MODULE 1

1. What is Computer Graphics? Explain the application of Computer Graphics. → 1
2. Explain the operation of video monitors based on standard CRT design. → 5
3. Differentiate raster scan displays and random scan displays. → 7
4. Explain the following → 8
   a) Color CRT Monitors
   b) Flat panel Displays
5. Explain the Architecture of a simple raster graphics system and a raster graphics system with a display processor. → 14
6. Explain the input devices. → 16
7. Explain the display window management using GLUT. → 20
8. What is coordinate reference frames, screen coordinates, absolute and relative coordinates? → 21
9. Explain the OpenGL functions for point and line. → 22
10. Explain the DDA line drawing algorithm. → 26
11. Explain the Bresenham's Line drawing algorithm. → 29
12. Explain the Bresenham's Midpoint circle drawing algorithm. → 33
13. Explain the Point attribute functions. → 35
14. List the OpenGL line attribute functions → 36
15. List and explain OpenGL point and line primitive with example → 38
16. Explain Cathode Ray Tube with diagram. → 40
17. With a neat diagram, explain Refresh Cathode Ray tubes → 42
18. Implement an OpenGL program for Bresenham's line drawing algorithm. → 46
19. Write short note on basic OpenGL syntax → 51
20. Explain properties of circle → 52
21. Implement midpoint circle draw in OpenGL → 53
22. Implement an OpenGL program to display points and lines along with its attribute functions included. → 55
23. Write Bresenham's line drawing Algorithm for |m| < 1.0 .Digitalize the line with endpoints (20,10) (30,18) → 57
24. Given a circle with radius=10 demonstrate the midpoint circle algorithm by determining positions along circle octant with first Quadrant from x=0 to x=y(Assume circle center is positioned at origin). → 59
25. Apply Bresenham's Line drawing algorithm for the given end points
    a)(30,20) and (40, 28) b)(0,0) to (5,4) c) (0,0) to (5,6) → 61

MODULE 2

26. With neat diagram, explain the two commonly used algorithms for identifying interior areas of a plane figure. → 63
27. Explain two dimensional viewing transformation pipeline. → 66
28. Show that successive scaling is multiplicative. → 69
29. Show that successive translations are additive. → 71
30. Design a polygon ABC – A(3,2), B(6,2) & C(6,6) rotate in anticlockwise direction by 30 degree by keeping C fixed. → 73
31. What are world coordinates and view port coordinates? Explain 2D viewing transformation pipeline. → 71
32. Explain the scan-line polygon fill algorithm. → 75

Prof Shankar R https://hemanthrajhemu.github.io https://shankarrajagopal.github.io/

Shankar R
Asst Professor,
CSE, BMSIT&M

iii) Virtual-Reality Environments

A more recent application of CG is in the creation of VR environments in which a user can interact with the objects in a 3D-scene. Specialised hardware devices provide 3D viewing effects and allows the user to pick up objects in a scene.

iv) Data Visualisation

Producing graphical representations for scientific, engineering and medical data sets and processes is another fairly new application of CG which is generally referred as scientific visualisation.

v) 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, physiological functions, population trends or equipment such as color coded diagram can help trainees to understand the operations of a system.

vi) Computer Art:

Both fine and commercial artists make use of CG methods. They now have available a variety of computer methods and tools, including specialised hardware, commercial software packages such as Lumena, CAD packages and animation systems.

Prof Shankar R
https://hemanthrajhemu.github.io
https://shankarrajagopal.github.io/

Shankar R
Asst Professor,
CSE, BMSIT&M

vii) Movies and
viii) Games (Entertainment)

Television productions, motion pictures, music videos and the gaming industry use CG methods. Sometimes graphics methods images are combined with live actors and scenes, and sometimes films are completely generated using computer rendering and animation techniques

ix) GUI

It is very common now for Application software to provide a GUI. A major component of Graphical interface is a window manager that allows a user to display multiple display windows.

x) Image Processing

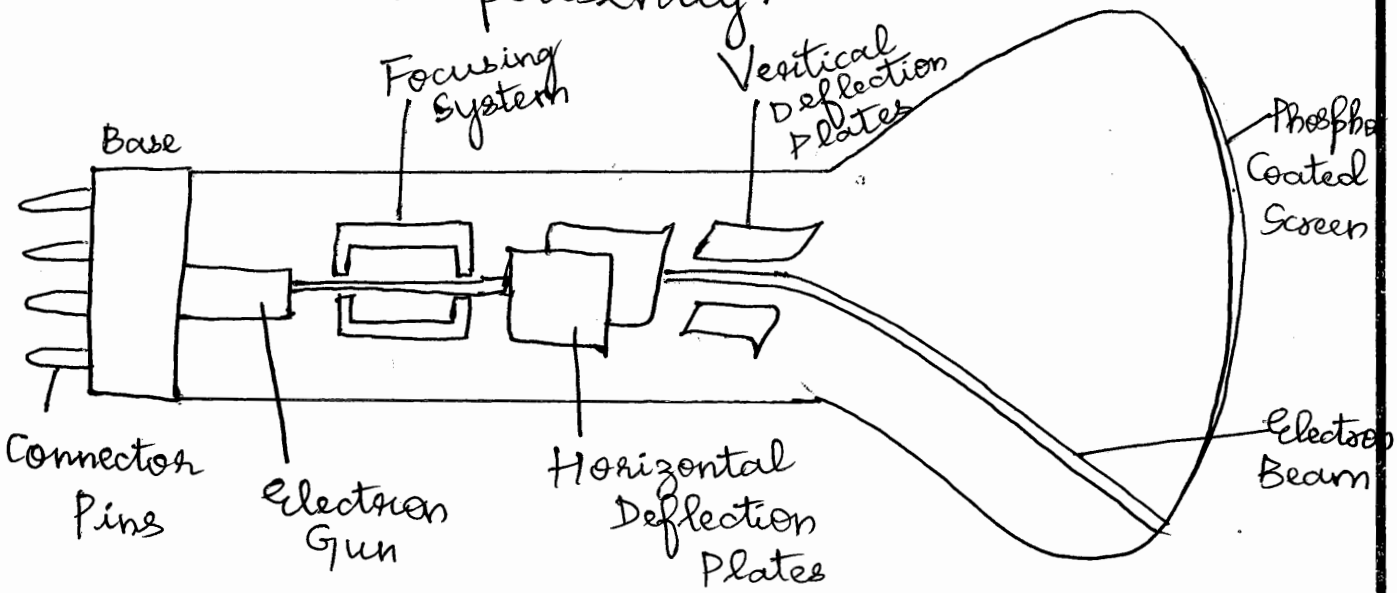
The modification or interpretation of existing pictures such as photographs and TV scans is called Image Processing. Although methods in CG & Image Processing overlap, 2 areas are concerned with fundamentally different operations. In CG, a computer is used to create a picture. Image Processing techniques, on the other hand are used to improve quality of image, analyse image or recognise visual patterns for robotic applications. Image processing methods are often used in CG & CG methods are applied in Image Processing

Prof Shankar R https://hemanthrajhemu.github.io
https://shankarrajagopal.github.io/

Question 2: Explain the operation of video monitors based on standard CRT design.

Shankar R
Asst Professor,
CSE, BMSIT&M

Solution: CRT design video monitors is still the most common video display device presently.



Electrostatic deflection of electron beam in a CRT

An electron gun emits a beam of electrons, which passes through focusing and deflection systems and hits on phosphor-coated screen. The no. of points displayed on a CRT is referred to as resolution (eg. 1024 x 768). Different phosphors emit small light spots of different colors, which can combine to form a range of colors.

A common method for colour CRT display is the shadow-mask method.

Shankar R
Asst Professor,
CSE. BMSIT&M

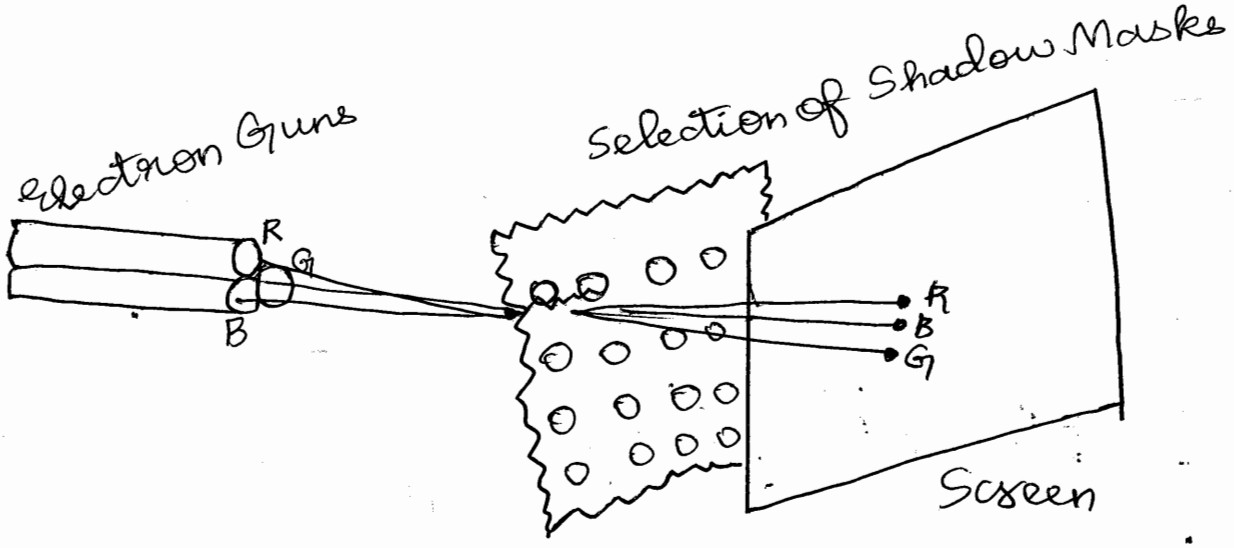


Illustration of a shadow-mask CRT

CRT Phosphor Screen: The screen is coated with phosphor, 3 colors for a color monitor, 1 for monochrome.

