these parameters for our first example programs.
- We perform the GLUT initialization with the statement **glutInit (&argc, argv);**

Step 2: Title

- We can state that a display window is to be created on the screen with a given caption for the title bar. This is accomplished with the function

  glutCreateWindow ("An Example OpenGL Program");
- where the single argument for this function can be any character string that we want to use for the display-window title.

Step 3: Specification of the display window

- Then we need to specify what the display window is to contain.
- For this, we create a picture using OpenGL functions and pass the picture definition to the GLUT routine glutDisplayFunc, which assigns our picture to the display window.
- Example: suppose we have the OpenGL code for describing a l ine segment in a procedure called lineSegment.

- Then the following function call passes the line -segment description to the display window: **glutDisplayFunc (lineSegment);**

Step 4: one more GLUT function

- But the display window is not yet on the screen.
- We need one more GLUT function to complete the window-processing operations.
- After execution of the following statement, all display windows that we have created, including their graphic content, ar e now activated:

glutMainLoop ( );

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

Step 5: these parameters using additional GLUT functions

- glutInitWindowPosition(50,100);
- glutInitWindowSize(400,300);

Coordinate Reference Frames

To describe a picture, we first decide upon

- A convenient Cartesian coordinate system, called the world-coordinate reference frame, which could be either 2D or 3D.

- We then describe the objects in our picture by giving their geometric specifications in terms of positions in world coordinates.
- Example: We define a straight-line segment with two endpoint positions, and a polygon is specified with a set of positions for its vertices.
- These coordinate positions are stored in the scene description along with other info about the objects, such as their color and their coordinate extents
- Co-ordinate extents :Co-ordinate extents are the minimum and maximum x, y, and z values for each object.

- A set of coordinate extents is also described as a bounding box for an object.
- Ex:For a 2D figure, the coordinate extents are sometimes called its bounding rectangle. Objects are then displayed by passing the scene description to the viewing routines which identify visible surfaces and map the objects to the frame buffer positions and then on the video monitor.
- The scan-conversion algorithm stores info about the scene, such as color values, at the appropriate locations in the frame buffer, and then the scene is displayed on the output device.

Screen co-ordinates:

- on a video monitor are referenced in integer screen coordinates, Locations which correspond to the integer pixel positions in the frame buffer.
- Scan-line algorithms for the graphics primitives use the coordinate descriptions to determine the locations of pixels
- Example: given the endpoint coordinates for a line segment, a display algorithm must calculate the positions for those pixels that lie along the line path between the endpoints.
- Since a pixel position occupies a finite area of the screen, the finite size of a pixel must be taken into account by the implementation algorithms.

- For the present, we assume that each integer screen position references the centre of a pixel area.
- Once pixel positions have been identified the color values must be stored in the frame buffer

Assume we have available a low-level procedure of the form

i)setPixel (x, y);

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

ii)getPixel (x, y, color);

Retrieves the current frame-buffer setting for a pixel location;

Parameter color receives an integer value corresponding to the combined RGB bit codes stored for the specified pixel at position (x,y).

Additional screen-coordinate information is needed for 3D scenes.

For a two-dimensional scene, all depth values are 0.

Absolute and Relative Coordinate Specifications ***Absolute***

coordinate:

So far, the coordinate references that we have discussed are stated as absolute coordinate values.

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

Relative coordinates:

However, some graphics packages also allow positions to be specified using relative coordinates.

This method is useful for various graphics applications, such as producing drawings with pen plotters, artist's drawing and painting systems, and graphics packages for publishing and printing applications.

Taking this approach, we can specify a coordinate position as an offset from the last position that was referenced (called the current position).

