

Module 1 Overview: Computer Graphics and OpenGL

what is computer graphics

Define Computer Graphics

Computer graphics is an art of drawing pictures, lines, charts, etc. using computers with the help of programming. Computer graphics image is made up of number of pixels. Pixel is the smallest addressable graphical unit represented on the computer screen.

List and explain any six applications of computer graphics (10 Marks)

Applications of Computer Graphics

discuss applications of computer graphics

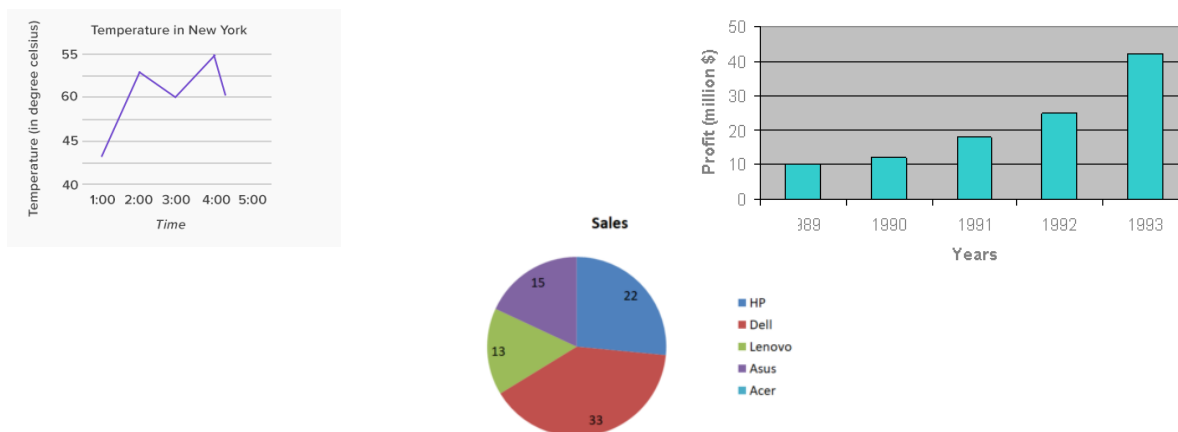
- Graphs and Charts
- Computer-Aided Design
- Virtual-Reality Environments
- Data Visualizations
- Education and Training
- Computer Art
- Entertainment
- Image Processing
- Graphical User Interfaces

Graphs and Charts

An early application for computer graphics is the display of simple data graphs usually plotted on a character printer. Data plotting is still one of the most common graphics application.

Graphs & charts are commonly used to summarize functional, statistical, mathematical, engineering and economic data for research reports, managerial summaries and other types of publications.

Typically examples of data plots are line graphs, bar charts, pie charts, surface graphs, contour plots and other displays showing relationships between multiple parameters in two dimensions, three dimensions, or higher-dimensional spaces.



A line graph is a type of chart used to show information that changes over time.

A bar chart or bar graph is a chart or graph that presents categorical data with rectangular bars with heights or lengths proportional to the values that they represent.

Pie Chart. Pie Chart: a special chart that uses "pie slices" to show relative sizes of data.

The Surface graph or series is used to represent a set of three-dimensional data as a mesh surface.

Computer-Aided Design

A major use of computer graphics is in design processes-particularly for engineering and architectural systems.

CAD, computer-aided design or CADD, computer-aided drafting and design methods are now routinely used in the automobiles, aircraft, spacecraft, computers, home appliances.

Circuits and networks for communications, water supply or other utilities are constructed with repeated placement of a few geographical shapes.

Animations are often used in CAD applications. Real-time, computer animations using wire-frame shapes are useful for quickly testing the performance of a vehicle or system.

Software packages for CAD applications typically provide the designer with a multiwindow environment,

The various windows can show enlarged sections or different views of objects

wire-frame displays allow the designer to see into the interior of the vehicle and to watch the behavior of inner components during motion.

When object designs are complete, or nearly complete, realistic lighting conditions and surface rendering are applied to produce displays that will show the appearance of the final product.

Virtual-Reality Environments

user can interact with the objects in a three-dimensional scene.

Specialized hardware devices provide three-dimensional viewing effects and allow the user to “pick up” objects in a scene.

Animations in virtual-reality environments are often used to train heavy equipment operators or to analyze the effectiveness of various cabin configurations and control placements.

Data Visualizations

Producing graphical representations for scientific, engineering, and medical data sets and processes generally referred to as scientific visualization.

business visualization is used in connection with data sets related to commerce, industry, and other nonscientific areas

Researchers, analysts, and others often need to deal with large amounts of information or to study the behavior of highly complex processes

satellite cameras and other recording sources are amassing large data files faster than they can be interpreted. Scanning these large sets of numbers to determine trends and relationships is a tedious and ineffective process. But if the data are converted to a visual form, the trends and patterns are often immediately apparent

In data visualization large amounts of information or to study the behavior of highly complex processes. There are many different kinds of data sets, and effective visualization schemes depend on the characteristics of the data.

Education and Training

Computer-generated models of physical, financial, political, social, economic, and other systems are often used as educational aids.

Models of physical processes are represented as the color-coded diagram which can help trainees to understand the operation of a system.

For some training applications, special hardware systems are designed.

Examples of such specialized systems are the simulators for practice sessions or training of ship captains, aircraft pilots, heavy-equipment operators, and air traffic-control personnel.

Although automobile simulators can be used as training systems, they are commonly employed to study the behavior of drivers in critical situations. Driver reactions in various traffic conditions can then be used as a basis for optimizing vehicle design to maximize traffic safety.

Computer Art

The picture is usually painted electronically on a graphics tablet using a stylus, which can simulate different brush strokes, brush widths and colors.

Both **fine art** and **commercial art** make use of computer-graphics methods. commercial software packages such as Lumina can be used to draw images.

Using a paintbrush program, a cartoonist created the characters

“automatic art” without intervention from the artist can be created.

“mathematical” art is generated by using math equations.

Commercial art also uses these “painting” techniques for **generating logos** and other designs, page layouts combining text and graphics.

TV advertising spots and for producing television commercials generally employ computer art.

Morphing is a technique where one object is transformed (metamorphosed) into another.

Entertainment

Television production, motion pictures, and music videos routinely use computer graphics methods.

Many TV series regularly employ computer-graphics methods to produce **special effects**.

computer graphics **to generate buildings, terrain features,** or other **backgrounds for a scene** are used.

computer-generated figures of people, animals, or cartoon characters with the live actors in a scene or to transform an actor’s face into another shape can be employed.

Photo-realistic techniques are employed in such films to give the computer-generated “actors”.

Image Processing

Image processing is a method to perform **some operations on an image**, in order to get an enhanced image or to extract some useful information from it.

It is used to improve picture quality, analyze images, or recognize visual patterns. image-processing methods are often used in computer graphics.

Medical applications also make extensive use of image-processing techniques for picture enhancements in tomography and in simulations of surgical operations.

It is also used in computed X-ray tomography(CT), position emission tomography(PET),and computed axial tomography(CAT).

Graphical User Interfaces

It is common now for applications software to provide a graphical user interface (GUI).

A major component of a graphical interface is a window manager that allows a user to display multiple, rectangular screen areas, called display windows.

Each screen display area can contain a different process, showing graphical or nongraphical information, and various methods can be used to activate a display window

Using an interactive pointing device, such as mouse, we can active a display window

An icon is a graphical symbol that is often designed to suggest the option it represents.

The advantages of icons are that they take up less screen space than corresponding textual descriptions and they can be understood more quickly if well designed.

The icons represent options for painting, drawing, zooming, typing text strings.

Define the following terms with respect to computer graphics

Bitmap

pics map

aspect ratio

frame buffer

VIDEO DISPLAY DEVICES

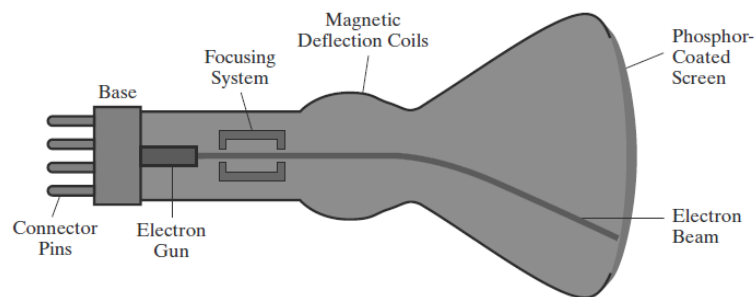
Typically, the primary output device in a graphics system is a video monitor. Video displays can be categorized in to two categories based on design as

- cathode-ray tube (CRT) design
- flat-panel displays

Explain refresh cathode ray tube with example (10 marks)

Mention raster scan display also

Refresh Cathode-Ray Tubes



Explain video display devices – a. CRT b. Raster Scan c. Random Scan d. Flat panel

A beam of electrons (cathode rays), 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. Because the light emitted by the phosphor fades very rapidly, some method is needed for maintaining the screen picture.

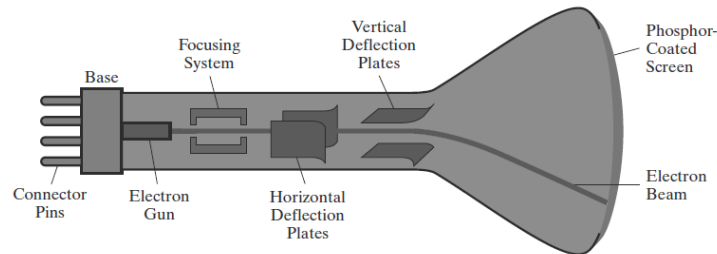
Most common method now employed for maintaining phosphor glow is to redraw the picture repeatedly by quickly directing the electron beam back over the same screen points. This type of display is called a **refresh CRT**, and the frequency at which a picture is redrawn on the screen is referred to as the **refresh rate**.

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.

The vacuum inside the CRT envelope, the free, negatively charged electrons are then accelerated toward the phosphor coating by a high positive voltage. The accelerating voltage can be generated with a positively charged metal coating on the inside of the CRT envelope near the phosphor screen, or an accelerating anode can be used to provide the positive voltage.

Focusing 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 an equilibrium position. As the beam moves to the outer edges of the screen, displayed images become blurred.

Cathode-ray tubes are now commonly constructed with magnetic-deflection coils mounted on the outside of the CRT envelope.



Persistence is defined as the time that it takes the emitted light from the screen to decay to one-tenth of its original intensity. Lower-persistence phosphors require higher refresh rates to maintain a picture **on** the screen without flicker.

The intensity is greatest at the center of the spot, and it decreases with a Gaussian distribution out to the edges of the spot.

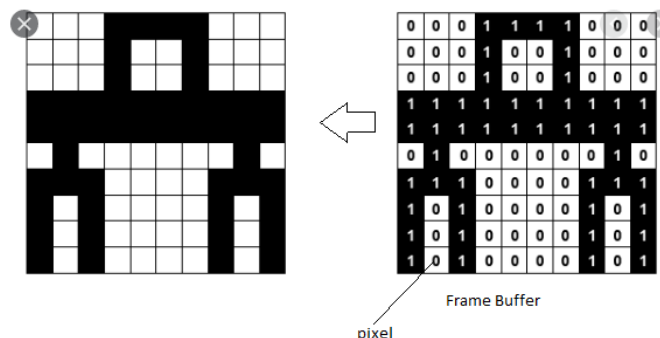
The maximum number of points that can be displayed without overlap on a CRT is referred to as the **resolution**. A more precise definition of resolution is the number of points per centimeter that can be plotted horizontally and vertically, although it is often simply stated as the total number of points in each direction.