For a color monitor, three guns light up red, green or blue phosphors.

Intensity is controlled by the amount of time at a specific phosphor location.

The light emitted by phosphor fades very rapidly, so it needs to redraw the picture repeatedly.

Prof Shankar R https://hemanthrajhemu.github.io
https://shankarrajagopal.github.io/

Shankar R
Asst Professor,
CSE, BMSIT&M

3] Differentiate raster scan displays and random scan displays.

	Random Scan Display.	Raster Scan Display.
1]	In vector scan dislay the beam is moved between the end points of the graphics primitives.	1. In raster scan display the beam is moved all over the screen one scan line at a time, from top bottom and then back to top.
2]	Vector display flickers when the number of primitives in the buffer becomes too large	2. In raster display, the refresh process is independent of the complexity of the image
3	Scan conversion is not required	3. Graphics primitives are specified in terms of their endpoints and must be scan converted into their corresponding pixels in the frame buffer.
4.	Scan conversion hardware is not required.	4. Because each primitive must be scan-converted, real time dynamics is far more computational and requires seperate scan conversion hardware.
5.	Vector display draws a continuous and smooth lines	5. Raster display can display mathematically smooth lines, polygons, and boundaries of curved primitives only by approximating them with pixels on the raster grid
6.	Cost is more	6. Cost is low.
7.	Vector display only draws lines and characters	7. Raster display has ability to display areas filled with solid colours or patterns

Prof Shankar R https://hemanthrajhemu.github.io https://shankarrajagopal.github.io/

Shankar R
Asst Professor,
CSE, BMSIT&M

4] Explain the following:
a. Color CRT Monitors
b. Flat panel Displays.

A CRT monitor displays color pictures by using a combination of phosphors that emit different-colored light. By combining the emitted light from the different phosphors, a range of colors can be generated. The two basic techniques for producing color displays with a CRT are the beam-penetration method and the shadown-mask method.

1] The Beam-penetration method for displaying color pictures has been used with random-scan monitors. Two layers of phosphor, usually red and green, are coated onto the inside of the CRT screen, and the displayed colour depends on how far the electron beam penetrates into the phosphor layer.

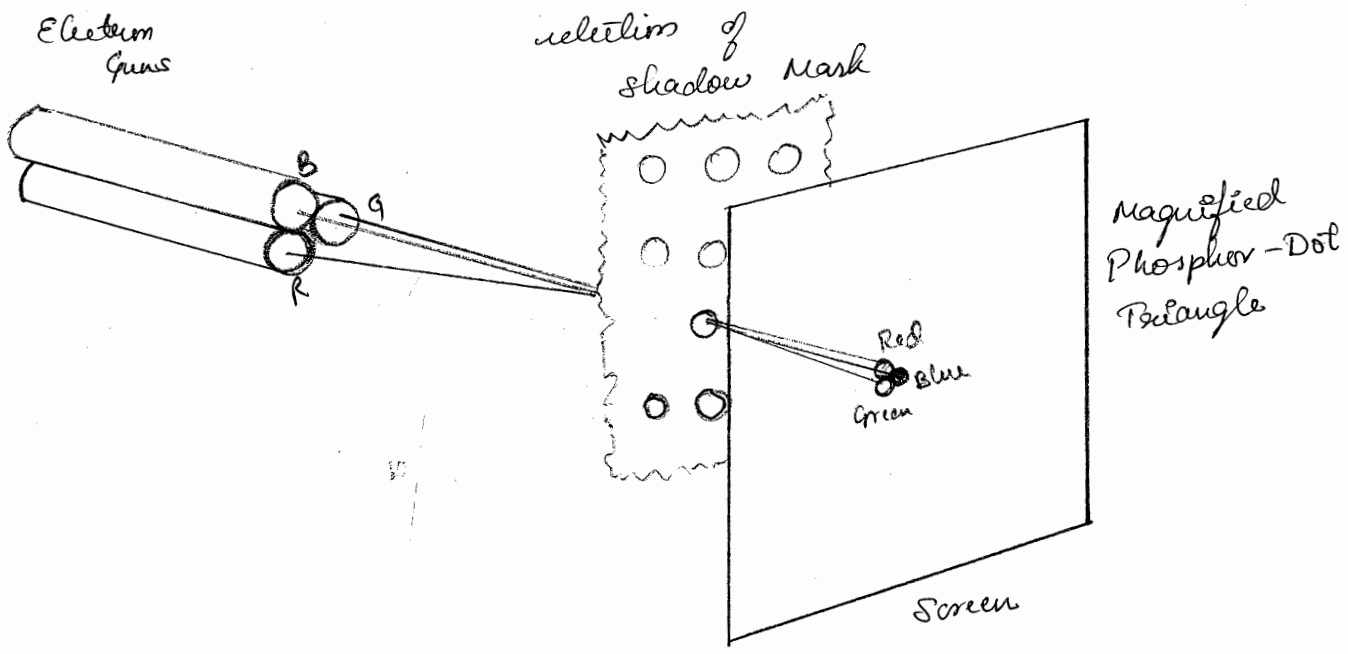
- A beam of slow electrons excites only the outer red layer. A beam of very fast electrons penetrates through the red layer and excites the inner green layer.
- At intermediate beam speeds, combinations of red and green light are emitted to show two additional colors, orange and yellow. The speed of the electrons, and hence the screen color at any point, is controlled by the beam-acceleration voltage. Beam penetration has been an inexpensive way to produce color in random-scan monitors, but only four colors are possible, and the quality of pictures is not as good as with other methods.

2] Shadow-mask methods are commonly used in raster scan systems because they produce a much wider range of colors than the beam penetration method. A shadow mask CRT has three phosphor color dots at each pixel position. One phosphor dot emits a red light, another emits a green light and the third emits 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.

Prof Shankar R
https://hemanthrajhemu.github.io
https://shankarrajagopal.github.io/

Shankar R
Asst Professor,
CSE, BMSIT&M

- 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. The phosphor dots in the triangles are arranged so that each electron beam can activate only its corresponding color dot when it passes through the shadow mask. Another configuration for the three electron guns is an in-line arrangement in which the three electron guns, and the corresponding red-green-blue color dots on the screen, are aligned along one scan line instead of in a triangular pattern. This in-line arrangement of electron guns is easier to keep in alignment and is commonly used in high-resolution color CRT's.
- We obtain color variations in a shadow-mask CRT by varying the intensity of levels of the three electron beams. By turning off the red and green guns, we get only the color coming from the blue phosphor. Other combinations of beam intensity produce the three colors into one composite. The color we see depends on the amount of excitation of the red, green, and blue phosphors. A white (or gray) area is the result of activating all three dots with equal intensity.



Shadow-mask CRT

Prof Shankar R https://hemanthrajhemu.github.io
https://shankarrajagopal.github.io/

Shankar R
Asst Professor,
CSE, BMSIT&M

Flat - Panel Displays.

- The term 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 ~~power requirements compared to a CRT~~ is that they are thinner than CRT's and we can hang them on walls or wear them on wrists.
- Current uses for flat-panel displays include small TV monitors, calculators, pocket video games, laptop computers, armrest viewing of movies on airlines, as advertisement boards in elevators, and as graphics displays in applications requiring rugged, portable monitors

flat-panel displays into two categories.

1] emissive displays.
2] nonemissive displays.

(i) The emissive displays are devices that convert electrical energy into light Plasma panels, thin-film electroluminescent displays, and light-emitting diodes are examples of emissive displays. Flat CRTs have also been devised, in which electron beams are accelerated parallel to the screen, then deflected 90° to the screen. But flat CRTs have not proved to be as successful as other emissive devices.

(ii) Non-emissive displays (or nonemitters) use optical effects to convert sunlight or light from some other source into graphics patterns. The most important example of a non-emissive flat-panel display is a liquid-crystal device.

1 a) Plasma panels, also called gas-discharge displays are 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 ribbons is built into the other glass panel.

Prof Shankar R https://hemanthrajhemu.github.io
https://shankarrajagopal.github.io/

Shankar R
Asst Professor,
CSE, BMSIT&M

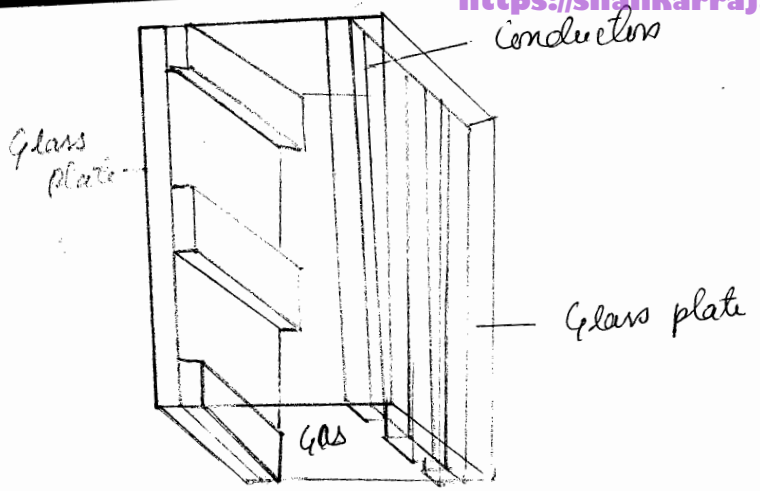
- Firing voltages applied to a pair of horizontal and vertical conductors cause the gas at the intersection of the two conductors to break down into a glowing plasma of electrons and ions.
- Picture definition is stored in a refresh buffer, and the firing voltages are applied to refresh the pixel positions 60 times per second.
- Alternating-current methods are used to provide faster application of the firing voltages, and thus brighter displays.
- Separation between pixels is provided by the electric field of the conductors.

