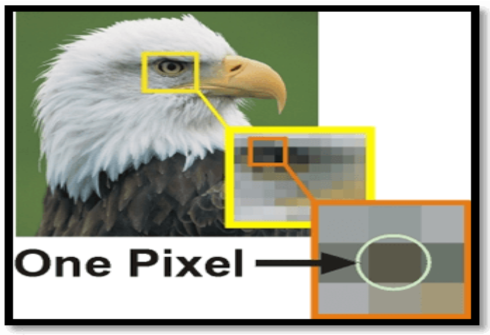
MODULE 1

1.1Basics of Computer Graphics:

- Computer graphics is an art of drawing pictures, lines, charts, etc. using computers with the help of programming.
- Computer graphics objects are presented as a collection of discrete pictures of elements (Pixels)

DOT = **PIXEL** = **IMAGE PIXEL**



- Pixel is the smallest addressable graphical unit represented on the computer screen / smallest screen element.
- It is the creation and manipulation of graphic images by means of a computer (edit using hardware devices).

Advantages of CG:

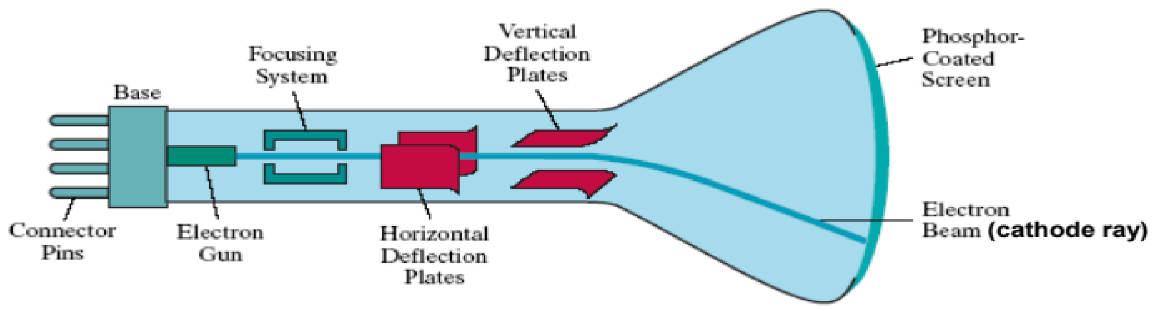
- High quality graphics displays on personal computers.
- Provides the tools to produce the picture.
- Produce animations using 2D, 3D transformation.
- Using motion dynamics tools, user can make objects stationary and the viewer moving around them.
- Using update dynamic tool, it is possible to change the shape, color or other properties of object.
- Large amount of data is rapidly converted into bar charts, pie charts, and graphs.

1.2 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.


Base
Focusing System
Vertical Deflection Plates
Phosphor-Coated Screen
Connector Pins
Electron Gun
Horizontal Deflection Plates
Electron Beam **(cathode ray)**

✓ 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.

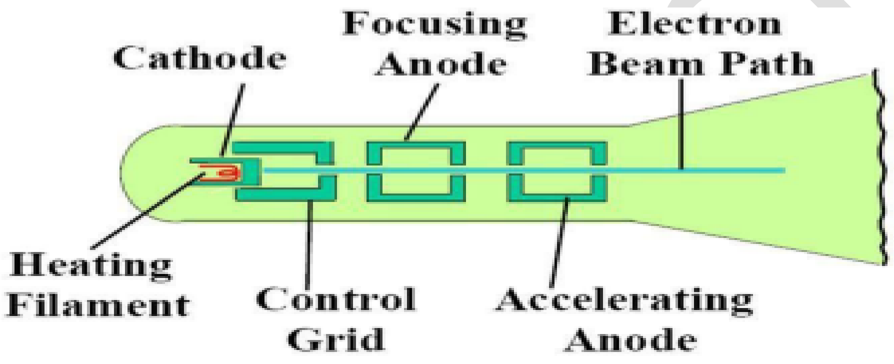
✓ The most common method now employed for maintaining phosphor glow is to redraw the picture repeatedly by quickly directing the electron beam back over the same screen points. This type of display is called a refresh CRT.

✓ The frequency at which a picture is redrawn on the screen is referred to as the refresh rate.

1. **Technology**: CRT monitors work by using an electron gun to shoot a beam of electrons onto a phosphorescent screen. The beam scans across the screen horizontally and vertically, illuminating pixels and creating images. The intensity of the electron beam determines the brightness of each pixel.
2. **Resolution**: CRT monitors were available in various resolutions, ranging from standard definition (SD) to high definition (HD), although their maximum resolutions were generally lower compared to modern displays.
3. **Color Depth**: CRT monitors supported different color depths, typically ranging from 8-bit to 24-bit color. Higher color depths allowed for more accurate color representation and smoother gradients.
4. **Refresh Rate**: One notable advantage of CRT monitors was their ability to achieve high refresh rates, often exceeding 100 Hz. Higher refresh rates reduced flicker and provided smoother motion, making CRT monitors popular for gaming and animation.

5. **Aspect Ratio**: CRT monitors typically had a 4:3 aspect ratio, although widescreen CRTs also existed. This aspect ratio was standard for many years until widescreen displays became more prevalent in the LCD era.
6. **Gamma Correction**: CRT monitors required gamma correction to accurately represent images. Gamma correction ensured that the luminance response of the monitor matched the input signal, resulting in accurate color reproduction.
7. **Screen Size and Weight**: CRT monitors were available in various screen sizes, ranging from small desktop monitors to large, bulky CRT TVs. However, larger CRT monitors were heavy and cumbersome, making them less portable compared to modern flat-panel displays.
8. **Phosphor Persistence**: One drawback of CRT monitors was phosphor persistence, which caused images to linger on the screen briefly after the electron beam moved away. This persistence could lead to motion blur, especially in fast-paced gaming or animation