Raster-Scan Displays

The most common type of graphics monitor employing a CRT is the raster-scan display.

In a raster-scan system, the electron beam is swept across the screen, one row at a time, from top to bottom. Each row is referred to as a scan line.

Picture definition is stored in a memory area called the **refresh buffer or frame buffer**, where the term frame refers to the total screen area.



Each screen spot that can be illuminated by the electron beam is referred to as a **pixel** or pel (shortened forms of picture element). It is the smallest element of an image. Since the refresh buffer is used to store the set of screen color values, it is also sometimes called a color buffer. Also, other kinds of pixel information, besides color, are stored in buffer locations, so all the different buffer areas are sometimes referred to collectively as the **“frame buffer.”**

The number of bits per pixel in a frame buffer is sometimes referred to as **depth of the buffer**. In the above example the depth is 1, since one bit is used to save a single pixel. Image is a Black and white Image A frame buffer with one bit per pixel is commonly called a **bitmap**, and a frame buffer with multiple bits per pixel is a **pixmap**.

Gray scale images are the ones where each pixel will have value between 0 to 255 (8 bits per pixel). Zero represents black and 255 white.

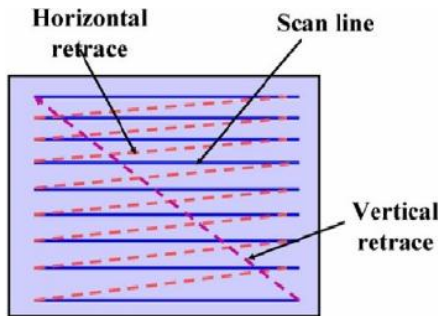
We can use 24 bits to represent each pixel

8 bits for Red + 8 bits for Green + 8 bits for Blue

These images are referred as True color images.

Display processor will read pixel values from frame buffer and display the contents on the screen. Property of CRT is that the phosphorous will emit light only for certain duration of time and hence the frame buffer must be displayed for multiple number of times to avoid flicker. Minimum number of times the frame buffer needs to be displayed is 60 times, it is also referred as 60Hz.

At the end of each scan line, the electron beam returns to the left side of the screen to begin displaying the next scan line. This is called the **horizontal retrace**. at the end of each frame, the electron beam returns to the top left corner of the screen, referred as **vertical retrace** to begin the next frame.



Aspect ratio defined as the number of pixel columns divided by the number of scan lines that can be displayed by the system.

RANDOM-SCAN DISPLAYS

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

Refresh rate on a random-scan system depends on the number of lines to be displayed on that system. Picture definition is now stored as a set of line-drawing commands in an area of memory referred to as the display list, refresh display file, vector file, or display program. To display a specified picture, the system cycles through the set of commands in the display file, drawing each component line in turn. After all line-drawing commands have been processed, the system cycles back to the first line command in the list.

Random-scan systems were designed for line-drawing applications, such as architectural and engineering layouts, and they cannot display realistic shaded scenes.

vector displays generally have higher resolutions than raster systems. Also, vector displays produce smooth line drawings because the CRT beam directly follows the line path.

difference between random and raster

Compare Random with Raster Scan Display(3 M)

Differentiate between Random and Raster Scan Display

| Raster Scan Display | Random Scan Display |
|---|------------------------------------|
| Its resolution is low | It has high Resolution |
| It is less expensive | It is more expensive |
| Solid pattern is easy to fill | Solid pattern is tough to fill |
| Refresh rate does not depend on the picture | Refresh rate depends on resolution |

Definitions

- o Resolution
- o Pixel
- o Aspect ratio
- o Intensity
- o Horizontal and vertical deflection.

| | |
|---|---|
| It uses interlacing | It does not use interlacing method. |
| It is suitable for realistic display. Modification requires processing | It is restricted to line drawing applications Modification is easy |

COLOR CRT MONITORS

A CRT monitor displays color pictures by using a combination of phosphors that emit different-colored light. The emitted light from the different phosphors merges to form a single perceived color, which depends on the particular set of phosphors that have been excited.

Color CRT are implemented using two methods

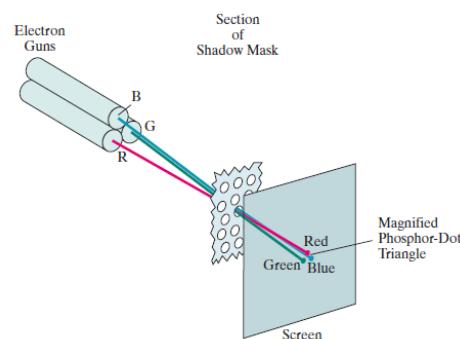
- beam-penetration
- Shadow-mask

One way to display color pictures is to coat the screen with layers of different colored phosphors. The emitted color depends on how far the electron beam penetrates the phosphor layers. This approach, called the **beam-penetration**,

Beam penetration has been an inexpensive way to produce color, but only a limited number of colors are possible, and picture quality is not as good as with other methods.

Shadow-mask methods are commonly used in raster-scan systems (including color TV) since they produce a much wider range of colors than the beam penetration method. This approach is based on the way that we seem to perceive colors as combinations of red, green, and blue components, called the **RGB color model**. Thus, a shadow-mask CRT uses three phosphor color dots at each pixel position. One phosphor dot emits a red light, another emits a green light, and the third emits a blue light.

This type of CRT has three electron guns, one for each color dot, and a shadow-mask grid just behind the phosphor-coated screen. The three electron beams are deflected and focused as a group onto the shadow mask, which contains a series of holes aligned with the phosphor-dot patterns. When the three beams pass through a hole in the shadow mask, they activate a dot triangle, which appears as a small color spot on the screen.



Flat Panel Display

The term flat panel display refers to a class of video device that have reduced volume, weight & power requirement compared to a CRT.

As flat panel display is thinner than CRTs, we can hang them on walls or wear on our wrists.

Flat panel display can be divided into two categories

Emissive displays: - the emissive display or emitters are devices that convert electrical energy into light. For Ex. Plasma panel, thin film electroluminescent displays and light emitting diodes.

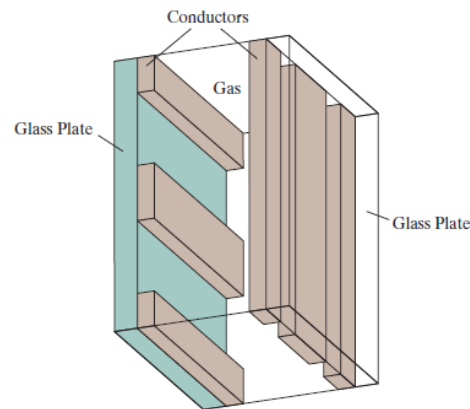
Non emissive displays: - non emissive display or non emitters use optical effects to convert sunlight or light from some other source into graphics patterns. For Ex. LCD (Liquid Crystal Display).

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.

Picture definition is stored in a refresh buffer and the firing voltages are applied to refresh the pixel positions, 60 times per second.

Alternating current methods are used to provide faster application of firing voltages and thus brighter displays.

Separation between pixels is provided by the electric field of conductor.

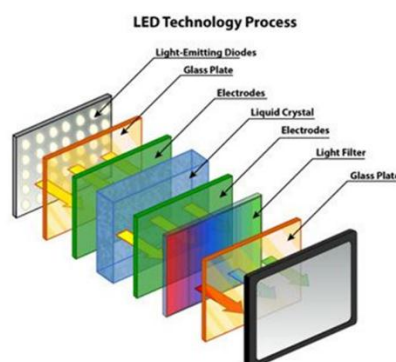
One disadvantage of plasma panels is they were strictly monochromatic device that means shows only one color other than black like black and white.

Light Emitting Diode (LED)

A matrix of diodes is arranged to form pixel positions in the displays.

The picture definition is stored in a refresh buffer.

Information is read from the refresh buffer and converted to voltage levels that are applied to the diodes to produce light patterns in the display.



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.

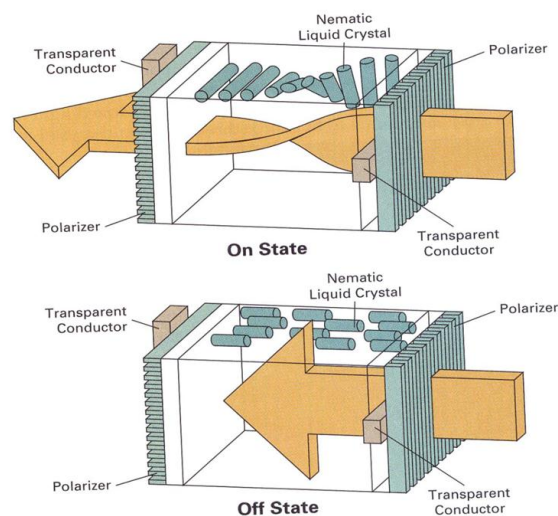
It consists of two glass plates each with light polarizer at right angles to each other sandwich the liquid crystal material between the plates.

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.

In the ON state polarized light passing through material is twisted so that it will pass through the opposite polarizer.

In the OFF state it will reflect back towards source.

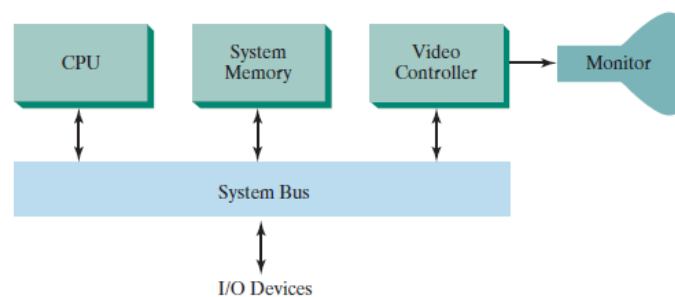


RASTER-SCAN SYSTEMS

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.

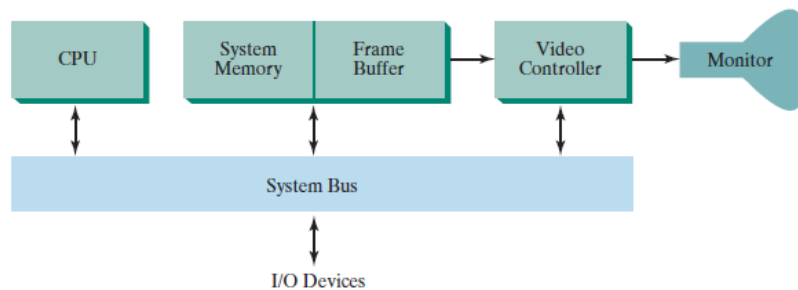
Organization of a simple raster system is shown in below Figure.



The figure below shows Architecture of a raster system with a fixed portion of the system memory reserved for the frame buffer.

video controller is given direct access to the frame-buffer memory.

discuss 1.video controller
2.display processor 3.frame
buffer 4.coordinate reference
frames

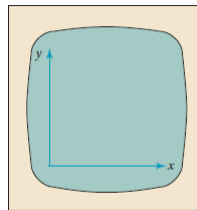


Frame-buffer locations, and the corresponding screen positions, are referenced in the Cartesian coordinates.

