Trinity College of Engineering & Research
Department of Computer Engineering

Computer Graphics

Prof. Prasad I. Bhosle
Assist. Professor

1.1 Image and Objects

In an **image**-based **graphic**, the places where colors are measured are fixed, while the colors at these positions are different.

A simple form of **object**-based **graphic** is a 2D vector **graphics image**, where the **picture** is represented by a list of 2D **objects** like lines and circles.

1.1.2 Pixel and Resolution

The smallest element of a digitized image

Megapixel - a unit equal to 1 million pixels

Resolution - the number of Pixels in an image.

1.1.3 Text Mode Graphics

- Alternatively known as **character mode** or **alphanumeric mode**, **text mode** is a display mode divided into rows and columns of boxes showing only alphanumeric characters.
- **Text mode** is a mode of a software program where only text is displayed.

1.1.5 Bitmap and Vector Based Graphics

- **A bitmap (also called "raster")** graphic is created from rows of different colored pixels that together form an image.
- In their simplest form, bitmaps have only two colors, with each pixel being either black or white.
- Examples of bitmap graphic formats include GIF, JPEG, PNG, TIFF, XBM, BMP, and PCX as well as bitmap.
- **Vector (also known as "object-oriented")** graphics are constructed using mathematical formulas describing shapes, colors, and placement.
- Rather than a grid of pixels, a vector graphic consists of shapes, curves, lines, and text which together make a picture.
- While a bitmap image contains information about the color of each pixel, a vector graphic contains instructions about where to place each of the components.

1.1.6Application of Computer Graphics

- **Computer-Aided Design for engineering and architectural systems etc.**
- **Presentation Graphics**

To produce illustrations which summarize various kinds of data. Except 2D, 3D graphics are good tools for reporting more complex data.

- **Computer Art**

Painting packages are available. For films, 24 frames per second are required. For video monitor, 30 frames per second are required.

- **Entertainment**

Motion pictures, Music videos, and TV shows, Computer games

- **Education and Training**

Training with computer-generated models of specialized systems such as the training of ship captains and aircraft pilots.

- **Visualization**

For analyzing scientific, engineering, medical and business data or behavior. Converting data to visual form can help to understand mass volume of data very efficiently.

- **Image Processing**

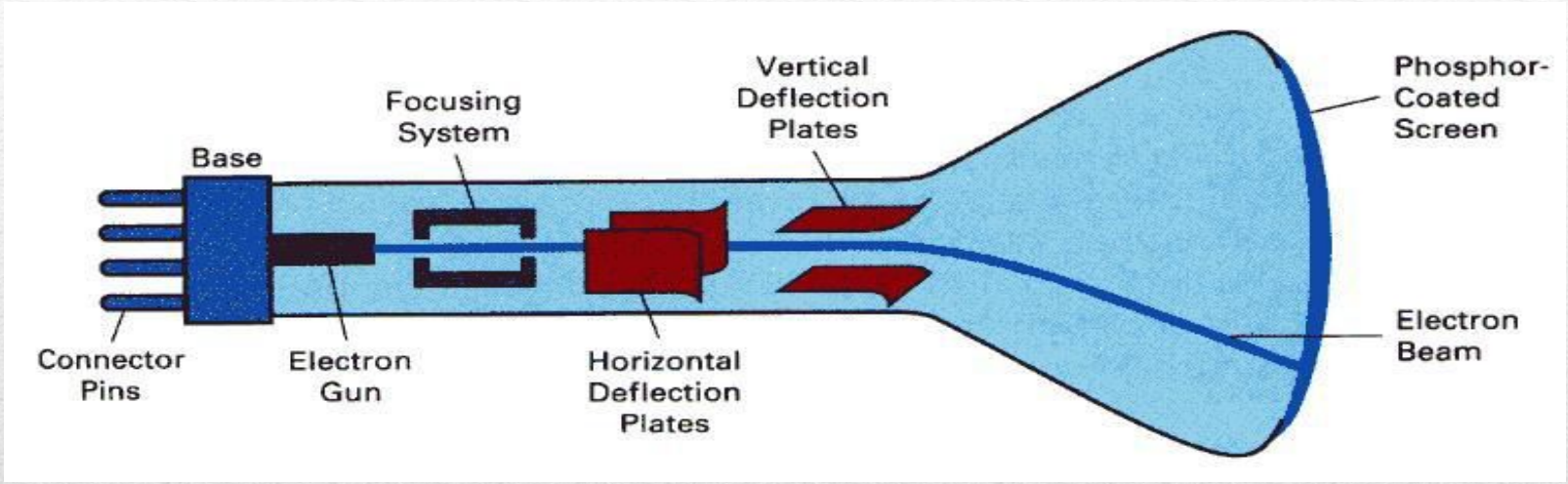
Image processing is to apply techniques to modify or interpret existing pictures. It is widely used in medical applications.