1 b] Thin-film electroluminescent displays are similar in construction to a plasma panel.
- The difference is that the region between the glass plates is filled with a phosphor, such as zinc sulfide doped with manganese, instead of a gas.
- When a sufficiently high voltage is applied to a pair of crossing electrodes, the phosphor becomes a conductor in the area of the intersection of the two electrodes.
- Electrical energy is then absorbed by the manganese atoms, which then release the energy as a spot of light similar to the glowing plasma effect in a plasma panel.
- Electroluminescent displays require more power than plasma panels, and good color and gray scale displays are hard to achieve.

1 c) Light-emitting diodes (LED). A matrix of diodes is arranged to form the pixel positions in the display, and picture definition is stored in a refresh buffer.
- As in scan-line refreshing of a CRT, information is read from the refresh buffer and converted to voltage levels that are applied to the diodes to produce the light patterns in the display.

Prof Shankar R
https://hemanthrajhemu.github.io
https://shankarrajagopal.github.io/

Shankar R
Asst Professor,
CSE, BMSIT&M



Basic design of a plasma-panel display device

2 a] Liquid-crystal displays (LCDs) are commonly used in small systems, such as calculators and portable, laptop computers.

- These non-emissive devices produce a picture by passing polarized light from the surroundings or from an internal light source through a liquid crystal material that can be aligned to either block or transmit the light.
- The term liquid crystal refers to the fact that these compounds have a crystalline arrangement of molecules, yet they flow like a liquid.
- Flat-panel displays commonly use nematic (threadlike) liquid-crystal compounds that tend to keep the long axes of the rod-shaped molecules aligned.
- A flat-panel display can then be constructed with a nematic liquid-crystal ~~compounds that tend to keep the long axes of the~~ as demonstrated
- Two glass plates, each containing a light polarizer at right angles to the other plate, sandwich the liquid-crystal material.
- Rows of horizontal transparent conductors are built into one glass plate, and columns of vertical conductors are put into the other plate.

Prof Shankar R https://hemanthrajhemu.github.io https://shankarrajagopal.github.io/

Shankar R
Asst Professor,
CSE, BMSIT&M

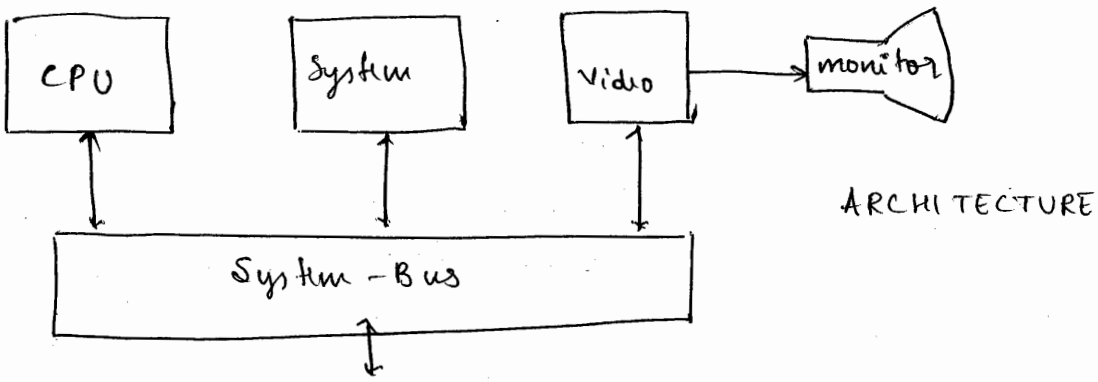
- The intersection of two conductors defines a pixel position.
- Normally, the molecules are aligned as shown in the "on state". The Polarized light passing through the material is twisted so that it will pass through the opposite polarizer.
- The light is then reflected back to the viewer. To turn off the pixel, we apply a voltage to the two intersecting conductors to align the molecules so that the light is not twisted. This type of flat-panel device is referred to as a passive-matrix LCD
- Picture definitions are stored in a refresh buffer, and the screen is refreshed at the rate of 60 frames per second, as in the emissive devices

Prof Shankar R https://hemanthrajhemu.github.io
https://shankarrajagopal.github.io/

Shankar R
Asst Professor,
CSE, BMSIT&M

5. Explain the architecture of a simple raster graphics system and a raster graphics system with a display processor.

Interactive raster-graphics system typically employ several processing units. In addition to the central processing unit, a special-purpose processor, called video controller or display controller, is used to control the operation of the display device. Organisation of a simple raster system is shown below.

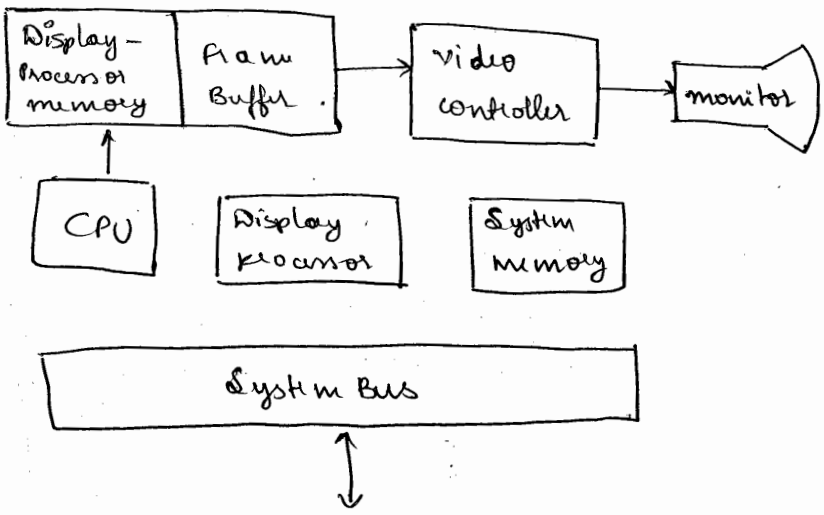


ARCHITECTURE

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, more sophisticated raster systems employ other processors as coprocessors and accelerators to implement various graphic operation.

Prof Shankar R https://hemanthrajhemu.github.io
https://shankarrajagopal.github.io/

Raster Scan Display Processor :- The following figure shows one way to organize the components of a raster system that contains separate display processor, sometimes referred to as a graphics controller or a display co-processor. 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 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. Graphics commands specifying straight lines and other geometric objects are scan conversion. Graphics commands specifying straight lines and other geometric objects are scan con-verted into a set of discrete points, corresponding to screen pixel positions.

Prof Shankar R https://hemanthrajhemu.github.io
https://shankarrajagopal.github.io/

Shankar R
Asst Professor,
CSE, BMSIT&M

Display processors are also designed to perform a number of additional operations. These function include generating various line styles, displaying color areas, and applying transformations to the objects in a scene. Also, display processors are typically designed to interface with interactive input devices, such as a mouse.

6. Explain the input devices.

Graphics workstations make use of various devices for data input. Most systems have keyboards and mouse while some other system have trackball, spaceball, joystick, button boxes, touch panels, image scanners and voice systems.

Keyboard :-

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

Prof Shankar R https://hemanthrajhemu.github.io
https://shankarrajagopal.github.io/

Button Boxes and Dials :-

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

mouse Devices :-

- Mouse is a hand-held device, usually moved around on a flat surface to position the screen cursor, whether of rollers on the bottom of the mouse used to record the amount and direction of movement.
- Some of the mouses uses optical sensors, which detects movement across the horizontal and vertical grid lines.
- since a mouse can be picked up and put down, it is used for making relative changes in the positon of the screen.

Trackballs and spaceballs :-

- A trackball is a ball devices that can be rotated with the fingers or palm of the hand to produce screen cursor movement.
- Laptop keyboards are equipped with a trackball to eliminate the extra space required by a mouse.
- Spaceball is an extension of two-dimensional trackball concept.

Prof Shankar R https://hemanthrajhemu.github.io https://shankarrajagopal.github.io/

Shankar R
Asst Professor,
CSE, BMSIT&M

Data Gloves:

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

Digitizers:

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

Image Scanners:

- Drawing, graphs, photographs or text can be stored for computer processing with an image scanner by passing an optical scanning mechanism over the information to be stored.
- Once we have the representation of the pictures, then we can apply various image-processing method to modify the

representation of the picture and various editing operations can be performed on the stored documents.

Touch panels:

- It allows displayed object or screen positions to be selected with the touch of a finger.
- Touch panel is used for the selection of processing options that are represented as a menu graphical icons.

Voice systems:

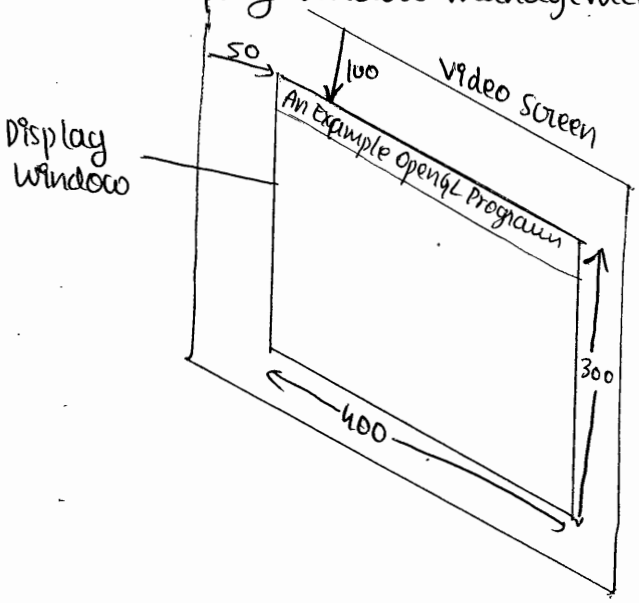
- Speech recognizers are used with some graphics work-stations as input devices for voice commands. The voice system input can be used to initiate operations or to enter data.