In an application (user) program, we use the commands within a graphics software package to set coordinate positions for displayed objects relative to the origin of the

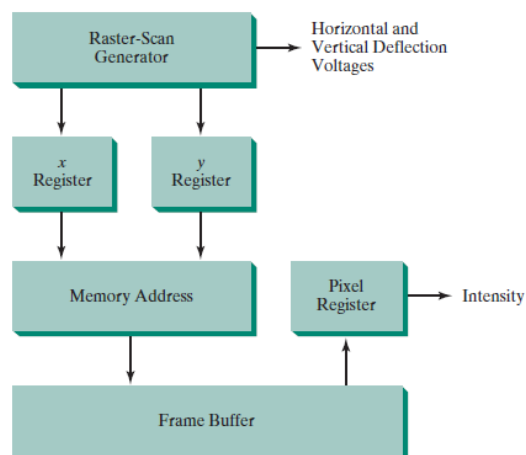
The coordinate origin is referenced at the lower-left corner of a screen display area by the software commands, although we can typically set the origin at any convenient location for a particular application.

Figure shows a two-dimensional Cartesian reference frame with the origin at the lower left screen corner



Pixel positions are then assigned integer x values that range from 0 to x_{\max} across the screen, left to right, and integer y values that vary from 0 to y_{\max} , bottom to top.

the basic refresh operations of the video controller is shown in the below figure



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.

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.

Since the screen must be refreshed at a rate of at least 60 frames per second, the simple procedure illustrated in above figure may not be accommodated by RAM chips if the cycle time is too slow.

To speed up pixel processing, video controllers can retrieve multiple pixel values from the refresh buffer on each pass. When group of pixels has been processed, the next block of pixel values is retrieved from the frame buffer.

A video controller can be designed to perform a number of other operations.

For various applications, the video controller can retrieve pixel values from different memory areas on different refresh cycles. This provides a fast mechanism for generating real-time animations.

Another video-controller task is the transformation of blocks of pixels, so that screen areas can be enlarged, reduced, or moved from one location to another during the refresh cycles.

In addition, the video controller often contains a lookup table, so that pixel values in the frame buffer are used to access the lookup table. This provides a fast method for changing screen intensity values.

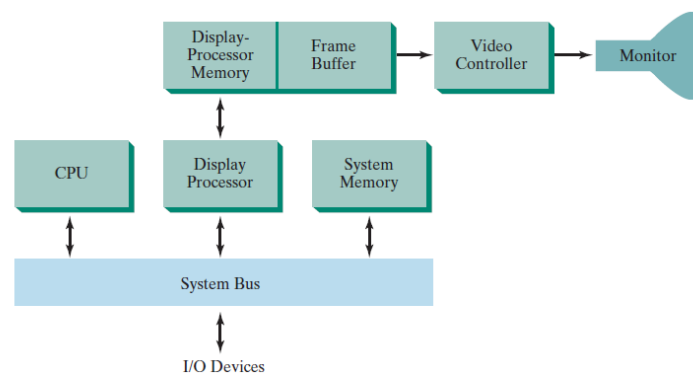
Finally, some systems are designed to allow the video controller to mix the framebuffer image with an input image from a television camera or other input device.

With a neat diagram, explain the architecture of Raster-Scan with Display Processor(10 M)

Explanation must include Run length encoding

Raster-Scan Display Processor

To have faster display of the frame buffer contents and to free the processor from activity of display, separate display processor was integrated to the architecture.



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.

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**.

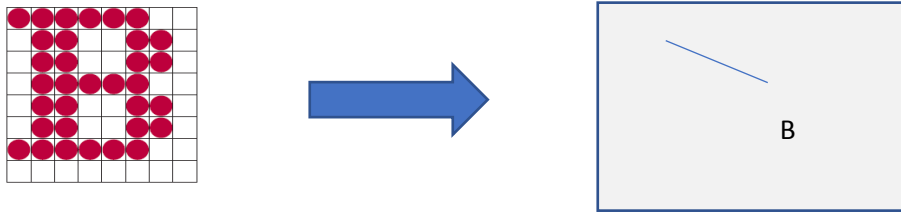
Example 1: displaying a line

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. The array size for character grids can vary from about 5 by 7 to 9 by 12 or more for higher-quality displays.

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



Additional operations of Display processors:

Display processors are also designed to perform a number of additional operations.

These functions include generating various line styles (dashed, dotted, or solid), displaying color areas, and applying transformations to the objects in a scene.

Display processors are typically designed to interface with interactive input devices, such as a mouse.

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. For example let us assume that the raster scan values are as follows

| | | | | | | | | | |
|-----|-----|-----|---|---|---|-----|-----|-----|-----|
| 255 | 255 | 255 | 0 | 0 | 0 | 100 | 125 | 125 | 125 |
|-----|-----|-----|---|---|---|-----|-----|-----|-----|

Usually in images the pixel values will be almost same in subsequent indexes. Linked list can be used to store the above information



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.

In Disadvantages of encoding:

The disadvantages of encoding runs are that color changes are difficult to record and storage requirements increase as the lengths of the runs decrease.

In addition, it is difficult for the display controller to process the raster when many short runs are involved.

Moreover, the size of the frame buffer is no longer a major concern, because of sharp declines in memory costs instead of having one pixel, if we consider regions as Matrix then we refer it as cell encoding.

GRAPHICS WORKSTATIONS

- Graphics workstations range from small general-purpose computer systems to multi-monitor facilities, often with ultra-large viewing screens.

- For a personal computer, screen resolutions vary from about 640 by 480 to 1280 by 1024, and diagonal screen lengths measure from 12 inches to over 21 inches
- Commercial workstations can also be obtained with a variety of devices specific applications
- High-definition graphics systems, with resolutions upto 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, system for stereoscopic viewing, and multi-channel wide-screen system.
- A multi-panel display can be used to show a large view of a single scene or several individual images.
- Large graphics displays can also be presented on curved viewing screens,
- A 360 degree paneled viewing system used in NASA control-tower simulator, which is used for training and for testing ways to solve air-traffic and runway problems at airports.

INPUT DEVICES what are different input devices

Keyboards, Button Boxes, and Dials

Alphanumeric **keyboard** on a graphics system is used primarily as a device for entering text strings, issuing certain commands, and selecting menu options.

The

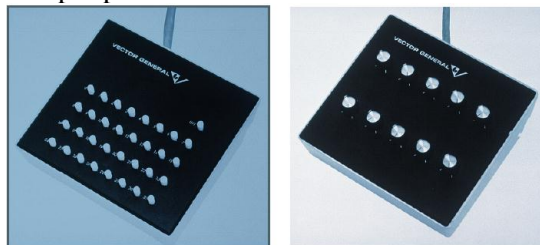
keyboard is an efficient device for inputting such non graphic data as picture labels associated with a graphics display. Keyboards can also be provided with features to facilitate entry of screen coordinates, menu selections, or graphics functions.

Cursor-control keys and function keys are common features on general purpose keyboards.

Function keys allow users to select frequently accessed operations with a single keystroke, and cursor-control keys are convenient for selecting a displayed object or a location by positioning the screen cursor.

Button Boxes

Buttons and switches are often used to input predefined functions.



Dials

Devices for entering scalar values.

Numerical values within some defined range are selected for input with dial rotations.

A potentiometer is used to measure dial rotation, which is then converted to the corresponding numerical value.



Mouse Devices

moved around on a flat surface to position the screen cursor. Wheels or rollers on the bottom of the mouse can be used to record the amount and direction of movement.

Another method for detecting mouse motion is with an optical sensor. For some optical systems, the mouse is moved over a special mouse pad that has a grid of horizontal and vertical lines. it is used for making relative changes in the position of the screen cursor. Additional features can be included in the basic mouse design to increase the number of allowable input parameters. The Z mouse



has three buttons, a thumbwheel on the side, a trackball on the top, and a standard mouse ball underneath. This design provides six degrees of freedom to select spatial positions, rotations, and other parameters. With the Z mouse, we can select an object displayed on a video monitor, rotate it, and move it in any direction. We could also use the Z mouse to navigate our viewing position and orientation through a three-dimensional scene. Applications of the Z mouse include virtual reality, CAD, and animation.

Trackballs



A **trackball** is a ball device that can be rotated with the fingers or palm of the hand to produce screen-cursor movement. Potentiometers, connected to the ball, measure the amount and direction of rotation. Laptop keyboards are often equipped with a trackball to eliminate the extra space required by a mouse.

An extension of the two-dimensional trackball concept is the **spaceball**, which provides six degrees of freedom. Spaceballs are used for three-dimensional positioning and selection operations in virtual-reality systems, modeling, animation, CAD, and other applications.

Joysticks

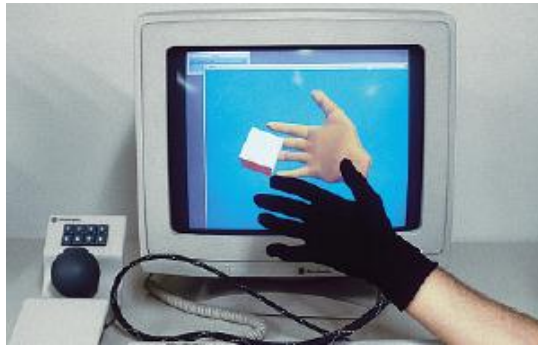
Another positioning device is the **joystick**, which consists of a small, vertical lever (called the stick) mounted on a base.

We use the joystick to steer the screen cursor around. The distance that the stick is moved in any direction from its center position corresponds to the relative screen-cursor movement in that direction.

Potentiometers mounted at the base of the joystick measure the amount of movement, and springs return the stick to the center position when it is released.

In another type of movable joystick, the stick is used to activate switches that cause the screen cursor to move at a constant rate in the direction selected. Eight switches, arranged in a circle, are sometimes provided so that the stick can select anyone of eight directions for cursor movement.

Data Gloves



Data Gloves can be used to grasp a “virtual object”

The glove is constructed with a series of sensors that detect hand and finger motions.

Electromagnetic coupling between transmitting antennas and receiving antennas are used to provide information about the position and orientation of the hand.

Input from the glove is used to position or manipulate objects in a virtual scene. A two-dimensional projection of the scene can be viewed on a video monitor, or a three-dimensional projection can be viewed with a headset.

Digitizers

A common device for drawing, painting, or interactively selecting positions is a **digitizer**. These devices can be designed to input coordinate values in either a two-dimensional or a three-dimensional space.

In engineering or architectural applications, a digitizer is often used to scan a drawing or object and to input a set of discrete coordinate positions.

One type of digitizer is the **graphics tablet** (also referred to as a data tablet), which is used to input two-dimensional coordinates by activating a hand cursor or stylus at selected positions on a flat surface.

Many graphics tablets are constructed with a rectangular grid of wires embedded in the tablet surface. Electromagnetic pulses are generated in sequence along the wires, and an electric signal is induced in a wire coil in an activated stylus or hand-cursor to record a tablet position.

Image Scanners

image scanner by passing an optical scanning mechanism over the information to be stored. The gradations of gray scale or color are then recorded and stored in an array.

Once we have the internal representation of a picture, we can apply transformations to rotate, scale, or crop the picture to a particular screen area.

We can also apply various image-processing methods to modify the array representation of the picture. For scanned text input, various editing operations can be performed on the stored documents. Scanners are available in a variety of sizes and capabilities.

Touch Panels

touch panels allow displayed objects or screen positions to be selected with the touch of a finger. A typical application of touch panels is for the selection of processing options that are represented as a menu of graphical icons

Existing screens can be adapted for touch input by fitting a transparent device containing a touch-sensing mechanism over the video monitor screen. Touch input can be recorded using optical, electrical, or acoustical methods.

Optical touch panels employ a line of infrared light-emitting diodes (LEDs) along one vertical edge and along one horizontal edge of the frame. Light detectors are placed along the opposite vertical and horizontal edges. These detectors are used to record which beams are interrupted when the panel is touched. The two crossing beams that are interrupted identify the horizontal and vertical coordinates of the screen position selected.

