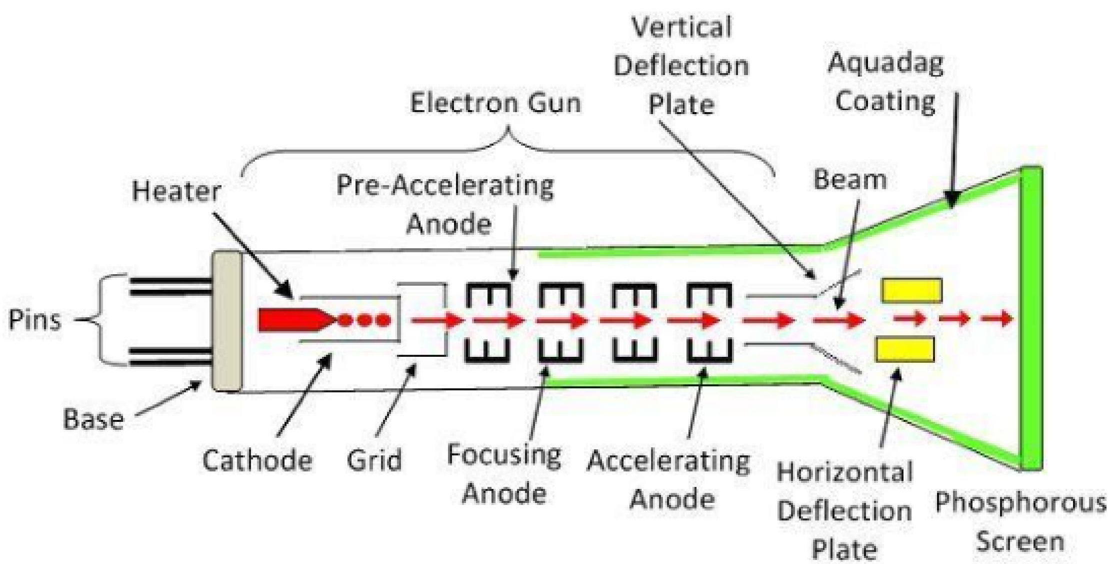
Computer-Graphics-Module-1-Important-Topics

For more notes visit

https://rtpnotes.vercel.app

1. CRT Working

- The CRT is a display screen which produces images in the form of the video signal.
- It is a type of vacuum tube which displays images when the electron beam through electron guns are strikes on the phosphorescent surface.



How CRT Works

- **Electron Gun**: This is the starting point. The electron gun generates a stream of electrons (tiny negatively charged particles) by heating a filament.
- **Beam Creation and Acceleration**: The electrons are formed into a narrow beam and accelerated towards the screen.
-
  - **Deflection System**: This system uses electric or magnetic fields to steer the electron

beam so it can move back and forth across the screen.

- - **Phosphorescent Screen**: The inside of the screen is coated with phosphor, a material that glows when hit by the electron beam, creating the visible image.
- The movement of the electron beam back and forth and up and down on the screen paints the image we see.

Main Components of a CRT

- **Electron Gun**:
  - **Heater and Cathode**: Produces electrons through heating (thermionic emission).
- **Control Grid**: Regulates the number of electrons, affecting the brightness of the image.
- **Focusing System**: Focuses the electron beam into a fine point to create a sharp image.
- **Deflection Yoke**:
  - Creates magnetic or electric fields to steer the electron beam.
  - Controls where the beam hits the screen, forming the image.
- **Phosphorescent Screen**:
  - - Coated with phosphor.
  - - Glows when hit by the electron beam, creating the image.