Prof Shankar R https://hemanthrajhemu.github.io
https://shankarrajagopal.github.io/

Shankar R
Asst Professor,
CSE, BMSIT&M

7) Explain the display window management using GLUT.

→



- We perform the GLUT initialization with the statement

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

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

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

where the single argument for this function can be any character string that we want to use for the display-window title.

- The following function call passes the line-segment description to the display window

```
glutDisplayFunc (lineSegment);
```

- `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 mouse or keyboard.

- `glutInitWindowPosition (50,100);`

The following statement specifies that the upper-left corner of the display window should be placed 50 pixels to the right of the left edge of the screen and 100 pixels down from the top edge of the screen.

- `glutInitWindowSize (400,300);`

the glutInitWindowSize function is used to set the initial pixel width and height of the display window.

- `glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);`

the following command specifies that a single refresh buffer is to

Prof Shankar R https://hemanthrajhemu.github.io https://shankarrajagopal.github.io/

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.

Shankar R
Asst Professor,
CSE, BMSIT&M

8) What is coordinate reference frames, screen coordinates, absolute and relative coordinates?

→ Coordinate Reference frames:-

- To describe a picture, the world-coordinate reference frame (2D or 3D) must be selected.
- each object has geometrical coordinate as well as other attributes (like color)
- A rectangular region completely enclosing an object is characterized by its coordinate extents. This region is frequently called a bounding box, or in 2D, a bounding rectangle.

Screen Coordinates:-

- Screen coordinates identify pixels on the screen of the display device.
- The "y" coordinate identifies a particular scan line, while the "x" coordinate identifies a particular pixel in the scan line.
- Hardware typically considered the upper left corner as position (0,0), but this is easily transformed by software.
- We will assume the coordinates identify the center of a pixel area.

Absolute and Relative Coordinates:-

- The coordinates we've been using have all been absolute.
- It would be easy, however, to specify a current position and then use coordinates that are relative to that current position.
- for example, if the current position was (5,2), then the relative coordinate (-2,4), would refer to the same pixel as the absolute coordinate (3,6).
- Relative coordinates are useful if we wish to draw several copies of an object with many geometric components. We describe the object using coordinates relative to some point. Then drawing the object several times with different current positions would yield several identical copies.

Prof Shankar R https://hemanthrajhemu.github.io
https://shankarrajagopal.github.io/

Shankar R
Asst Professor,
CSE, BMSIT&M

9. Explain the openGL functions for point and line

Ans: We use glVertex*(): function to state the co-ordinate values for a single position.

Here the asterisk(*) indicates that suffix codes are required for this function

The form for an openGL specification of a point position is

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

coordinates are given as integer pairs

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

Alternatively we can specify points in arrays 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();
```

We can give the co-ordinates as explicit floating-point values

```
glBegin (GL_POINTS);
  glvertex3f(-78.05, 909.72, 14.60);
  glvertex3f(261.91, -5200.67, 188.33);
glEnd();
```

We can also define a C++ class or structure for specifying point positions in various dimensions

```
class wcPt2D{
  public:
    GLfloat x, y;
};
```

Using this class definition, we could specify a 2-D, world co-ordinate point position with the statements

```
wcPt2D pointPos;
pointpos.x = 120.75;
pointpos.y = 45.30;
glBegin (GL_POINTS);
```

Prof Shankar R https://hemanthrajhemu.github.io
https://shankarrajagopal.github.io/

Shankar R
Asst Professor,
CSE, BMSIT&M

```
glVertex2f(pointPos.x, pointPos.y);
glEnd();
```

we can use the openGL point-plotting functions within a C++ procedure to implement the setPixel command.

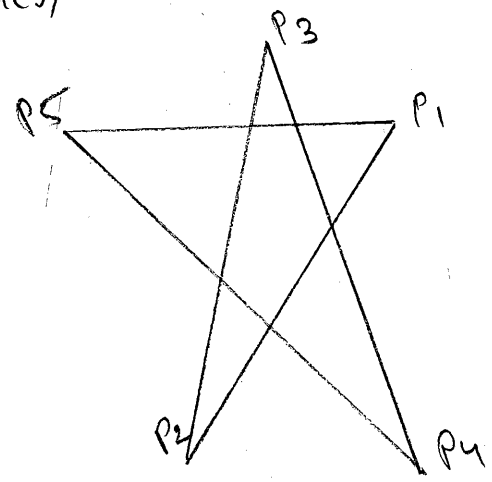
OpenGL line functions:

A set of straight line segments between each successive pair of endpoints in a list is generated using the primitive line constant GL_LINES.

If a set of points are repeated then the figure might result in a closed structure

example:

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



GL_LINE_STRIP is another function used to display a line. The first line segment in the polyline is displayed between the first end-point and second endpoint. The second line segment is between the second endpoint and third endpoint and this goes on so on and so forth.

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

The third openGL line primitive is GL_LINE_LOOP, which produces a closed polyline

example:

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

Prof Shankar R
https://hemanthrajhemu.github.io
https://shankarrajagopal.github.io/

Shankar R
Asst Professor,
CSE, BMSIT&M

10. Explain the DDA line drawing algorithm

The digital differential analyzer (DDA) is a scan-conversion line algorithm based on calculating either δy or δx

$$\delta y = m \cdot \delta x$$

$$\delta x = \frac{\delta y}{m}$$

We consider first a line with positive slope. If the slope is less than or equal to 1, we sample at unit x intervals ($\delta x = 1$) and compute successive y values as

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

Subscript k takes integer values starting from 0, for the first point, and increases by 1 until the final endpoint is reached.

For lines with positive slope greater than 1.0 we reverse the roles of x and y. That is, we sample at until y intervals ($\delta y = 1$) and calculate consecutive x values as

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

each x value is rounded to the nearest pixel

Prof Shankar R https://hemanthrajhemu.github.io
https://shankarrajagopal.github.io/

position along the current y scan line

If the starting endpoint is at the right then either we have $\delta x = -1$ and

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

or we have $\delta y = -1$ with

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

similarly calculations are carried out using $y_{k+1} = y_k + m$ and $x_{k+1} = x_k - \frac{1}{m}$

If the absolute value of the slope is less than 1 and the starting endpoint is at the left we set $\delta x = 1$ and calculate y values with $y_{k+1} = y_k + m$

When the starting endpoint is at the right we set $\delta x = -1$ and obtain y positions using $y_{k+1} = y_k - m$. For a negative slope with absolute value greater than 1 we use $\delta y = -1$ and $x_{k+1} = x_k - \frac{1}{m}$ or we use $\delta y = 1$ and $x_{k+1} = x_k - \frac{1}{m}$

The algorithm is summarized as following:

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

Prof Shankar R https://hemanthrajhemu.github.io
https://shankarrajagopal.github.io/

Shankar R
Asst Professor,
CSE, BMSIT&M