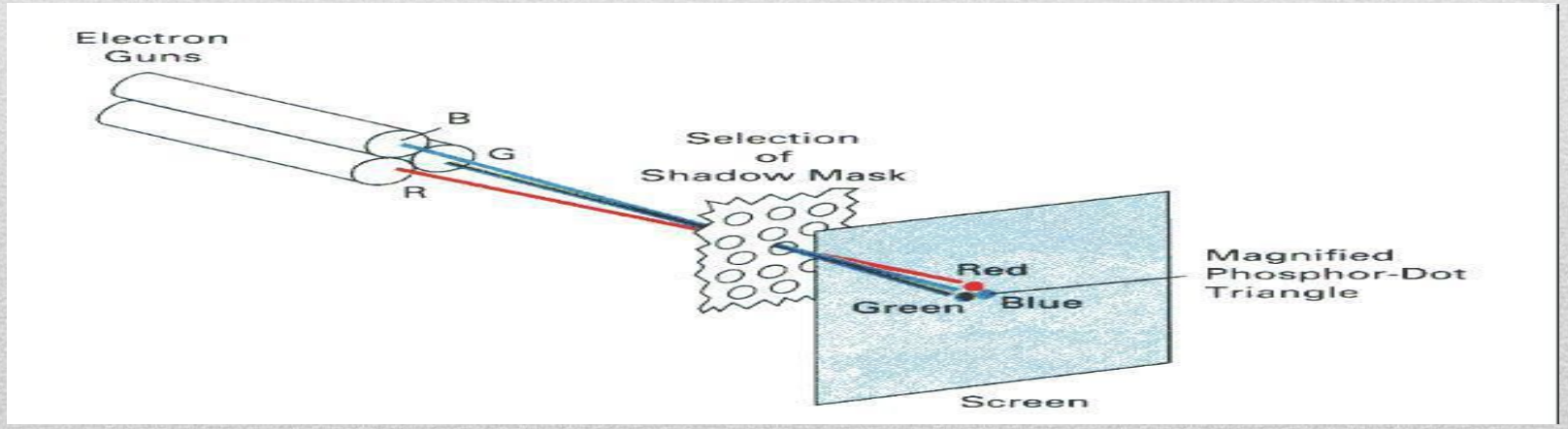
- **Graphical User Interface**

Multiple window, icons, menus allow a computer setup to be utilized more efficiently.

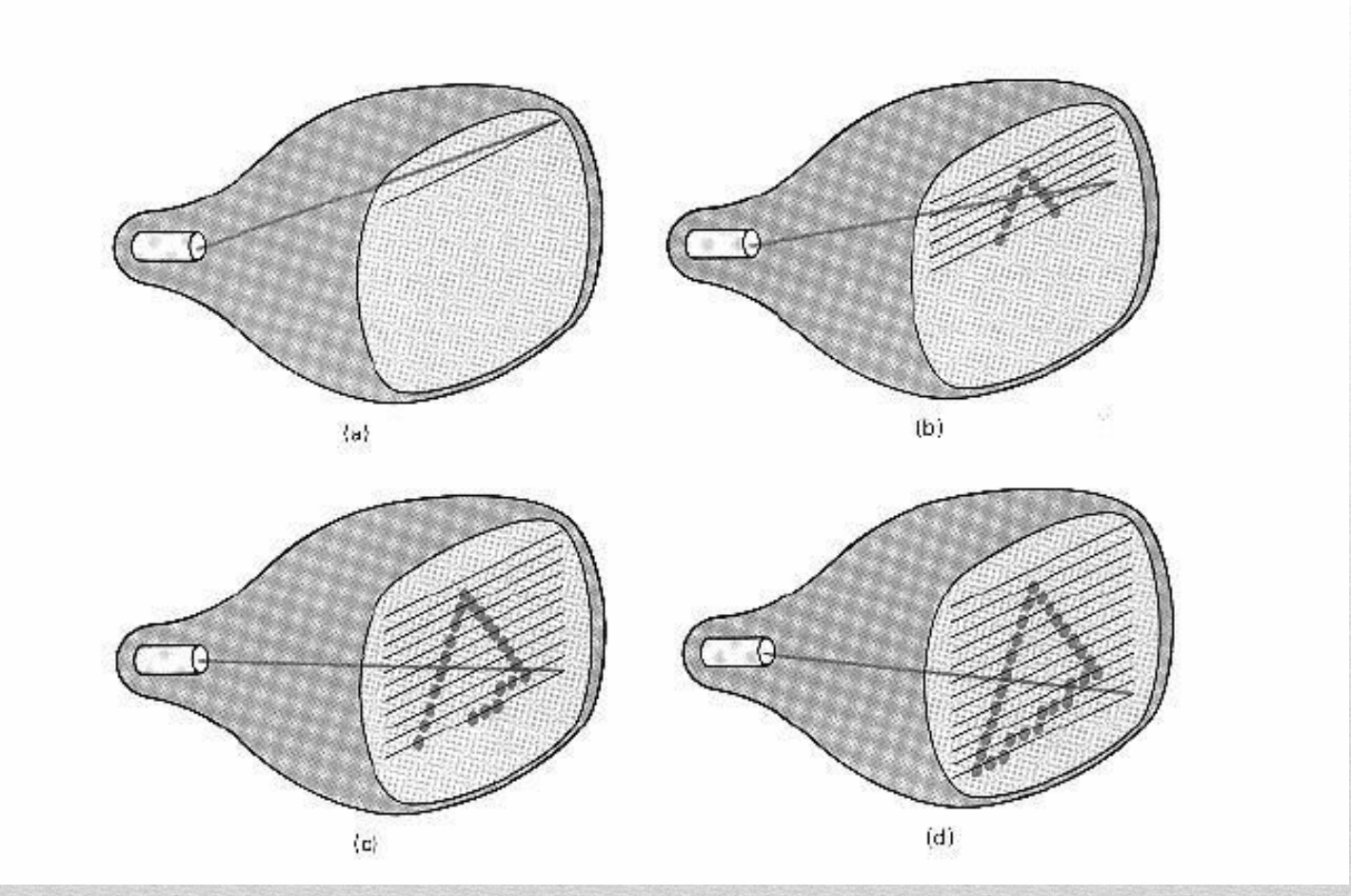
1.2 Display Devices:

1.2.1 Cathode-Ray Tubes (CRT)

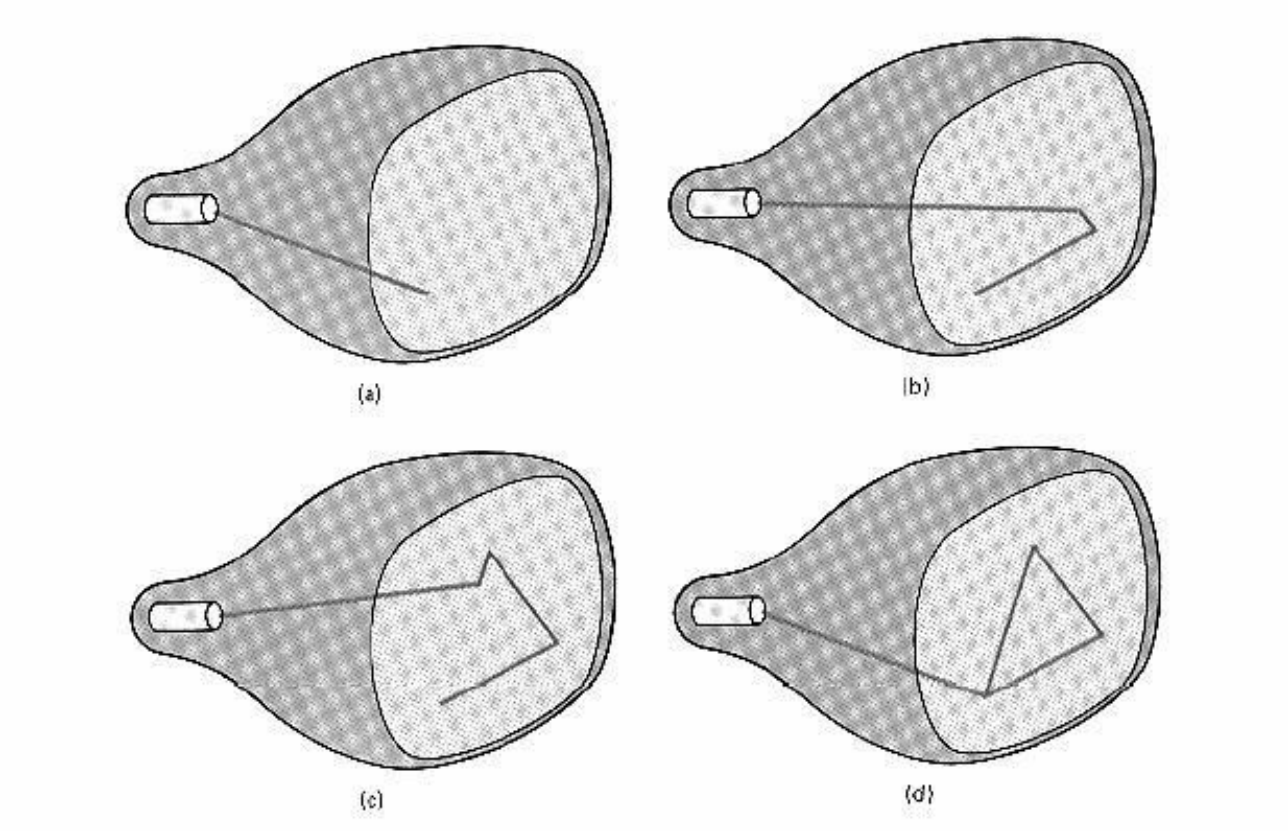




1.2.2 Raster-Scan



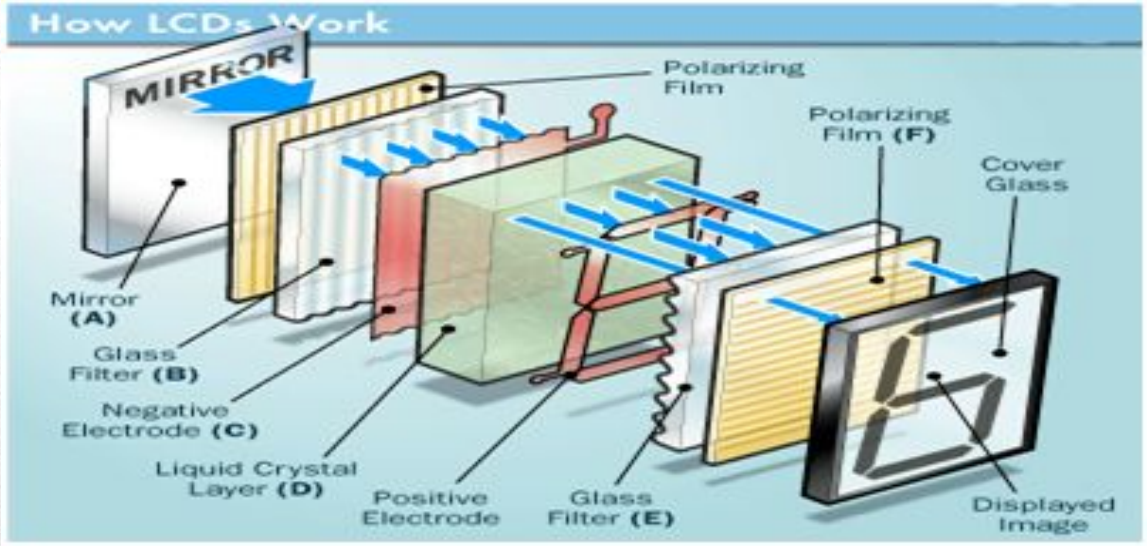
1.2.3 Random Scan Display



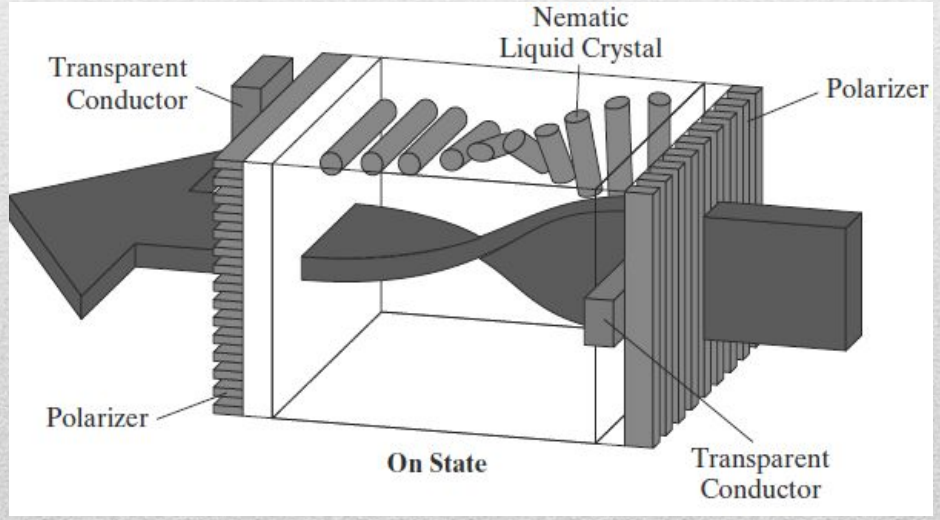
1.2.4 Flat-Panel Displays

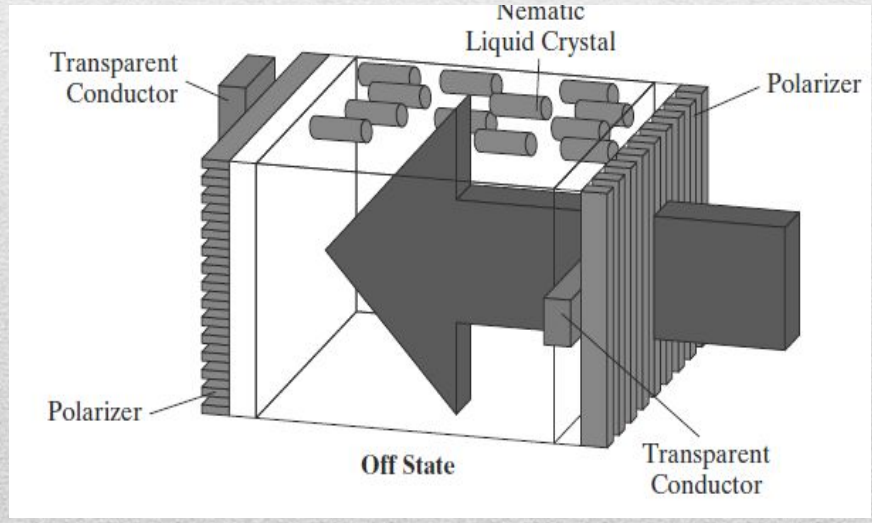
- **flat-panel display** refers to a class of video devices that have reduced volume, weight, and power requirements compared to a CRT.
- A significant feature of flat-panel displays is that they are thinner than CRTs, and we can hang them on walls or wear them on our wrists.
- Some additional uses for flat-panel displays are as small TV monitors, calculator screens, pocket video-game screens, laptop computer screens, advertisement boards in elevators, and graphics displays in applications, portable monitors.
- flat-panel displays into two categories: **emissive displays** and **nonemissive displays.**
- **Emissive display:** devices that convert electrical energy into light (**emitter**). E.g. plasma, LED
- **Nonemissive displays :** use optical effects to convert sunlight or light from some other source into graphics patterns. (**nonemitters) e.g.** liquid-crystal device. LCD

1.2.5 LED and LCD Displays