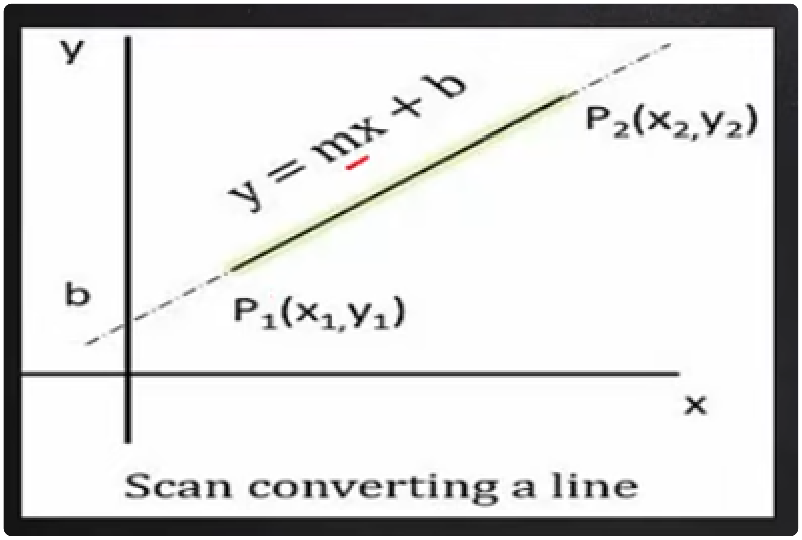
Principle of Operation

- **Cathode and Anode**: When high voltage is applied, the cathode (negative plate) emits electrons. These electrons are attracted to the anode (positive plate) and pass through it, continuing towards the screen.
- **Phosphor Coating**: When the electron beam hits the phosphor coating on the screen, it lights up, creating an image.
- **Deflection**: The direction of the electron beam can be controlled by magnetic or electric fields, allowing the beam to be directed to different parts of the screen to create the full image.

---

2. DDA,Bresenham's line drawing problem

- DDA Means Digital differential analyser
- We have 2 line drawing algorithm, Bressenhams and DDA Line drawing

- Both are based on straight lines **slope intercept form**
    - y = mx + b
    - m = slope
    - b = y intercept
- **How to find slope?**
    - 

    - Here there are 2 points
        - P1(x1, y1)
        - P2(x2,y2)
        - The slope **m = y2 - y1 / x2 - x1**
        - b = y1 - m * x1
        - m can be written as m = dy / dx
            - where dy = y2 - y1 and dy = x2 - x1

DDA Line drawing algorithm

m = dy/dx
dy = m.dx
dx = dy/m

Case 1: If slope (m) is less than or equal to 1

- When m is less than 1, it means
    - dy/dx < 1
    - dy < dx
- We have to find the next x point (xk + 1) and the corresponding y point

- $$x_{k+1} = x_k + 1$$
  $$y_{k+1} = y_k + m \quad \ldots\ldots\ldots\ldots\ldots(eq3)$$

Case 2: If slope (m) is greater than 1

- When m is greater than 1 it means
  - dy/dx > 1
  - dy > dx
- We have to find the next y point (yk + 1) and corresponding x
  - $$y_{k+1} = y_k + 1$$
    $$x_{k+1} = x_k + 1/m \quad \ldots\ldots\ldots\ldots\ldots(eq4)$$

Algorithm

```
#define Round(a)((int)(a+0.5))

void lineDDA(int xa, int ya, int xb, int yb) {
        int x = xb - xa, dy = yb - ya, steps, k;
        float xIncrement, yIncrement, x = ra, y=ya;
        if(abs(dx)>abs(dy))
                steps = abs(dx);
        else
                steps = abs(dy);
        xIncrement = dx/(float)steps;
        yIncrement = dy/(float)steps;

        setPixel(Round(x),Round(y));
        for(k = 0; k<steps;k++){
                x = x + xIncrement;
                y = y + yIncrement;
                setPixel(Round(x),Round(y));
        }
}
```

Problem

Draw line from (1,1) to (8,7) using DDA Algorithm

- Lets check the coordinates
  - y2 = 7
  - y1 = 1
  - x2 = 8
  - x1 = 1
- Based on the coordinates, lets calculate the slope
  - Slope m = (y2 - y1) / (x2 - x1) = (7 - 1) / (8 - 1)
  - Slope m = 6 / 7 = 0.9 < 1
- Which case does this satisfy?
  - It satisfies **Case 1: If slope (m) is less than or equal to 1**
  - increment $x_{k+1}=x_k+1$ and $y_{k+1}=y_k+m$