```
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));
    }
}
```

Prof Shankar R https://hemanthrajhemu.github.io
https://shankarrajagopal.github.io/

Shankar R
Asst Professor,
CSE, BMSIT&M

Q. Explain Bresenham's Line-Drawing algorithm.

Ans:- 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.

→ A computed line positions of (10.48, 20.51) is converted to pixel position (10, 21). This rounding of coordinates values to integers causes all but horizontal and vertical lines to be displayed with a stair-step appearance.

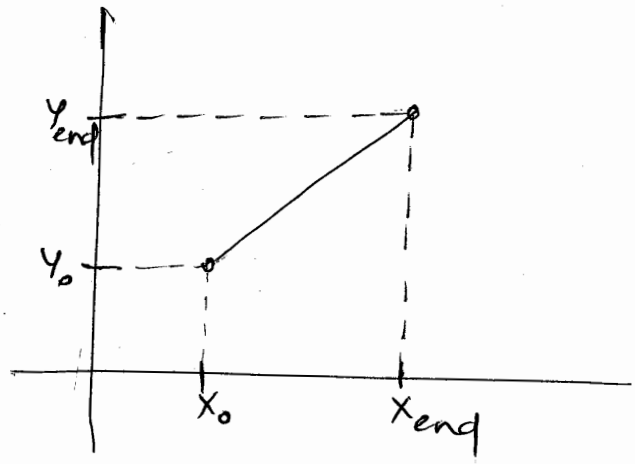
Line Equations:

⇒ The cartesian slope-intercept equation for a straight line is

$$y = mx + c \rightarrow (1)$$

with m as the slope of the line and c as the y-intercept.

⇒ Given that the two endpoints of a line segment are specified at positions (x_0, y_0) and (x_{end}, y_{end}), as shown in fig.



⇒ We determine values for the slope m and c with the following equations:

$$m = \frac{y_{end} - y_0}{x_{end} - x_0} \rightarrow (2)$$

$$c = y_0 - m x_0 \rightarrow (3)$$

⇒ we consider $m=1$, $m<1$ and $m>1$ cases:

(i) $m<1$

= x increments unitwisely

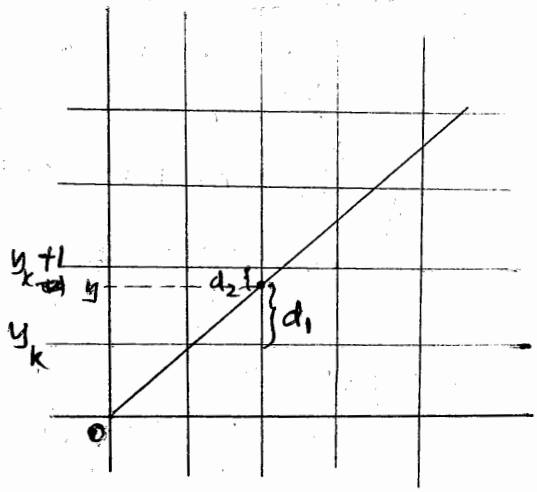
i.e., $x_{k+1} = x_k + 1$

then, $m = \frac{(y_{k+1} - y_k)}{(x_{k+1} - x_k)}$

or, $m = y_{k+1} - y_k$

or, $y_{k+1} = y_k + m \rightarrow (4)$

and $\boxed{y = m(x_k+1) + c} \rightarrow (5)$



$\Rightarrow d_1 = y - y_k$

sub. eqn (5) in d_1

$\boxed{d_1 = m(x_k+1) + c - y_k}$

$\Rightarrow d_2 = y_{k+1} - y$

$\boxed{d_2 = y_k - m(x_k+1) - c + 1}$

⇒ To determine the value from (y_k+1, y_k), we calculate decision parameter, (P_k)

$$\boxed{P_k = \Delta x(d_1 - d_2)} \rightarrow (6)$$

$\Rightarrow d_1 - d_2 = m(x_k+1) + c - y_k - y_k + m(x_k+1) + c - 1$

or, $d_1 - d_2 = 2m(x_k+1) - 2y_k + 2c - 1$

$\therefore P_k = \Delta x(2m(x_k+1) - 2y_k + 2c - 1)$

or, $\boxed{P_k = 2m\Delta x(x_k+1) - 2\Delta x y_k + 2\Delta x c - \Delta x} \rightarrow (7)$

↳ $P_k = 2\Delta y(x_k+1) - 2\Delta x y_k + 2\Delta x c - \Delta x$

⇒ P_k is the initial decision parameter in equation 7 and in order to find continuous decisions, we have to find the next P_k.

i.e., $\boxed{P_{k+1} = 2\Delta y(x_{k+1}+1) - 2\Delta x y_{k+1} + 2\Delta x c - \Delta x} \rightarrow (8)$

Prof Shankar R https://hemanthrajhemu.github.io https://shankarrajagopal.github.io/

Shankar R
Asst Professor,
CSE, BMSIT&M

⇒ For next decision parameter, we will be mathematically take differences between p_{k+1} in eqn ⑥ and p_k in eqn ⑦

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

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

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

and the preceeding decision parameter is

$$\boxed{p_{k+1} = p_k + 2\Delta y - 2\Delta x(y_{k+1} - y_k)} \rightarrow (9)$$

⇒ we know that the initial point to be plotted is (x_k, y_k) and let us substitute (x_k, y_k) in the initial decision parameter (p_k).

⇒ equation ⑦ becomes:

$p_k = 2\Delta y(x_k + 1) - 2\Delta x y_k + \Delta x(2(y - mx) - 1)$

or, $p_k = 2\Delta y(x_k + 1) - 2\Delta x y_k + \Delta x(2y - 2mx - 1)$

or, $p_k = 2\Delta y(x_k + 1) - 2\Delta x y_k + \Delta x(2y_k - 2mx_k - 1)$

or, $p_k = 2\Delta y(x_k + 1) - 2\Delta x y_k + \Delta x(2y_k) - 2\frac{\Delta y}{\Delta x}(\Delta x)x_k - \Delta x$

or, $p_k = 2\Delta y x_k + 2\Delta y - 2\Delta x y_k + 2\Delta x y_k - 2\Delta y x_k - \Delta x$

or, $\boxed{p_k = 2\Delta y - \Delta x} \rightarrow (10)$

Note: apply above p_k only once initially.

Conclusion :-

```
if (Pk ≥ 0)
  {
    x_{k+1} = x_k + 1
    y_{k+1} = y_k + 1
  }
else
  {
    x_{k+1} = x_k + 1
    y_{k+1} = y_k
  }
```

Case 2 : $m > 1$ → Y increments unitwisely i.e, $x_{k+1} = x_k + 1$

Similarly for $m > 1$ case and

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

$P_{k+1} = P_k + 2\Delta x - 2\Delta y (x_{k+1} - x_k)$

Conclusion :-

```
if (Pk ≥ 0)
  {
    y_{k+1} = y_k + 1
    x_{k+1} = x_k + 1
  }
else
  {
    y_{k+1} = y_k + 1
    x_{k+1} = x_k
  }
```

Case 3 : $m = 1$ → both x & y increments unitwisely.

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

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

Prof Shankar R https://hemanthrajhemu.github.io https://shankarrajagopal.github.io/

Shankar R
Asst Professor,
CSE, BMSIT&M

29. Explain the Midpoint circle algorithm.

Ans: Midpoint circle algorithm generates all points on a circle centered at the origin by incrementing all the way around circle.

→ The strategy is to select which of 2 pixels is closer to the circle by evaluating a function at the midpoint between the 2 pixels.

Eight way symmetry

The shape of the circle is similar in each quadrant. Therefore, if we determine the curve positions in the first quadrant, we can generate the circle positions in the second quadrant of my plane. The circle sections in the third and fourth quadrant can be obtained from sections in the 1st and 2nd quadrant by considering the symmetry along x-axis.

⇒ In this algorithm by evaluating a function at the midpoint between the 2 pixel, we can decide which pixel is closer to the circle.

⇒ Suppose the initial point is (x_k, y_k) and next point closer to it could be either (x_k+1, y_k) or (x_k+1, y_k-1)

⇒ Mid-point $= \left(\frac{x_k+1+x_k+1}{2}, \frac{y_k+y_k-1}{2}\right)$

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

⇒ As we have circle equation as

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

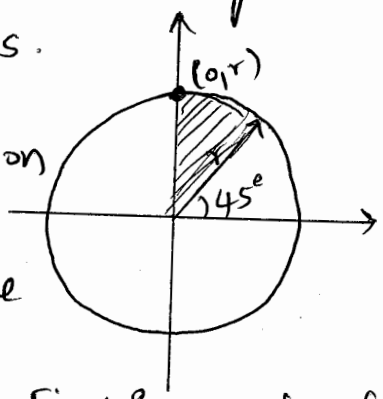


Fig: Symmetry of a circle. Calculation of a circle point (x,y) in one octant yields the circle points shown for the other seven octants.

Prof Shankar R https://hemanthrajhemu.github.io https://shankarrajagopal.github.io/

$$P_k = (x_k+1)^2 + (y_k - y_2)^2 - r^2 \rightarrow (1)$$

⇒ initial decision parameter and next could be

$$P_{k+1} = (x_{k+1}+1)^2 + (y_{k+1} - y_2)^2 - r^2$$

or,

$$P_{k+1} = (x_k+2)^2 + (y_k - y_2)^2 - r^2 \rightarrow (2)$$

⇒ TO get continuous decisions, next P_k could be

$$P_{k+1} - P_k = 2(x_k+1) + (y_{k+1}^2 - y_k^2) - (y_{k+1} - y_k) + 1$$

⇒ initial decision parameter at $(0,r)$ is

$$P_k = (0+1)^2 + (r - y_2)^2 - r^2$$

$$\text{or, } P_k = 1 + r^2 - r + \frac{1}{4} - r^2$$

$$\text{or, } P_k = \frac{5}{4} - r$$

$$\text{or, } P_k = 1.25 - r$$

$$\text{or } P_k = 1 - r$$

Conclusion:

```
if (Pk ≥ 0)
  { xk+1 = xk + 1
    yk+1 = yk - 1
  }
else
  { xk+1 = xk + 1
    yk+1 = yk
  }
```

Prof Shankar R
https://hemanthrajhemu.github.io
https://shankarrajagopal.github.io/

Shankar R
Asst Professor,
CSE, BMSIT&M

13. Explain point attribute Functions.

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.

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

We set the size for an OpenGL point with

```
glPointSize (size);
```

and the point is then displayed as a square block of pixels. Parameter size is assigned a positive floating-point value, which is rounded to an integer (unless the point is to be antialiased). 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 2x2 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 values for point size is 1.0.

Attribute functions may be listed inside or outside of a glBegin/glEnd pair. For 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:

Prof Shankar R https://hemanthrajhemu.github.io https://shankarrajagopal.github.io/

Shankar R
Asst Professor,
CSE, BMSIT&M

```
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();
```

14) List the OpenGL Line attributes Functions.

We can control the appearance of a straight-line segment in openGL, with three attribute settings.

1. Line color
2. line width
3. line style

OpenGL Line-width Function:

Line width is set in openGL with the function

```
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. However, when antialiasing is applied to the line, it's edges are smoothed to reduce the raster stair-step appearance and fractional

Prof Shankar R https://hemanthrajhemu.github.io https://shankarrajgopal.github.io/

Shankar R
Asst Professor,
CSE, BMSIT&M

width are possible. Some implementations of the line width function might support only a limited number of widths and some might not support widths other than 1-0.

OpenGL Line-Style Function

By default, a straightline segment is displayed as a solid line. However, we can also display dashed lines, dotted lines, or a line with a combination of dashes and dots, and 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.

```
glLineStipple (repeatFactor, pattern);
```

Before a line can be displayed in the current line-style pattern, we must activate the line-style feature of OpenGL. We accomplish with the following function.

```
glEnable (GL-LINE-STIPPLE);
```

If we forget to include this enable function, solid lines are displayed; that is the default pattern OXFFFF is used to display line segments. At any time, we can turn-off the line-pattern feature with

```
glDisable (GL-LINE-STIPPLE);
```

This replaces current line-style pattern with the default pattern.

