

Computer Graphics

Unit 1

Contents

- Introduction, graphics primitives - pixel, resolution, aspect ratio, frame buffer. Display devices, applications of computer graphics.
- Introduction to OpenGL - OpenGL architecture, primitives and attributes, simple modelling and rendering of two and three dimensional geometric objects, GLUT, interaction, events and call-backs picking.(Simple Interaction with the Mouse and Keyboard)
- Scan conversion: Line drawing algorithms: Digital Differential Analyzer (DDA), Bresenham. Circle drawing algorithms: DDA, Bresenham, and Midpoint.

Graphics Primitives

- Computer graphics primitives are basic geometric shapes or elements that serve as the building blocks for creating more complex images in computer graphics.
- Some common computer graphics primitives include:
 - **Points:** These are single pixels that represent a location in space.
 - **Lines:** Lines are sequences of connected pixels that extend in a particular direction. They can be defined by two endpoints or by a point and a direction vector.
 - **Line Segments:** These are finite sections of lines that have a definite starting and ending point.

- **Polygons:** Polygons are closed geometric shapes with straight sides. They can be regular (all sides and angles are equal) or irregular (sides and angles can vary). Common examples include triangles, rectangles, and pentagons.
- **Circles:** Circles are round shapes defined by a center point and a radius. They can be used to represent curves and arcs.
- **Ellipses:** Similar to circles, ellipses are elongated round shapes defined by a center point, major axis, and minor axis.
- **Curves:** Curves represent smooth or nonlinear paths. Bezier curves and splines are examples of commonly used curves in computer graphics.
- **Surfaces:** Surfaces are two-dimensional representations of shapes. These can be used to create 3D models by combining multiple surfaces.

Basic Concepts

- **Screen Size :** The physical dimensions of a screen. It is the length, in inches, of the screen from one corner to the diagonal corner.
- **Pixel:** Screens display images through pixels. A pixel, pel or dots, or picture element is a physical point in a raster image, or the smallest addressable element in raster display device; so it is the smallest controllable element of a picture represented on the screen.
 - Pixels are arranged in a grid to form images on screens, such as computer monitors, TVs, and mobile devices.
 - Pixels are not always the same size from device to device.

0 1 2 c w-1

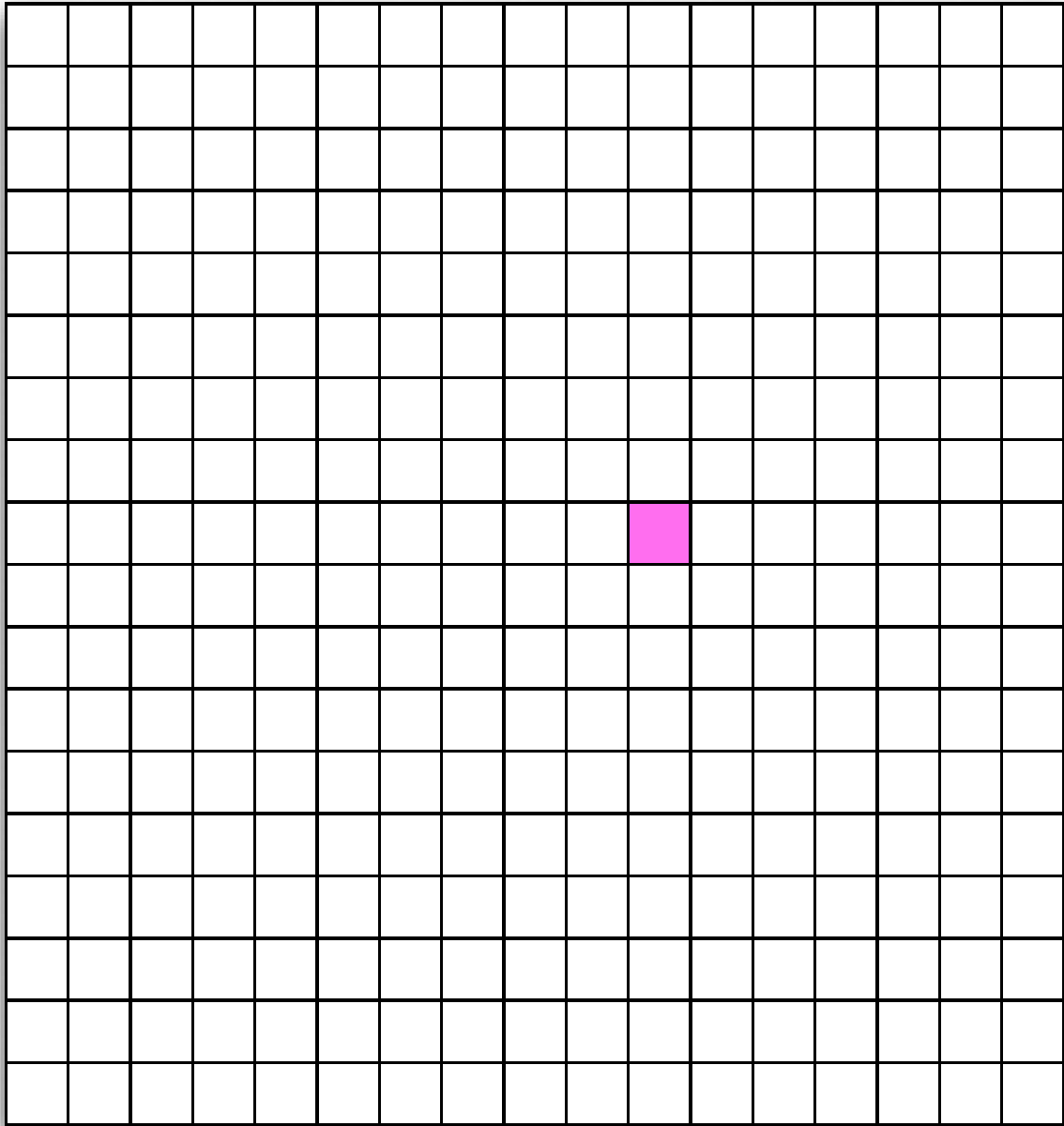
0

1

2

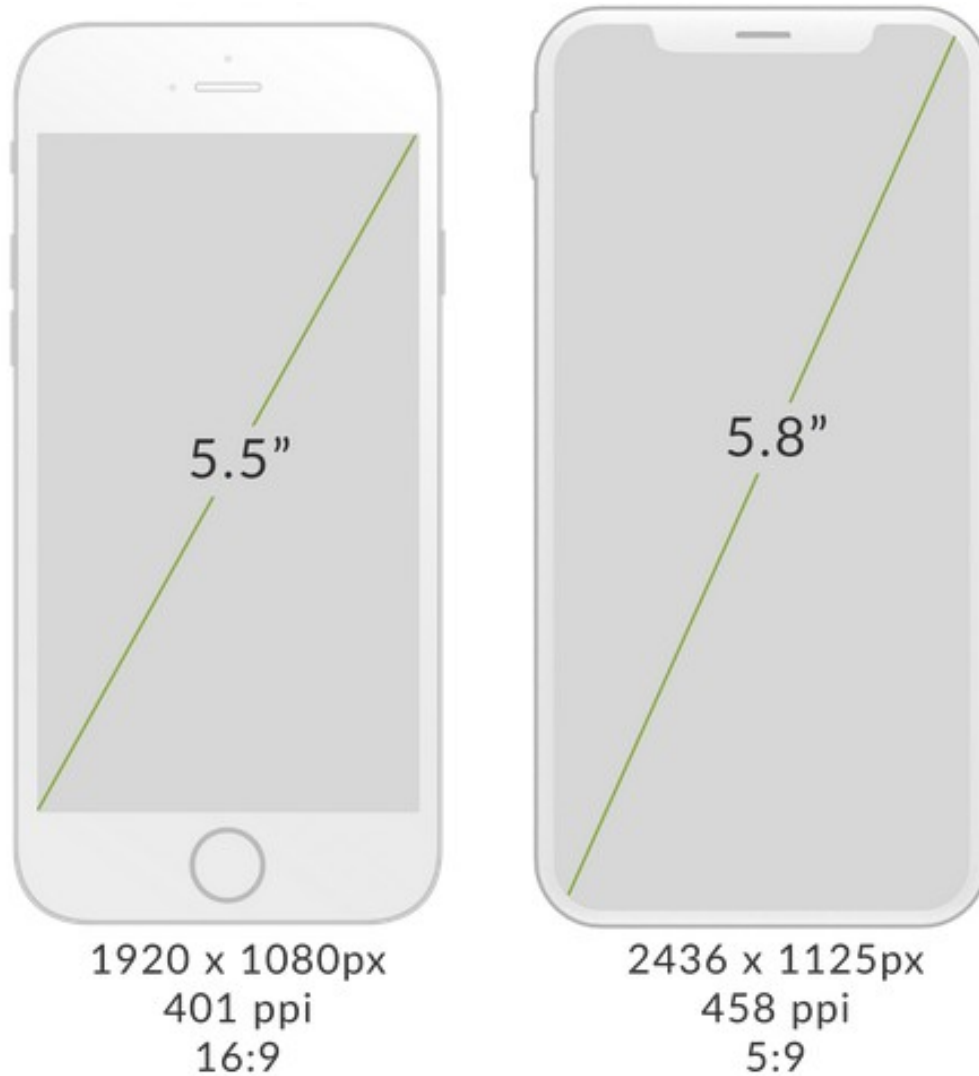
r

h-1



- **Resolution:** This is the number of pixels displayed on the screen.
 - It is often formatted as width x height or pixels per inch.
 - Because pixels aren't always the same size, it is possible to have two devices with the same screen size and different resolutions.
 - Resolution affects the clarity and level of detail in images.
 - A higher resolution image has more pixels and therefore can display finer details, but it may also require more processing power and storage space.