An electrical touch panel is constructed with two transparent plates separated by a small distance. One of the plates is coated with a conducting material, and the other plate is coated with a resistive material. When the outer plate is touched, it is forced into contact with the inner plate. This contact creates a voltage drop across the resistive plate that is converted to the coordinate values of the selected screen position.

In acoustical touch panels, high-frequency sound waves are generated in horizontal and vertical directions across a glass plate. Touching the screen causes part of each wave to be reflected from the finger to the emitters. The screen position at the point of contact is calculated from a measurement of the time interval between the transmission of each wave and its reflection to the emitter.

Light Pen

Such pencil-shaped devices are used to select screen positions by detecting the light coming from points on the CRT screen. They are sensitive to the short burst of light emitted from the phosphor coating at the instant the electron beam strikes a particular point.

An activated light pen, pointed at a spot on the screen as the electron beam lights up that spot, generates an electrical pulse that causes the coordinate position of the electron beam to be recorded.

Disadvantages

Prolonged use of the light pen can cause arm fatigue.

light pens require special implementations for some applications since they cannot detect positions within black areas light.

pens sometimes give false readings due to background lighting in a room.

GRAPHICS SOFTWARE

There are two broad classifications for computer-graphics software:

- Special purpose packages and
- general programming packages

Special-purpose packages are designed for nonprogrammers who want to generate pictures, graphs, or charts in some application area without worrying about the graphics procedures that might be needed to produce such displays.

Examples: painting programs, business, medical, and engineering CAD systems.

General programming package provides a library of graphics functions that can be used in a programming language such as C, C++, Java, or Fortran.

Examples: OpenGL, VRML (Virtual-Reality Modeling Language), Java 2D, and Java 3D.

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

Explain with diagram the different cartesian reference frames are used in the process of constructing and displaying the scenes (8 M)

Coordinate Representations

To generate a picture using a programming package, we first need to give the geometric descriptions of the objects that are to be displayed. These descriptions determine the locations and shapes of the objects. For example line is defined by two vertex points.

Cartesian-coordinate reference frame refers to area where the vertex can be mentioned.

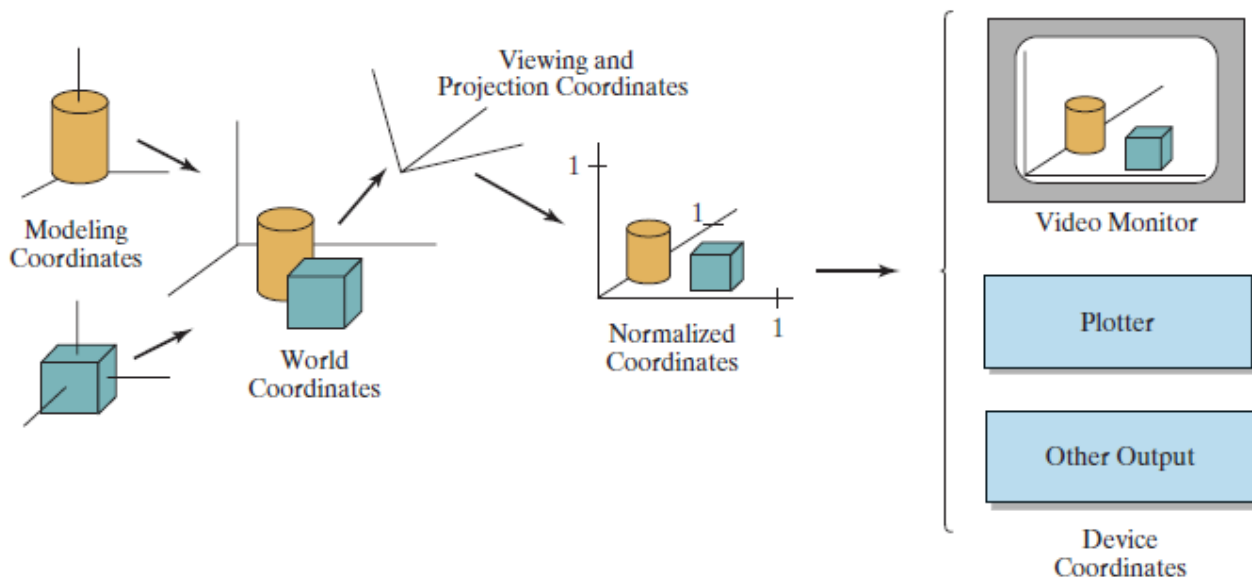
Different Cartesian reference frames are used in the process of constructing and displaying a scene

Stage 1: coordinate reference frame for each object. These reference frames are called **modeling coordinates**. Description of how line must be mentioned itself is referred as modelling coordinates.

Stage 2: Once the individual object shapes have been specified, we can construct ("model") a scene by placing the objects into appropriate locations within a scene reference frame called **world coordinates**. Orthogonal area is an example of world coordinates.

This step involves the transformation of the individual modeling-coordinate frames to specified positions and orientations within the world-coordinate frame.

Geometric descriptions in modeling coordinates and world coordinates can be given in any convenient floating-point or integer values, without regard for the constraints of a particular output device.



Stage 3: After all objects are defined in world coordinate, it is processed through various routines on to one or more output-device reference frames for display. This process is called the **viewing pipeline**.

World coordinate positions are first converted to viewing coordinates

Then object locations are transformed to a two-dimensional projection of the scene, which corresponds to what we will see on the output device.

The scene is then stored in **normalized coordinates**, where each coordinate value is in the range from -1 to 1. Normalized coordinates are also referred to as *normalized device coordinates*, since using this representation makes a graphics package independent of the coordinate range for any specific output device.

Identify visible surfaces and eliminate picture parts outside of the bounds for the viewer want to show on the display device.

Stage 4: Finally, the picture is scan converted into the refresh buffer of a raster system for display. The coordinate systems for display devices are generally called **device coordinates**, or **screen coordinates**.

An initial modeling-coordinate position (x_{mc}, y_{mc}, z_{mc}) in this illustration is transferred to world coordinates, then to viewing and projection coordinates, then to left-handed normalized coordinates, and finally to a device-coordinate position (x_{dc}, y_{dc}) with the sequence:

$$(x_{mc}, y_{mc}, z_{mc}) \rightarrow (x_{wc}, y_{wc}, z_{wc}) \rightarrow (x_{vc}, y_{vc}, z_{vc}) \rightarrow (x_{pc}, y_{pc}, z_{pc}) \\ \rightarrow (x_{nc}, y_{nc}, z_{nc}) \rightarrow (x_{dc}, y_{dc})$$

Device coordinates (x_{dc}, y_{dc}) are integers within the range $(0, 0)$ to (x_{max}, y_{max}) for a particular output device.

INTRODUCTION TO OPENGL

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

Basic OpenGL Syntax opengl is cross language, cross platform API used for rendering 2D and 3D vector graphics.

Function names in the **OpenGL basic library** (also called the **OpenGL core library**) are prefixed with **gl** and each component word within a function name has its first letter capitalized.

glBegin, glClear, glCopyPixels, glPolygonMode

Certain functions require arguments to be passed and all these are symbolic constants. All such constants begin with the uppercase letters GL. In addition, component words within a constant name are written in capital letters, and the underscore () is used as a separator between all component words in the name.

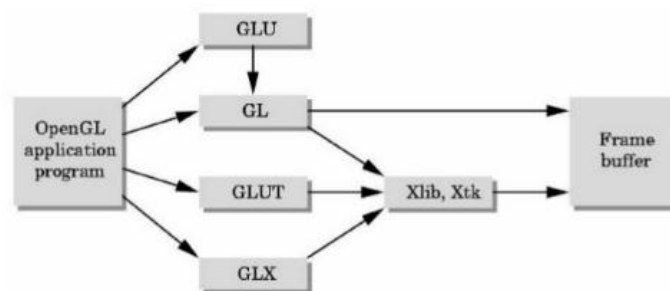
GL_RGB, GL_CCW, GL_POLYGON

OpenGL uses special built-in, data-type names, such as

GLbyte, GLshort, GLint, GLfloat, GLdouble, GLboolean

What is open GL with the help of block diagram explain library organization of open GL program and give general structure of open GL program

OpenGL provides a powerful but primitive set of rendering commands, and all higher-level drawing must be done in terms of these commands.



Related Libraries

The OpenGL Utility (GLU) provides routines for setting up viewing and projection matrices

gluCylinder, gluLookAt

There is also an object oriented toolkit based on OpenGL, called Open Inventor, which provides routines and predefined object shapes for interactive three-dimensional applications.

The **OpenGL Extension to the X Window System (GLX)** provides a set of routines that are prefixed with the letters **glX**. Apple systems can use the **Apple GL (AGL)** interface for window-management operations. Function names for this library are prefixed with **agl**.

Header Files

In all of our graphics programs, we will need to include the header file for the OpenGL core library. For most applications we will also need GLU

```
#include <GL/gl.h>
#include <GL/glu.h>
#include <GL/glut.h>
```

In addition, we will often need to include header files that are required by the C++ code. For example,

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
```

Display-Window Management Using GLUT

To get started, we can consider a simplified, minimal number of operations for displaying a picture.

GLUT is initialized with the statement

```
glutInit (&argc, argv);
```

Display window is created on the screen with a given caption for the title bar. This is accomplished with the function

```
glutCreateWindow ("An Example OpenGL Program");
```

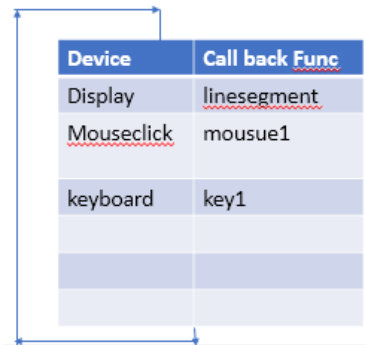
where the single argument for this function can be any character string we want to use for the display-window title. Then we need to specify what the display window is to contain. For this, we create a picture using OpenGL functions and pass the picture definition to the GLUT routine `glutDisplayFunc`.

glutDisplayFunc (lineSegment);

to keep events in a loop, the following glut function is used

glutMainLoop ();

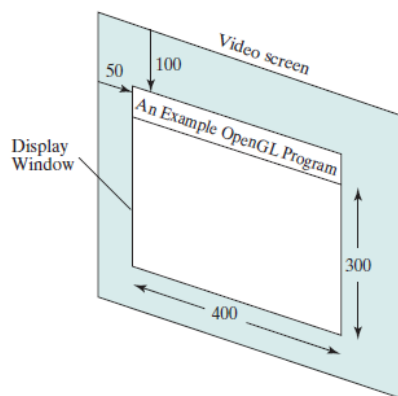
It displays the initial graphics and puts the program into an infinite loop that checks for input from devices such as a mouse or keyboard.



| Device | Call back Func |
|-------------------|--------------------|
| Display | <u>linesegment</u> |
| <u>Mouseclick</u> | mousue1 |
| keyboard | key1 |
| | |
| | |

We use the **glutInitWindowPosition** function to give an initial location for the top left corner of the display window. This position is specified in integer screen coordinates, whose origin is at the upper-left corner of the screen.

glutInitWindowPosition (50, 100);



Similarly, the **glutInitWindowSize** function is used to set the initial pixel width and height of the display window

```
glutInitWindowSize (400, 300);
```

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

For example, the following command specifies that a single refresh buffer is to be used for the display window and that the RGB (red, green, blue) color mode is to be used for selecting color values.

glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);

The values of the constants passed to this function are combined using a logical *or* operation.

A Complete OpenGL Program

For the display window, we can choose a background color.

glClearColor (1.0, 1.0, 1.0, 0.0);

The first three arguments in this function set each of the red, green, and blue component colors to the value 1.0. Thus we get a white color for the display window.

One use for the alpha value is as a “blending” parameter. When we activate the OpenGL blending operations, alpha values can be used to determine the resulting color for two overlapping objects.

Although the **glClearColor** command assigns a color to the display window, it does not put the display window on the screen. To get the assigned window color displayed, we need to invoke the following OpenGL function.

glClear (GL_COLOR_BUFFER_BIT);

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

In addition to setting the background color for the display window, we can choose a variety of color schemes for the objects

glColor3f (1.0, 0.0, 0.0);

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

We can set the projection type (mode) and other viewing parameters that we need with the following two functions.

glMatrixMode (GL_PROJECTION);

gluOrtho2D (0.0, 200.0, 0.0, 150.0);

This specifies that an orthogonal projection is to be used to map the contents of a two-dimensional (2D) rectangular area of world coordinates to the screen, and that the *x*-coordinate values within this rectangle range from 0.0 to 200.0 with *y*-coordinate values ranging from 0.0 to 150.0.

Finally, we need to call the appropriate OpenGL routines to create our line segment.