LCD Layered Diagram

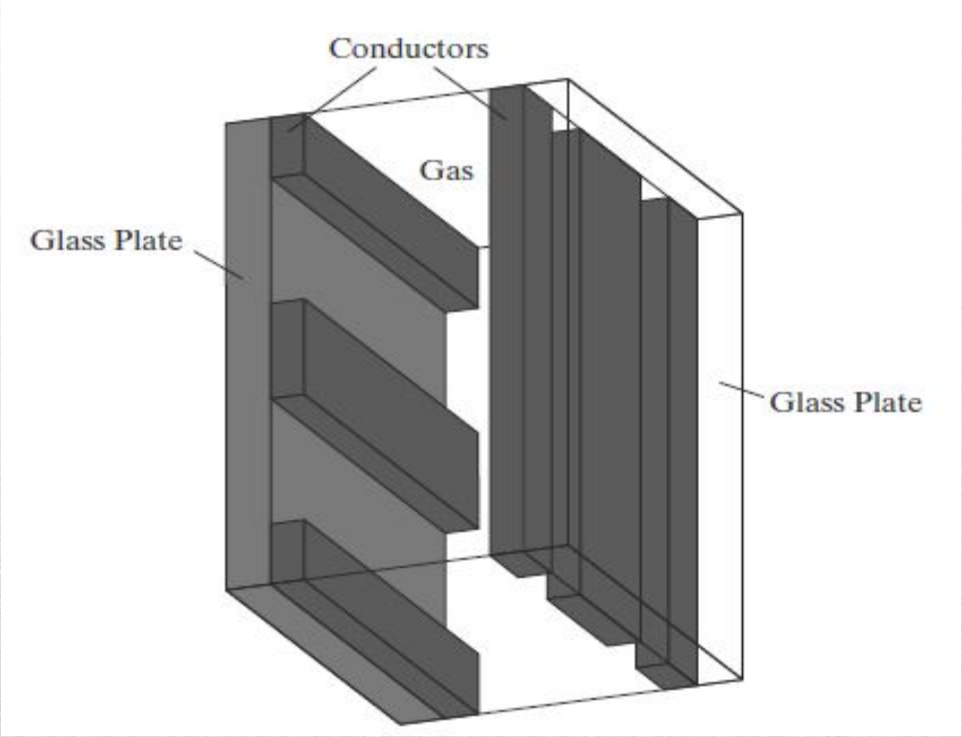




LCD Working Diagram

Click here for Vedio on LCD https://youtu.be/OPZDvGermEI?feature=shared

1.2.6 Plasma Displays



Click here for Vedio on Plasma Display https://youtu.be/Jfpe9txZ4ec?feature=shared

https://youtu.be/8lkuqybOjBw?feature=shared

1.3 Output primitive

Graphic SW and HW provide subroutines to describe a scene in terms of basic geometric structures called output primitives.

Output primitives are combined to form complex structures

Simplest primitives – Point(pixel)

– Line segment

Converting output primitives into frame buffer updates. Choose which pixels contain which intensity value.

Constraints – Straight lines should appear as a straight line

– primitives should start and end accurately

– Primitives should have a consistent brightness along their length

– They should be drawn rapidly

Algorithms are used to plot primitive

1. Line drawing: Digital Differential Analyzer(DDA), Bresenham's Line Algorithm
2. Circle Generation: DDA, Bresenham's Line Algorithm
3. Polygon : Scan line, fill methods, flood fill algorithm.

GL (Graphics Library), OpenGL library are provide package to generate graphics.