Specifying a Two-Dimensional World-Coordinate Reference Frame in OpenGL

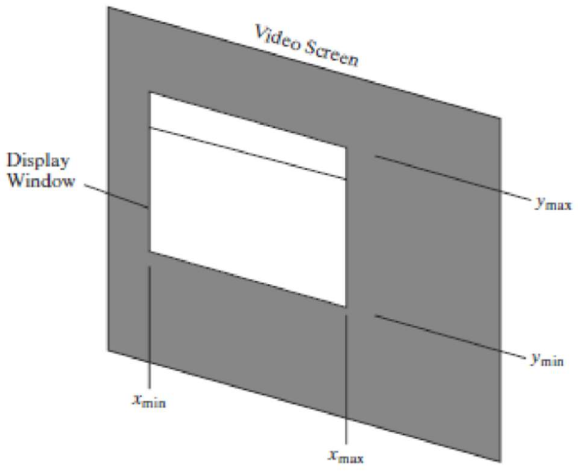
- The gluOrtho2D command is a function we can use to set up any 2D Cartesian reference frames.
- The arguments for this function are the four values defining the x and y coordinate limits for the picture we want to display.
- Since the gluOrtho2D function specifies an orthogonal projection, we need also to be sure that the coordinate values are placed in the OpenGL projection matrix.
- In addition, we could assign the identity matrix as the projection matrix before defining the world-coordinate range.
- This would ensure that the coordinate values were not accumulated with any values we may have previously set for the projection matrix.
- Thus, for our initial two-dimensional examples, we can define the coordinate frame for the screen display window with the following statements
- ***glMatrixMode (GL_PROJECTION);***

glLoadIdentity ( );

gluOrtho2D (xmin, xmax, ymin, ymax);

The display window will then be referenced by coordinates (xmin, ymin) at the lower-

left corner and by coordinates (xmax, ymax) at the upper-right corner, as shown in Figure below



- We can then designate one or more graphics primitives for display using the coordinate reference specified in the gluOrtho2D statement.
- If the coordinate extents of a primitive are within the coordinate range of the display window, all of the primitive will be displayed.
- Otherwise, only those parts of the primitive within the display-window coordinate limits

- will be shown.
- Also, when we set up the geometry describing a picture, all positions for the OpenGL primitives must be given in absolute coordinates, with respect to the reference frame defined in the gluOrtho2D function.

OpenGL Point Functions

- The type within glBegin() specifies the type of the object and its value can be as follows:

  GL_POINTS
- Each vertex is displayed as a point.
- The size of the point would be of at least one pixel.
- Then this coordinate position, along with other geometric descriptions we may have in our scene, is passed to the viewing routines.
- Unless we specify other attribute values, OpenGL primitives are displayed with a default size and color.
- The default color for primitives is white, and the default point size is equal to the size of a single screen pixel

  Syntax:

Case 1:

glBegin (GL_POINTS);

glVertex2i (50, 100);

glVertex2i (75, 150); glVertex2i (100, 200); glEnd ( );

Case 2: we could specify the coordinate values for the preceding points in arrays such as int point1 [ ] = {50, 100}; int point2 [ ] = {75, 150}; int point3 [ ] = {100, 200}; and call the OpenGL functions for plotting the three points as

***glBegin* (GL_POINTS);**

glVertex2iv (point1); glVertex2iv(point2);

glVertex2iv (point3);

glEnd ( );

Case 3:

specifying two point positions in a three dimensional world reference frame. In this case, we give the coordinates as explicit floating-point values:

glBegin (GL_POINTS);

glVertex3f (-78.05, 909.72, 14.60);

glVertex3f (261.91, -5200.67, 188.33);

glEnd ( );

Algorithm of Digital Differential Analyzer (DDA) Line Drawing

The Digital Differential Analyzer helps us to interpolate the variables on an interval from one point to another point. We can use the digital Differential Analyzer algorithm to perform rasterization on polygons, lines, and triangles.

Digital Differential Analyzer algorithm is also known as an **incremental method** of scan conversion. In this algorithm, we can perform the calculation in a step by step manner. We use the previous step result in the next step.