```
glBegin (GL_LINES);
    glVertex2i (180, 15);
    glVertex2i (10, 145);
glEnd ();
```

program to draw a line

```
#include <GL/glut.h>
void init ()
{
    glMatrixMode (GL_PROJECTION); // Set projection parameters.
    glLoadIdentity()
    gluOrtho2D (0, 200, 0, 200);
    glMatrixMode (GL_MODELVIEW);
}
void lineSegment (void)
{
    glClearColor (1, 1, 1, 0); // Set display-window color to white.
    glClear (GL_COLOR_BUFFER_BIT); // Clear display window.
    glColor3f (1, 0, 0); // Set line segment color to red.
    glBegin (GL_LINES);
        glVertex2i (150, 150); // Specify line-segment geometry.
        glVertex2i (20, 20);
    glEnd ();
    glFlush (); // Process all OpenGL routines as quickly as possible.
}
void main (int argc, char** argv)
{
    glutInit (&argc, argv); // Initialize GLUT.
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB); // Set display mode.
    glutInitWindowPosition (50, 100); // Set top-left display-window position.
    glutInitWindowSize (500, 500); // Set display-window width and height.
    glutCreateWindow ("An Example OpenGL Program"); // Create display window.
    init (); // Execute initialization procedure.
    glutDisplayFunc (lineSegment); // Send graphics to display window.
    glutMainLoop (); // Display everything and wait.
```

}

COORDINATE REFERENCE FRAMES

To describe a picture, we first decide upon a convenient Cartesian coordinate system, called the **world-coordinate reference frame**, which could be either two dimensional or three-dimensional. We then describe the objects, For example 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 information about the objects, such as their color and their **coordinate extents**, which 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.

Objects are then displayed by passing the scene information to the viewing routines, which identify visible surfaces and ultimately map the objects to positions on the video monitor through scan conversion.

Screen Coordinates

Locations on a video monitor are referenced in integer screen coordinates, which correspond to the pixel positions in the frame buffer. Pixel coordinate values give the scan line number (the y value) and the column number (the x value along a scan line). Hardware processes, such as screen refreshing, typically address pixel positions with respect to the top-left corner of the screen. Scan lines are then referenced from 0, at the top of the screen, to some integer value, y_{max}, at the bottom of the screen, and pixel positions along each scan line are numbered from 0 to x_{max}, left to right.

Scan-line algorithms for the graphics primitives use the defining coordinate descriptions to determine the locations of pixels that are to be displayed. For 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.

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

For this purpose, we will assume that

we have available a low-level procedure of the form

setPixel (x, y);

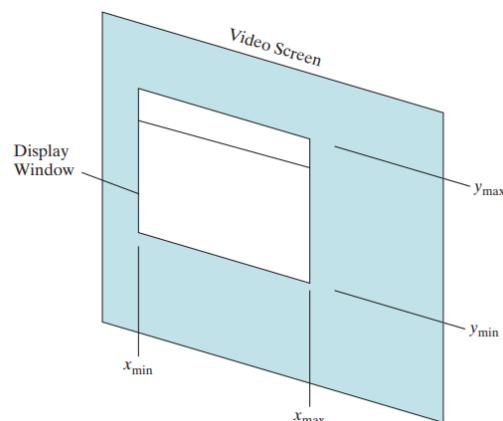
This procedure stores the current color setting into the frame buffer at integer position (x, y), relative to the selected position of the screen-coordinate origin. We sometimes also will want to be able to retrieve the current frame-buffer setting for a pixel location. So we will assume that we have the following low-level function for obtaining a frame-buffer color value.

getPixel (x, y, color);

SPECIFYING A TWO-DIMENSIONAL WORLD-COORDINATE REFERENCE FRAME IN OpenGL

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

```
glMatrixMode (GL_PROJECTION);
glLoadIdentity ( );
gluOrtho2D (xmin, xmax, ymin, ymax)
```



OpenGL POINT FUNCTIONS

Point can be specified in OpenGL as

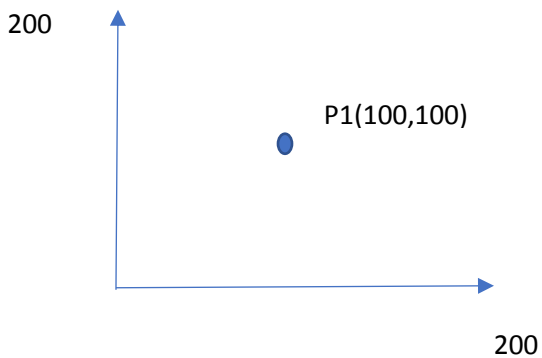
```
glBegin (GL_POINTS);
    glVertex* ( );
glEnd ( );
```

`glVertex* ();` where * can be represented as **ntv**

n: number of dimension. 2D or 3D

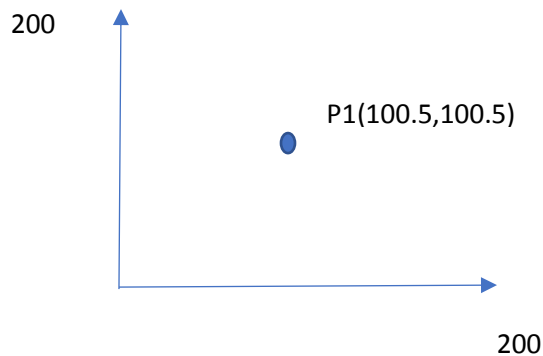
t: represents the data type , **i** (integer), **s** (short), **f** (float), and **d** (double).

v-vector



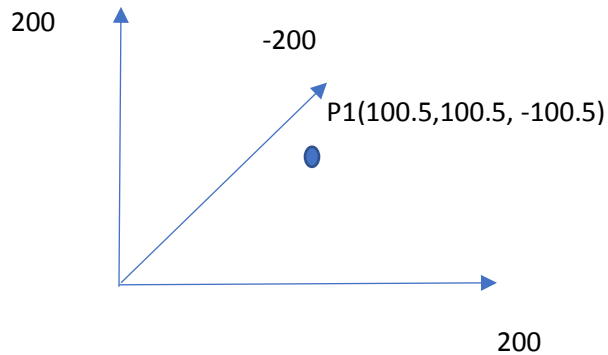
In order to display this point which is 2 dimensional and has integer value , vertex can be defined as

`glVertex2i(100,100)`



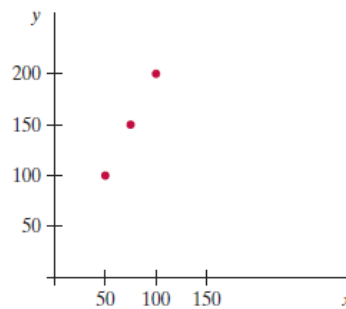
P1 point is 2 dimensional and has floating value value and hence vertex can be defined as

`glVertex2f(100.5,100.5)`



P1 point is 3 dimensional and has floating value value and hence vertex can be defined as
`glVertex3f(100.5,100.5,-100.5)`

To display the following points the code segment will be



```
glBegin (GL_POINTS);
    glVertex2i (50, 100);
    glVertex2i (75, 150);
    glVertex2i (100, 200);

glEnd ( );
```

Alternatively, 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 ( );
```

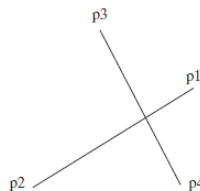
OpenGL LINE FUNCTIONS

Graphics packages typically provide a function for specifying one or more straight-line segments, where each line segment is defined by two endpoint coordinate positions.

A set of straight-line segments between each successive pair of endpoints is generated using the primitive line constant **GL_LINES**. For example

```
glBegin (GL_LINES);
    glVertex2iv (p1);
    glVertex2iv (p2); // first line from p1 to p2
    glVertex2iv (p3);
    glVertex2iv (p4); // second line from p3 to p4
    glVertex2iv (p5);
glEnd ( );
```

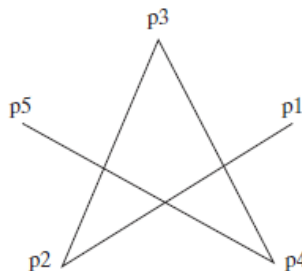
last vertex is ignored because it does not have end vertex to draw.



GL_LINE_STRIP

the display is a sequence of connected line segments between the first endpoint in the list and the last endpoint. It will draw the first line from p1 to p2. Once vertex p3 is mentioned, it will draw from p2 to p3. Similarly rest of the lines are drawn.

```
glBegin (GL_LINE_STRIP);
    glVertex2iv (p1);
    glVertex2iv (p2);
    glVertex2iv (p3);
    glVertex2iv (p4);
    glVertex2iv (p5);
glEnd ( );
```



GL_LINE_LOOP

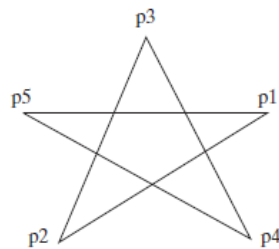
Exactly similar to **GL_LINE_STRIP**, only difference is that the last point is connected to the first point. For the above example p5 gets connected to p1.

```
glBegin (GL_LINE_LOOP);
    glVertex2iv (p1);
```

```

glVertex2iv (p2);
glVertex2iv (p3);
glVertex2iv (p4);
glVertex2iv (p5);
glEnd ( );

```

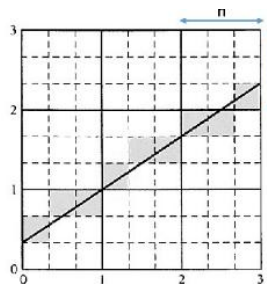


LINE-DRAWING ALGORITHMS

A straight-line segment in a scene is defined by the coordinate positions for the endpoints of the segment. To display the line on a raster monitor, the graphics system must first project the endpoints to integer screen coordinates and determine the nearest pixel positions along the line path between the two endpoints.

Then the line color is loaded into the frame buffer at the corresponding pixel coordinates. Reading from the frame buffer, the video controller plots the screen pixels.

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

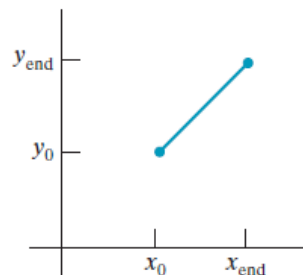


Line Equations

Equation of the line can be written as

$$y = m \cdot x + b \quad \text{--eq1}$$

with m as the slope of the line and b is the constant. Given that the two end points of a line segment are specified at positions (x_0, y_0) and $(x_{\text{end}}, y_{\text{end}})$,



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

$$m = \frac{y_{\text{end}} - y_0}{x_{\text{end}} - x_0}$$

$$b = y_0 - m \cdot x_0$$

-----eq 2 & eq 3

What is DDA ? what is dda algorithm? Give steps involved in dda algorithm. What are advantages and disadvantages of dda algorithm.