- C++ illustrate applications of OpenGL and the general algorithms for implementing graphics functions.
- OpenGL is a software interface that allows the programmer to create 2D and 3D graphics images. OpenGL is both a standard API and the implementation of that API. You can call the functions that comprise OpenGL from a program you write and expect to see the same results no matter where your program is running.
- OpenGL is independent of the hardware, operating, and windowing systems in use. The fact that it is windowing-system independent, makes it portable.
- OpenGL program must interface with the windowing system of the platform where the graphics are to be displayed. Therefore, a number of windowing toolkits have been developed for use with OpenGL.
- OpenGL's rendering commands, however are "primitive". You can tell the program to draw points, lines, and polygons, and you have to build more complex entities upon these.

GLUT (pronounced like the *glut* in *gluttony*)

- It is the OpenGL Utility Toolkit, a window system independent toolkit for writing OpenGL programs.
- It implements a simple windowing application programming interface (API) for OpenGL.
- GLUT makes it considerably easier to learn about and explore OpenGL programming.
- GLUT provides a portable API so you can write a single OpenGL program that works on both Win32 PCs and X11 workstations.
- GLUT is designed for constructing small to medium sized OpenGL programs.
- While GLUT is well-suited to learning OpenGL and developing simple OpenGL applications,
- The GLUT library has both C, C++ (same as C), FORTRAN, programming bindings.
- The GLUT source code distribution is portable to nearly all OpenGL implementations for the X Window System and Windows 95 and NT.
- GLUT also works well with Brian Paul's Mesa, a freely available implementation of the OpenGL API

Input and Interaction

Objectives are to learn about: Introducing variety of devices that are used for interaction. Learn about two different perspectives from which the input devices are considered:

1) The way that the physical devices can be described by the real-world properties,
2) The way that these devices appear to the application program. How a client-server networks and graphics work.

Input Devices

There are two different ways to look at the devices:

Physical devices: keyboard or mouse, and discuss how they work

- There are two primary types of physical devices:
  1) pointing devices 2) keyboard devices

Logical devices: the way application programs look at the devices.

- A logical device is characterized by its high-level interface with the user program, rather than by its physical characteristics.
- In C input and output are done using, printf, scanf, getchar, and putchar.
- In computer graphics, the use of logical devices is slightly more complex. This is because the forms that input can take are more varied than the strings of bits or characters.

2.1 Basic Concepts in line Drawing.

- A line in Computer graphics is a portion of straight line that extends indefinitely in opposite direction.
- It is defined by its two end points.
- Its density should be independent of line length.
- The slope intercept equation for a line:

$$y = mx + b \qquad (1)$$

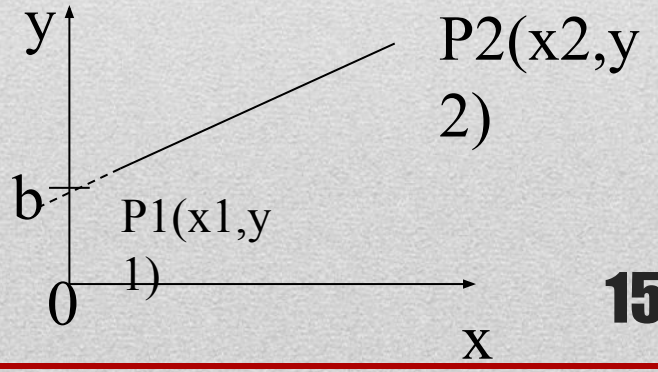
where, **m** = Slope of the line
b = the y intercept of a line

- The two endpoints of a line segment are specified at positions **(x1,y1)** and **(x2,y2).**
- We can determine the value for slope m & b intercept as

$$m = y2-y1/x2-x1$$

$$\text{i.e. } m = \Delta y / \Delta x \qquad (2)$$

- The challenge is to find a way to calculate the next x,y position by previous one as quickly as possible.



2.1.1 *Digital Differential analyzer (DDA)*

- The Digital differential analyzer (DDA) algorithm is an incremental scan-conversion method.
- Such an approach is characterized by performing calculations at each step using results from the preceding step.
- (x1,y1) (x2,y2) are the end points and dx, dy are the float variables.
- Where dx= abs(x2-x1) and dy= abs(y2-y1)
- Limitations of DDA:

(1) The rounding operation & floating point arithmetic are time consuming procedures.

(2) Round-off error can cause the calculated pixel position to drift away from the true line path for long line segment.

Steps of DDA Algorithm :

(i) If dx >=dy then
length = dx
else
length = dy
endif

(ii) dx = (x2-x1)/length
dy = (y2-y1)/length

(iii) x = x1 + 0.5* Sign(dx)
y = y1 + 0.5* Sign(dy)

Here, Sign() function will make the algorithm to work in all quadrant. It returns -1,0, 1 depending on whether its argument is <0, =0, >0 respectively.

(iv) i = 0

(v) $x_i = x + dx$
$y_i = y + dy$

(vi) Plot $((x_i), (y_i))$

(vii) i = i + 1

(viii) If i <= length then go to step (v)

(ix) Stop

Example of DDA

Scan convert a line having end points (3,2) & (4,7) using DDA.