Operation of an electron gun with an accelerating anode



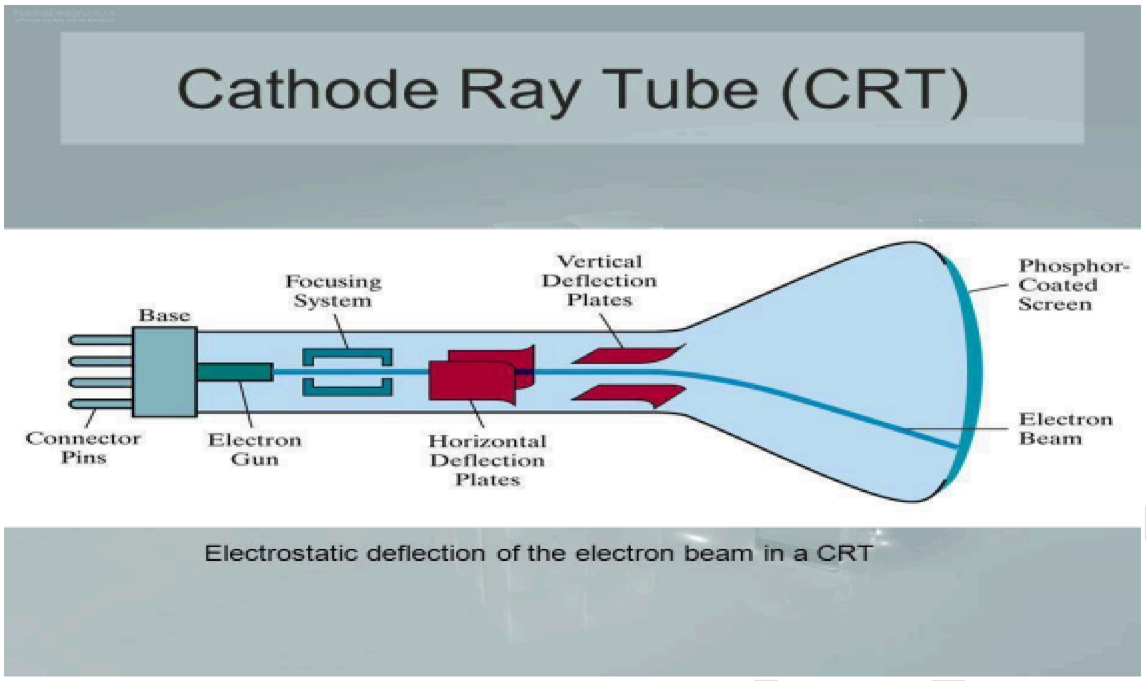
- ✔ The primary components of an electron gun in a CRT are the heated metal cathode and a control grid.
- ✔ The heat is supplied to the cathode by directing a current through a coil of wire, called the filament, inside the cylindrical cathode structure. This causes electrons to be “boiled off” the hot cathode surface.
- ✔ Inside the CRT envelope, the free, negatively charged electrons are then accelerated toward the phosphor coating by a high positive voltage.
- ✔ Intensity of the electron beam is controlled by the voltage at the control grid.
- ✔ Since the amount of light emitted by the phosphor coating depends on the number of electrons striking the screen, the brightness of a display point is controlled by varying the voltage on the control grid.
- ✔ The focusing system in a CRT forces the electron beam to converge to a small cross

section as it strikes the phosphor and it is accomplished with either electric or magnetic fields.

- ✔ With electrostatic focusing, the electron beam is passed through a positively charged metal cylinder so that electrons along the center line of the cylinder are in equilibrium position.
- ✔ Deflection of the electron beam can be controlled with either electric or magnetic fields.
- ✔ Cathode-ray tubes are commonly constructed with two pairs of magnetic-deflection coils
- ✔ One pair is mounted on the top and bottom of the CRT neck, and the other pair is mounted on opposite sides of the neck.
- ✔ The magnetic field produced by each pair of coils results in a traverse deflection force that is perpendicular to both the direction of the magnetic field and the direction of travel of the electron beam.
- ✔ Horizontal and vertical deflections are accomplished with these pair of coils.

Electrostatic deflection of the electron beam in a CRT

- ✔ When electrostatic deflection is used, two pairs of parallel plates are mounted inside the CRT envelope where, one pair of plates is mounted horizontally to control vertical deflection, and the other pair is mounted vertically to control horizontal deflection.
- ✔ Spots of light are produced on the screen by the transfer of the CRT beam energy to the phosphor.
- ✔ When the electrons in the beam collide with the phosphor coating, they are stopped and their kinetic energy is absorbed by the phosphor.
- ✔ Part of the beam energy is converted by the friction in to the heat energy, and the remainder causes electros in the phosphor atoms to move up to higher quantum-energy levels.
- ✔ After a short time, the “excited” phosphor electrons begin dropping back to their stable ground state, giving up their extra energy as small quantum of light energy called photons.


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.

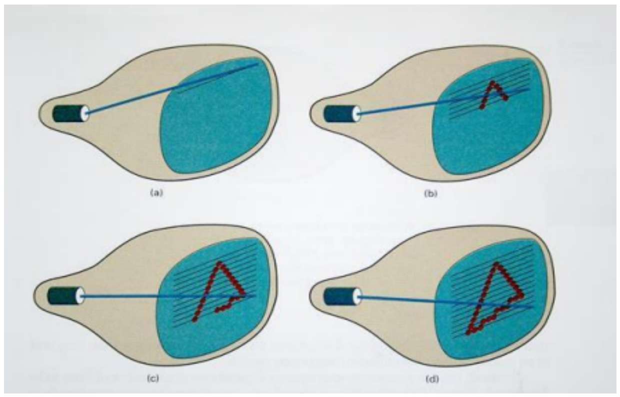
✓ Resolution of a CRT is dependent on the type of phosphor, the intensity to be displayed, and the focusing and deflection systems. High-resolution systems are often referred to as high-definition systems.

1.2.1 Raster-Scan Displays and Random Scan Displays

i)Raster-Scan Displays

- The electron beam is swept across the screen one row at a time from top to bottom.
- As it moves across each row, the beam intensity is turned on and off to create a pattern of illuminated spots.

- This scanning process is called refreshing. Each complete scanning of a screen is normally called a frame.
- The refreshing rate, called the frame rate, is normally 60 to 80 frames per second, or described as 60 Hz to 80 Hz.
- Grid of Pixels: The display screen is divided into a grid of tiny squares called pixels (short for "picture elements"). Each pixel can emit light independently, allowing for the creation of images.
- Scan Pattern: The raster scan pattern starts from the top-left corner of the screen and moves horizontally across each row, illuminating pixels one by one. Once it reaches the end of a row, it moves to the next row below, repeating the process until the entire screen is covered.
- Refresh Rate: The speed at which the scan pattern moves across the screen determines the refresh rate of the display. A higher refresh rate results in smoother motion and reduced flicker.
- Color and Intensity: Each pixel's color and intensity are controlled by the graphics hardware and software. In color displays, each pixel typically consists of multiple color components (e.g., red, green, and blue) that can be mixed to produce a wide range of colors.
- Frame Buffer: The image to be displayed is stored in a memory area called the frame buffer. As the raster scan progresses, the pixel values from the frame buffer are read and sent to the display hardware to illuminate the corresponding pixels on the screen.
- Picture definition is stored in a memory area called the frame buffer.
- This frame buffer stores the intensity values for all the screen points. Each screen point is called a pixel (picture element).
- Property of raster scan is Aspect ratio, which defined as number of pixel columns divided by number of scan lines that can be displayed by the system.



Case 1: In case of black and white systems

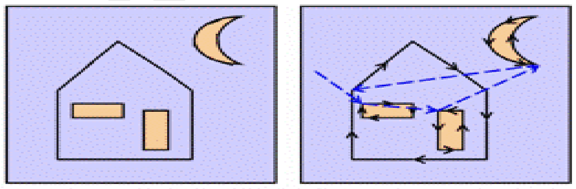
- On black and white systems, the frame buffer storing the values of the pixels is called a bitmap. Each entry in the bitmap is a 1-bit data which determine the on (1) and off (0) of the intensity of the pixel.