As we know the general equation of the straight line is:
$y = mx + c$

Here, **m** is the slope of **(x_1, y_1)** and **(x_2, y_2)**.
$m = (y_2 – y_1)/ (x_2 – x_1)$

Now, we consider one point **(x_k, y_k)** and **(x_{k+1}, y_{k+1})** as the next point.
Then the slope **$m = (y_{k+1} – y_k)/ (x_{k+1} – x_k)$**

Case 1: If **$m < 1$**
Then **x** coordinate tends to the Unit interval.
$x_{k+1} = x_k + 1$
$y_{k+1} = y_k + m$

Case 2: If **$m > 1$**
Then **y** coordinate tends to the Unit interval.
$y_{k+1} = y_k + 1$
$x_{k+1} = x_k + 1/m$

Case 3: If **$m = 1$**
Then **x and y** coordinate tend to the Unit interval.
$x_{k+1} = x_k + 1$
$y_{k+1} = y_k + 1$

We can calculate all intermediate points with the help of above three discussed cases.

Step 1: Start.
Step 2: We consider Starting point as **(x_1, y_1)**, and endingpoint **(x_2, y_2)**.

Step 3: Now, we have to calculate ▲**x** and ▲**y.**

$\blacktriangle x = x_2-x_1$

$\blacktriangle y = y_2-y_1$

$m = \blacktriangle y/\blacktriangle x$

Step 4: Now, we calculate three cases.

If **$m < 1$**
Then **x** change in Unit Interval
y moves with deviation

$(x_{k+1}, y_{k+1}) = (x_k+1, y_k+1)$

If **$m > 1$**
Then **x** moves with deviation **y**
change in Unit Interval

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

If **$m = 1$**
Then **x** moves in Unit Interval
y moves in Unit Interval

$(x_{k+1}, y_{k+1}) = (x_k+1, y_k+1)$

Step 5: We will repeat step 4 until we find the ending point of the line. **Step 6:** Stop.

Example: A line has a starting point (1,7) and ending point (11,17). Apply the Digital Differential Analyzer algorithm to plot a line.

Solution: We have two coordinates,

Starting Point = **(x_1, y_1) = (1,7)**

Ending Point = **(x_2, y_2) = (11,17)**

Step 1: First, we calculate **▲x**, **▲y** and **m.**

▲x = $x_2 - x_1$ = 11-1 = 10
▲y = $y_2 - y_1$ = 17-7 = 10
m = ▲y/▲x = 10/10 = 1

Step 2: Now, we calculate the number of steps.

▲x = ▲y = 10
Then, the number of steps = **10**

Step 3: We get **m = 1,** Third case is satisfied.

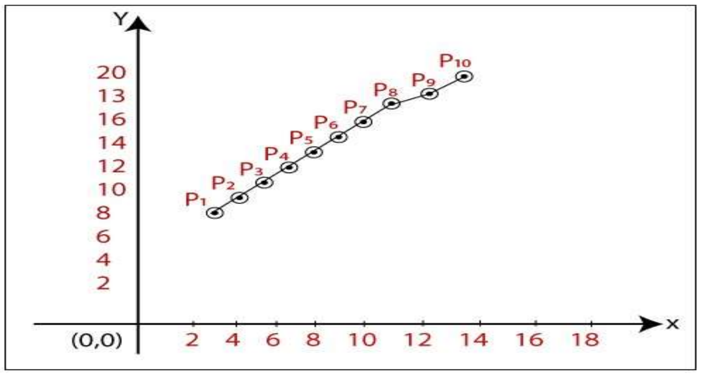
Now move to next step.

x_k	y_k	x_{k+1}	y_{k+1}	(x_{k+1}, y_{k+1})
1	7	2	8	(2, 8)
		3	9	(3, 9)
		4	10	(4, 10)
		5	11	(5, 11)
		6	12	(6, 12)

	7	13	(7, 13)
	8	14	(8, 14)
	9	15	(9, 15)
	10	16	(10, 16)
	11	17	(11, 17)

Step 4: We will repeat step 3 until we get the endpoints of the line.

Step 5: Stop.