Solution:

i) dx= x2 - x1 = 4-3 = 1, dy= y2 - y1 = 7-2 = 5

As, dx < dy then

length = y2-y1 = 5

ii) dx = (x2-x1)/ length; dx=(4-3)/5; dx=1/5; dx=0.2

dy = (y2-y1)/ length; dy=(7-2); dy= 5/5; dy= 1

iii) x=x1+0.5* Sign(0.2); x0= 3+0.5 x0=3.5

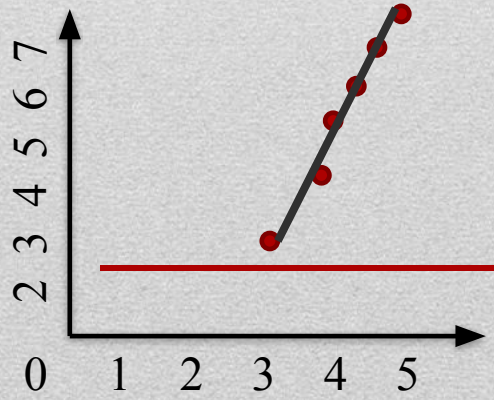
y=y1+0.5* Sign(1); y0=2+0.5 y0=2.5

x1=3.5+0.2 x1=3.7

y1=3+1 y1=4

iv) Follow step no iii) until i<Length; i<=5

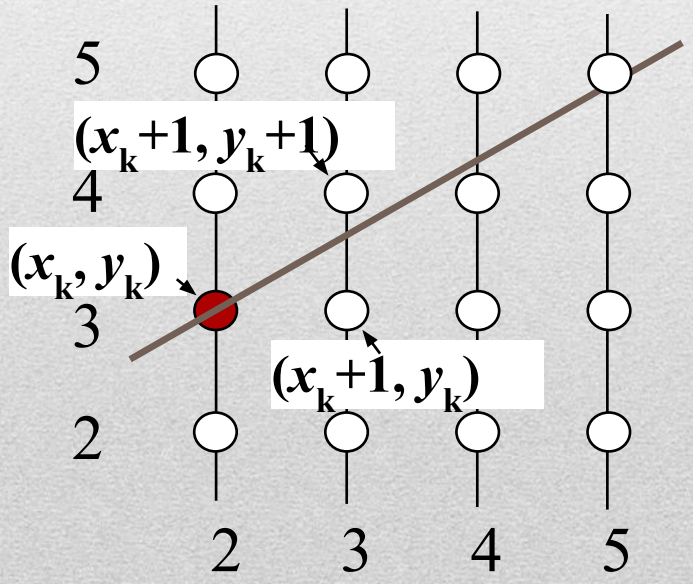
x1	y1	x2	y2	L	dx	dy	i	x	y	Result	Plot
3	2	4	7	5	0.2	1	0	3.5	2.5	3.5,2.5	4,3
							1	3.7	4	3.4,4	3,4
							2	3.9	5	3.6,5	4,5
							3	4.1	6	3.8,6	4,6
							4	4.3	7	4,7	4,7
							5	4.5	8	4.2,8	4,8



2.2 The Bresenham Line Algorithm

- The Bresenham algorithm is another incremental scan conversion algorithm
- The big advantage of this algorithm is that it uses only integer calculations move across the *x* axis in unit intervals and at each step choose between two different *y* coordinates.
- The Bresenham line algorithm has the following advantages:
  - An fast incremental algorithm

  Uses only integer calculations.



For example, from position (2, 3) we have to choose between (3, 3) and (3, 4)

We would like the point that is closer to the original line

Steps of Bresenham Line Drawing Algorithm:

i) Calculate ΔX and ΔY from the given input.

These parameters are calculated as-

$$\Delta X = X_n - X_0$$

$$\Delta Y = Y_n - Y_0$$

ii) Calculate the decision parameter P_k.

It is calculated as-

$$P_k = 2\Delta Y - \Delta X$$

iii) Suppose the current point is (X_k, Y_k) and the next point is (X_{k+1}, Y_{k+1}). Find the next point depending on the value of decision parameter P_k.

Case 1: IF $P_k < 0$;

$$P_{k+1} = P_k + 2\Delta Y$$

$$X_{k+1} = X_k + 1$$

$$Y_{k+1} = Y_k$$

Case 2: IF $P_k >= 0$;

$$P_{k+1} = P_k + 2\Delta Y - 2\Delta X$$

$$X_{k+1} = X_k + 1$$

$$Y_{k+1} = Y_k + 1$$

iv) Keep repeating Step-03 until the end point is reached or number of iterations equals to (ΔX-1) times.

Example: Calculate the points between the starting coordinates (9, 18) and ending coordinates (14, 22)

Solution-

Starting coordinates = $(X_0, Y_0) = (9, 18)$

Ending coordinates = $(X_n, Y_n) = (14, 22)$

Step-01:

Calculate ΔX and ΔY from the given input.

$\Delta X = X_n - X_0 = 14 - 9 = 5$

$\Delta Y = Y_n - Y_0 = 22 - 18 = 4$

Step-02:

Calculate the decision parameter.

$P_k = 2\Delta Y - \Delta X; \quad P_k = 2 \times 4 - 5; \quad P_k = 3$

So, decision parameter $P_k = 3$

Step-03:

As $P_k >= 0$, so case-02 is satisfied.

Thus,

$P_{k+1} = P_k + 2\Delta Y - 2\Delta X = 3 + (2 \times 4) - (2 \times 5) = 1$

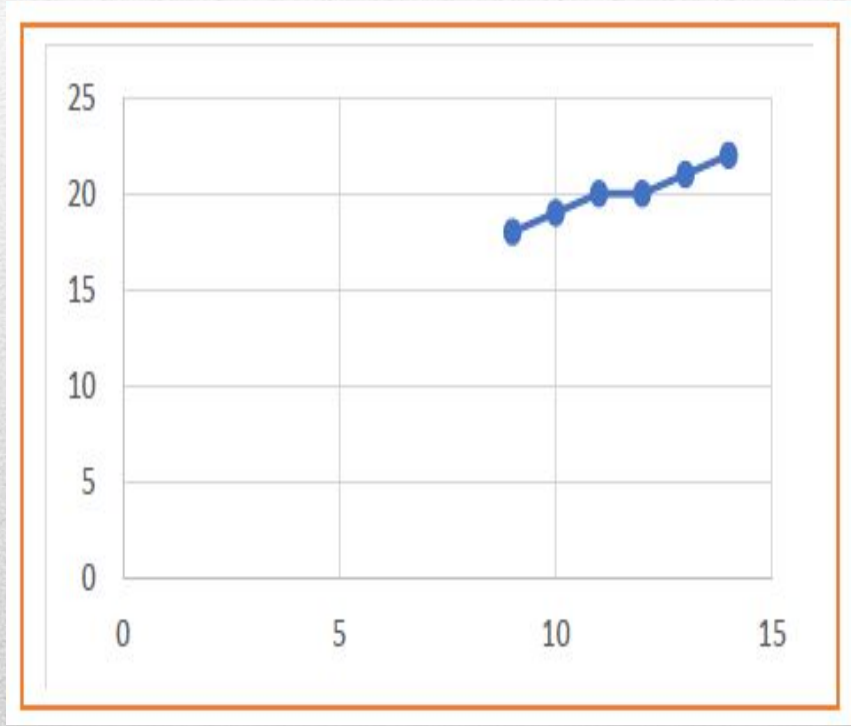
$X_{k+1} = X_k + 1 = 9 + 1 = 10$

$Y_{k+1} = Y_k + 1 = 18 + 1 = 19$

Similarly, Step-03 is executed until the end point is reached or number of iterations equals to 4 times.