Case 2: In case of color systems

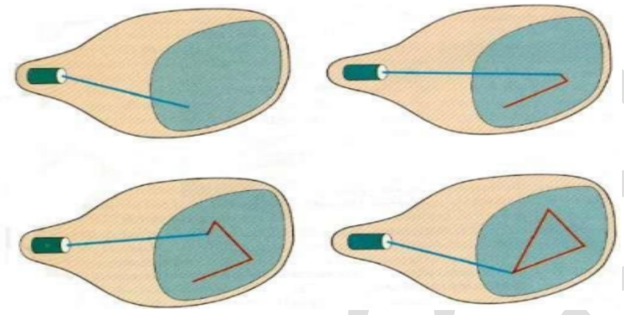
- On color systems, the frame buffer storing the values of the pixels is called a pixmap (Though now a days many graphics libraries name it as bitmap too).
- Each entry in the pixmap occupies a number of bits to represent the color of the pixel. For a true color display, the number of bits for each entry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

ii). Random-Scan Displays

- When operated as a random-scan display unit, a CRT has the electron beam directed only to those parts of the screen where a picture is to be displayed.
- Pictures are generated as line drawings, with the electron beam tracing out the component lines one after the other.



- For this reason, random-scan monitors are also referred to as vector displays (or stroke writing displays or calligraphic displays).
- The component lines of a picture can be drawn and refreshed by a random-scan system in any specified order.



- A pen plotter operates in a similar way and is an example of a random-scan, hard-copy device.
- 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 displays are designed to draw all the component lines of a picture 30 to 60 times each second, with up to 100,000 "short" lines in the display list.
- When a small set of lines is to be displayed, each refresh cycle is delayed to avoid very high refresh rates, which could burn out the phosphor.

Difference between Raster scan system and Random scan system

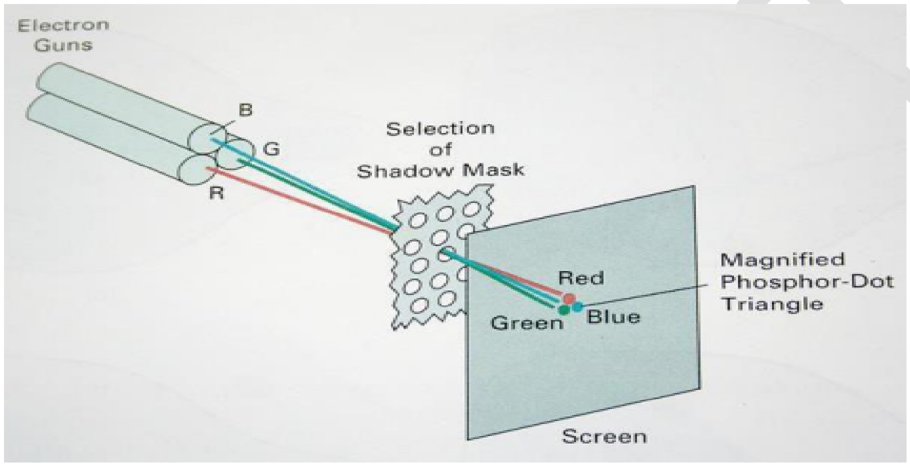
Base of Difference	**Raster Scan System**	**Random Scan System**
Electron Beam	The electron beam is swept across the screen, one row at a time, from top to bottom.	The electron beam is directed only on the parts of screen where a picture is to be drawn.
Resolution	Its resolution is poor because raster 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 contain shadow and color pattern.	These systems are designed for linedrawing and can't display realistic shaded scenes.
Draw an Image	Screen points/pixels are used to draw an image	Mathematical functions are used to draw an image

1.2.2 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 techniques

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 in to the phosphor layer.



- The beam of fast electron penetrates more and excites inner green layer while slow electron 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.

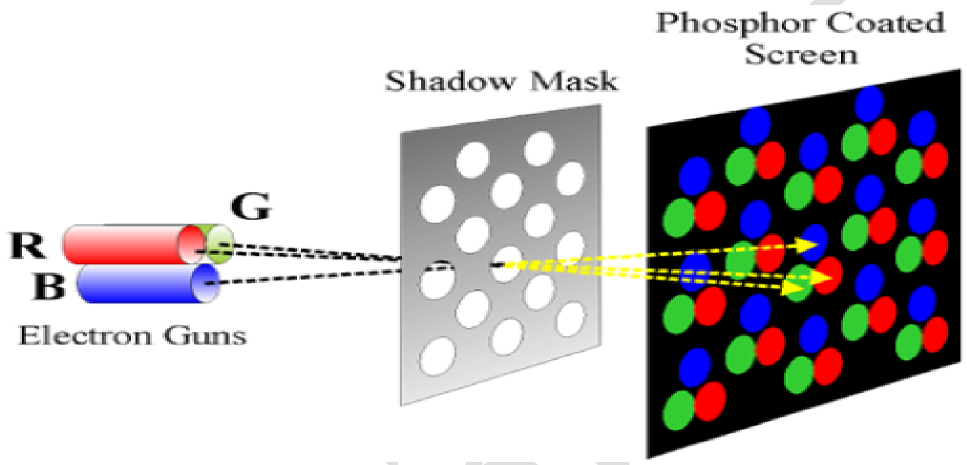
Disadvantages:

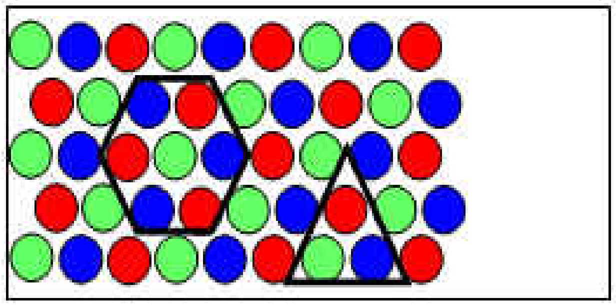
- It is a low cost technique to produce color in random scan monitors.
- It can display only four colors.
- Quality of picture is not good compared to other techniques

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.




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.
- By changing the intensity of the three electron beams we can obtain different colors in the shadow mask CRT.