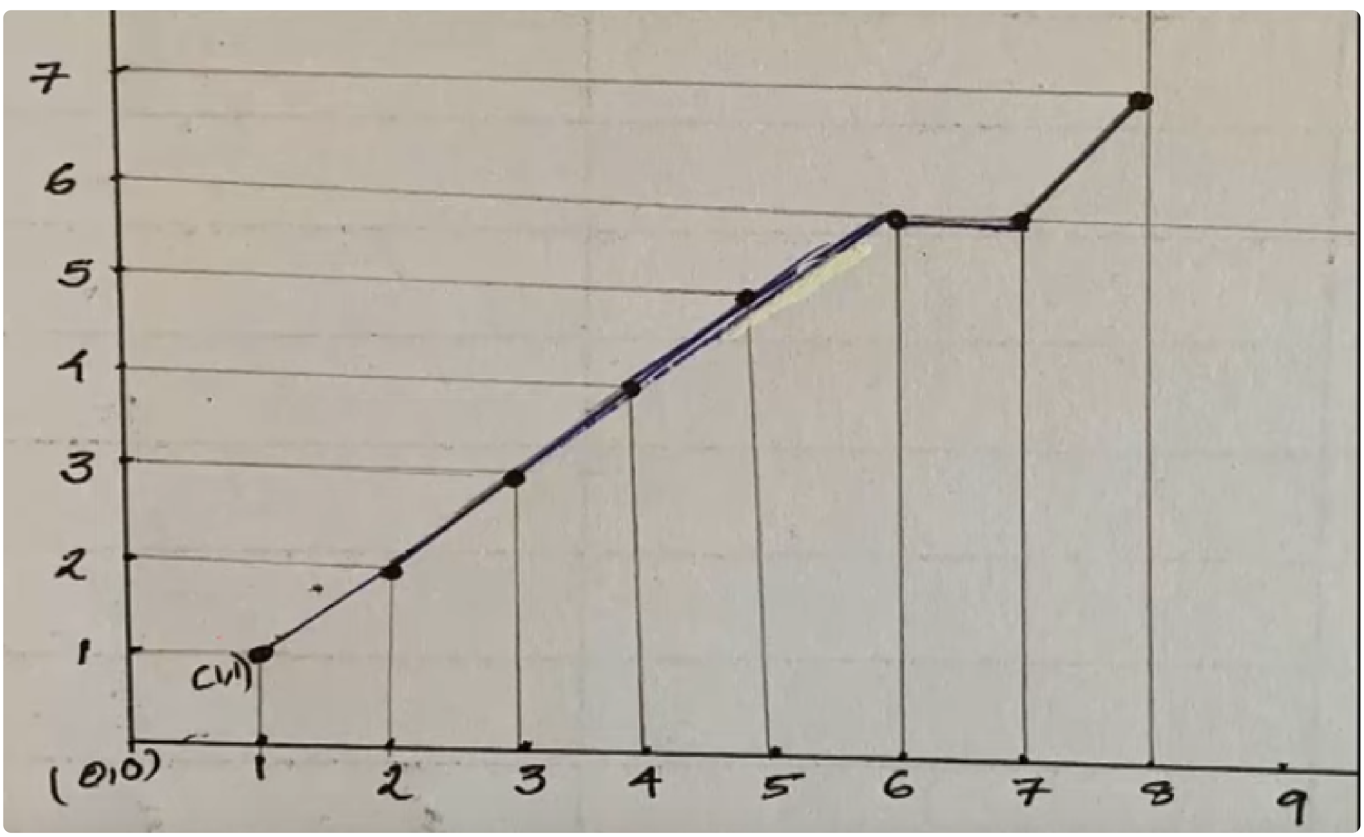
Lets start filling the table

- **Initally our point is (1,1)**
  - Iteration 1
    - Calculating X
      - $X_{k+1} = X_k + 1$
      - Here our current point X_k is 1
      - $X_{k+1} = 1 + 1 = 2$
    - Calculating Y
      - $Y_{k+1} = Y_k + m$
      - Y_k = 1
      - Value of m = 6 / 7 = 0.9
      - Y_k = 1 + 0.9 = 1.9
    - Rounding off the pixels
      - Round(2,1.9) = (2,2)
  - Iteration 2
    - Here our current point X_k is 2
    - $X_{k+1} = 2 + 1 = 3$
    - Y_k = 1.9

        - Y_k = 1.9 + 0.9 = 2.8
        - Round(3,2.8) = (3,3)
    - Similarly repeating until $X_{k+1} = 8$
- We get the following table at the end
    -

No	Xk+1	Yk+1	Pixel Plotted Round(Xk+1 , Yk+1)
1	1+1=2	1+6/7=1.9	Round(2,1.9)=(2,2)
2	2+1=3	1.9 +6/7=2.8	Round(3,2.8)=(3,3)
3	3+1=4	2.8 + 6/7=3.7	Round(4, 3.7)=(4,4)
4	4+1=5	3.7+6/7 =4.6	Round(5, 4.6)=(5,5)
5	5+1=6	4.6 +6/7=5.5	Round(6, 5.5)=(6,6)
6	6+1=7	5.5 +6/7=6.4	Round(7, 6.4)=(7,6)
7	7+1=8	6.4 + 6/7=7.3	Round(8, 7.3)=(8,7)

Lets draw the graph based on the table



Bresenhams Line drawing algorithm

Algorithm when |Slope| < 1

1. Input the two endpoints and store the left endpoint in (x0,y0)
2. Load (x0,y0) into the frame buffer, that is plot the first point
3. Calculate the constants dx,dy,2dy and 2dy - 2dx and get the first value for the decision parameter as
   1. p0 = 2dy - dx
4. At each Xk along the line, starting at k = 0, perform the following test
   1. If pk < 0, the next point to plot is (xk + 1, yk) and Pk+1 = Pk + 2dy
   2. else, the next point to plot is (xk + 1, yk + 1) and Pk+1 = Pk + 2dy - 2dx
5. Repeat step 4 dx times