(Number of iterations = $\Delta X - 1 = 5 - 1 = 4$)

P_k	P_{k+1}	X_{k+1}	Y_{k+1}
		9	18
3	1	10	19
1	-1	11	20
-1	7	12	20
7	5	13	21
5	3	14	22

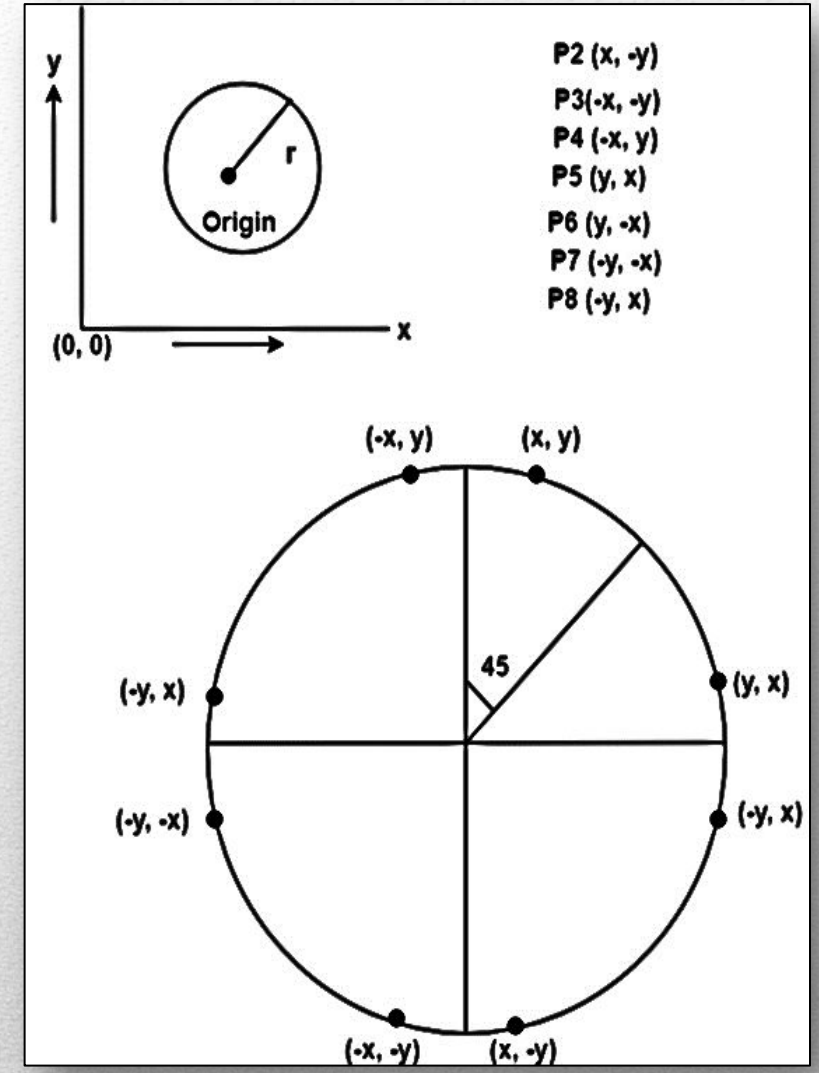


2.3 Circle Generating Algorithms

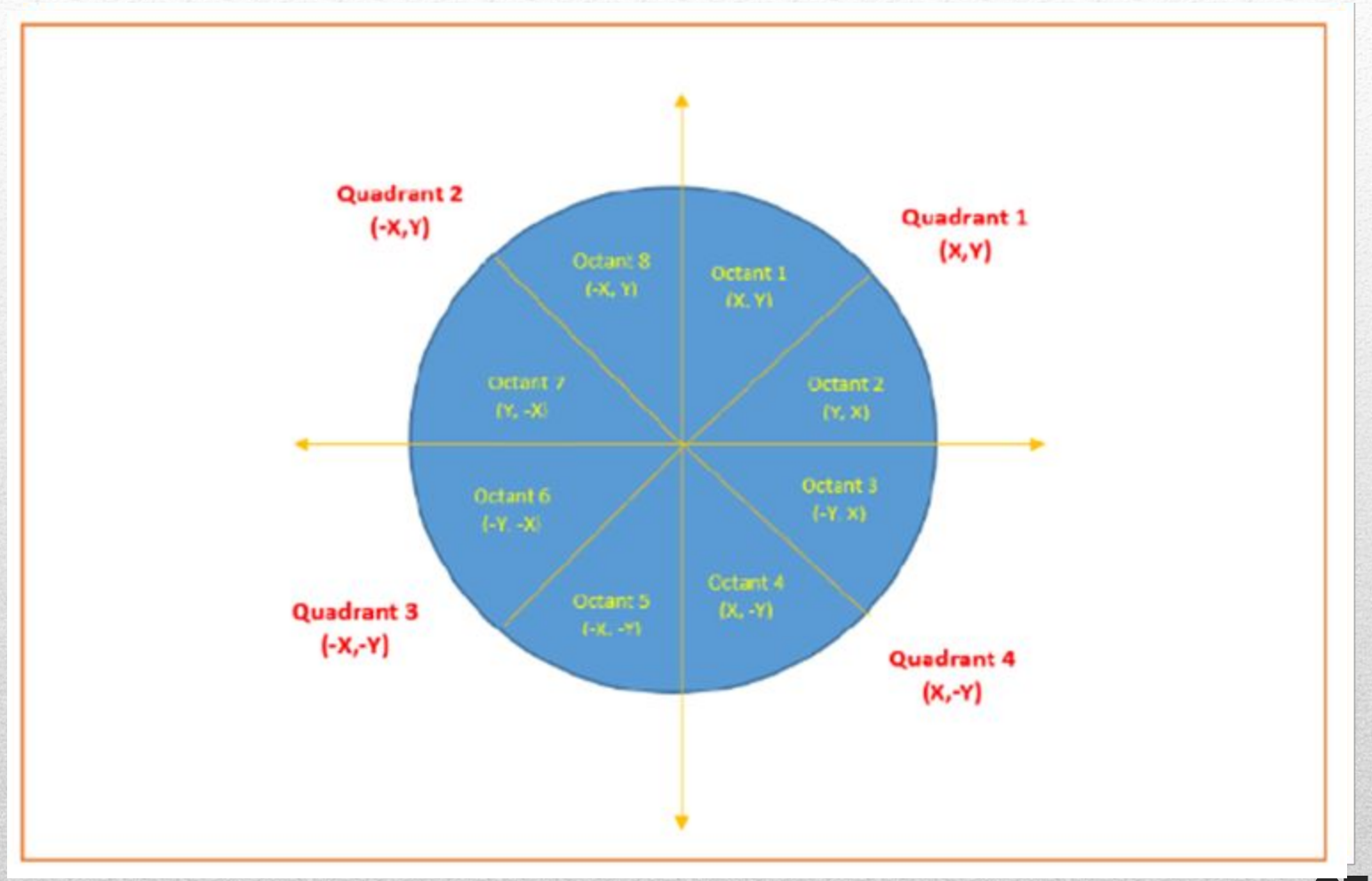
2.3.1 Symmetric of Cirlcle

- Circle is an eight-way symmetric figure. The shape of circle is the same in all quadrants.
- In each quadrant, there are two octants. If the calculation of the point of one octant is done, then the other seven points can be calculated easily by using the concept of eight-way symmetry.
- For drawing, circle considers it at the origin.
- If a point is $P_1(x, y)$, then the other seven points will beSo we will calculate only 45°arc. From which the whole circle can be determined easily.
- If we want to display circle on screen then the putpixel function is used for eight points as shown below:

putpixel (x, y, color)
putpixel (x, -y, color)
putpixel (-x, y, color)
putpixel (-x, -y, color)
putpixel (y, x, color)
putpixel (y, -x, color)
putpixel (-y, x, color)
putpixel (-y, -x, color)



2.3.2 Bresenham Circle Drawing Algorithm-



2.3.2 Bresenham Circle Drawing Algorithm-

- The points for other octacts are generated using the eight symmetry property.
- Given the centre point and radius of circle,
- Bresenham Circle Drawing Algorithm attempts to generate the points of one octant.

Bresenham Circle Drawing Algorithm involves the following steps

Step-01: Assign the starting point coordinates (X0, Y0) as-

$X_0 = 0;$ $Y_0 = R$

Step-02: Calculate the value of initial decision parameter P0 as-

$P_0 = 3 - 2 * R$

Step-03: Suppose the current point is (X_k, Y_k) and the next point is (X_{k+1}, Y_{k+1}).

Find the next point of the first octant depending on the value of decision parameter Pk.