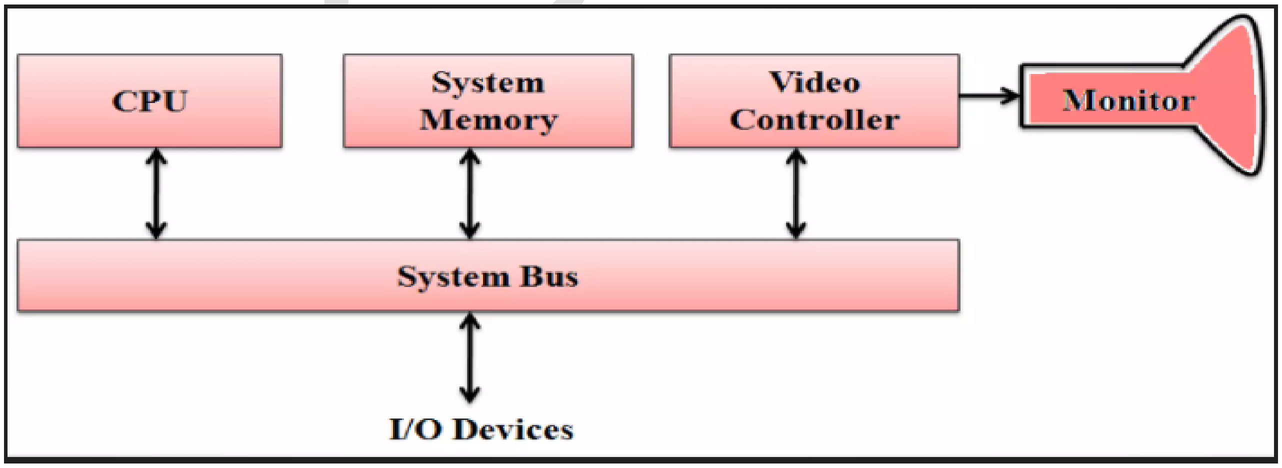
1.2.3 Flat Panel Display

- 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
- Since we can even write on some flat panel displays they will soon be available as pocket notepads.
- We can separate flat panel display in two categories:
  1. 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.
  2. 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).

Three- Dimensional Viewing Devices ↓ Graphics monitors for the display of three-dimensional scenes have been devised using a technique that reflects a CRT image from a vibrating, flexible mirror As the varifocal mirror vibrates, it changes focal length.

Organization of a simple Raster system

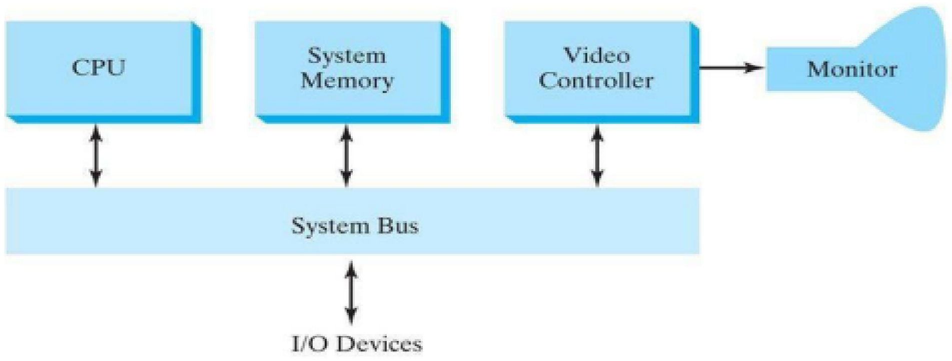
1.



The basic architecture of a computer system with a focus on the components involved in displaying graphics on a monitor

1.3 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, is called to control the operation of the display device.



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

- Here, the frame buffer can be anywhere in the system memory, and the video controller accesses the frame buffer to refresh the screen
- In addition to the video controller, raster systems employ other processors as coprocessors and accelerators to implement various graphics operations.
- **CPU (Central Processing Unit)**: The central processing unit is the primary component responsible for executing instructions and performing calculations in the computer system. It processes data and controls the overall operation of the system.
- **System Memory**: System memory, also known as RAM (Random Access Memory), is used to temporarily store data and instructions that the CPU needs to access quickly. It holds the operating system, application programs, and data being processed.
- **System Bus**: The system bus is a communication pathway that connects the CPU, system memory, and other components of the computer system. It allows for the transfer of data and instructions between different parts of the system.
- **Video Controller**: The video controller, also known as the graphics card or GPU (Graphics Processing Unit), is responsible for generating and controlling the visual output displayed on the monitor. It processes graphics data, performs calculations, and sends signals to the monitor to create images.

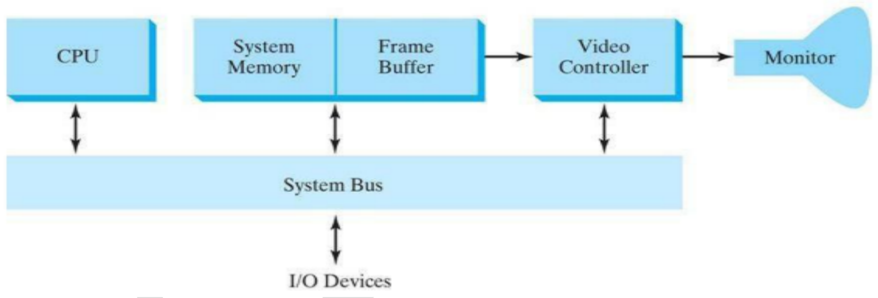
- **I/O Devices**: Input/output (I/O) devices include peripherals such as keyboards, mice, printers, and storage devices. They allow users to input data into the computer system and receive output from it.
- **Monitor**: The monitor is the output device that displays visual information generated by the computer system. It receives signals from the video controller and uses them to create images on the screen for users to view. Overall, this diagram provides a simplified overview of how the CPU, system memory, video controller, and other components work together to produce graphics output on a monitor in a computer system.

1.4.1 Video controller:

✔ The figure below shows a commonly used organization for raster systems.

✔ A fixed area of the system memory is reserved for the frame buffer, and the video controller is given direct access to the frame-buffer memory.

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

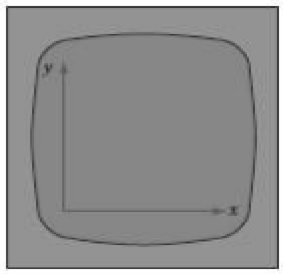


Cartesian reference frame:

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

Working:

- Figure shows a two-dimensional Cartesian reference frame with the origin at the lower left screen corner.
- The screen surface is then represented as the first quadrant of a two-dimensional system with positive x 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.

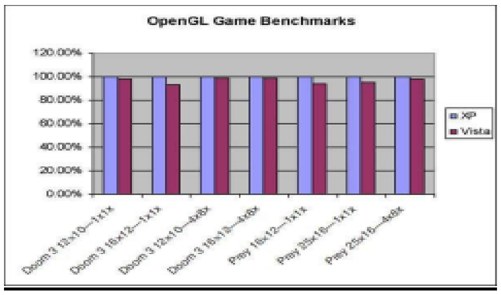


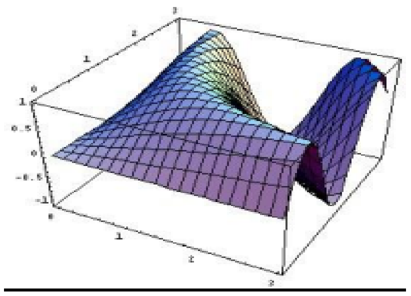
Advantages of video controller:

- 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.