- **Aspect Ratio:** Aspect ratio measures width to height ratio of screen.
- For example, if a computer graphic has an aspect ratio of 3:1, this means the width of the graphic is three times of the height of the image.
- Common aspect ratios include 4:3 (standard for older TVs and monitors), 16:9 (widescreen HD), and 21:9 (ultrawide).
- The aspect ratio is significant because it affects how images are displayed on different screens. For instance, a 16:9 aspect ratio corresponds to a widescreen display, while a 4:3 aspect ratio is more square.



Screen sizes, resolutions, pixels, and aspect ratios

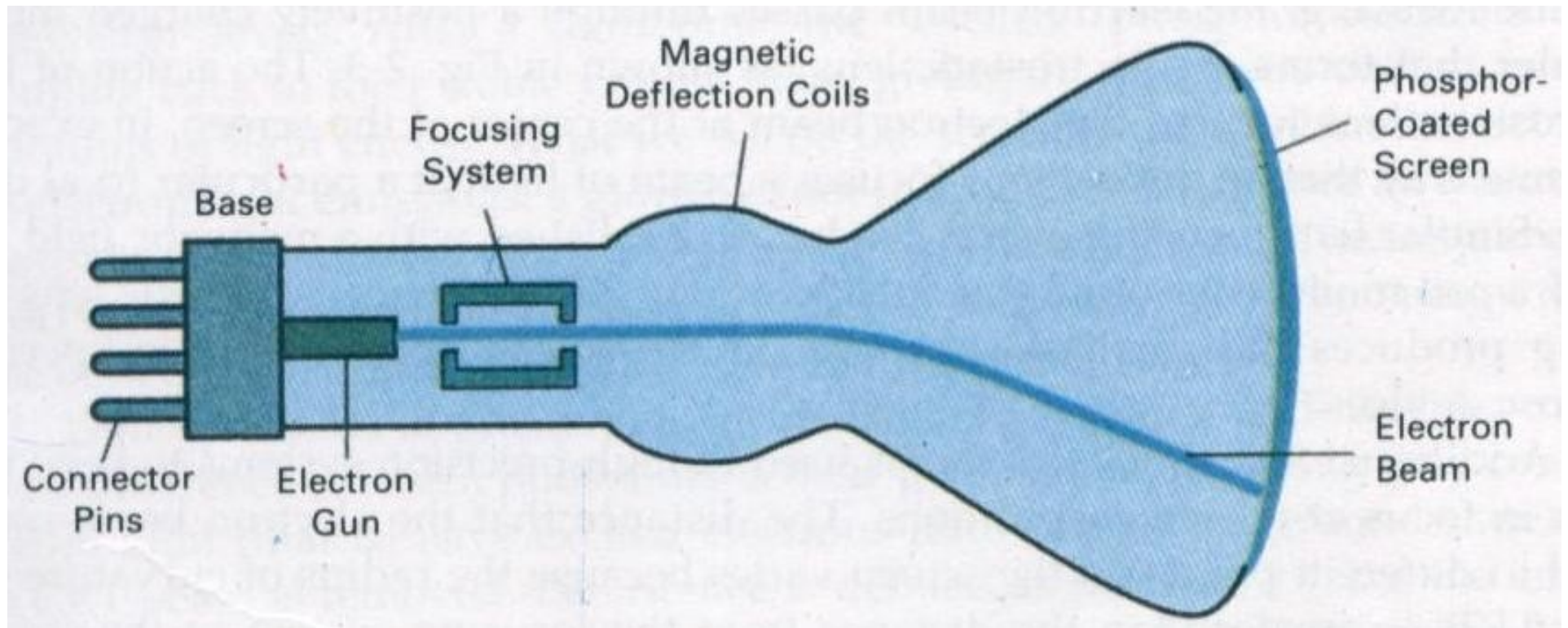
Frame Buffer

- A frame buffer (Refresh Buffer) is a portion of computer memory used to store and manage the data necessary for displaying images or video.
- The frame buffer stores pixel values for each position on the screen. Each pixel's value represents its color and intensity.
- The frame buffer essentially acts as a 2D array or grid, where each cell corresponds to a pixel on the display.

- The contents of the frame buffer are periodically read by the display hardware to illuminate the corresponding pixels on the physical display. This process happens rapidly, creating the illusion of continuous motion.
- The frame buffer's capacity (the number of pixels it can store) and bit depth (the number of bits used to represent the color of each pixel) directly impact the quality and complexity of the images that can be displayed. Higher resolutions and more colors require larger frame buffer capacities.

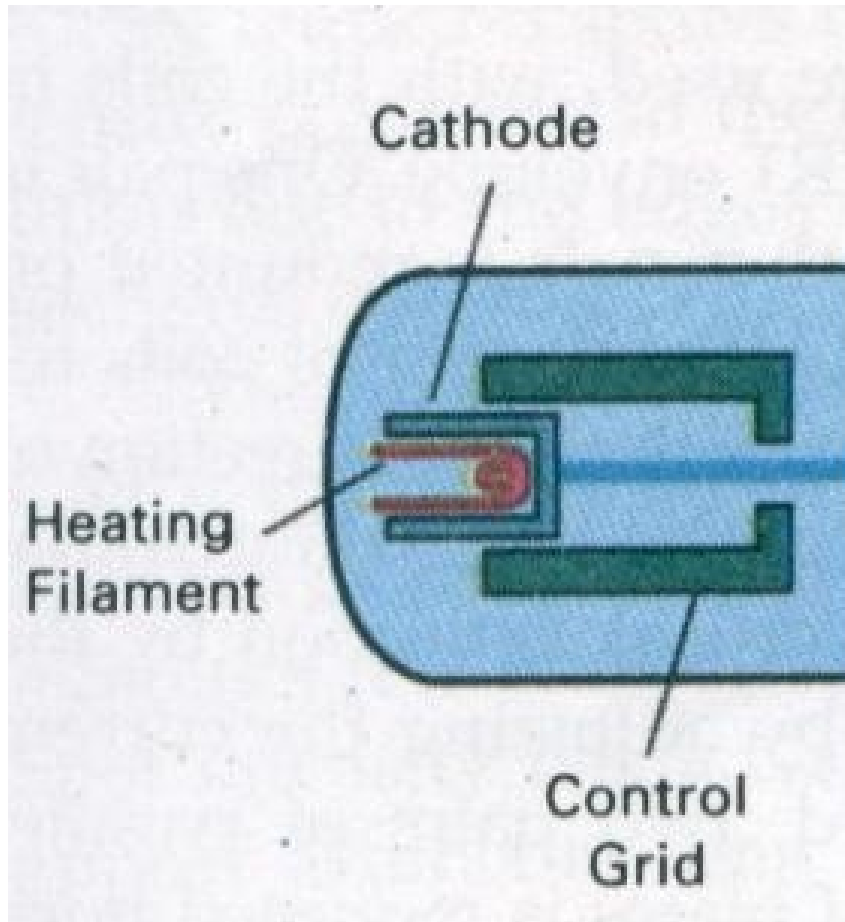
Display Devices

CRT



- The electron gun emits a beam of electrons (**cathode rays**).
- The electron beam passes through focusing and deflection systems that direct it towards specified positions on the **phosphor-coated screen**.
- When the beam hits the screen, the phosphor emits a small spot of light at each position contacted by the electron beam.