Prof. Shankar R https://hemanthrajhemu.github.io
https://shankarrajagopal.github.io/

Shankar R
Asst Professor,
CSE, BMSIT&M

(15) List & explain OpenGL point & line primitive.

→ OpenGL primitives are displayed with default size & color. Default color: white & Default point size is equal to size of single screen pixel.

→ glVertex*(); → used to state coordinate values for single position.

→ Calls to glVertex functions must be placed betn glBegin() & glEnd().

→ glBegin() is used to identify kind of o/p primitive i.e. to be displayed & glEnd takes no argument.

→ For point plotting, glBegin() is symbolic constant GL_POINTS.

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

→ glVertex() specifies coordinates for any point position. The 2nd suffix code is used to specify numerical values of coordinates.

(Eg)
```
#include<GL/glut.h>
void display()
{ glPointSize(20); glBegin(GL_POINTS);
  glVertex2f(-0.5,-0.5); glVertex2f(0.5,0.5);
  glEnd(); glFlush(); }
void main(int argc, char **argv)
{ glutInit(&argc, argv); glutCreateWindow("4 Points");
  glutDisplayFunc(display); glutMainLoop(); }
```

Prof Shankar R https://hemanthrajhemu.github.io
https://shankarrajagopal.github.io/

→ The displayed color of designated point position is controlled by current color values in state list. Color is specified with glColor().

→ Set the size for opengl point with glPointSize(size);

→ Graphics pkgs provide function for specifying 1/more straight line segments where each line segment is defined by 2 endpoints coordinate positions.

→ GL_LINES:
A set of straight-line segments o/p each successive pair of endpoints in list is generated using this primitive line constant.

→ (eg) glBegen(GL_LINES);
glVertex2iv(p1); glVertex2iv(p2);
glVertex2iv(p3); glVertex2iv(p4);
glVertex2iv(p5); glEnd();

→ GL_LINE_STRIP:
Display is a sequence of connected line segments o/p 1st endpoint in list & last endpoint which is polyline.

(eg) glBegen(GL_LINE_STRIP);
glVertex2iv(p1); glVertex2iv(p2); glVertex2iv(p3);
glVertex2iv(p4); glVertex2iv(p5);
glEnd();

2.

Prof Shankar R https://hemanthrajhemu.github.io
https://shankarrajagopal.github.io/

Shankar R
Asst Professor
CSE, BMSITM

→ GL_LINE_LOOP:

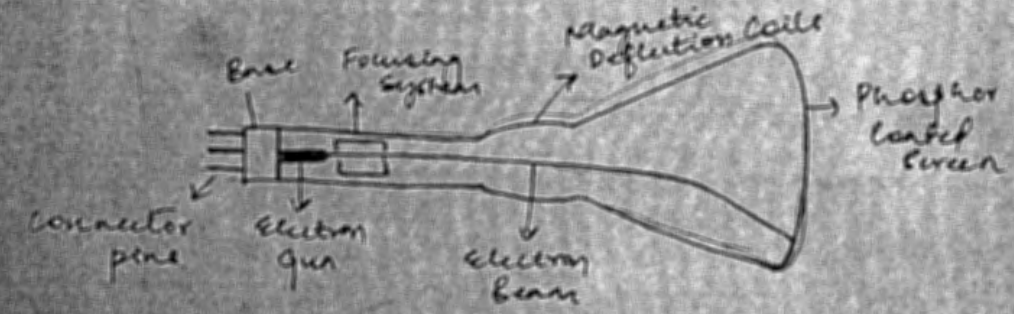
An additional line is drawn to connect last coordinate position & first position which produces closed polyline.

(eg) glBegin(GL_LINE_LOOP);
glVertex2iv(p1); glVertex2iv(p2); glVertex2iv(p3);
glVertex2iv(p4); glVertex2iv(p5); glEnd();

→ We can control appearance of straight line segment in OpenGL with 3 attributes settings: line color, line width & line style.

(16) Explain Cathode Ray Tube with diagram.

→ Operation of most video monitors is based on standard cathode Ray Tube (CRT).



Magnetic Deflection CRT

→ Beam of electrons emitted by electron gun passes through focusing & deflection system

Prof Shankar R https://hemanthrajhemu.github.io
https://shankarrajagopal.github.io/

that direct beam towards specified positions on phosphour coated screen.

→ Phosphor then emits small spot of light at each position contacted by e^- beam & light emitted fades very rapidly.

→ One way to maintain screen picture is to store picture information as charge distribution with CRT in order to keep phosphorous activated.

→ Common method in maintaining glow is to redraw picture repeatedly by quickly directing e^- beam back over same screen points. This type of display is called refresh CRT.

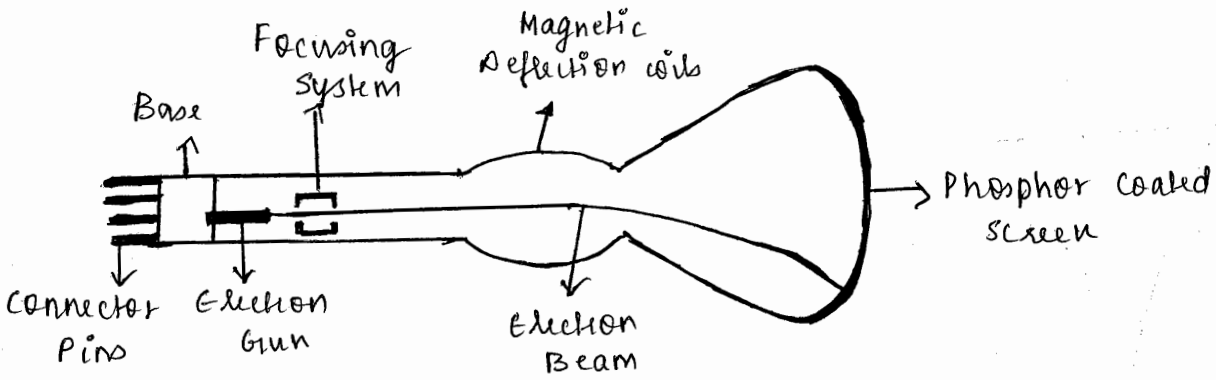
→ Frequency at which picture is redrawn on screen is known as Refresh Rate.

Prof Shankar R https://hemanthrajhemu.github.io
https://shankarrajagopal.github.io/

Shankar R
Asst Professor,
CSE, BMSIT&M

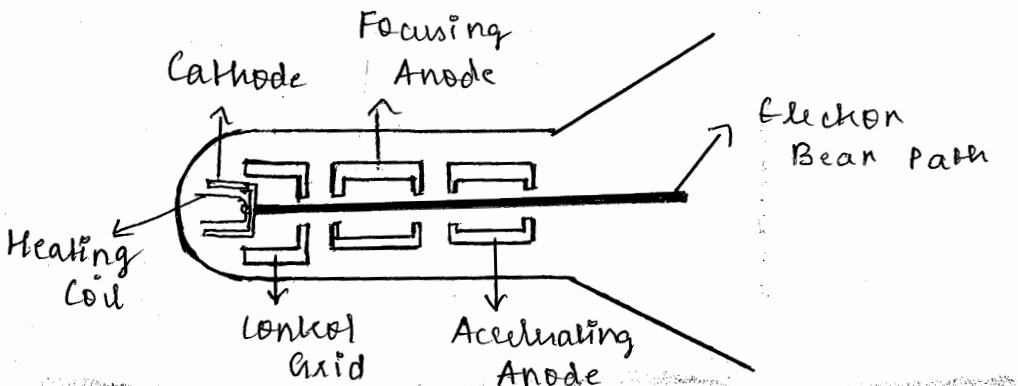
17) With a neat diagram, explain Refresh Cathode Ray Tubes.

Any.



- 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 light at each position contacted by 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 <u>refresh CRT</u>.
- The frequency at which a picture is redrawn on the screen is refered to as the <u>refresh rate</u>.

→ <u>Operation of an electron gun with an accelerating anode</u>



42 48

- The primary components of an electron gun in CRT are the heated metal cathode and a control grid.
- The heat 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 envelop, 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 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 transverse 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 pairs of coils.

Prof Shankar R https://hemanthrajhemu.github.io
https://shankarrajagopal.github.io/

18) Implement an OpenGL program for Bresenham's line drawing algorithm.

Shankar R
Asst Professor,
CSE, BMSIT&M

Soln.

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

1. Input the two line endpoints and store the left endpoint in (x_0, y_0).
2. Set the color for frame-buffer position (x_0, y_0); 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.