Problem

Draw line using Bresenhams Line Drawing Algorithm, The end points are (20,10) and (30,18)

1. Slope dy/dx -> (y2-y1) / (x2-x1) -> (18-10) / (30-20) -> 0.8 < 1
2. dy = 8, dx = 10, 2dy = 8, 2dx = 20, 2dy-2dx = 16 -20 = -4
3. Iteration 1
   1. Calculating decision parameter
      - p0 = 2dy - dx
      - p0 = 2 x 8 - 10 = 6
   2. Check the condition
      1. If pk < 0, the next point to plot is (xk + 1, yk) and Pk+1 = Pk + 2dy
      2. otherwise, the next point to plot is (xk + 1, yk + 1) and Pk+1 = Pk + 2dy - 2dx
   3. Here p0 = 6 > 1
      1. Next point to plot is (xk + 1,yk + 1)
         1. (20 + 1, 10 + 1)
         2. (21,11)
      2. Pk+1 = Pk + 2dy - 2dx
         1. 6 + (-4) = 2
4. Iteration 2
   1. pk = 2

    2. pk > 1
    3. Next point
        1. (21 + 1, 11 + 1)
        2. (22,12)
    4. Pk+1 = 2 - 4 = -2
5. Iteration 3
    1. pk = -2
    2. pk < 1
    3. Next point
        1. (xk + 1, yk)
        2. Pk+1 = Pk + 2dy
    4. (22 + 1, 12) = (23,12)
    5. Pk+1 = -2 + 16 = 14
6. Repeating the steps, we get

K	P_k	(X_{k+1}, y_{k+1})
0	6	(21,11)
1	(6+-4)→2 >0	(22,12)
2	(2+-4)→-2 <0	(23,12)
3	(-2+16)→14	(24,13)
4	(14+-4)→10 >0	(25,14)
>14 5	(10+-4)→6	(26,15)
6	(6+-4)→2	(27,16)
7	10+-4 = 6>0 (2+-4)→-2 +16	(28,16)
8	(-2+16)→14	(29,17)
9	(14+-4)→10	(30,18)

Advantages of Bresenhams Line Drawing Algorithm

- Easy to implement
- Fast and incremental

- It Executes fast but less faster than the DDA algorithm
- The points generated by this algorithm are more accurate than DDA algorithm

Disadvantages of Bresenhams line drawing algorithm

- It also does not provide snooth lines though accuracy have been improved
- Less faster than DDA

---

3. Midpoint Circle drawing Algorithm

Algorithm

- Input radius r and circle centre (x0,y0)
- Obtain first point on the circumference of circle centered on the origin as (x0,y0) = (0,r)
- Calculate the initial value of decision parameter as P0 = 5/4 -r
- At each x_k position , strting at k=0 , perform the following test : if $p_k<0$, the next point along the circle centered on (0,0) is(x_k+1,y_k) and

  $$P_{k+1} = P_k+2x_k+3$$

  Otherwise next point along the circle is (x_k+1,y_k-1) and

  $$P_{k+1} = P_k+2x_k-2y_{k+}\ 5$$

- Move each calculated pixel position(x,y) onto the circular path centered on (x_c,y_c) and plot the coordinate values :

  $$x=x+x_c \qquad y=y+y_c$$

Problem

Given a circle of radius r = 10, determine positions along circle octant in the first quadrant from x= 0 to x = y

- **Initial value of decision parameter is 5/4 -r**
  - Rounding 5/4 to 1
  - **P0 = 1- r = 1-10 = -9**
- **Initial points**
  - Initial point is (x0,y0) = (0,10)

- 2Xo = 0, 2Yo = 20
- Iteration 1
  - Here p = -9 < 0
    - if $p_k<0$, the next point along the circle centered on (0,0) is(x_k+1,y_k) and
      $$P_{k+1} = P_k+2x_k+3$$
    - Pk+1 = -9 + 0 + 3 = -6
    - (xk + 1, yk) = (1,10)
- Iteration 2
  - pk = -6 < 0
  - Pk+1 = -6 + 2 x 1 + 3 = -6 + 5 = -1
  - (2,10)
- Iteration 3
  - pk = -1 < 0
  - Pk+1 = -1 + 2x2 + 3 = -1 + 7 = 6
  - (3,10)
- Iteration 4
  - pk = 6 > 0
  - next point along the circle is (x_k+1,y_k-1) and
    $$P_{k+1} = P_k+2x_k-2y_{k+} 5$$
  - Pk+1 = 6 + 6 - 20 + 5 = -3
  - (3,9)
- Repeating the steps until x = y
- We will finally get
k	pk	(x_{k+1},y_{k+1})	$2\ x_k +3$	$2x_k-2y_k +5$
0	-9	(1,10)	3	
1	-9+3=-6	(2,10)	5	
2	-6+5=-1	(3,10)	7	
3	-1+7=6	(4 , 9)		6-20+5=-9
4	6-9= -3	(5,9)	11	
5	-3+11 =8	(6 ,8)		10-18+5 =-3
6	8 -3 =5	(7,7) →x=y ,then stop		