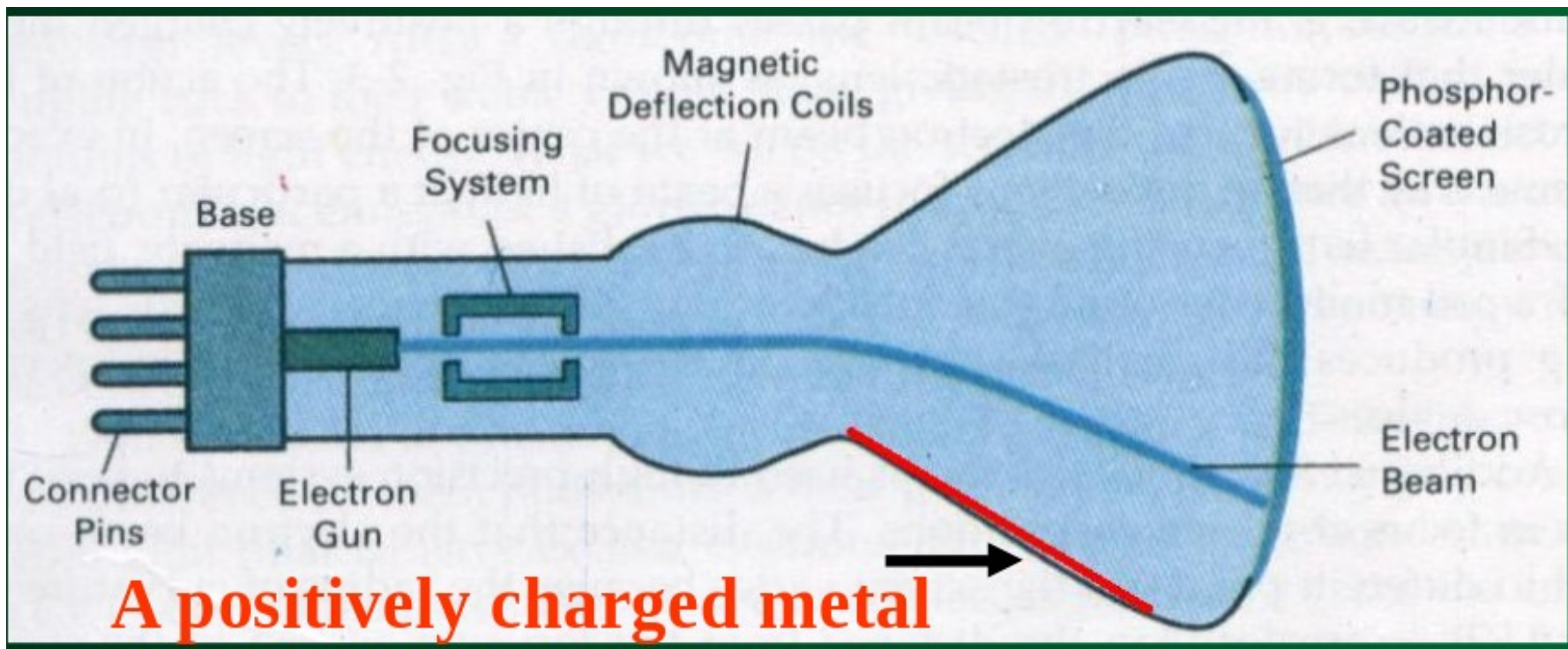
Electron Gun



- Heat is supplied to the cathode by the **filament**.
- The free electrons are then accelerated toward the phosphor coating by a **high positive voltage**.

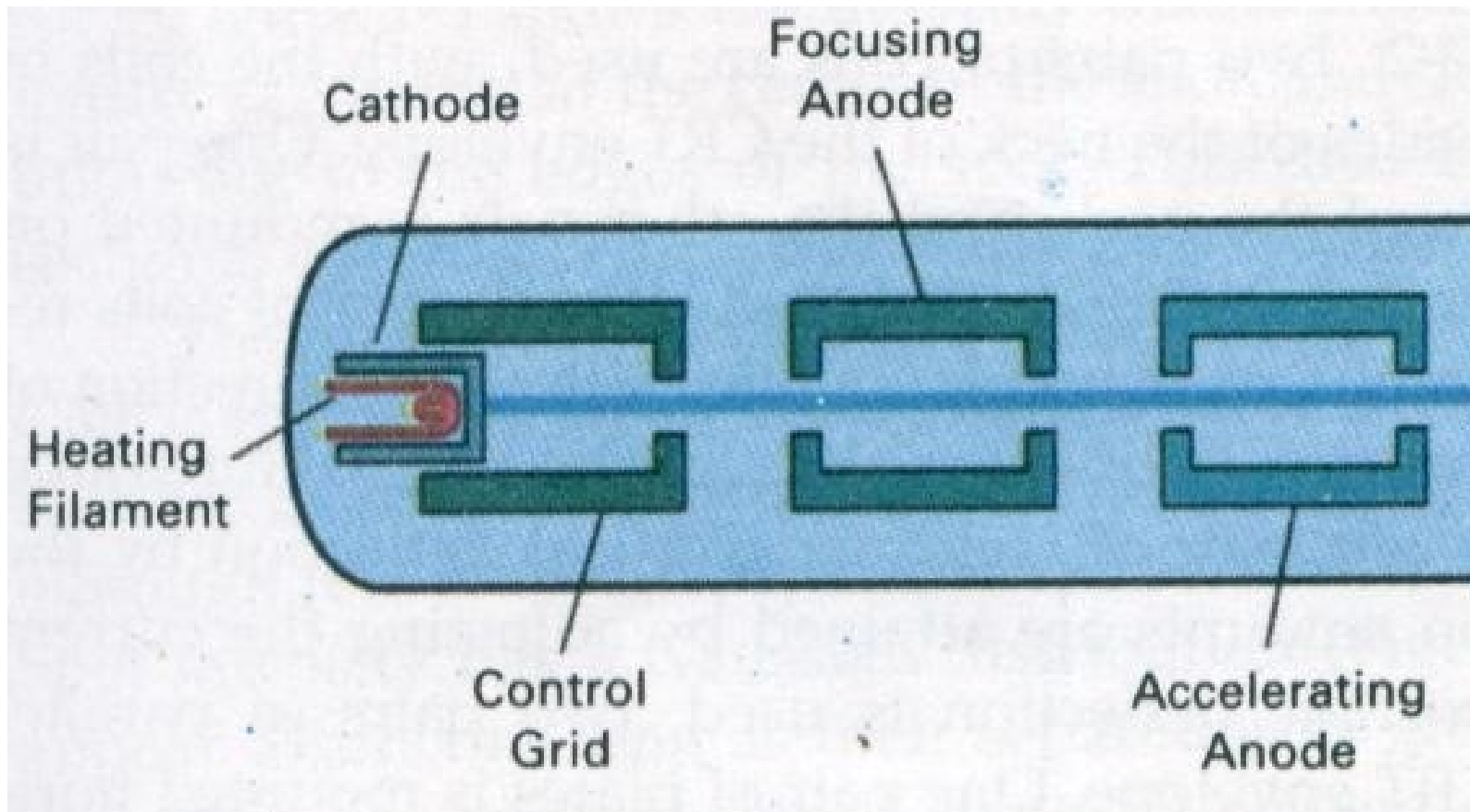
High positive voltage

A positively charged metal coating on the inside of the CRT envelope near the phosphor screen.