DDA Algorithm

The digital differential analyzer (DDA) is a scan-conversion line algorithm based on calculating either Δy or Δx ,

Δx is nothing but small increment that must be added to get the next point

x_k is the first point and to get the next point x_{k+1}

$$x_{k+1} = x_k + \Delta x \quad \text{- eq 1}$$

similarly, y_k is the first point and to get the next point y_{k+1}

$$y_{k+1} = y_k + \Delta y \quad \text{- eq 2}$$

also

$$m = (y_2 - y_1) / (x_2 - x_1)$$

similar to the points (x_k, y_k) and (x_{k+1}, y_{k+1}) the slope of the line can be rewritten as

$$m = (y_{k+1} - y_k) / (x_{k+1} - x_k) \text{ but from eq 1 \& eq 2 } \Delta x = x_{k+1} - x_k \text{ and } \Delta y = y_{k+1} - y_k$$

$$m = \Delta y / \Delta x \quad \text{-eq 3}$$

$$\Delta x = \Delta y / m \quad \text{-eq 4}$$

$$\Delta y = \Delta x * m \quad \text{-eq 5}$$

successive steps can be calculated as

$$y_{k+1} = y_k + \Delta y, \text{ but from equation 5 } \Delta y = \Delta x * m$$

$$y_{k+1} = y_k + \Delta x * m \quad \text{-eq 6}$$

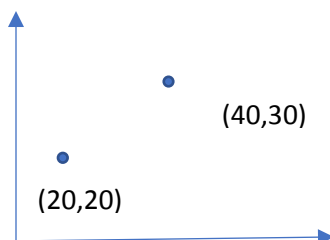
similarly

$$x_{k+1} = x_k + \Delta x, \text{ but from equation 4 } \Delta x = \Delta y / m$$

$$x_{k+1} = x_k + \Delta y / m \quad \text{-eq 7}$$

There can be following three cases to discuss:

Case 1: $m < 1$



In the above example P1(20,20) (x1,y1) and P2(40,30)(x2,y2)

Then the slope $m = (y_2 - y_1) / (x_2 - x_1)$

$$m = (30 - 20) / (40 - 20)$$

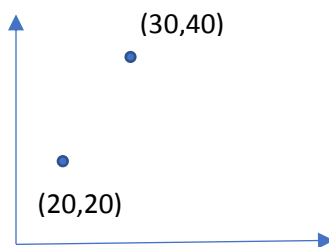
$$m = 10 / 20 = 0.5$$

If $m < 1$, then larger increment in x and hence $\Delta x = 1$, substituting in equation in 1 and 6 we get

$$x_{k+1} = x_k + 1 \quad -$$

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

Case 2: $m > 1$



In the above example P1(20,20) (x1,y1) and P2(30,40)(x2,y2)

Then the slope $m = (y_2 - y_1) / (x_2 - x_1)$

$$m = (40 - 20) / (30 - 20)$$

$$m = 20 / 10 = 2$$

If $m > 1$, then larger increment in y and hence $\Delta y = 1$, substituting in equation in 2 and 7 we get

$$x_{k+1} = x_k + 1/m \quad -$$

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

Case 3: $m = 1$

Then

$$x_{k+1} = x_k + 1 \quad -$$

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

However, for negative value slopes. We follow the same procedure as above. So, only the sampling unit Δx and Δy become -1.

It can be generalized as follows

Case 1: If the slope is less than or equal to 1 ($|m| \leq 1$). Then the coordinate x is sampled at unit intervals ($\Delta x = 1$). While each successive value of y is computed as

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

Case 2: For slope greater than 1 ($|m| > 1$), then y is sampled at unit intervals ($\Delta y = 1$), and x values are calculated as

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

for negative value slopes. We follow the same procedure as above. So, only the sampling unit Δx and Δy become -1.

```
void lineDDA (int x0, int y0, int xEnd, int yEnd)
{
    int dx = xEnd - x0, dy = yEnd - y0, steps, k;
    float xIncrement, yIncrement, x = x0, y = y0;
    if (fabs (dx) > fabs (dy))
        steps = fabs (dx);
    else
        steps = fabs (dy);
    xIncrement = float (dx) / float (steps);
    yIncrement = float (dy) / float (steps);
    setPixel (round (x), round (y));
    for (k = 0; k < steps; k++) {
        x += xIncrement;
        y += yIncrement;
        setPixel (round (x), round (y));
    }
}
```

Draw a line from P(1,1) to Q(5,9) using DDA algorithm

Solution

P(1,1) => $x_1 = 1, y_1 = 1$

Q(5,9) => $x_2 = 5, y_2 = 9$

Now lets calculate the difference:

$dx = x_2 - x_1 = 5 - 1 = 4$

$dy = y_2 - y_1 = 9 - 1 = 8$

if (fabs (dx) > fabs (dy)) $4 > 8$

steps = fabs (dy);

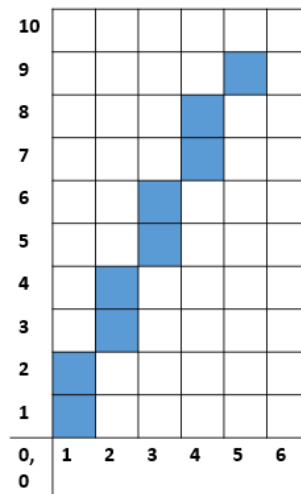
x=1

y=1

xIncrement = float (dx) / float (steps);=4/8=0.5

yIncrement = float (dy) / float (steps);=8/8=1

| Steps | X | Y | X = X + xInc | Y = Y + yInc | Plot(Floor(X), Floor(Y)) |
|---------|--------------|--------------|-----------------|-----------------|---------------------------------|
| Initial | X = x1 =1 | Y = y1 =1 | | | Plot(1,1) |
| 1 | 1 | 1 | 1 + 0.5 = 1.5 | 1 + 1 = 2 | Plot(1,2) |
| 2 | 1.5 | 2 | 1.5 + 0.5 = 2 | 2 + 1 = 3 | Plot(2,3) |
| 3 | 2 | 3 | 2 + 0.5 = 2.5 | 3 + 1 = 4 | Plot(2,4) |
| 4 | 2.5 | 4 | 2.5 + 0.5 = 3 | 4 + 1 = 5 | Plot(3,5) |
| 5 | 3 | 5 | 3 + 0.5 = 3.5 | 5 + 1 = 6 | Plot(3,6) |
| 6 | 3.5 | 6 | 3.5 + 0.5 = 4 | 6 + 1 = 7 | Plot(4,7) |
| 7 | 4 | 7 | 4 + 0.5 = 4.5 | 7 + 1 = 8 | Plot(4,8) |
| 8 | 4.5 | 8 | 4.5 + 0.5 = 5 | 8 + 1 = 9 | Plot(5,9) |



Disadvantages of DDA Line Drawing Algorithm

1. The accumulation of roundoff error in successive additions of the floating-point increment can cause the calculated pixel positions to drift away from the true line path for long line segments further.
2. Moreover, the rounding operations and floating-point arithmetic are time-consuming.

With necessary steps explain Bresenham's Line Algorithm (5 M)

Mention the diagram, Assumptions, there is no need to explain the steps , directly write the decision equations and the algorithm .

DERIVATION OF THE BRESENHAM'S LINE ALGORITHM

Assumptions:

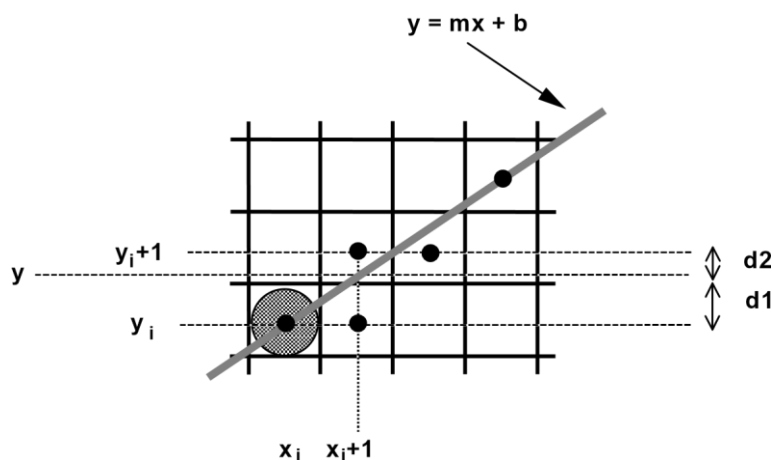
input: line endpoints at $(X1, Y1)$ and $(X2, Y2)$

$X1 < X2$

line slope $0 < m \leq 1$

x coordinate is incremented in steps of 1, y coordinate is computed

generic line equation: $y = mx + b$



derive decision parameter by using Bresenham's method used to generate a straight line segment with slope < 1 .

Derivation

Assume that we already have a location of pixel (x_i, y_i) and have plotted it. The question is, what is the location of the next pixel.

Geometric location of the line at x-coordinate $x_{i+1} = x_i + 1$ is:

$$y = m(x_i + 1) + b \quad (1)$$

where:

$$m = \Delta y / \Delta x \text{ (slope)} \quad (2)$$

b – intercept

$$\Delta x = X2 - X1 \text{ (from the assumption above that } X1 < X2 \text{)} \quad (3)$$

$$\Delta y = Y2 - Y1$$

Define:

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

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

Calculate:

$$\begin{aligned} d1 - d2 &= m(x_i + 1) + b - y_i - y_i - 1 + m(x_i + 1) + b \\ &= 2m(x_i + 1) - 2y_i + 2b - 1 \end{aligned} \quad (4)$$

$$\text{if } d1 - d2 < 0 \text{ then } y_{i+1} = y_i \quad (5)$$

$$\text{if } d1 - d2 > 0 \text{ then } y_{i+1} = y_i + 1 \quad (6)$$

We want integer calculations in the loop, but m is not an integer. Looking at definition of m ($m = \Delta y / \Delta x$) we see that if we multiply m by Δx , we shall remove the denominator and hence the floating point number.

For this purpose, let us multiply the difference ($d1 - d2$) by Δx and call it p_i :

$$p_i = \Delta x (d1 - d2)$$

The sign of p_i is the same as the sign of $d1 - d2$, because of the assumption (3).

Expand p_i :

$$\begin{aligned} p_i &= \Delta x (d1 - d2) \\ &= \Delta x [2m(x_i + 1) - 2y_i + 2b - 1] && \text{from (4)} \\ &= \Delta x [2 \cdot (\Delta y / \Delta x) \cdot (x_i + 1) - 2y_i + 2b - 1] && \text{from (2)} \\ &= 2 \cdot \Delta y \cdot (x_i + 1) - 2 \cdot \Delta x \cdot y_i + 2 \cdot \Delta x \cdot b - \Delta x && \text{result of multiplication by } \Delta x \\ &= 2 \cdot \Delta y \cdot x_i + 2 \cdot \Delta y - 2 \cdot \Delta x \cdot y_i + 2 \cdot \Delta x \cdot b - \Delta x \\ &= 2 \cdot \Delta y \cdot x_i - 2 \cdot \Delta x \cdot y_i + \underline{2 \cdot \Delta y + 2 \cdot \Delta x \cdot b - \Delta x} \end{aligned} \quad (7)$$

Note that the underlined part is constant (it does not change during iteration), we call it c , i.e. $c = 2 \cdot \Delta y + 2 \cdot \Delta x \cdot b - \Delta x$

Hence we can write an expression for p_i as:

$$p_i = 2 \cdot \Delta y \cdot x_i - 2 \cdot \Delta x \cdot y_i + c \quad (8)$$

Because the sign of p_i is the same as the sign of $d1 - d2$, we could use it inside the loop to decide whether to select pixel at $(x_i + 1, y_i)$ or at $(x_i + 1, y_i + 1)$. Note that the loop will only include integer arithmetic. There are now 6 multiplications, two additions and one selection in each turn of the loop.