1.2 Applications of Computer Graphics:

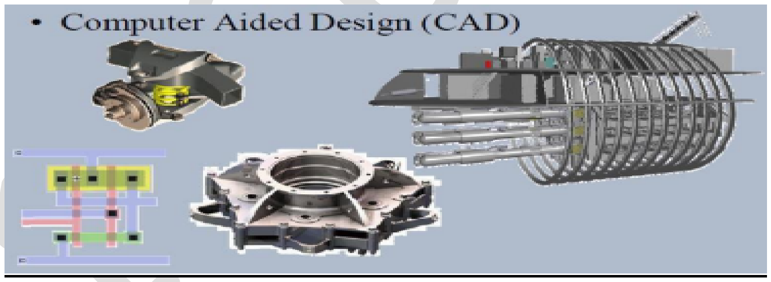
a. Graphs and Charts





- 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 applications.
- 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

b. Computer-Aided Design



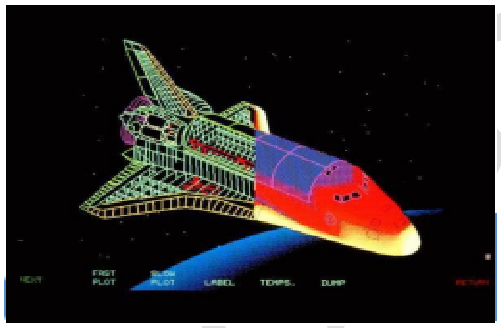
- A major use of computer graphics is in design processes-particularly for engineering and architectural systems.
- CAD, computer-aided design or CADD, computer-aided drafting and design methods are now routinely used in the automobiles, aircraft, spacecraft, computers, home appliances.
- Circuits and 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 wireframe shapes are useful for quickly testing the performance of a vehicle or system.

c. Virtual Reality Environment

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

- With virtual-reality systems, designers and others can move about and interact with objects in various ways. Architectural designs can be examined by taking simulated "walk" through the rooms or around the outsides of buildings to better appreciate the overall effect of a particular design.
- With a special glove, we can even "grasp" objects in a scene and turn them over or move them from one place to another.

d. Data Visualizations

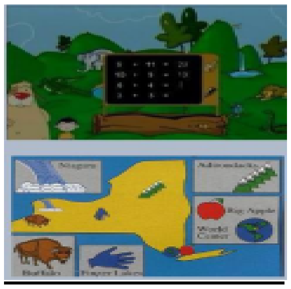


- Producing graphical representations for scientific, engineering and medical data sets and processes is another fairly new application of computer graphics, which is generally referred to as scientific visualization. And the term business visualization is used in connection with data sets related to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

higher-order tensors.

e. Education and Training

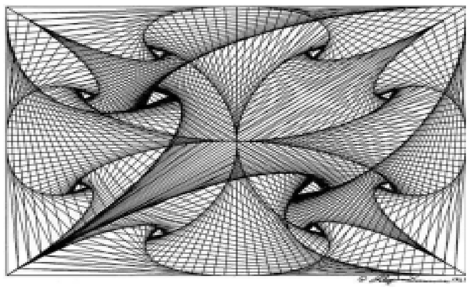
- Computer generated models of physical, financial, political, social, economic & other systems are often used as educational aids.



- Models of physical processes physiological functions, equipment, such as the color-coded diagram as shown in the figure, can help trainees to understand the operation of a system.
- For some training applications, special hardware systems are designed. Examples of such specialized systems are the simulators for practice sessions , aircraft pilots, air traffic-control personnel.
- Some simulators have no video screens, for eg: flight simulator with only a control panel for instrument flying.

f. 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.
- Fine artists use a variety of other computer technologies to produce images. To create pictures the artist uses a combination of 3D modeling packages, texture mapping, drawing programs and CAD software etc.

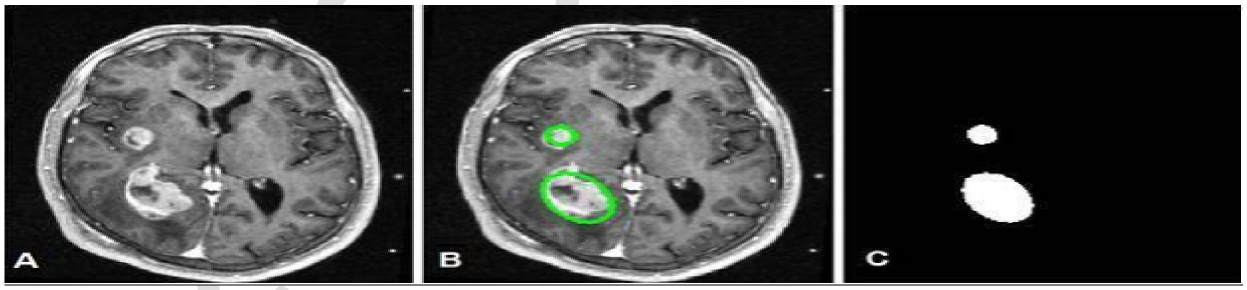


- Commercial art also uses theses "painting" techniques for generating logos & other designs, page layouts combining text & graphics, TV advertising spots & other applications.
- A common graphics method employed in many television commercials is morphing, where one object is transformed into another.

g. Entertainment

- Television production, motion pictures, and music videos routinely a computer graphics method.
- Sometimes graphics images are combined a live actors and scenes and sometimes the films are completely generated a computer rendering and animation techniques.
- Some television programs also use animation techniques to combine computer generated figures of people, animals, or cartoon characters with the actor in a scene or to transform an actor's face into another shape.

h. Image Processing

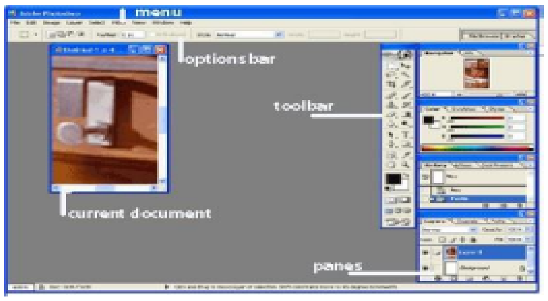


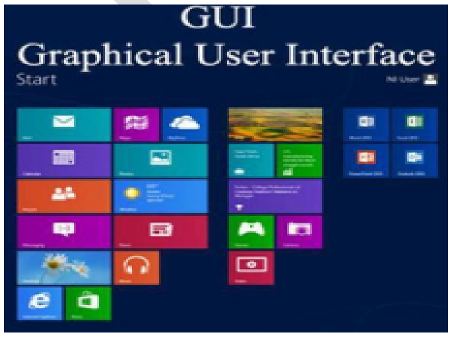
- The modification or interpretation of existing pictures, such as photographs and TV scans is called image processing.
- Methods used in computer graphics and image processing overlap, the two areas are concerned with fundamentally different operations.
- Image processing methods are used to improve picture quality, analyze images, or recognize visual patterns for robotics applications.