- An implementation of Bresenham's line algorithm for slopes in the range $0 < m < 1.0$ is given in the following procedure. Endpoint pixel position for the line are passed to this procedure, and pixels are plotted from the left endpoint to the right endpoint.

i)

```
#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;
```

Prof Shankar R https://hemanthrajhemu.github.io https://shankarrajagopal.github.io/

```
/* 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);
}
}
```

(or)

ii)

```
#include <GL/glut.h>
#include <stdio.h>
int x1, y1, x2, y2;
void draw_pixel(int x, int y)
{
    glColor3f(1.0, 0.0, 0.0);
    glBegin(GL_POINTS);
        glVertex2i(x, y);
    glEnd();
}
```

Prof Shankar R https://hemanthrajhemu.github.io https://shankarrajagopal.github.io/

Shankar R
Asst Professor,
CSE, BMSIT&M

```
void bresenhams_line_draw (int x1, int y1, int x2,
                           int y2)
{ int dx = x2 - x1;
  int dy = y2 - y1;
  int m = dy / dx;
  if (m < 1)
    { int decision_parameter = 2 * dy - dx;
      int x = x1;
      int y = y1;
      if (dx < 0)
          { x = x2;
            y = y2;
            x2 = x1;
          }
      draw_pixel(x,y);
      while (x < x2)
        {
          if (decision_parameter >= 0)
              { x = x + 1;
                y = y + 1;
                decision_parameter = decision_parameter +
                                     2 * dy - 2 * dx * (y+1-y);
              }
          else
           { x = x + 1;
             y = y;
             decision_parameter = decision_parameter + 2 * dy
                                  - 2 * dx * (y-y);
           }
          draw_pixel (x,y);
        }
    }
  else if (m > 1)
    { int decision_parameter = 2 * dx - dy;
```

Prof Shankar R https://hemanthrajhemu.github.io
https://shankarrajagopal.github.io/

```
int x = x1;
int y = y1;
if (dy < 0)
   { x = x2;
     y = y2;
     y2 = y1;
   }
draw_pixel (x,y);
while (y < y2)
 { if (decision_parameter >= 0)
     {
        x = x + 1;
        y = y + 1;
        decision_parameter = decision_parameter + 2 * dx -
                             2 * dy * (x+1-x);
     }
   else
     { y = y + 1;
       x = x ;
       decision_parameter = decision_parameter + 2 * dx -
                            2 * dy * (x-x);
     }
   draw_pixel (x,y);
 }
}
else if (m == 1)
 { int x = x1;
   int y = y1;
   draw_pixel (x, y);
   while (x < x2)
     {
        x = x + 1;
        y = y + 1;
        draw_pixel (x,y);
     }
 }
}
```

Prof Shankar R
https://hemanthrajhemu.github.io
https://shankarrajagopal.github.io/

Shankar R
Asst Professor,
CSE, BMSIT&M

```
void init ()
 {
    glClearColor (1,1, 1, 1);
    gluOrtho2D (0.0, 500.0, 0.0, 500.0);
 }
void display()
 {
    glClear (GL_COLOR_BUFFER_BIT);
    bresenhams_line_draw (x1, y1, x2, y2);
    glFlush();
 }
int main (int argc, char ** argv)
{
   printf ("Enter start points (x1, y1)\n");
   scanf ("%d %d", &x1, &y1);
   printf ("Enter end points (x2, y2)\n");
   scanf ("%d %d", &x2, &y2);
   glutInit (&argc, argv);
   glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
   glutInitWindowSize (500,500);
   glutInitWindowPosition (220,200);
   glutCreateWindow ("Bresenhams Line Drawing");
   init();
   glutDisplayFunc (display);
   glutMainLoop();
}
```

Shankar R
Asst Professor,
CSE, BMSIT&M

Q19. Write short note on basic OpenGL Syntax.

Ans:- Basic Syntax :-

- In order to write our first OpenGL program, there are some things that we should know.
- All functions in OpenGL use the prefix "gl"
- Functions from GLU and GLUT have the prefixes "glu" and "glut" respectively.
- All constants in OpenGL use the prefix "GL" i.e glBegin(GL_POLYGON);

Code example:-

```
int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
    glutInitWindowSize(500,500);
    glutInitWindowPosition(220,200);
    glutCreateWindow("Bresenham's Line Drawing");

    init();
    glutDisplayFunc(display);
    glutMainLoop();
}
```

Prof Shankar R
https://hemanthrajhemu.github.io
https://shankarrajagopal.github.io/

Q20, Explain properties of Circle.

Shankar R
Asst Professor,
CSE, BMSIT&M

Ans:- Propertise of the Circle in computer graphics.

i) Circle is defined as a set of points that are all at a given distance r from a centre position (x_c, y_c).

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

iii) Bresenham's line algorithm for raster display is adapted to circle generation by setting up the decision parameters for finding the closet pixel for each sampling step.

iv) 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

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

v) Expressing circle equation in parametric polar form yields the pair of equations

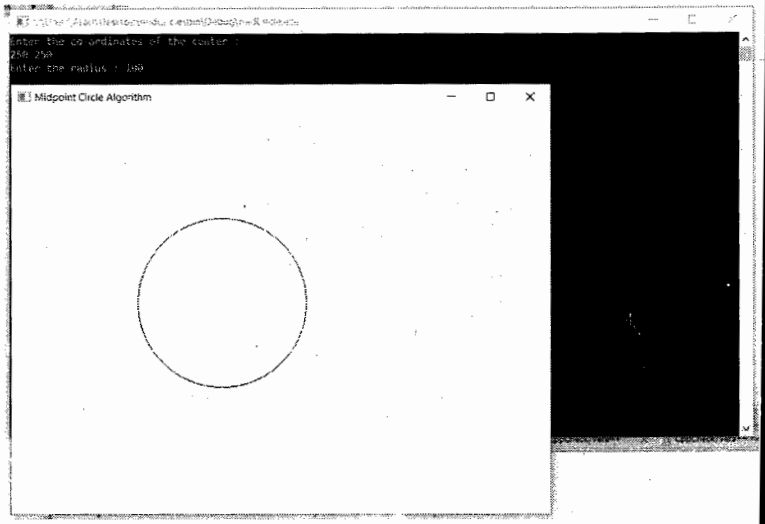
$$x = x_c + r\cos\theta$$
$$y = y_c + r\sin\theta$$

Prof Shankar R https://hemanthrajhemu.github.io https://shankarrajagopal.github.io/

21. Implement midpoint circle draw in OpenGL.

Shankar R
Asst Professor,
CSE, BMSIT&M

```
#include<stdio.h>
#include<GL/glut.h>
int  x1, y1, r;
void draw_pixel(int x, int y)
{
      glBegin(GL_POINTS);
      glVertex2i(x+x1, y+y1);
      glEnd();
}
void init()
{
     glClearColor(1.0,1.0,1.0,0.0);
     glColor3f(0.0, 0.0, 0.0);
     glPointSize(4.0);
     glMatrixMode(GL_PROJECTION);
     glLoadIdentity();
     gluOrtho2D(0.0, 640.0, 0.0, 480.0);
}
void midCircleAlgo()
{
     int x=0;
     int y=r;
     float decision = 5/4 - r;
     draw_pixel(x, y);
     while(y > x)
     {
          if(decision < 0)
          {
               x++;
               decision += 2*x + 1;
          }
          else
```



Prof Shankar R https://hemanthrajhemu.github.io
https://shankarrajagopal.github.io/

Shankar R
Asst Professor,
CSE, BMSIT&M

```
        {
            y--;
            x++;
            decision += 2*(x-y)+1;
        }
        draw_pixel(x,y);
        draw_pixel(x, -y);
        draw_pixel(-x, y);
        draw_pixel(-x,-y);
        draw_pixel(y, x);
        draw_pixel(-y, x);
        draw_pixel(y,-x);
        draw_pixel(-y,-x);
    }
}
void display()
{
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(0.0, 0.0, 0.0);
    glPointSize(1.0);
    midCircleAlgo();
    glFlush();
}
void main(int argc, char** argv)
{
    printf("Enter the co-ordinates of the center:\n");
    scanf("%d %d", &x1, &y1);
    printf("Enter the radius: ");
    scanf("%d", &r);
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(640, 480);
    glutInitWindowPosition(100, 150);
    glutCreateWindow("Midpoint Circle Algorithm");
    init();
    glutDisplayFunc(display);
    glutMainLoop();
}
```

Prof Shankar R https://hemanthrajhemu.github.io
https://shankarrajagopal.github.io/

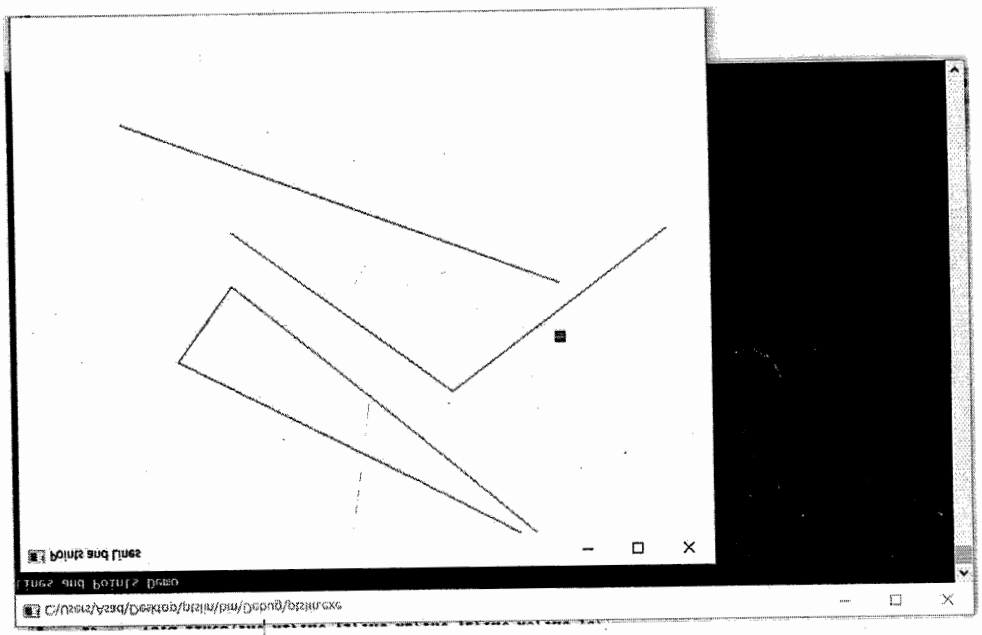
Shankar R
Asst Professor,
CSE, BMSIT&M

22. Implement an OpenGL program to display points and lines along with its attribute functions included.

```
#include<stdio.h>
#include<GL/glut.h>
void point(int x, int y)
{
    glBegin(GL_POINTS);
    glVertex2i(x, y);
    glEnd();
}
void init()
{
    glClearColor(1.0,1.0,1.0,1.0);
    glColor3f(0.0,0.0,0.0);
    glPointSize(10.0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0,640.0,0.0,480.0);
}
void lines(int x1, int y1, int x2, int y2, int x3, int y3)
{
    glVertex2i(x1, y1);
    glVertex2i(x2, y2);
    glVertex2i(x3, y3);
    glEnd();
}
void display()
{
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1.0, 0.0, 0.0);
    glPointSize(1.0);
    point(   ,   );
```