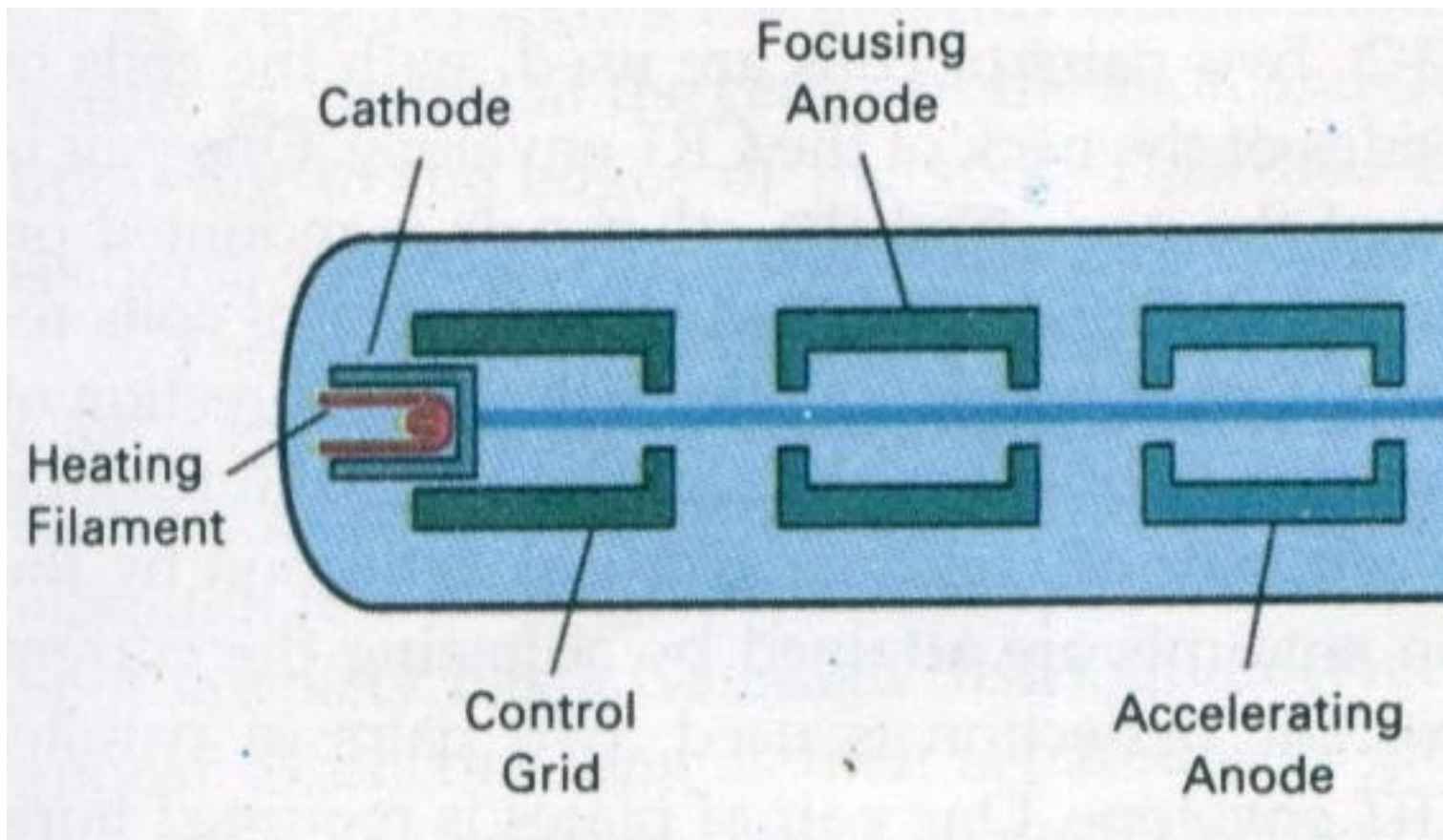
High positive voltage

Accelerating anode



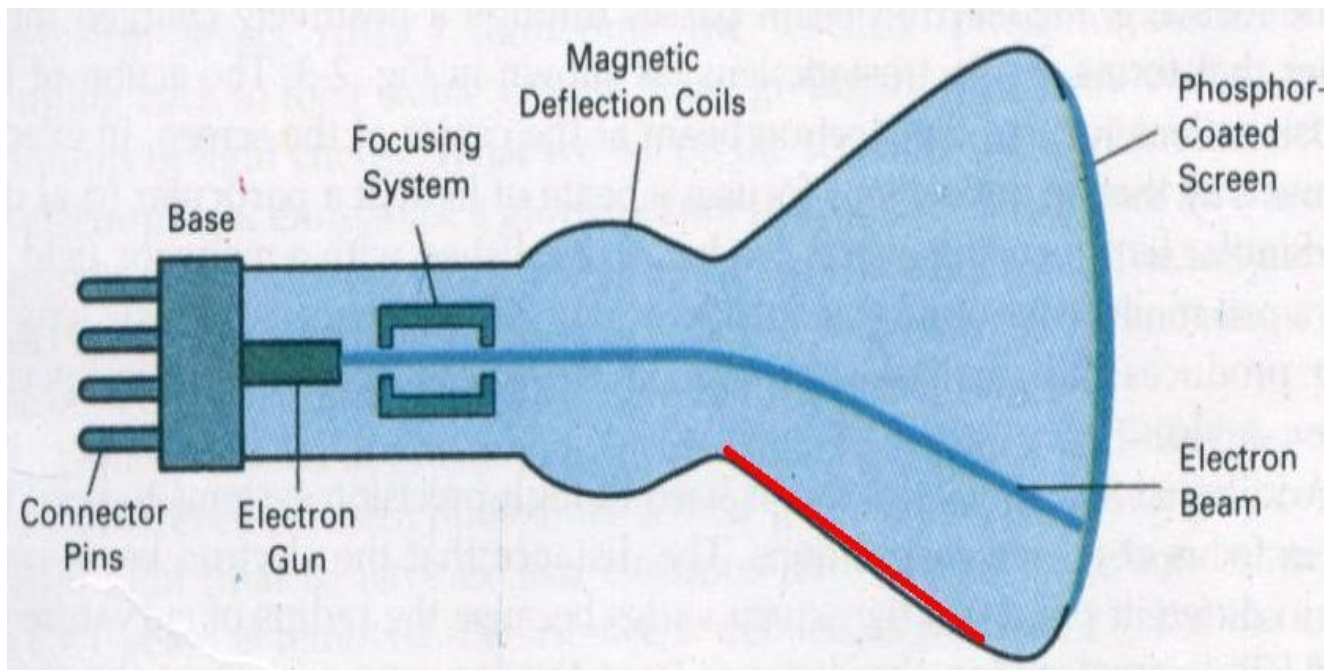
Control grid

Intensity of the electron beam is controlled by setting voltage level on the control grid.



Focusing system

The **focusing system** is needed to force the electron beam to converge into a small spot as it strikes the phosphor.



1) **Electrostatic focusing** is commonly used in computer graphics monitor.

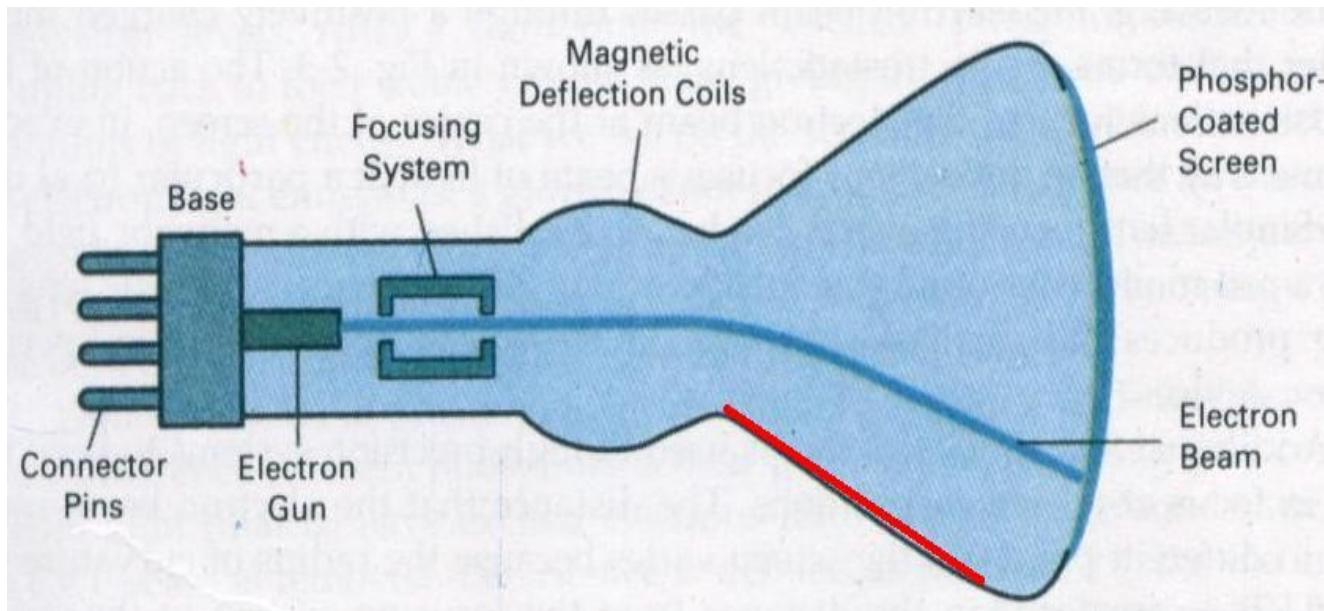
With electrostatic focusing, the electron beam passes through a positively charged metal cylinder that forms an **electrostatic lens**.

2) Similar lens focusing effects can be accomplished with a **magnetic field** set up by a coil mounted around the outside of the CRT envelope.

Deflection Systems

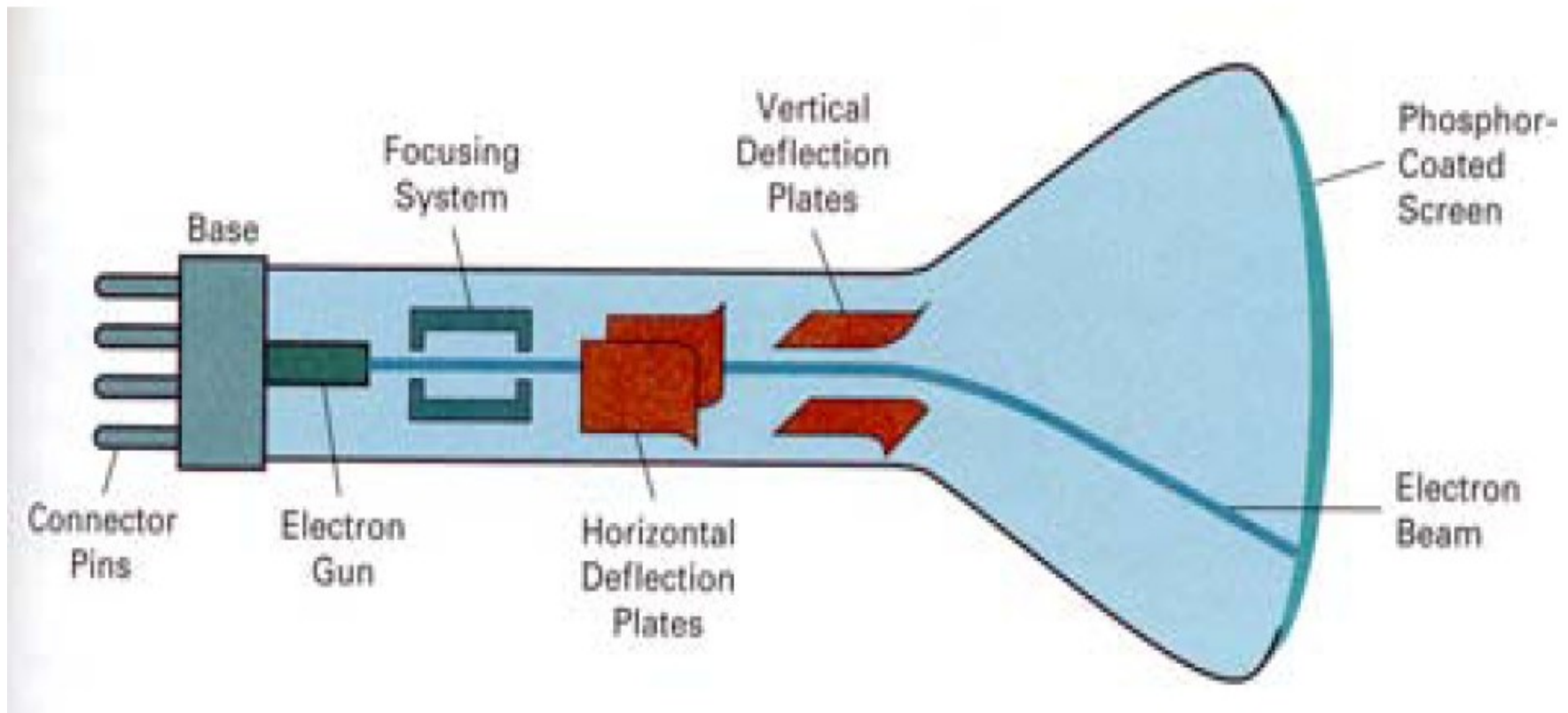
Deflection of the electron beam can be controlled either with **electric fields** or with **magnetic fields**.

1) **The magnetic deflection** coils mounted on the outside of the CRT envelope.