4. Bresenham's Circle drawing algorithm

Algorithm

1. Input radius r and circle centre (Xc,Yc), then set the coordinates for the first point on the circumference of a circle centered on the origin as (x0,y0) = (0,r)
2. Calculate the initial decision parameter as d0 = 3-2r
3. At each xi, from i = 0 perform the following

   If $d_i<0$,next point to plot along the circle centered on (0,0) is (x_{i+1},y_i) and

   $$d_{i+1}=d_i+4x_i+6 .$$

   Otherwise ,

   Next point to plot is (x_{i+1},y_{i-1}) and

   $$d_{i+1}=d_i+4(xi-yi)+10 .$$

4. Determine the symmetry points in the other seven octants
5. Move each calculated pixel position (x,y) onto the circular path centered at (xc,yc) and plot the coordinate values x= x+Xc and y= y+yc
6. Repeat steps 3 through 5 until x >=y

Problem

**Given a circle radius r = 10, determine positions along the circle octant in the fits quadrant fro

- **The initial value of the decision parameter is d0 = 3 - 2r**
  - d0 = 3 - 2 x 10 = -17
- Initial point is (0,10)
  - Here d0 is < 0
    - **Case1: di<0**

      $x_{i+1}=x_i+1$

      $y_{i+1}=y_i$

      $d_{i+1}=d_i +4x_i+6$
    - xi+1 = 0 + 1
    - yi + 1 = 10
    - (1,10)

        - di+1 = -17 + 4x0 + 6 = -11
- 2nd Iteration
    - di < 0
    - (2,10)
    - di+1 = -11 + 4x1 + 6 = -1
- 3rd Iteration
    - di < 0
    - (3,10)
    - di+1 = -1 + 4x2 + 6 = 13
- 4th iteration
    - di > 0
        - **Case2: di>=0**

          $x_{i+1}=x_i+1$

          $y_{i+1}=y_i-1$

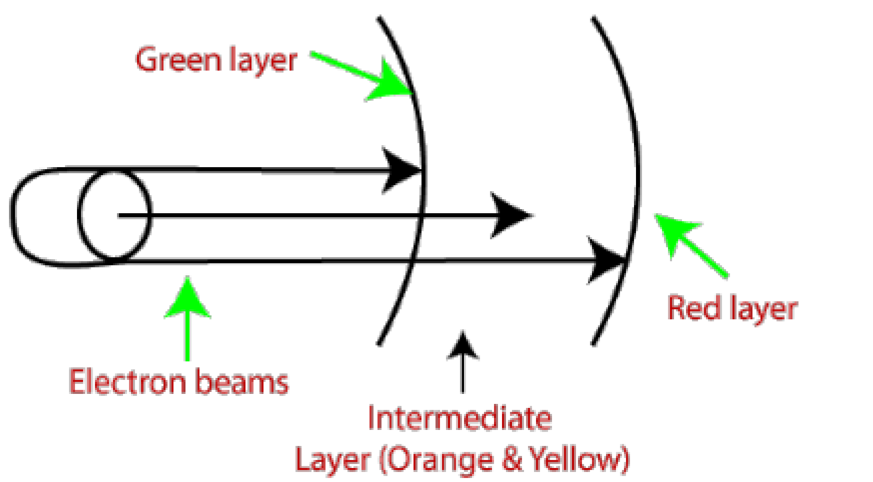
          $\mathbf{d_{i+1}=d_i+4x_i-4y_i+10}$
    - xi+1 = 3 + 1 = 4
    - yi+1 = 10 - 1
    - (4,9)
    - di+1 = 13 + 4x3 - 4x10 + 10 = -5
- Repeating until x = y, we get

k	dk	(x_{k+1},y_{k+1})	$4x_i+6$	4(xi-yi)+10
0	-17	(1,10)	4x0+6=6	
1	-17+6=-11	(2,10)	4x1+6=10	
2	-11+10=-1	(3,10)	4x2+6=14	
3	-1+14=13	(4,9)		4(3-10)+10=-18
4	13-18=-5	(5,9)	4x4+6=22	
5	-5+22=17	(6,8)		4(5-9)+10=-6
6	117-6=11	(7,7) x=y then stop		