The coordinates of drawn line are-

P_1 = (2, 8)
P_2 = (3, 9)
P_3 = (4, 10)
P_4 = (5, 11)
P_5 = (6, 12)
P_6 = (7, 13)
P_7 = (8, 14)
P_8 = (9, 15)
P_9 = (10, 16)
P_{10} = (11, 17)

Algorithm of Bresenham's Line Drawing Algorithm

Step 1: Start.

Step 2: Now, we consider Starting point as (x_1, y_1) and ending point (x_2, y_2). **Step 3:** Now, we have to calculate ▲x and ▲y.

$$\blacktriangle x = x_2 - x_1$$
$$\blacktriangle y = y_2 - y_1$$
$$m = \blacktriangle y / \blacktriangle x$$

Step 4: Now, we will calculate the decision parameter p_k with following formula.

$p_k = 2\blacktriangle y - \blacktriangle x$

Step 5: The initial coordinates of the line are (x_k, y_k), and the next coordinates are (x_{k+1}, y_{k+1}). Now, we are going to calculate two cases for decision parameter p_k **Case 1:** If

$$p_k < 0$$

Then

$$p_{k+1} = p_k + 2\blacktriangle y$$
$$x_{k+1} = x_k + 1$$
$$y_{k+1} = y_k$$

Case 2: If

$$p_k >= 0$$

Then

$$p_{k+1} = p_k + 2\blacktriangle y - 2\blacktriangle x$$
$$x_{k+1} = x_k + 1$$
$$y_{k+1} = y_k + 1$$

Step 6: We will repeat step 5 until we found the ending point of the line and the total number of iterations = **▲x-1.**

Step 7: Stop.

Example: A line has a starting point (9,18) and ending point (14,22). Apply the Bresenham's Line Drawing algorithm to plot a line.

Solution: We have two coordinates,

Starting Point = $(x_1, y_1) = (9,18)$

Ending Point = $(x_2, y_2) = (14,22)$

Step 1: First, we calculate ▲x, ▲y.

$$\blacktriangle x = x_2 - x_1 = 14-9 = 5$$
$$\blacktriangle y = y_2 - y_1 = 22-18 = 4$$

Step 2: Now, we are going to calculate the decision parameter (p_k) $p_k = 2\blacktriangle y - \blacktriangle x$

$$= 2 \times 4 - 5 = 3$$

The value of $p_k = 3$

Step 3: Now, we will check both the cases.

If $p_k >= 0$ Then **Case 2** is satisfied.

Thus

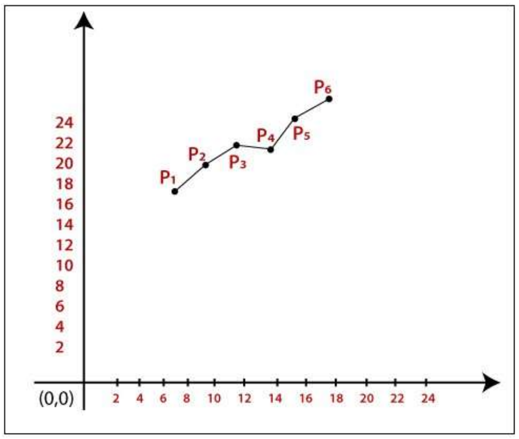
$p_{k+1} = p_k + 2\blacktriangle y - 2\blacktriangle x = 3 + (2 \times 4) - (2 \times 5) = 1$
$x_{k+1} = x_k + 1 = 9 + 1 = 10$
$y_{k+1} = y_k + 1 = 18 + 1 = 19$

Step 4: Now move to next step. We will calculate the coordinates until we reach the end point of the line.

$\blacktriangle x - 1 = 5 - 1 = 4$

p_k	p_{k+1}	x_{k+1}	y_{k+1}
		9	18
3	1	10	19
1	-1	11	20
-1	7	12	20
7	5	13	21
5	3	14	22

Step 5: Stop.