2) Electrostatic deflection: Two pairs of parallel plates are mounted inside the CRT envelope.

- One pair of plates is mounted horizontally to control the **vertical deflection**, and the other pair is mounted vertically to control **horizontal deflection**.
- The beam is deflected horizontally by applying an electric field between a pair of plates to its left and right, and vertically by applying an electric field to plates above and below.



Phosphor Coating

- Various phosphors are available depending upon the needs of the display application.
- The brightness, color, and persistence of the illumination depends upon the type of phosphor used on the CRT screen.
- Phosphors are available with persistences ranging from less than one microsecond to several seconds

3 types of displays:

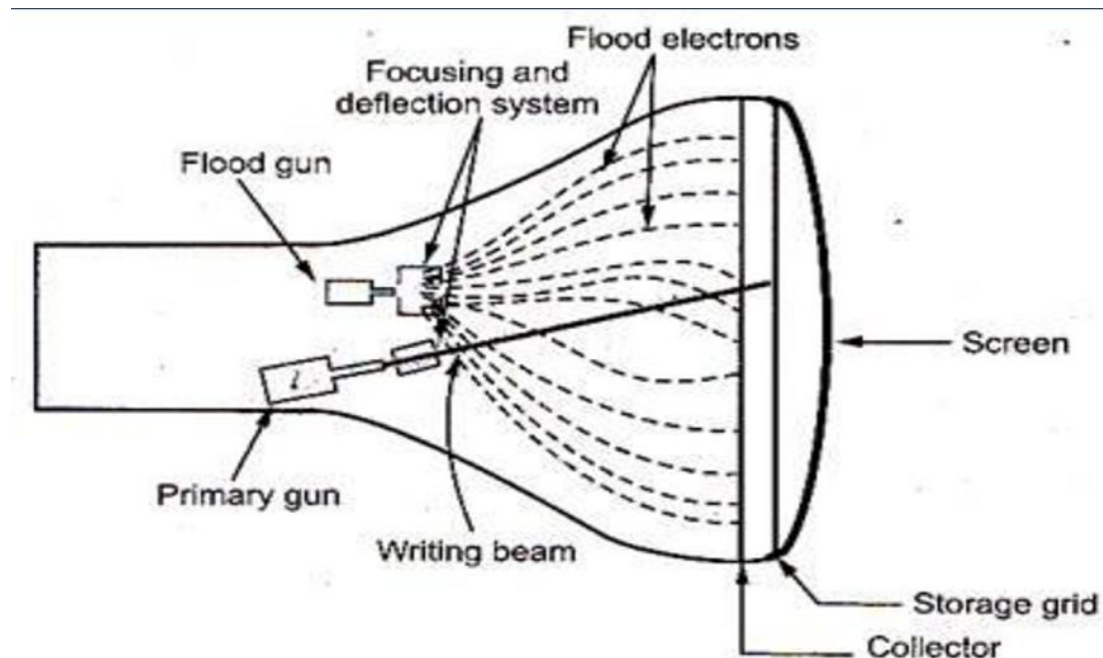
1. Direct View Storage Tube

2. Random Scan Displays

3. Raster Scan Displays

1. Direct View Storage Tube

- DVST is a CRT with **highly persistent phosphor**.
- A direct-view storage tube (DVST) stores the picture information as a charge distribution just behind the phosphor-coated screen.
- Two electron guns are used in a DVST. One, the primary gun, is used to store the picture pattern; the second, the flood gun, maintains the picture display as shown in the figure below.
- The term "storage grid" refers to the ability of the screen to retain the image which has been projected against it, thus avoiding the need to rewrite the image constantly.



Advantages

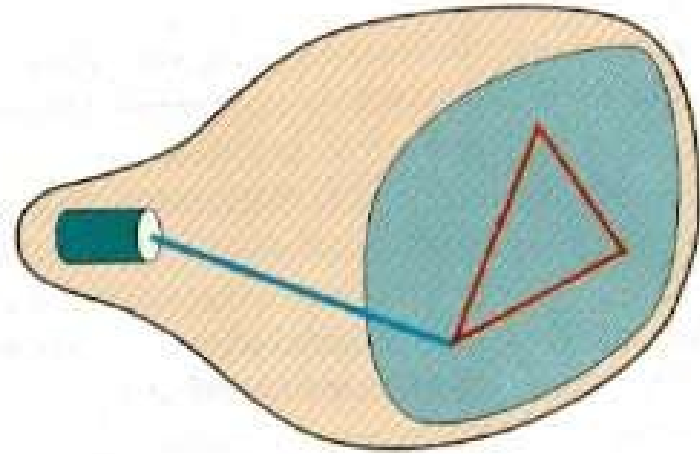
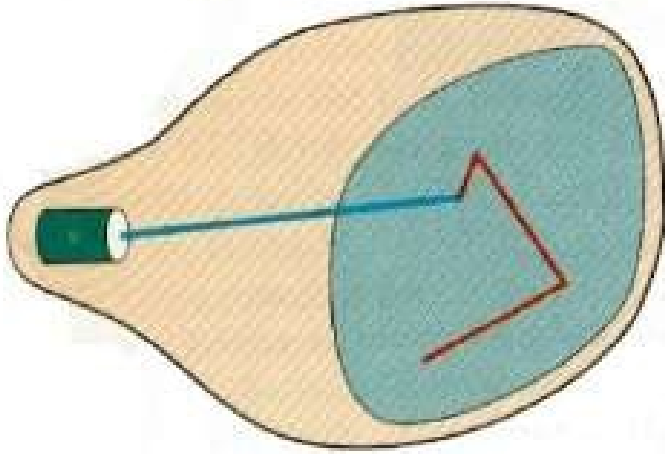
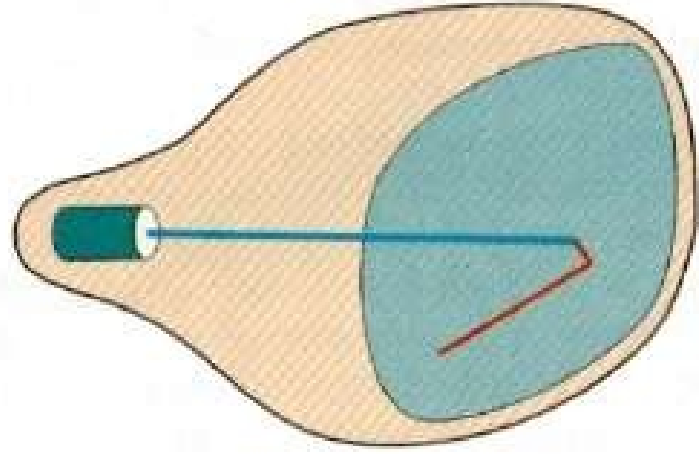
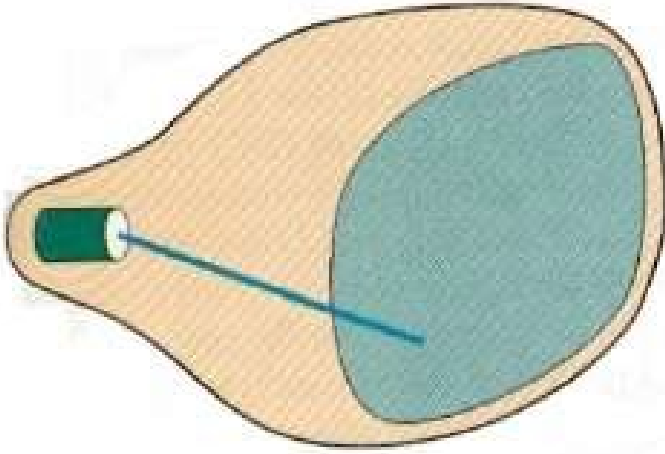
- No refreshing is needed.
- Very complex pictures can be displayed at very high resolution without flicker.