5. Explain the beam penetration and shadow mask method for displaying color pictures.

Beam penetration method

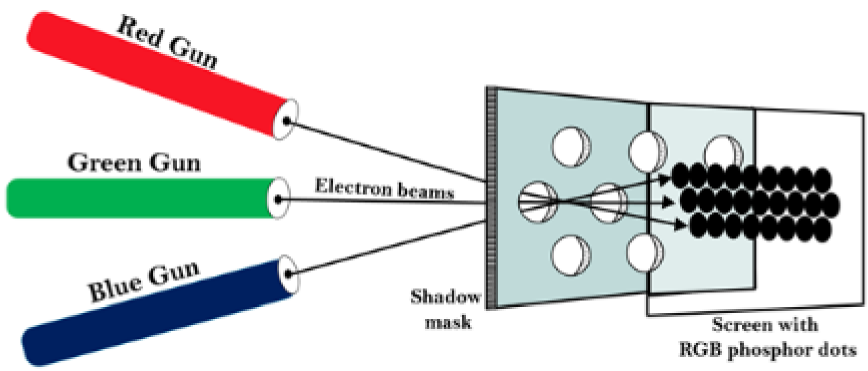
- The **Beam-Penetration** method has been used with random-scan monitors.
- In this method, the CRT screen is coated with two layers of phosphor, red and green and the displayed color depends on how far the electron beam penetrates the phosphor layers
- This method produces four colors only, **red, green, orange and yellow**.
- A beam of slow electrons excites the outer red layer only; hence screen shows red color only.
- A beam of high-speed electrons excites the inner green layer. Thus screen shows a green color.

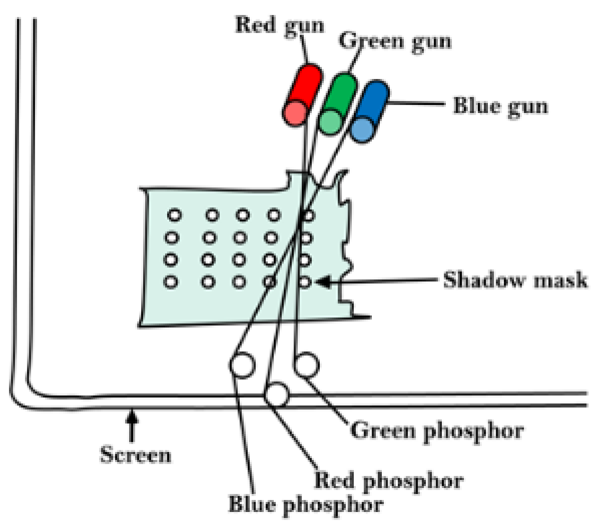


Shadow-Mask Method

- Produces a much wider range of colors than the beam-penetration method.
- It is used in the majority of color TV sets and monitors.
- A shadow mask CRT has **3 phosphor color dots** at each pixel position.
  - These emit red, green and blue light respectively
- This type of CRT has 3 electron guns, one for each color dot and a shadow mask grid just behind the phosphor coated screen.