However, we can do better than this, by defining p_i recursively.

$$p_{i+1} = 2 \cdot \Delta y \cdot x_{i+1} - 2 \cdot \Delta x \cdot y_{i+1} + c \quad \text{from (8)}$$

$$p_{i+1} - p_i = 2 \cdot \Delta y \cdot x_{i+1} - 2 \cdot \Delta x \cdot y_{i+1} + c \\ - (2 \cdot \Delta y \cdot x_i - 2 \cdot \Delta x \cdot y_i + c)$$

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

$$x_{i+1} - x_i = 1 \text{ always}$$

$$p_{i+1} - p_i = 2 \Delta y - 2 \Delta x \cdot (y_{i+1} - y_i)$$

Recursive definition for p_i :

$$p_{i+1} = p_i + 2 \Delta y - 2 \Delta x \cdot (y_{i+1} - y_i)$$

If you now recall the way we construct the line pixel by pixel, you will realise that the underlined expression: $y_{i+1} - y_i$ can be either 0 (when the next pixel is plotted at the same y- coordinate, i.e. $d1 - d2 < 0$ from (5)); or 1 (when the next pixel is plotted at the next y- coordinate, i.e. $d1 - d2 > 0$ from (6)). Therefore the final recursive definition for p_i will be based on choice, as follows (remember that the sign of p_i is the same as the sign of $d1 - d2$): if $p_i < 0$, $p_{i+1} = p_i + 2 \Delta y$ because $2 \Delta x \cdot (y_{i+1} - y_i) = 0$ if $p_i > 0$, $p_{i+1} = p_i + 2 \Delta y - 2 \Delta x$ because $(y_{i+1} - y_i) = 1$

At this stage the basic algorithm is defined. We only need to calculate the initial value for parameter p_0 .

$$p_i = 2 \cdot \Delta y \cdot x_i - 2 \cdot \Delta x \cdot y_i + 2 \cdot \Delta y + 2 \cdot \Delta x \cdot b - \Delta x \quad \text{from (7)}$$

$$p_0 = 2 \cdot \Delta y \cdot x_0 - 2 \cdot \Delta x \cdot y_0 + 2 \cdot \Delta y + 2 \cdot \Delta x \cdot b - \Delta x \quad (9)$$

For the initial point on the line: $y_0 = mx_0 + b$ therefore

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

Substituting the above for b in (9) we get:

$$p_0 = 2 \cdot \Delta y \cdot x_0 - 2 \cdot \Delta x \cdot y_0 + 2 \cdot \Delta y + 2 \Delta x \cdot [y_0 - (\Delta y / \Delta x) \cdot x_0] - \Delta x$$

$$= 2 \cdot \Delta y \cdot x_0 - 2 \cdot \Delta x \cdot y_0 + 2 \cdot \Delta y + 2 \Delta x \cdot y_0 - 2 \Delta x \cdot (\Delta y / \Delta x) \cdot x_0 - \Delta x \quad \text{simplify}$$

$$= 2 \cdot \Delta y \cdot x_0 - 2 \Delta x \cdot y_0 + 2 \cdot \Delta y + 2 \Delta x \cdot y_0 - 2 \Delta y \cdot x_0 - \Delta x \quad \text{regroup}$$

$$= 2 \Delta y \cdot x_0 - 2 \Delta y \cdot x_0 - 2 \Delta x \cdot y_0 + 2 \Delta x \cdot y_0 + 2 \cdot \Delta y - \Delta x \quad \text{simplify}$$

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

Develop Bresenham line-drawing algorithm (5M)

```

/* 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 p = 2 * dy - dx;
    int twoDy = 2 * dy, twoDyMinusDx = 2 * (dy - dx);
    int x, y;
    /* 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);
    while (x < xEnd) {
        x++;
        if (p < 0)
            p += twoDy;
        else {
            y++;
            p += twoDyMinusDx;
        }
        setPixel (x, y);
    }
}

```

With the help of suitable example demonstrate the working principle of Bresenham line-drawing algorithm for different slopes.

Algorithm for $m < 1$ is explained earlier

For slope $m > 1$, dx and dy will be interchanged

Initial decision will be

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

```

while (y < yEnd) {
    y++;
    if (p < 0)
        p += twoDx; // p=p+2 * Δx
    else {
        x++;
        p += twoDxMinusDy; // p=p+2*(dx-dy)
    }
    setPixel (x, y);
}

```

For negative slopes, the procedures are similar, except that now one coordinate decreases as the other increases. Finally, special cases can be handled separately: Horizontal lines ($dy = 0$), vertical lines ($dx = 0$), and diagonal lines ($|dx| = |dy|$) can each be loaded directly into the frame buffer without processing them through the line-plotting algorithm.

Draw a line from P(5,5) to Q(13,9) using Bresenham line-drawing algorithm

$$dx = 13 - 5 = 8$$

$$dy = 9 - 5 = 4$$

$$m = 0.5$$

$$p_0 = 2 * dy - dx = 0$$

$$p = 2 * (dy - dx) = -8$$

$$p = 2dy = 8$$

$$x = 5; y = 5$$

| steps | x | y | p | xincr | yincr | next p | Plot(x,y) |
|---------------------------------|-----|-----|-----------------------|-------|-------|---------------------------|------------|
| Initial | x=5 | y=5 | $p = 2 * dy - dx = 0$ | | | 0 | Plot(5,5) |
| 1 while (x < xEnd) | 5 | 5 | 0 | 6 | 6 | $P = 0 + 2(dy - dx) = -8$ | Plot(6,6) |
| 2 | 6 | 6 | -8 | 7 | 6 | $P = p + 2dy = 0$ | Plot(7,6) |
| 3 | 7 | 6 | 0 | 8 | 7 | $P = 0 + 2(dy - dx) = -8$ | Plot(8,7) |
| 4 | 8 | 7 | -8 | 9 | 7 | $P = p + 2dy = 0$ | Plot(9,7) |
| | 9 | 7 | 0 | 10 | 8 | $P = -8$ | Plot(10,8) |
| 6 | 10 | 8 | -8 | 11 | 8 | $P = 0$ | Plot(11,8) |
| 7 | 11 | 8 | 0 | 12 | 9 | $P = -8$ | Plot(12,9) |
| 8 | 12 | 9 | -8 | 13 | 9 | $P = 0$ | Plot(13,9) |

Draw line from P(5,2) to Q(2,-1)

$$dx = -3$$

$$dy = -3$$

$$m = 1$$

$$x = 2$$

$$x_{end} = 5$$

$$p_0 = 2 * \text{abs}(dx) - \text{abs}(dy) = 6 - 3 = 3$$

$$p = 2 * (\text{abs}(dx) - \text{abs}(dy)) = 0$$

$$p = 2 * \text{abs}(dy) = 6$$

$$x = 2; y = -1$$

| steps | x | y | p | xincr | yincr | next p | Plot(x,y) |
|---------|-----|-------|------------------------|-------|-------|--------|------------|
| Initial | x=2 | y= -1 | $p = 2 * dy - dx = -3$ | | | 3 | Plot(2,-1) |

| | | | | | | | |
|---------------------------------|---|----|---|---|---|-------------------------|-----------|
| 1 while (x < xEnd) | 2 | -1 | 3 | 3 | 0 | $P=3+2(dx-dy)$ $= 3$ | Plot(3,0) |
| 2 | 3 | 0 | 3 | 4 | 1 | $P=3+2(dx-dy)$ $= 3$ | Plot(4,1) |
| 3 | 4 | 1 | 3 | 5 | 2 | $P=3+2(dx-dy)$ $= 3$ | Plot(5,2) |

Displaying Polylines

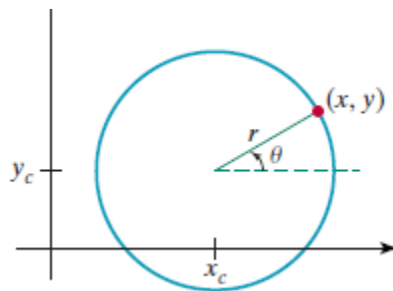
Implementation of a polyline procedure is accomplished by invoking a line drawing routine $n - 1$ times to display the lines connecting the n endpoints.

CIRCLE-GENERATING ALGORITHMS

Properties of Circles

A circle (Fig. 3-16) is defined as the set of points that are all at a given distance r from a center position (x_c, y_c) . For any circle point (x, y) , this distance relationship is expressed by the Pythagorean theorem as

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



We could use this equation to calculate the position of points on a circle circumference by stepping along the x axis in unit steps from $x_c - r$ to $x_c + r$ and calculating the corresponding y values at each position as

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

Disadvantage is that it involves considerable computation at each step and spacing between plotted pixel positions is not uniform.

Circle can also be generated using the equation

$$x = x_c + r \cos(\theta)$$

$$y = y_c + r \sin(\theta)$$

code segment for drawing a circle

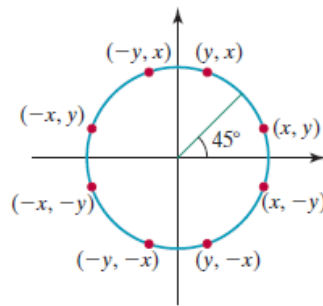
```
void display()
{
    glClearColor(1,1,1,1);
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1,0,0);
    glBegin(GL_LINE_LOOP);
    for(degree=0; degree<=360; degree+=5)
    {
        x=r*cos(degree)+x_c;
```

```

        y=r*sin(degree)+yc;
        glVertex2f(x,y);
    }
    glEnd();
    glFlush();
}

```

this approach is also time consuming. we can reduce computations by considering the symmetry of circle. The shape of the circle is similar in each quadrant. Therefore, if we determine the curve positions in the first quadrant, we can generate rest of the points in other quadrants.



```

void eight_way_sym(float x, float y)
{
    glBegin(GL_POINTS);
        glVertex2f(x, y);
        glVertex2f(y, x);
        glVertex2f(-x, y);
        glVertex2f(-y, x);
        glVertex2f(-x, -y);
        glVertex2f(-y, -x);
        glVertex2f(x, -y);
        glVertex2f(y, -x);
    glEnd();
    glFlush();
}

void display()
{
    glClearColor(1,1,1,1);
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1,0,0);
    for(theta=0;theta<45;theta++)
    {
        x=xc+r*cos(theta);
        y=yc+ r*sin(theta);
        eight_way_sym(x, y);
    }
}

```

With the help of suitable example demonstrate bresenham's circle drawing algorithm

Midpoint Circle Algorithm

To apply the midpoint method, we define a circle function as

$$f_{\text{circ}}(x, y) = x^2 + y^2 - r^2$$

Explain mid point algorithm

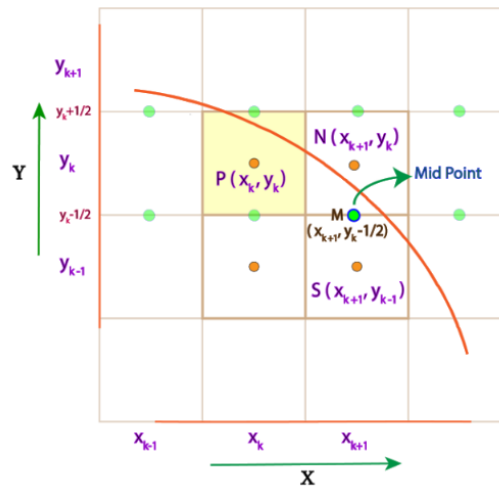
Any point (x, y) on the boundary of the circle with radius r satisfies the equation $f_{\text{circ}}(x, y) = 0$. If the point is in the interior of the circle, the circle function is negative. And if the point is outside the circle, the circle function is positive. To summarize, the relative position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

$$f_{\text{circ}}(x, y)$$

< 0 , if (x, y) is inside the circle boundary

$= 0$, if (x, y) is on the circle boundary

> 0 , if (x, y) is outside the circle boundary



let us assume we have plotted Pixel P whose coordinates are (X_k, Y_k)

Now we need to determine the next pixel.