Disadvantages

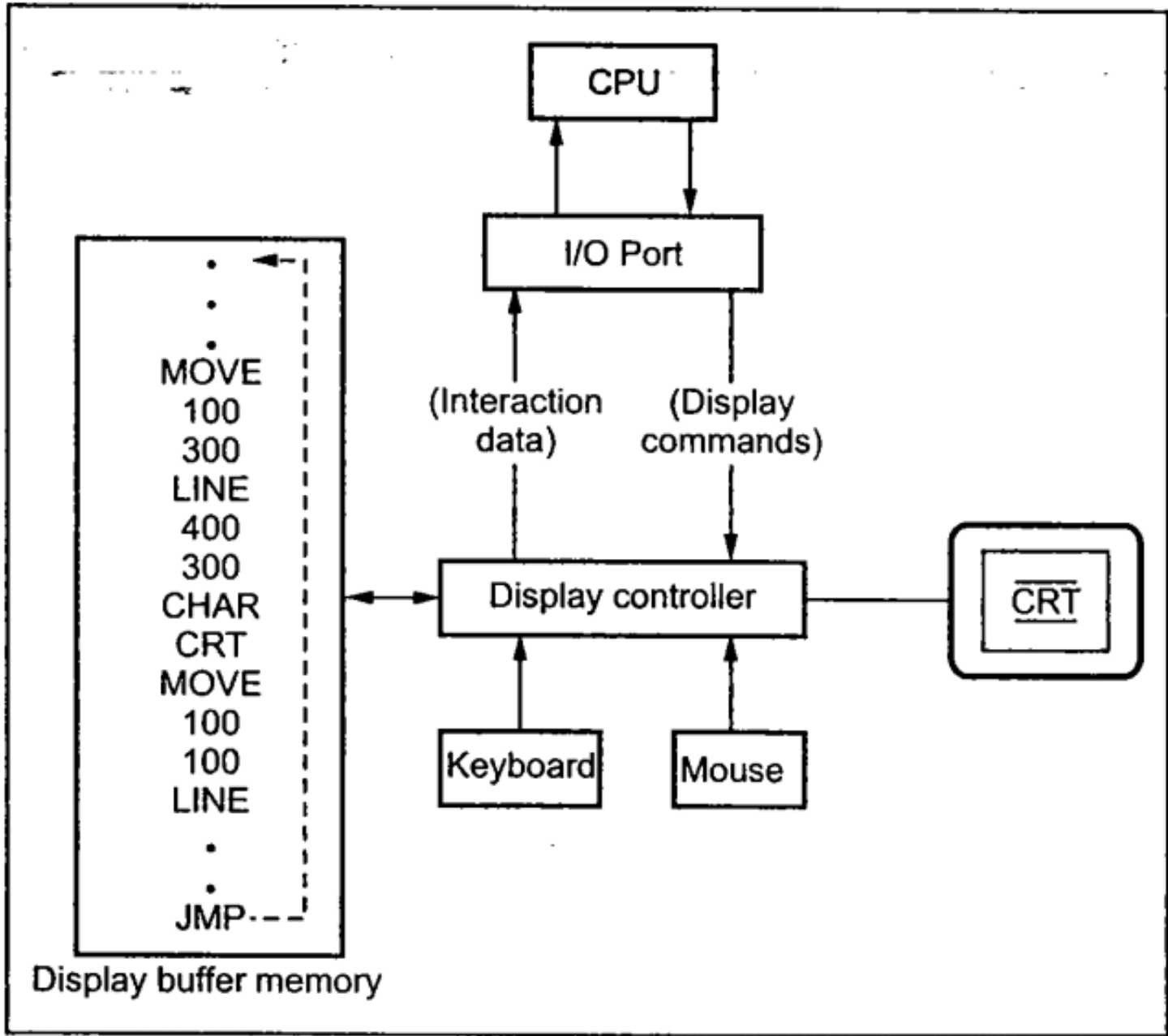
- They ordinarily do not display color.
- Selected part of the picture can not be erased. Modifying any part of image requires redrawing of entire image.
- No animation in DVST. The erasing and redrawing process can take several seconds for complex pictures.

2. Random Scan Displays

- In a random scan display, a CRT has the electron beam directed only to the parts of the screen where a picture is to be drawn.
- Random scan monitors draw a picture one line at a time.
- Random scan display is also called as **Vector display, Stroke – writing or calligraphic displays.**



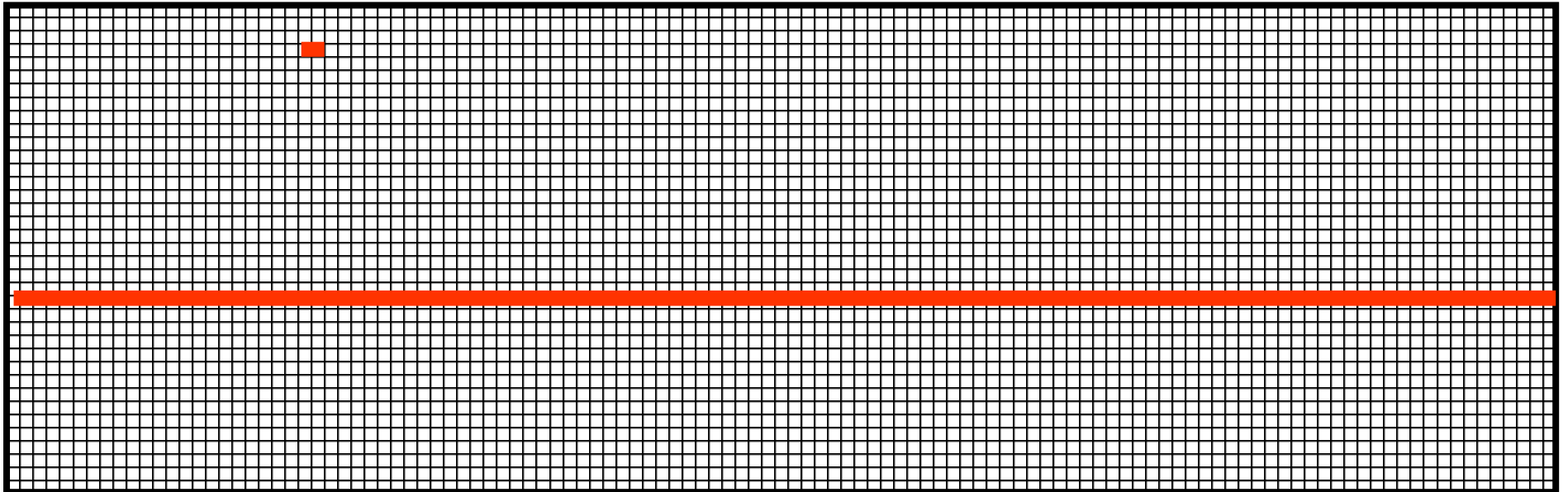
- Picture definition is stored as a set of line-drawing commands in an area of memory referred to as the **Display file (refresh display file / display buffer memory)**
- To display a specified picture, the system cycles through the set of commands in the display buffer memory, drawing each component line in turn.
- After all the line-drawing commands are processed, the system cycles back to the first line command in the list.
- Random-scan displays are designed to draw all the component lines of a picture 30 to 60 times each second.



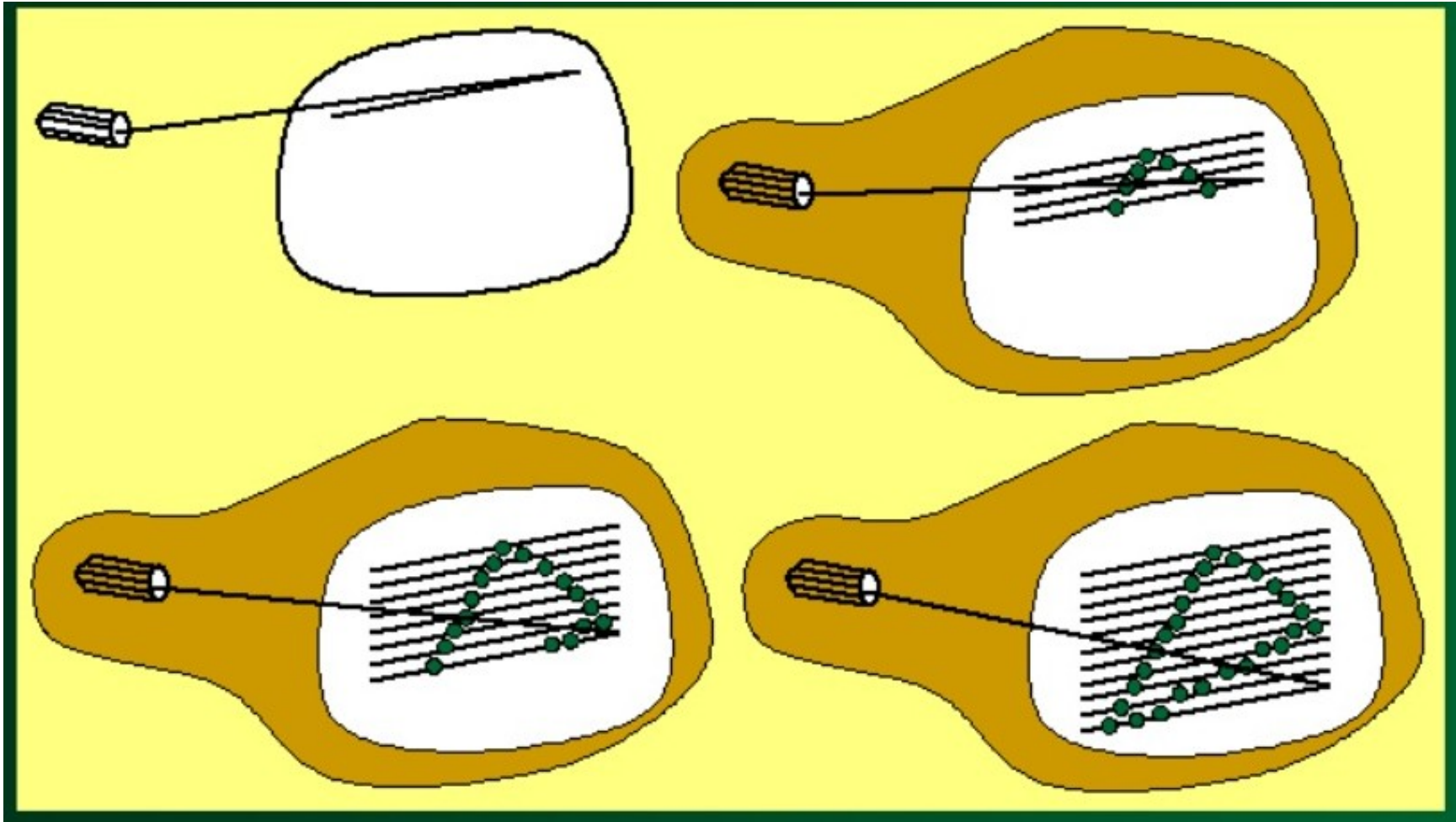
- Random scan displays have higher resolution than raster systems.
- Random displays produce smooth line drawing while a raster system produces jagged lines.
- Random scan displays are designed for line-drawing applications and can not display realistic shaded scenes.