- Shadow mask grid is pierced with small round holes in a triangular pattern



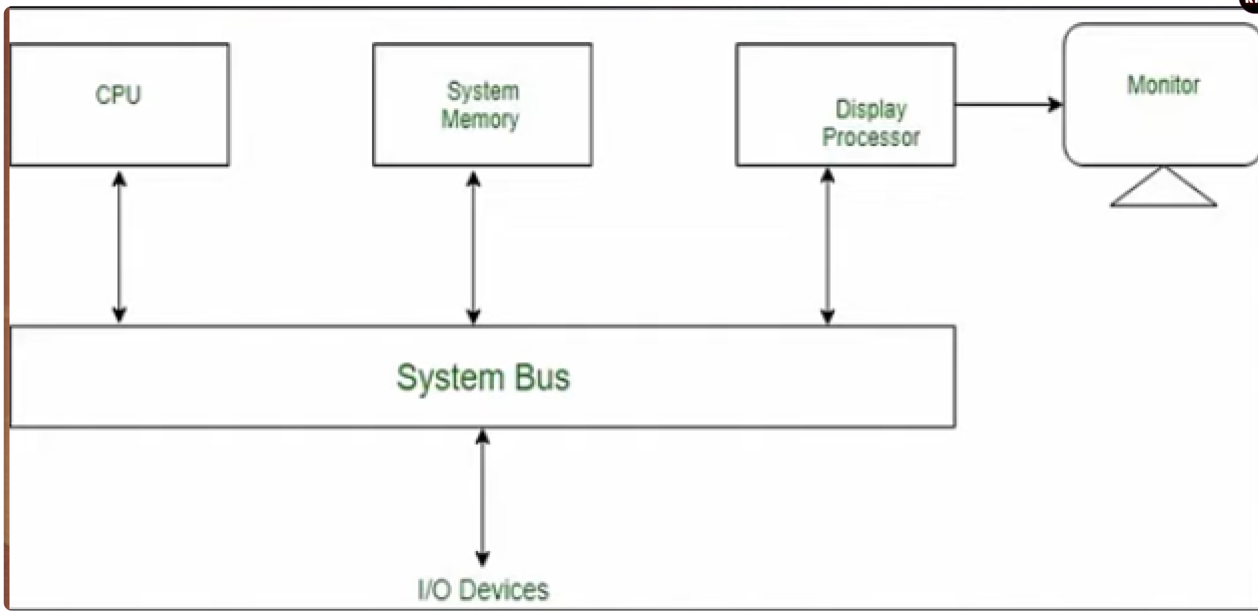


The Shadow mask CRT

6. *Random scan and raster scan system*

Random scan

- In random scan display, the electron beam is directed only to the areas of the screen where a picture has to be drawn
- It is also called vector display, as it draws pictures one line at a time
- It can draw and refresh component lines of a picture in any specified sequence. Input in the form of an application program is stored in the system memory along with a graphics package