We have chosen octet 2 where circle is moving forward and downwards so y can never be increased, either it can be same or decremented. Similarly x will always be increasing as circle is moving forward too.

So y is needed to be decided.

Now we need to decide whether we should go with point N or S.

For that decision Mid Point circle drawing technique will us decide our next pixel whether it will be N or S.

As x_{k+1} is the next pixel of x_k therefore we can write,

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

$$\text{And similarly } y_{k-1} = y_k - 1$$

Let M is the midpoint between $N(x_{k+1}, y_k)$ and $S(x_{k+1}, y_{k-1})$

And coordinates of point (M) are

$$M = \frac{(N+S)}{2}$$

$$M = (N(x_{k+1}, y_k) + S(x_{k+1}, y_{k-1}))/2$$

$$M = (x_{k+1} + x_{k+1})/2, (y_k + y_{k-1})/2$$

$$M = (x_k + 1 + x_k + 1)/2, (y_k + y_k - 1)/2$$

$$M = (x_k + 1, y_k - 1/2)$$

Equation of Circle with Radius r

$$(x-h)^2 + (y-k)^2 = r^2$$

When coordinates of centre are at Origin i.e., $(h=0, k=0)$

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

Function of Circle Equation

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

Function of Midpoint $M(x_{k+1}, y_k - 1/2)$ in circle equation

$$F(M) = (x_{k+1})^2 + (y_k - 1/2)^2 - r^2$$

The above equation is the decision parameter p_k

$$p_k = (x_{k+1})^2 + (y_k - 1/2)^2 - r^2 \quad \dots(i)$$

To find out the next decision parameter we need to get p_{k+1}

$$p_{k+1} = (x_{k+1}+1)^2 + (y_{k+1} - 1/2)^2 - r^2$$

Now,

$$\begin{aligned} p_{k+1} - p_k &= (x_{k+1}+1)^2 + (y_{k+1} - 1/2)^2 - r^2 - (x_{k+1})^2 - (y_k - 1/2)^2 + r^2 \\ &= ((x_{k+1}+1)^2 + (y_{k+1} - 1/2)^2 \\ &\quad - (x_{k+1})^2 - (y_k - 1/2)^2 \\ &= (x_{k+1})^2 + 1 + 2(x_{k+1}) + y_{k+1}^2 + (1/4) - y_{k+1} \\ &\quad - (x_{k+1})^2 - y_k^2 - (1/4) + y_{k+1} \\ &= 2(x_{k+1}) + y_{k+1}^2 - y_k^2 - y_{k+1} + y_{k+1} + 1 \\ &= 2(x_{k+1}) + (y_{k+1}^2 - y_k^2) - (y_{k+1} - y_k) + 1 \\ p_{k+1} &= p_k + 2(x_{k+1}) + (y_{k+1}^2 - y_k^2) - (y_{k+1} - y_k) + 1 \quad \dots(ii) \end{aligned}$$

let us conclude the initial decision parameter, For that we have to choose coordinates of starting point i.e. $(0, r)$

if it is inserted in equation (i)

$$p_k = (x_{k+1})^2 + (y_k - 1/2)^2 - r^2$$

$$p_0 = (0+1)^2 + (r - 1/2)^2 - r^2$$

$$p_0 = 1 + r^2 + 1/4 - r - r^2$$

$$p_0 = 1 + 1/4 - r \quad \dots(\text{initial decision parameter})$$

If the radius r is specified as an integer, we can simply round p_0 to

$$p_0 = 1 - r \quad (\text{for } r \text{ an integer})$$

Now If $p_k \geq 0$ that means midpoint is outside the circle and S is closest pixel so we will choose $S(x_{k+1}, y_{k-1})$

That means $y_{k+1} = y_{k-1}$

Putting coordinates of S in (ii) then,

$$\begin{aligned} p_{k+1} &= p_k + 2(x_{k+1}) + (y_{k-1}^2 - y_k^2) - (y_{k-1} - y_k) + 1 \\ &= p_k + 2(x_{k+1}) + (y_k - 1)^2 - y_k^2 - ((y_k - 1) - y_k) + 1 \end{aligned}$$

$$\begin{aligned}
 &= p_k + 2(x_k+1) + y_k^2 + 1 - 2y_k - y_k^2 - y_k + 1 + y_k + 1 \\
 &= p_k + 2(x_k+1) - 2y_k + 2 + 1 \\
 &= p_k + 2(x_k+1) - 2(y_k - 1) + 1
 \end{aligned}$$

As we know $(x_k+1 = x_{k+1})$ and $(y_k-1 = y_{k-1})$

Therefore,

$$P_{k+1} = P_k + 2x_{k+1} - 2y_{k-1} + 1$$

And if $P_k < 0$ that means midpoint is inside the circle and N is closest pixel so we will choose N (x_{k+1}, y_k)

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

Now put coordinates of N in (ii)

$$\begin{aligned}
 P_{k+1} &= P_k + 2(x_k+1) + (y_k^2 - y_k^2) - (y_k - y_k) + 1 \\
 &= P_k + 2(x_k+1) + (y_k^2 - y_k^2) - (y_k - y_k) + 1 \\
 &= P_k + 2(x_k+1) + 1
 \end{aligned}$$

as $x_k+1 = x_{k+1}$, therefore,

$$P_{k+1} = P_k + 2x_{k+1} + 1$$

Midpoint Circle Algorithm

Input radius r and circle center (x_c, y_c) , then set the coordinates for the first point on the circumference of a circle centered on the origin as $(x_0, y_0) = (0, r)$

2. Calculate the initial value of the decision parameter as

$$p_0 = 1 - r$$

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

$$P_{k+1} = P_k + 2x_{k+1} - 2y_{k-1} + 1$$

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

4. Determine symmetry points in the other seven octants.

5. Move each calculated pixel position (x, y) onto the circular path

centered at (x_c, y_c) and plot the coordinate values:

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

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

Given a circle radius $r = 10$, demonstrate the midpoint circle algorithm

The initial value of the decision parameter is

$$p_0 = 1 - r = -9$$

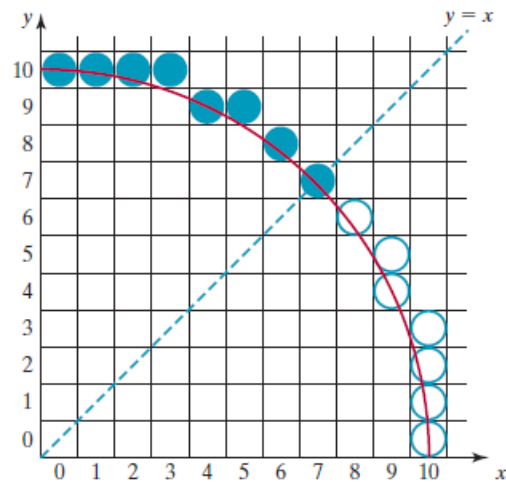
initial increment terms for calculating the decision parameters are

$$2x_0 = 2*0=0, 2y_0 = 2*10=20$$

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

Plot(0,10)

| k | p_k | (x_{k+1}, y_{k+1}) | $2x_{k+1}$ | $2y_{k+1}$ | p_{k+1} |
|---|-------|----------------------|------------|------------|--|
| 0 | -9 | (1,10) | 2 | 20 | $P_k + 2x_{k+1} + 1$ $-9+2+1$ |
| 1 | -6 | (2,10) | 4 | 20 | $-6+4+1$ |
| 2 | -1 | (3,10) | 6 | 20 | $-1+6+1$ |
| 3 | 6 | (4,9) | 8 | 18 | $P_k + 2x_{k+1} - 2y_{k-1} + 1$ $6+8-18+1$ |
| 4 | -3 | (5,9) | 10 | 18 | $P_k + 2x_{k+1} + 1$ $-3+10+1$ |
| 5 | 8 | (6,8) | 12 | 16 | $P_k + 2x_{k+1} - 2y_{k-1} + 1$ $8+12-16+1$ |
| 6 | 5 | (7,7) | 14 | 14 | $5+14+14+1$ |



Program segment of mid point circle drawing algorithm

```

void drawcircle(int xc, int yc, int radius)
{
    int x = radius;
    int y = 0;
    int p = 1-radius;
    eight_way_sym(x+xc,y+yc)

    while (x >= y)
    {
        x-=1;
        if (p < 0)
            p += 2*x + 1;

        else
        {
            y -= 1;
            p+= 2*x - 2*y + 1; // Pk+ 2x_{k+1} - 2y_{k-1} + 1
        }
    }
}

```

```

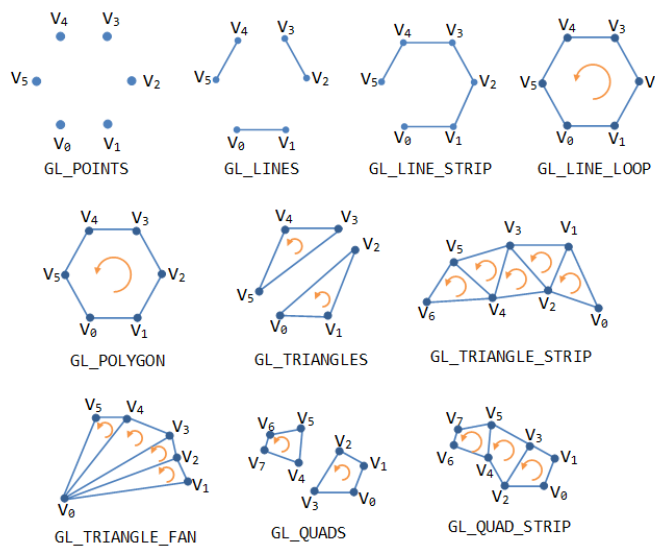
    eight_way_sym(x+xc,y+yc);
}
}

void display()
{
    glClearColor(1,1,1,1);
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1,0,0);
    drawcircle(5,5,10);
}

```

List and explain various open GL primitives and its attribute function (examples of few must be mentioned, detailed explanation must be taken from the above contents).

Primitive functions include



which are mentioned in between glBegin() and glEnd();

**6.Explain following functions
---->refer last question in qb 1 picture**

Attribute functions

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)

glColor*

glClearColor(1.0, 1.0, 1.0, 0.0);

point and line attributes

glPointSize (size); // A point size of 1.0 (the default value) displays a single pixel, and a point size of 2.0 displays a 2

by 2 pixel array. **2. Given a circle radius r(any value), at origin, using midpoint circle algorithm**

glLineWidth (width); **determine pixel position in first quadrant and plot the graph.**

3. what is interlaced display?