3. Raster Scan displays

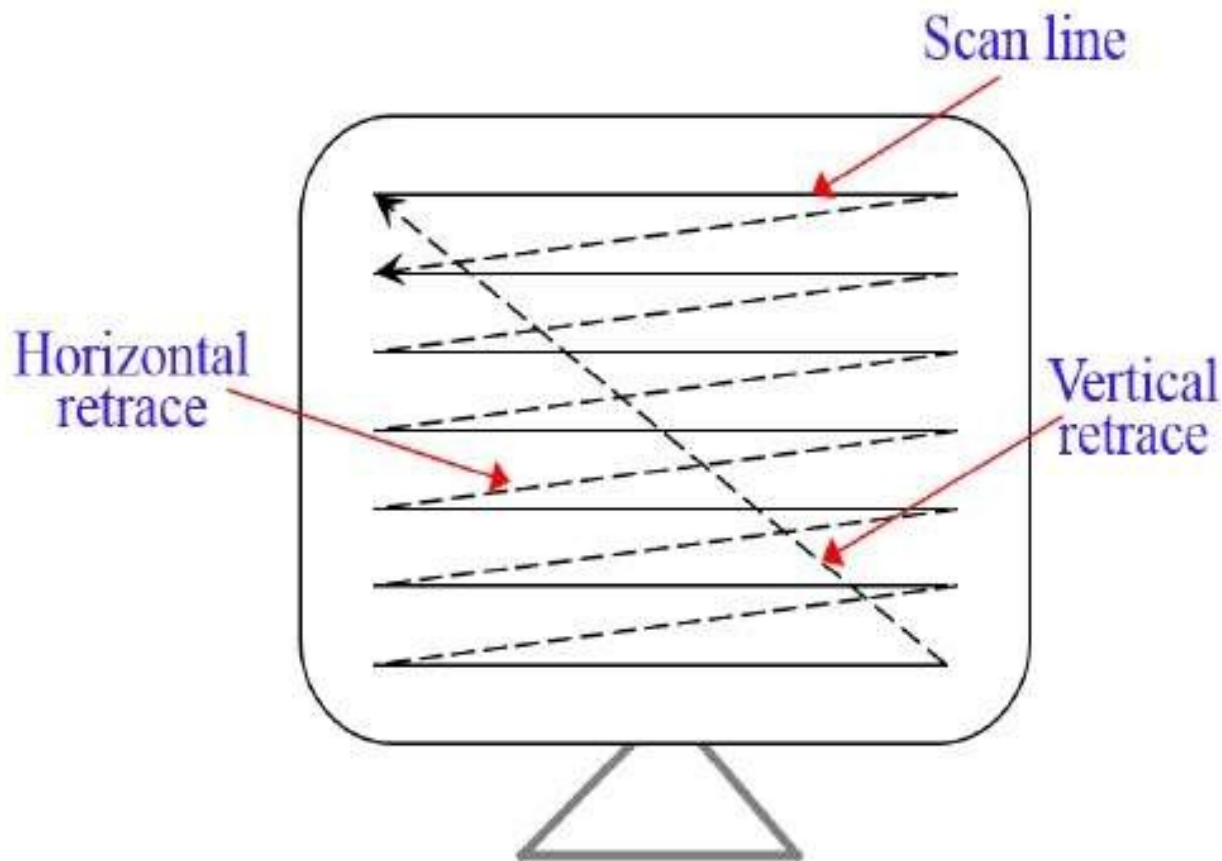
- **Raster:** A rectangular array of points or dots
- **Pixel:** One dot or picture element of the raster
- **Scan Line:** A row of pixels
- Raster Scan is the representation of images as a collection of **pixels**.



- In a raster scan system, the electron beam is swept across the screen, one row at a time from top to bottom.
- As the electron beam moves across each row, the beam intensity is turned on and off to create a pattern of illuminated spots.
- Picture definition is stored in memory area called the **Refresh Buffer or Frame Buffer**. This memory area holds the set of intensity values for all the screen points.
- Stored intensity values are then retrieved from the refresh buffer and “painted” on the screen one row (scan line) at a time.



- Each screen point is referred to as a **pixel (picture element) or pel.**
- At the end of each scan line, the electron beam returns to the left side of the screen to begin displaying the next scan line.



- A black-and-white system: each screen point is either on or off, so only one bit per pixel is needed to control the intensity of screen positions.

On a black-and-white system with one bit per pixel, the frame buffer is called **bitmap**.

- For system with multiple bits per pixel, the frame buffer is called **pixmap**.

- A raster system produces jagged lines.
- A raster system requires larger file size as compared to random display system.

Application of Computer Graphics

- **Entertainment and Media:**

- Video Games
- Movies and Animation
- Virtual Reality (VR) and Augmented Reality (AR)

- **Design and Visualization:**

- Graphic Design: Computer graphics are used in designing logos, posters, brochures, and other marketing materials.
- Architectural Visualization: Architects use computer graphics to create detailed 3D models and visualizations of buildings and interiors.
- Product Design: Graphics aid in creating and visualizing product prototypes and designs.

- **Engineering and Manufacturing:**

- CAD (Computer-Aided Design): Engineers use computer graphics for designing and modeling complex machinery and structures.
- Simulation: Graphics play a role in simulating real-world scenarios and testing designs before physical implementation.

- **Medical Imaging and Healthcare:**

- Medical Visualization: Graphics are used to visualize medical data from imaging technologies like MRI, CT scans, and X-rays, assisting in diagnosis and treatment planning.
- Surgical Simulation: Graphics-based simulations help train surgeons and practice complex procedures in a virtual environment.

- **Education and Training:**

- Educational Software: Computer graphics aid learning by providing interactive simulations, virtual labs, and visual explanations.
- Training Simulations: Graphics-based simulations are used in fields like aviation, military, and healthcare for training purposes.

- **Scientific Visualization:**

- Data Visualization: Graphics help researchers and scientists visualize complex data sets and patterns, aiding in analysis and discovery.
- Astrophysics and Molecular Modeling: Graphics are used to visualize large-scale astrophysical phenomena and molecular structures.

- **Advertising and Marketing:**

Digital Advertising: Graphics enhance online advertising with visually engaging content.

- **Digital Art:**

Artists create digital paintings, illustrations, and multimedia art using computer graphics tools.

- **Geographical Information Systems (GIS):**

Mapping and GIS: Computer graphics are used to create digital maps, visualize geographic data, and analyze spatial relationships.

Scan Conversion

- The process of representing continuous graphics object as a collection of discrete pixels is called **Scan Conversion**.
- In simple words, scan conversion is to figure out which pixels to fill in order to generate picture on a screen. While **shading** determines a color for each filled pixel.
- It is the responsibility of graphics system or the application program to convert each primitive from its geometric definition into a set of pixels. This conversion task is generally referred to as a **scan conversion** or **rasterization**.