Shankar R
Asst Professor,
CSE, BMSIT&M

```
        glBegin(GL_LINES);
        Lines(
        glBegin(GL_LINE_STRIP);
        Lines(
        glBegin(GL_LINE_LOOP);
        Lines(
        glFlush();
}
void main(int argc, char** argv)
{
        printf("Lines and Points Demo");
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
        glutInitWindowSize(640, 480);
        glutInitWindowPosition(100, 150);
        glutCreateWindow("Points and Lines");
        init();
        glutDisplayFunc(display);
        glutMainLoop();
}
```



Prof Shankar R https://hemanthrajhemu.github.io
https://shankarrajagopal.github.io/

Shankar R
Asst Professor,
CSE, BMSIT&M

Q.23) Bressenham's Line drawing Algorithm for $|m|<1.0$. Digitize the line with end points (20,10) (30,18).

Ans: Algorithm is given as:

Step 1: Declaration of variables

x1, y1 – start points
x2, y2 – end points
x, y – current point
p – decision parameter
dx, dy – difference in X & Y coordinates

Step 2: Calculation

$dx = x2 - x1$

$dy = y2 - y1$

$p = 2dx - dy$ (initially)

Step 3: Initialization

$x = x1$

$y = y1$

Step 4: Plotting the points

if ($p < 0$)

$p = p + 2dx$

$x = x$

$y = y + 1$

if ($p > 0$)

$p = p + 2dx - 2dy$

$x = x + 1$

$y = y + 1$

Prof Shankar R https://hemanthrajhemu.github.io https://shankarrajagopal.github.io/

Example: (20,10) to (30,18)

Shankar R
Asst Professor,
CSE, BMSIT&M

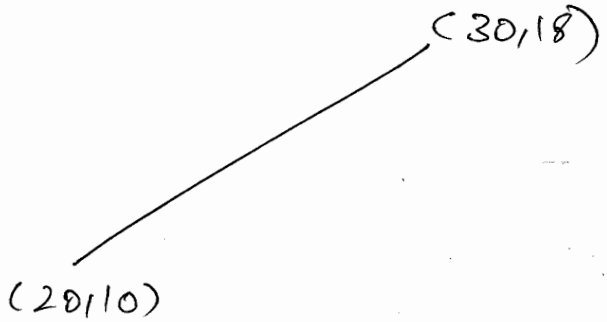
$dx = 10$

$dy = 8$

$2dy - dx = 6$

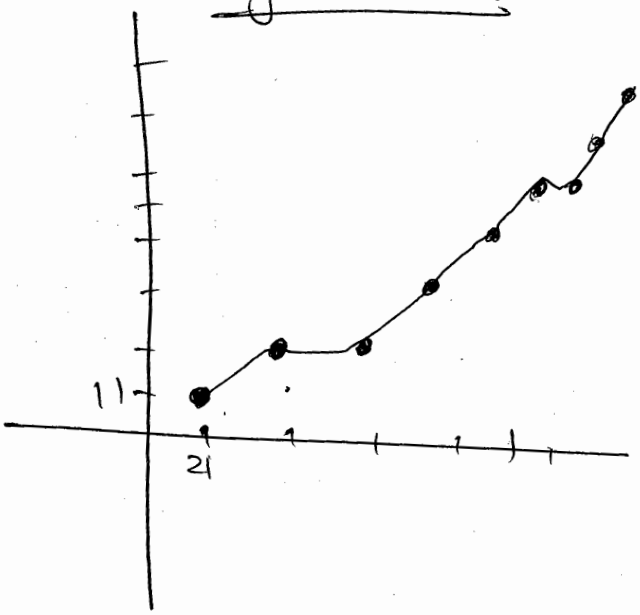
$2dy = 16$

$2(dy - dx) = -4$



i	di	(x_{i+1}, y_{i+1})
0	6	(21,11)
1	2	(22,12)
2	-2	(23,12)
3	14	(24,13)
4	-6	(25,14)
5	2	(26,15)
6	-2	(27,16)
7	14	(28,16)
8	10	(30,18)

Digitization:



Prof Shankar R https://hemanthrajhemu.github.io
https://shankarrajagopal.github.io/

Shankar R
Asst Professor,
CSE, BMSIT&M

Q.24) Given a circle with radius = 10, demonstrate the midpoint circle algorithm by determining the positions along circle octact with first quadrant x=0 to x=y (Assume circle centre is positioned at origin)

ANS As we know circle is symmetric.
We can just plot the values in the first quadrant.

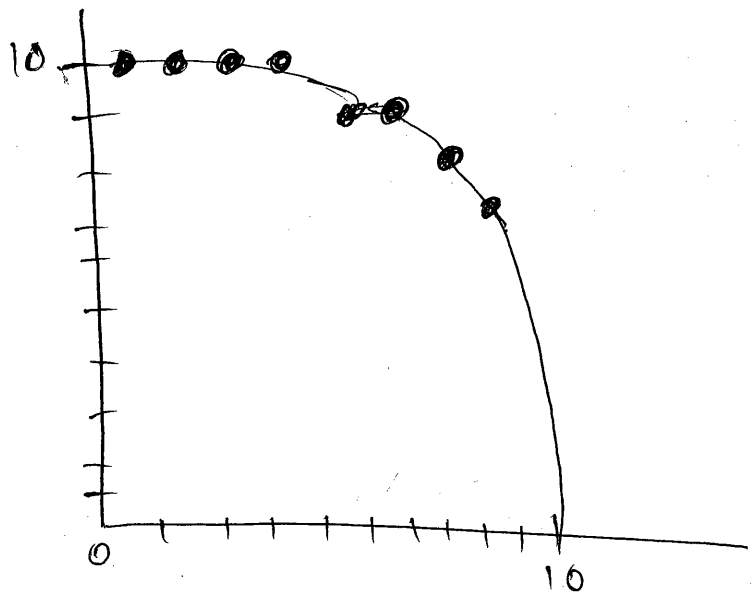
Initial decision parameter = $5/4 - r$

if ($p_k < 0$)

$x = x_k + 1$

$y = y_k$

$p_{k+1} = p_k + 2x_k + 1 + 1 - 2y_k + 1$.



Prof Shankar R
https://hemanthrajhemu.github.io
https://shankarrajagopal.github.io/

Bresenham's Circle Algorithm

Shankar R
Asst Professor,
CSE, BMSIT&M

(r=10, h, k=100)

Step no.	Number	x	y	decision parameter
	1	0	10	-8.75
	2	1	10	-4.75
	3	2	10	1.25
	4	3	9	-9.75
	5	4	9	0.25
	6	5	8	-4.75
	7	6	8	9.25
	8	7	7	10.25

Prof Shankar R https://hemanthrajhemu.github.io
https://shankarrajagopal.github.io/

Shankar R
Asst Professor,
CSE, BMSIT&M

25) Apply Bresenham's Line drawing algorithm for the given end points.

(a) (30, 20) and (40, 28) (b) (0,0) to (5,4) (c) (0,0) to (5,6)

Ans:- (a) (30, 20) and (40, 28)

$$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{28-20}{40-30} = \frac{4}{5} = 0.8 < 1.$$

dp: $2dy - dx = 6$., applying the formulae if $dp > 0$

$dp = dp + 2dy - 2dx(y+1-y)$ else $dp = dp + 2dy$.

x'	y'	x	y	decision parameter
		30	20	6
31	21	31	21	2
32	22	32	22	-2
33	22	33	22	14
34	23	34	23	10
35	24	35	24	6
36	25	36	25	2
37	26	37	26	-2
38	26	38	26	14
39	27	39	27	10
40	28	40	28	6

(b) (0,0) to (5,4)

$$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{4-0}{5-0} = \frac{4}{5} = 0.8 < 1$$

$$dp = 2dy - dx = 2(4) - 5 = 3.$$

going further,

if $dp > 0$:- decision parameter = decision parameters $+ 2dy - 2dx(y+1-y)$

else if $dp < 0$:- $dp = dp + 2dy - 2dx(y-y)$

Prof Shankar R https://hemanthrajhemu.github.io
https://shankarrajagopal.github.io/

Shankar R
Asst Professor,
CSE, BMSIT&M

x'	y'	x	y	decision parameter
-	-	0	0	3
1	1	1	1	1
2	2	2	2	-1
3	2	3	2	7
4	3	4	3	5
5	4	5	4	end

(c) (0,0) to (5,6).

$$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{6-0}{5-0} = \frac{6}{5} \; ; \; 1.2 > 1$$

$$dp = 2dx - dy = 2(5) - 6 = 4.$$

applying the formula

if $dp > 0$: $dp = dp + 2dx - 2dy(x+1-x)$; $x = x+1$

else $dp < 0$: $dp = dp + 2dx - dy(x-x)$ $x = x$

x'	y'	x	y	decision parameter
-	-	0	0	4
1	1	1	1	2
2	2	2	2	0
3	3	3	3	-2
3	4	3	4	8
4	5	4	5	6
5	6	5	6	end