- Image processing methods are often used in computer graphics, and computer graphics methods are frequently applied in image processing.
- Medical applications also make extensive use of image processing techniques for picture enhancements in tomography and in simulations and surgical operations.
- It is also used in computed X-ray tomography (CT), position emission tomography(PET),and computed axial tomography(CAT).

i. Graphical User Interfaces

- It is common now for applications software to provide graphical user interface (GUI).
- A major component of graphical interface is a window manager that allows a user to display multiple, rectangular screen areas called display windows.





- Each screen display area can contain a different process, showing graphical or 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 on some systems by positioning the screen cursor within the window display area and pressing the left mouse button.

2 Introduction To OpenGL

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

Basic OpenGL Syntax

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

- For eg:- **glBegin, glClear, glCopyPixels, glPolygonMode**
- Symbolic constants that are used with certain functions as parameters are all in capital letters, preceded by "GL", and component are separated by underscore.
- Eg: -GL_2D, GL_RGB, GL_CCW,GL_POLYGON,GL_AMBIENT_AND_DIFFUSE.
- The OpenGL functions also expect specific data types. For example, an OpenGL function parameter might expect a value that is specified as a 32-bit integer. But the size of an integer specification can be different on different machines.
- To indicate a specific data type, OpenGL uses special built-in, data-type names, such as **GLbyte, GLshort, GLint, GLfloat, GLdouble, Glboolean**

Related Libraries

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

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

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

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

  Eg:- Apple GL

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

Header Files

- In all graphics programs, we will need to include the header file for the OpenGL core library.
- In windows to include OpenGL core libraries and GLU we can use the following header files:- #include<window.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
- In Apple OS X systems, the header file inclusion statement will be **#include<GLUT/glut.h>**

Display-Window Management Using GLUT

- We can consider a simplified example, minimal number of operations for displaying a picture.

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 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 line 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, are now activated:

glutMainLoop ( );

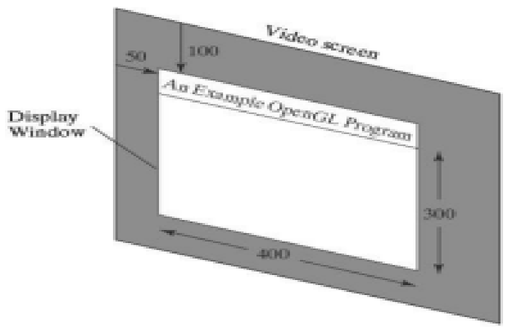
- This function must be the last one in our program. It displays the initial graphics and puts the program into an infinite loop that checks for input from devices such as a mouse or keyboard

Step 5: these parameters using additional GLUT functions

- Although the display window that we created will be in some default location and size, we can set these parameters using additional GLUT functions.

GLUT Function 1:

- We use the glutInit Window Position function to give an initial location for the upper 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



GLUT Function 2:

- After the display window is on the screen, we can reposition and resize it.

GLUT Function 3:

- 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.
- Arguments for this routine are assigned symbolic GLUT constants.
- Example: the following command specifies that a single refresh buffer is to be used for the display window and that we want to use the color mode which uses red, green, and blue (RGB) components to select color values:
  - glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
- The values of the constants passed to this function are combined using a logical or operation.
- Actually, single buffering and RGB color mode are the default options.
- But we will use the function now as a reminder that these are the options that are set for our display.
- Later, we discuss color modes in more detail, as well as other display options, such as double buffering for animation applications and selecting parameters for viewing three dimensional scenes.

A Complete OpenGL Program

- There are still a few more tasks to perform before we have all the parts that we need for a complete program.

Step 1: to set background color

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

- Using RGB color values, we set the background color for the display window to be white, with the OpenGL function: glClearColor (1.0, 1.0, 1.0, 0.0);
- The first three arguments in this function set the red, green, blue component colors to the values 1.0, giving us a white background color for the display window.
- If, instead of 1.0, we set each of the component colors to 0.0, we would get a black background.
- The fourth parameter in the glClearColor function is called the alpha value for the specified color. 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.
- An alpha value of 0.0 indicates a totally transparent object, and an alpha value of 1.0 indicates an opaque object.
- For now, we will simply set alpha to 0.0.
- Although the glClearColor command assigns a color to the display window, it does not put the display window on the screen.

Step 2: to set window color

- 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. (OpenGL has several different kinds of buffers that can be manipulated.

Step 3: to set color to object

- In addition to setting the background color for the display window, we can choose a variety of color schemes for the objects we want to display in a scene.
- For our initial programming example, we will simply set the object color to be a dark green glColor3f (0.0, 0.4, 0.2);
- The suffix 3f on the glColor function indicates that we are specifying the three RGB color components using floating-point (f) values.

- This function requires that the values be in the range from 0.0 to 1.0, and we have set red =0.0,green=0.4, and blue= 0.2

2.1 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:

- Locations on a video monitor are referenced in integer screen coordinates, 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

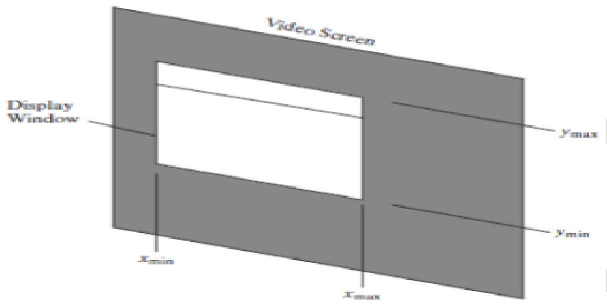
2.2 Specifying a Two-Dimensional World-Coordinate Reference Frame in OpenGL:

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

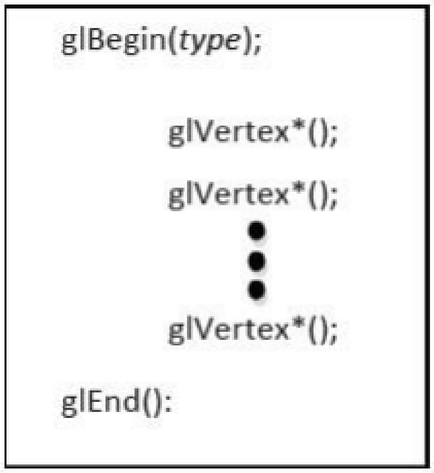


- We can then designate one or more graphics primitives for display using the coordinate reference specified in the gluOrtho2D statement.
- If the coordinate extents of a primitive are within the coordinate range of the display window, all of the primitive will be displayed.
- Otherwise, only those parts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

2.2.1 OpenGL Functions Geometric Primitives:

- It includes points, line segments, polygon etc.
- These primitives pass through geometric pipeline which decides whether the primitive is visible or not and also how the primitive should be visible on the screen etc.
- The geometric transformations such rotation, scaling etc can be applied on the primitives which are displayed on the screen.
- The programmer can create geometric primitives as shown below:
- where: glBegin indicates the beginning of the object that has to be displayed glEnd

indicates the end of primitive



```
glBegin(type);

        glVertex*();
        glVertex*();
            •
            •
            •
        glVertex*();

glEnd():
```

2.3 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 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:	Case 2:	Case 3:

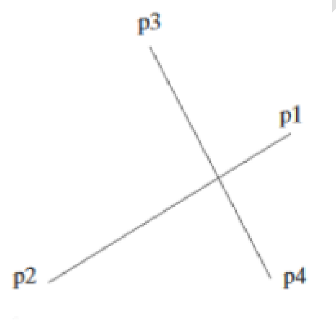
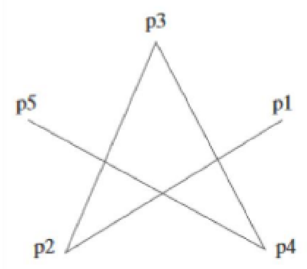
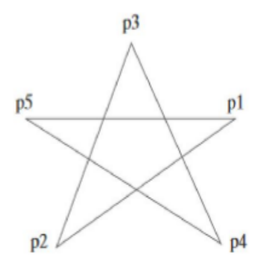
glBegin (GL_POINTS);
glVertex2i (50, 100);
glVertex2i (75, 150);
glVertex2i (100, 200);
glEnd ( );
	glBegin (GL_POINTS);
	glVertex2iv (point1);
	glVertex2iv (point2);
	glVertex2iv (point3);
	glEnd ( );
		glBegin (GL_POINTS);
		glVertex3f(-78.05, 909.72, 14.60);
		glVertex3f(261.91,-5200.67, 188.33);
		glEnd ( );

2.4 OpenGL LINE FUNCTIONS

- Primitive type is GL_LINES
- Successive pairs of vertices are considered as endpoints and they are connected to form an individual line segments.
- Note that successive segments usually are disconnected because the vertices are processed on a pair-wise basis.
- We obtain one line segment between the first and second coordinate positions and another line segment between the third and fourth positions.
- if the number of specified endpoints is odd, so the last coordinate position is ignored.

Case 1: : Lines	Case2:GL_LINE_STRIP: Successive vertices are connected using line segments. However, the final vertex is not connected to the initial vertex.	Case3: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);
glVertex2iv (p1);
glVertex2iv (p2);
glVertex2iv (p3);
glVertex2iv(p4);
glVertex2iv (p5);
glEnd ( );
	glBegin(GL_LINES_STRIP);
	glVertex2iv (p1);
	glVertex2iv (p2);
	glVertex2iv(p3);
	glVertex2iv(p4);
	glVertex2iv (p5);
	glEnd ( );
		glBegin(GL_LINES_LOOP);
		glVertex2iv(p1);
		glVertex2iv(p2);
		glVertex2iv(p3);
		glVertex2iv(p4);
		glVertex2iv (p5);
		glEnd ( );

1.4 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.
- For a raster system: Point size is an integer multiple of the pixel size, so that a large point is displayed as a square block of pixels

Opengl Point-Attribute Functions

Color:

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

Size:

- ❖ 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 antaliased).
- ❖ The number of horizontal and vertical pixels in the display of the point is determined by parameter size.
- ❖ Thus, a point size of 1.0 displays a single pixel, and a point size of 2.0 displays a 2×2 pixel array.
- ❖ If we activate the antialiasing features of OpenGL, the size of a displayed block of pixels will be modified to smooth the edges.
- ❖ The default value for point size is 1.0.

Example program:

- Attribute functions may be listed inside or outside of a glBegin/glEnd pair.
- Example: the following code segment plots three points in varying colors and sizes.
- The first is a standard-size red point, the second is a double-size green point, and the third is a triple-size blue point:

  Ex: glColor3f (1.0, 0.0, 0.0);

glBegin (GL_POINTS);

```
glVertex2i (50, 100);
glPointSize (2.0);
glColor3f (0.0, 1.0, 0.0);
glVertex2i (75, 150);
glPointSize (3.0);
glColor3f (0.0, 0.0, 1.0);
glVertex2i (100, 200);
glEnd ( );
```

1.5 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.
- That is, the magnitude of the horizontal and vertical separations of the line endpoints, deltax and deltay, are compared to determine whether to generate a thick line using vertical pixel spans or horizontal pixel spans.

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 dots.

- ❖ 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);

Pattern:

- Parameter pattern is used to reference a 16-bit integer that describes how the line should be displayed.
- 1 bit in the pattern denotes an "on" pixel position, and a 0 bit indicates an "off" pixel position.
- The pattern is applied to the pixels along the line path starting with the low-order bits in the pattern.
- The default pattern is 0xFFFF (each bit position has a value of 1),which produces a solid line.

repeatFactor

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

Polyline:

- With a polyline, a specified line-style pattern is not restarted at the beginning of each segment.

➔ It is applied continuously across all the segments, starting at the first endpoint of the polyline and ending at the final endpoint for the last segment in the series.

Example:

➔ For line style, suppose parameter pattern is assigned the hexadecimal representation 0x00FF and the repeat factor is 1.

➔ This would display a dashed line with eight pixels in each dash and eight pixel positions that are "off" (an eight-pixel space) between two dashes.

Activating line style:

➢ Before a line can be displayed in the current line-style pattern, we must activate the line-style 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);

➢ This replaces the current line-style pattern with the default pattern (solid lines).

Example Code:

```
typedef struct { float x, y; } wcPt2D;
wcPt2D dataPts [5];
void linePlot (wcPt2D dataPts [5])
{
int k;
glBegin (GL_LINE_STRIP);
for (k = 0; k < 5; k++)
glVertex2f (dataPts [k].x, dataPts [k].y);
glFlush ( );
glEnd ( );
}
```

Curve Attributes

➔ Parameters for curve attributes are the same as those for straight-line segments.

➔ We can display curves with varying colors, widths, dot-dash patterns, and available pen or brush options.

➔ Methods for adapting curve-drawing algorithms to accommodate attribute selections are similar to those for line drawing.

➔ Raster curves of various widths can be displayed using the method of horizontal or vertical pixel spans.

Case 1: Where the magnitude of the curve slope $|m| <= 1.0$, we plot vertical spans;

Case 2: when the slope magnitude $|m| > 1.0$, we plot horizontal spans.

Different methods to draw a curve:

Method 1: Using circle symmetry property, we generate the circle path with vertical spans in the octant from x = 0 to x = y, and then reflect pixel positions about the line y = x to y=0

Method 2: Another method for displaying thick curves is to fill in the area between two Parallel curve paths, whose separation distance is equal to the desired width. We could do this using the specified curve path as one boundary and setting up the second boundary either inside or outside the original curve path. This approach, however, shifts the original curve path either inward or outward, depending on which direction we choose for the second boundary.

Method 3:The pixel masks discussed for implementing line-style options could also be used in raster curve algorithms to generate dashed or dotted patterns

Method 4: Pen (or brush) displays of curves are generated using the same techniques discussed for straight-line segments.

Method 5: Painting and drawing programs allow pictures to be constructed interactively by using a pointing device, such as a stylus and a graphics tablet, to sketch various curve shapes.

Line Drawing Algorithm

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

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

✓ The Cartesian slope-intercept equation for a straight line is

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

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

✓ Given that the two endpoints of a line segment are specified at positions (x0,y0) and (xend, yend) ,as shown in fig.