- 

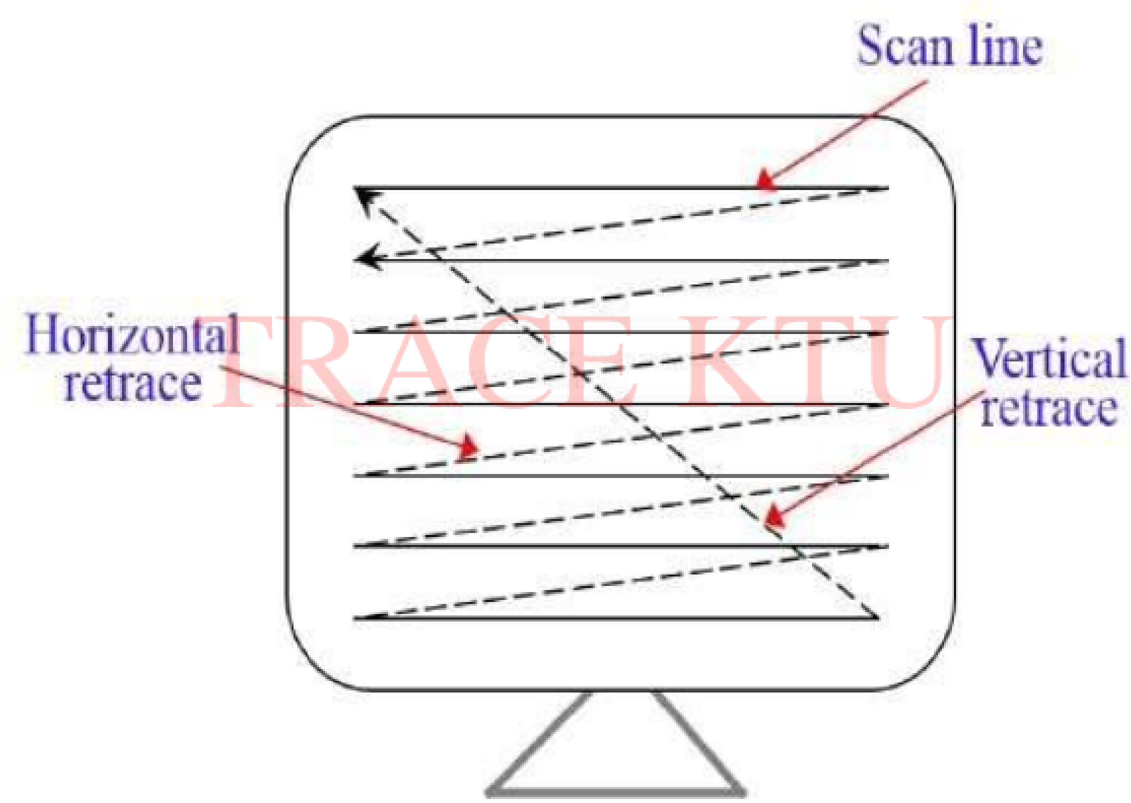
Raster scan system

Raster

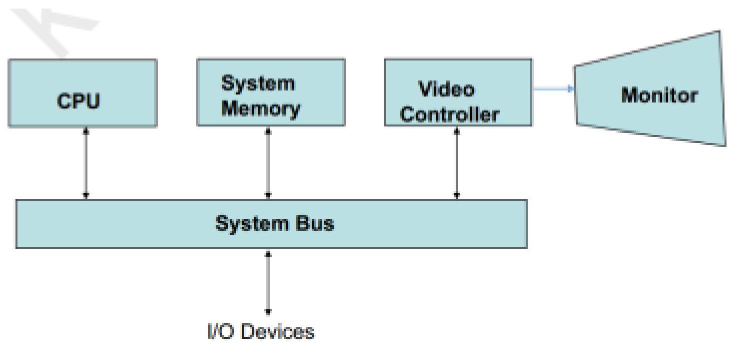
- Raster can be explained as a rectangular collection of dots or points plotted.
- An image is subdivided into various horizontal lines which are referred to as scan lines which are then further divided into different pixels which helps in the processing of an image.

Working of Raster Scan

- •In this system, a beam of an electron is moved across the screen. It moves from top to bottom considering one row at a time.
- As the beam of electron moves through each row, its intensity is alternatively turned on and off which helps to create a pattern of spots that are illuminated.
- When each scan of the line is refreshed it returns to the left side of the screen. This motion is known as **Horizontal retrace**.
- As a particular frame ends, the beam of electron moves to the left top corner of the screen to move to another frame. This motion is referred to as **Vertical retrace**



- The picture is then stored in an area of memory which is referred to as the **frame buffer** or **refresh buffer**.
  - The values stored in the buffer are then fetched and traced over scan lines one by one on the screen.
- •The image formed through this raster scan is known as a **raster image**



Architecture of Simple Raster graphics system

Difference between Raster Scan and Random Scan

Raster Scan	Random Scan
Resolution of raster scan is lesser or lower than	The resolution of random scan is higher

Raster Scan	Random Scan
random scan	than raster scan
Cost of raster scan is lesser than random scan	Its costlier than raster scan
It stores picture definition as a set of intensity values of the pixels in the frame buffe	It stores picture definition as a set of line commands in the refresh buffer
While in raster scan any alteration is not so easy	In a random scan, any alteration is easy in comparison of a raster scan.
The refresh rate is 60 to 80 frames per second and is independent of picture complexity	The refresh rate depends on the number of lines to be displayed
Eg: TV Sets	Eg: Pen Plotter

7. Frame buffer problems

1. Consider a raster system with a resolution of 1024x1024. What is the size of the raster needed to store 4bits per pixel. How much storage is needed if 8 bits per pixel are to be stored

- Resolution
  - No of pixels present in a screen
  - Consider a screen, The total number of pixels here is called resolution
  - 1024 x 1024
    - First 1024 means no of pixels in a horizontal line
    - Second 1024 means no of pixels in a vertical line
- The question asks what is the size of raster needed?
  - Here Raster means **we need to find the size of the frame buffer**
  - So what is the size of the frame buffer needed if a pixel stores 4 bits
    - 4 bits are used in a pixel to store the intensity value
- So one pixel = 4 bits
- We have 1024 x 1024 pixels
- Total number of bits = 1024 x 1024 x 4
- So the Frame buffer should have size 1024x1024x4 bits
- In bytes
  - One byte = 8 bits, dividing by 8
  - **1024x1024x4 / 8 bytes**

- How much storage is needed if 8 bits per pixel are to be stored?
- 1024 x 1024 x 8 bits
- **1024 x 1024 x 8 / 8 bytes**

2.Suppose you have a raster system designed using an 8 inches x 10 inches screen with a resolution of 100 pixels per inch in each direction. What frame buffer size is required if 6 bits are stored per pixel in the buffer?

- 1 inch = 100 pixels
- 8 inches = 800 pixels
- 10 inches = 1000 pixels
- So the resolution is 800 x 1000 pixels
- No of bits per pixel = 6
- Frame buffer size = 800x1000 x 6 bits
- = 800 x 1000 x 6 / 8 bits

3. How much time is spent scanning across each row of pixels during screen refresh on a raster system wiht a resilution of 1280 x 1024 and are the refresh rate of 60 frames per second

- 1 second = 60 frames
- 1 frame = 1/60 seconds
- Question asks about scanline, Here scanlines = 1024
- 1024 pixels = 1/60 seconds
- 1 scanline = 1/(60x1024)

---

8. List applications of Computer Graphics

Computer graphics is used in a lot of areas such as

1. Computer aided design
2. Presentation graphics
3. Computer art
4. Entertainment
5. Education and training

6. Visualization
7. Image processing
8. Graphical user interfaces