The Coordinates of drawn lines are-

$P_1 = (9, 18)$
$P_2 = (10, 19)$
$P_3 = (11, 20)$
$P_4 = (12, 20)$
$P_5 = (13, 21)$
$P_6 = (14, 22)$

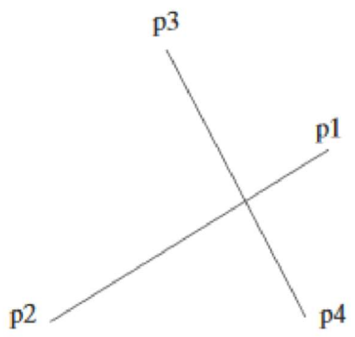
OpenGL LINE FUNCTIONS

Case 1: Lines *glBegin (GL_LINES); glVertex2iv (p1);*
glVertex2iv (p2);
glVertex2iv (p3);

glVertex2iv (p4);

glVertex2iv (p5);

glEnd ( );



Case 2: GL_LINE_STRIP:

Successive vertices are connected using line segments. However, the final vertex is not connected to the initial vertex.

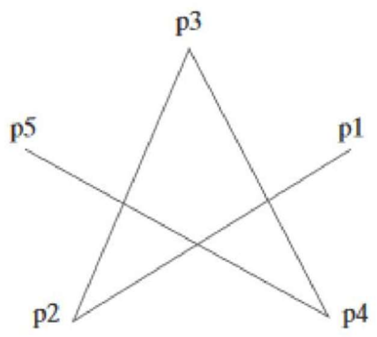
glBegin (GL_LINES_STRIP);

glVertex2iv (p1); glVertex2iv (p2);

glVertex2iv (p3); glVertex2iv (p4);

glVertex2iv (p5);

glEnd ( );



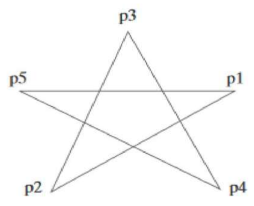
Case 3: GL_LINE_LOOP:

Successive vertices are connected using line segments to form a closed path or loop i.e., final vertex is connected to the initial vertex.

glBegin (GL_LINES_LOOP); glVertex2iv (p1); glVertex2iv (p2);

glVertex2iv (p3); glVertex2iv (p4); glVertex2iv (p5);

glEnd ( );



Point Attributes

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

In a state system: The displayed color and size of a point is determined by the current values stored in the attribute list.

Color components are set with RGB values or an index into a color table.

<u>Opengl Point-Attribute Functions</u> **<u>Color:</u>**

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

<u>Size:</u>

- We set the size for an OpenGL point with **glPointSize (size);** and the point is then displayed as a square block of pixels.
- Parameter size is assigned a positive floating-point value, which is rounded to an integer (unless the point is to be antialiased).

Line-Attribute Functions OpenGL

- In OpenGL straight-line segment with three attribute settings: line color, line-width, and line style.
- OpenGL provides a function for setting the width of a line and another function for specifying a line style, such as a dashed or dotted line.

OpenGL Line-Width Function

- Line width is set in OpenGL with the function
- Syntax: glLineWidth (width);
- We assign a floating-point value to parameter width, and this value is rounded to the nearest
- nonnegative integer.
- If the input value rounds to 0.0, the line is displayed with a standard width of 1.0, which is the default width.
- Some implementations of the line-width function might support only a limited number of widths, and some might not support widths other than 1.0.

OpenGL Line-Style Function

- By default, a straight-line segment is displayed as a solid line.
- But we can also display dashed lines, dotted lines, or a line with a combination of dashes and dot
- We can vary the length of the dashes and the spacing between dashes or dots.
- We set a current display style for lines with the OpenGL function: Syntax: glLineStipple (repeatFactor, pattern);

<u>Activating line style:</u>

- Before a line can be displayed in the current line-style pattern, we must activate the linestyle feature of OpenGL.
- **glEnable (GL_LINE_STIPPLE);**
- If we forget to include this enable function, solid lines are displayed; that is, the default pattern 0xFFFF is used to display line segments.
- At any time, we can turn off the line-pattern feature with **glDisable (GL_LINE_STIPPLE);**