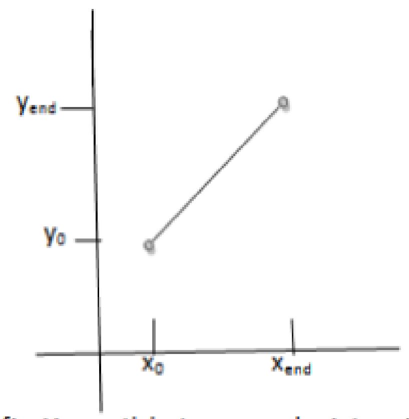


fig. Line path between endpoint positions (x_0,y_0) and (x_{end}, y_{end}).

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

m=(yend - y0)/(xend - x0) ---------------->(2)

b=y0 - m.x0------------- >(3)

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

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

δy=m. δx ---------------- >(4)

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

δx=δy/m----------------- >(5)

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

✓ For lines with slope magnitudes

➔ |m|<1, δx can be set proportional to a small horizontal deflection voltage with the corresponding vertical deflection voltage set proportional to δy from eq.(4)

➔ |m|>1, δy can be set proportional to a small vertical deflection voltage with the corresponding horizontal deflection voltage set proportional to δx from eq.(5)

➔ |m|=1, δx=δy and the horizontal and vertical deflections voltages are equal.

DDA Algorithm (DIGITAL DIFFERENTIAL ANALYZER)

In computer graphics, the Digital Differential Analyzer (DDA) algorithm is used for drawing lines between two given points on a raster grid (like a computer screen). Here's a step-by-step explanation of the DDA algorithm:

1. **Input**: Given two points (x1, y1) and (x2, y2) representing the endpoints of the line to be drawn.
2. **Calculate Differences**: Compute the differences in x and y coordinates between the two points:
   - Δx = x2 - x1
   - Δy = y2 - y1
3. **Determine Steps**: Determine the number of steps needed to reach from one point to the other. The number of steps is typically the maximum absolute difference between Δx and Δy. Let's denote it as `steps`.
4. **Calculate Incremental Values**: Calculate the incremental values for each step:
   - Δx_step = Δx / steps
   - Δy_step = Δy / steps
5. **Initialize Start Point**: Start from the first point (x1, y1).
6. **Drawing Loop**:
   - Iterate `i` from 1 to `steps`:
     - Calculate the coordinates of the next point along the line:
       - $x = x_1 + i \times \Delta x_{step}$
       - $y = y_1 + i \times \Delta y_{step}$
     - Round off the calculated coordinates to the nearest integer values to plot the pixel. If the fractional part of �*x* or �*y* is greater than or equal to 0.5, round up; otherwise, round down.
     - Plot the pixel at the rounded coordinates (x, y) on the screen.
7. **End**: Finish the line drawing process.

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

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

➔ DDA Algorithm has three cases so from equation i.e.., m=(yk+1 - yk)/(xk+1 - xk)

Case1:

if m<1, x increment in unit intervals

i.e.., xk+1=xk+1

then, m=(yk+1 - yk)/(xk+1 - xk)

m= yk+1 - yk

yk+1 = yk+ m----------- >(1)

➔ where k takes integer values starting from 0,for the first point and increases by 1 until final endpoint is reached. Since m can be any real number between 0.0 and 1.0,

Case2:
if m>1, y increment in unit intervals
i.e.., yk+1 = yk + 1
then, m= (yk + 1- yk)/(xk+1 - xk)
m(xk+1 - xk)=1
xk+1 =(1/m)+ xk ------------------------ (2)
Case3:
if m=1,both x and y increment in unit intervals
i.e..,xk+1=xk + 1 and yk+1 = yk + 1

Equations (1) and (2) are based on the assumption that lines are to be processed from the left endpoint to the right endpoint. If this processing is reversed, so that the starting endpoint is at the right, then either we have δx=-1 and
yk+1 = yk- m (3)
or(when the slope is greater than 1)we have δy=-1 with
xk+1 = xk- (1/m)--------------- (4)
➔ Similar calculations are carried out using equations (1) through (4) to determine the pixel positions along a line with negative slope. thus, if the absolute value of the slope is less than 1 and the starting endpoint is at left ,we set δx==1 and calculate y values with eq(1).
➔ when starting endpoint is at the right(for the same slope),we set δx=-1 and obtain y positions using eq(3).
➔ This algorithm is summarized in the following procedure, which accepts as input two integer screen positions for the endpoints of a line segment.
➔ if m<1,where x is incrementing by 1
yk+1 = yk + m
➔ So initially x=0,Assuming (x0,y0)as initial point assigning x= x0,y=y0 which is the starting point .

o Illuminate pixel(x, round(y))

o x1= x+ 1 , y1=y + 1

o Illuminate pixel(x1,round(y1))

o x2= x1+ 1 , y2=y1 + 1

o Illuminate pixel(x2,round(y2))

o Till it reaches final point.

➔ if m>1,where y is incrementing by 1 k+1 =(1/m)+ xk

➔ So initially y=0,Assuming (x0,y0)as initial point assigning x= x0,y=y0 which is the starting point .

o Illuminate pixel(round(x),y)

o x1= x+(1/m) ,y1=y

o Illuminate pixel(round(x1),y1)

o x2= x1+ (1/m) , y2=y1

o Illuminate pixel(round(x2),y2)

o Till it reaches final point.

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

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

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

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

```
#include <stdlib.h>
#include <math.h>
inline int round (const float a)
{
```

```
return int (a + 0.5);
}
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));
}}
```

Bresenham's Algorithm:

- It is an efficient raster scan generating algorithm that uses incremental integral calculations.
- Bresenham's line algorithm is an efficient method for drawing a straight line between two points on a raster grid (like a computer screen).
- The algorithm only uses integer arithmetic, making it fast and suitable for implementation on computers without floating-point hardware.
- It determines which pixels to turn on to approximate the line accurately.
- To illustrate Bresenham's approach, we first consider the scan-conversion process for lines with positive slope less than 1.0.
- Pixel positions along a line path are then determined by sampling at unit x intervals.

Starting from the left endpoint (x0, y0) of a given line, we step to each successive column (x position) and plot the pixel whose scan-line y value is closest to the line path.

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

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

1. Input the two line endpoints and store the left endpoint in (x0, y0).
2. Set the color for frame-buffer position (x0, y0); i.e., plot the first point.
3. Calculate the constants Δx, Δy, $2\Delta y$, and $2\Delta y - 2\Delta x$, and obtain the starting value for the decision parameter as $p_0 = 2\Delta y - \Delta x$
4. At each x_k along the line, starting at $k = 0$, perform the following test:

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

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

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

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

5. Repeat step 4 $\Delta x - 1$ more times.

Note:

If $|m|>1.0$, Then

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

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

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

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

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

Code:

```
#include <stdlib.h>
#include <math.h>
/* 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);
}
}
```