Follow the below two cases-

Case 1: IF Pk<0; $X_{k+1}=X_k+1;$ $Y_{k+1}=Y_k$; $P_{k+1}=P_k+4X_{k+1}+6$

Case 2: IF Pk>0; $X_{k+1}=X_k+1;$ $Y_{k+1}=Y_k - 1$; $Pk+1=Pk+4(X_{k+1}-Y_{k+1})+10$

Step-04:

If the given centre point (X0, Y0) is not (0, 0), then do the following and plot the point-

Xplot = Xc + X0

Yplot = Yc + Y0

Here, (Xc, Yc) denotes the current value of X and Y coordinates.

Step-05: Keep repeating Step-03 and Step-04 until Xplot = > Yplot.

Step-06: Step-05 generates all the points for one octant.

To find the points for other seven octants, follow the eight symmetry property of circle.

Example : Given the center point coordinates (10, 10) and radius as 10, generate all the points to form a circle using **Bresenham Circle Drawing Algorithm**

Solution- Centre Coordinates of Circle $(X_0, Y_0) = (10, 10)$
Radius of Circle = 10

Step-01: Assign the starting point coordinates (X_0, Y_0) as-
$X_0 = 0$; $Y_0 = R = 10$

Step-02:
Calculate the value of initial decision parameter P_0 as-
$P_0 = 3 – 2 \times R$; $P_0 = 3 – 2 \times 10$; $P_0 = -17$

Step-03:
As $P_{initial} < 0$, so case-01 is satisfied.
Thus,
$X_{k+1} = X_k + 1 = 0 + 1 = 1$
$Y_{k+1} = Y_k = 10$
$P_{k+1} = P_k + 4 \times X_{k+1} + 6 = -17 + (4 \times 1) + 6 = -7$

Step-04:
This step is applicable here as the given centre point coordinates is (10, 10).
$X_{plot} = X_c + X_0 = 1 + 10 = 11$
$Y_{plot} = Y_c + Y_0 = 10 + 10 = 20$

Step-05:
Step-03 and Step-04 are executed similarly until $X_{plot} => Y_{plot}$ as follows-

P_k	P_{k+1}	(X_{k+1}, Y_{k+1})	(X_{plot}, Y_{plot})
		(0, 10)	(10, 20)
-17	-7	(1, 10)	(11, 20)
-7	7	(2, 10)	(12, 20)
7	-7	(3, 9)	(13, 19)
-7	15	(4, 9)	(14, 19)
15	13	(5, 8)	(15, 18)
13	19	(6, 7)	(16, 17)
Algorithm Terminates These are all points for Octant-1.			