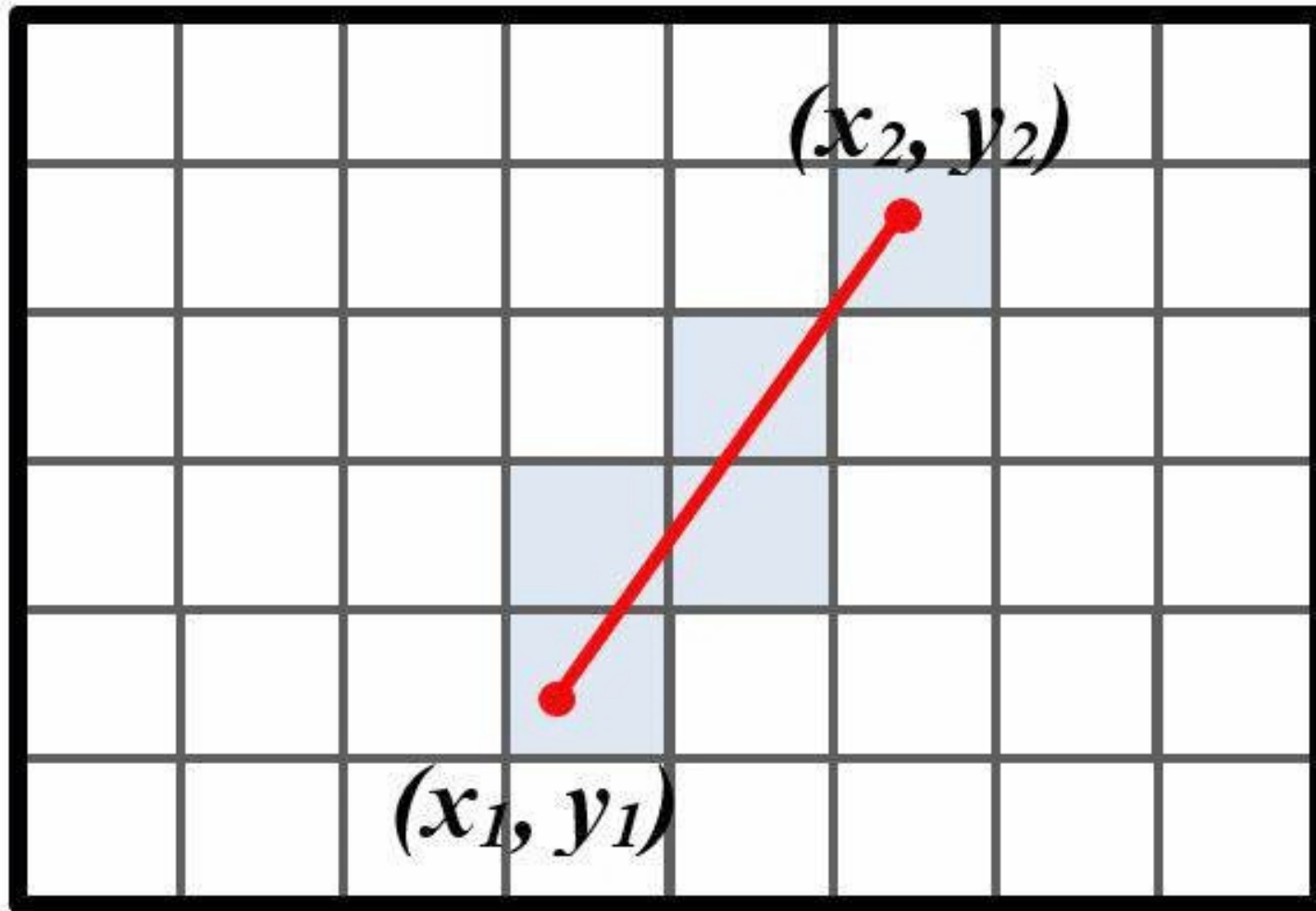
DDA Line Drawing algorithm

1. Take end points of the line (x_1, y_1) & (x_2, y_2)

```
2. Compute:  dx = x2-x1;
              dy = y2-y1;
              if (abs(dx) >= abs(dy))
                  len=abs(dx);
              else
                  len=abs(dy);
```

```
3. xin = dx / len;
   yin = dy / len;
```

```
4. x = x1;
   y = y1;
   int i=1;
   while (i<=len)
       {
           putpixel (Round(x),Round(y),RED);
           x=x+xin;
           y=y+yin;
           i++;
       }
```



Bresenham Line Drawing algorithm

- Take end points from user: (x_1, y_1) and (x_2, y_2)
- Initialize variables: $x=x_1$; $y=y_1$;
- $dx=abs(x_2-x_1)$; $dy=abs(y_2-y_1)$;
- $s_1=Sign(x_2-x_1)$; $s_2=Sign(y_2-y_1)$;
- Interchange dx and dy depending on slope of the line

If $dy > dx$ then

$temp=dx$

$dx=dy$

$dy=temp$

 Interchange=1

else

 Interchange=0

- $e = 2 * dy - dx$ //initialise error term

main loop

for i=1 to dx

setpixel(x,y);

while (e>0)

if Interchange=1 then

x=x+s1;

else

y=y+s2;

end if

e = e-2*dx

end while

if Interchange=1 then

y=y+s2;

else

x=x+s1;

end if

e = e+2*dy

next i;

end for

Bresenham Circle Drawing algorithm

1) Read the x and y coordinates of center: (**centx**, **centy**)

2) Read the radius of circle: (**r**)

3) Initialize,

$x = 0;$

$y = r;$

4) Initialize decision parameter: $p = 3 - (2*r)$

5) do {

setpixel(x,y);

If ($p < 0$)

{

$p = p + (4*x) + 6;$

}

else {

$p = p + [4*(x-y)] + 10;$

$y = y - 1;$

}

$x = x + 1;$

} while($x < y$)


```
putpixel(centx+x, centy-y, 4);  
putpixel(centx-x, centy-y, 4);  
putpixel(centx+x, centy+y, 4);  
putpixel(centx-x, centy+y, 4);  
putpixel(centx+y, centy+x, 4);  
putpixel(centx+y, centy-x, 4);  
putpixel(centx-y, centy+x, 4);  
putpixel(centx-y, centy-x, 4);
```

Midpoint Circle Drawing algorithm

1) Read the x and y coordinates of center: (**centx, centy**)

2) Read the radius of circle: (**r**)

3) Initialize,

$x = 0;$

$y = r;$

4) Initialize decision parameter: $\mathbf{p = 1 - r};$

5) do {

setpixel(x,y);

If ($p < 0$)

{

$p = p + 2 * x + 3;$

}

else {

$p = p + 2 * x - 2 * y + 5;$

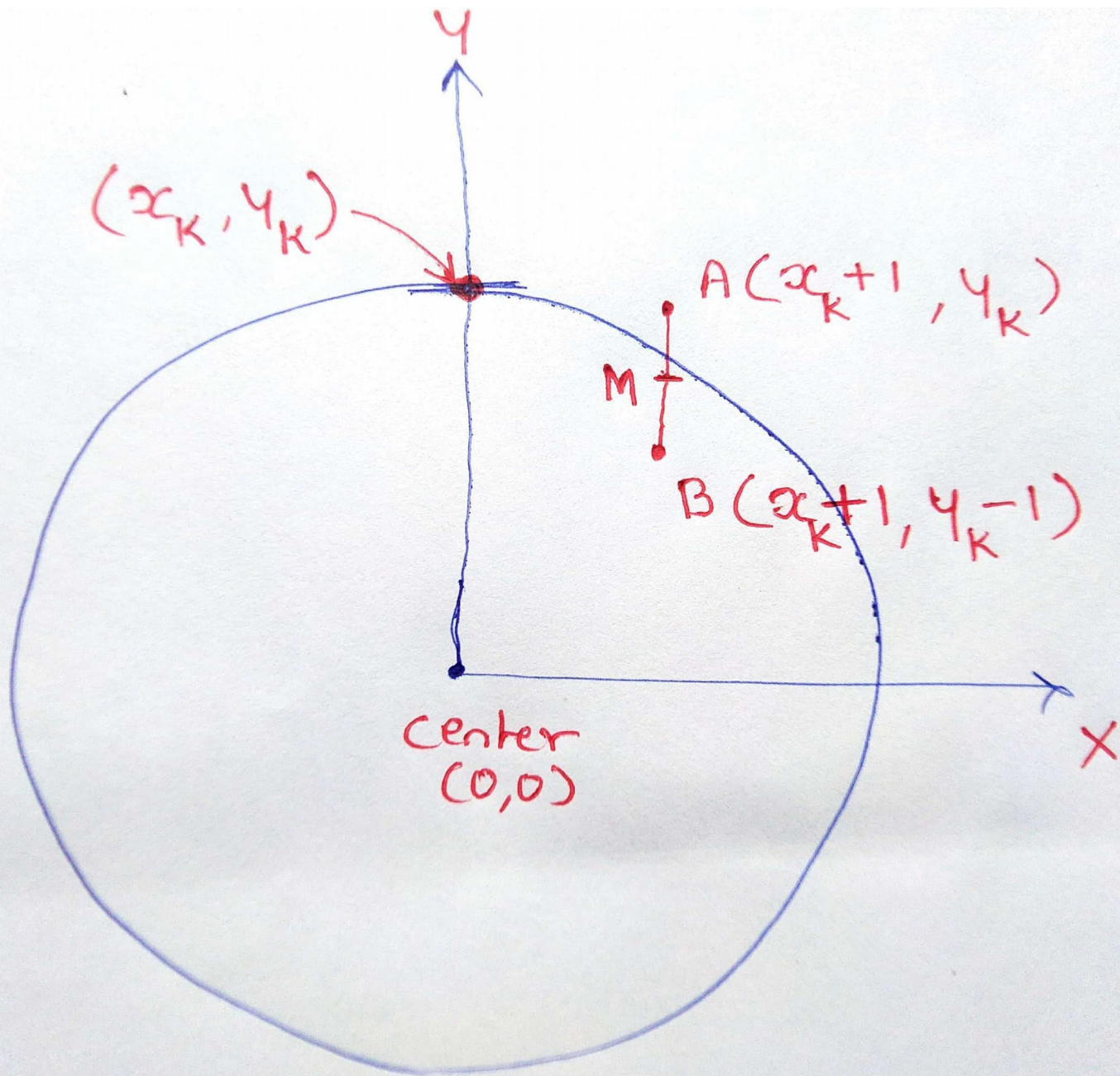
$y = y - 1;$

}

$x = x + 1;$

} while($x < y$)

```
putpixel(cen $x$ + $x$ , cen $y$ - $y$ , 4);  
putpixel(cen $x$ - $x$ , cen $y$ - $y$ , 4);  
putpixel(cen $x$ + $x$ , cen $y$ + $y$ , 4);  
putpixel(cen $x$ - $x$ , cen $y$ + $y$ , 4);  
putpixel(cen $x$ + $y$ , cen $y$ + $x$ , 4);  
putpixel(cen $x$ + $y$ , cen $y$ - $x$ , 4);  
putpixel(cen $x$ - $y$ , cen $y$ + $x$ , 4);  
putpixel(cen $x$ - $y$ , cen $y$ - $x$ , 4);
```



- Consider point (x_k, y_k) is on circle
- In octant 1, we have to decide next ~~px~~ pixel to be printed i.e. either pixel A or pixel B.
- Consider m is the midpoint of A & B. coordinates of m will be,

$$m \left(\frac{(x_k+1) + (x_k+1)}{2}, \frac{(y_k) + (y_k-1)}{2} \right)$$

$$= m \left(\underbrace{x_k+1}_{x \text{ coordinate}}, \underbrace{y_k - \frac{1}{2}}_{y \text{ coordinate}} \right)$$

we have eqⁿ of circle, at origin,

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

$$d_k = x^2 + y^2 - r^2 \quad (\text{decision parameter})$$

if ① $d_k = 0$: point lies on circle

② $d_k < 0$: point lies inside circle

③ $d_k > 0$: point lies outside circle

- If we put x & y coordinates of midpoint in above eqⁿ, we can check whether midpoint is on/inside/outside the circle.

- If midpoint is inside the circle, ~~point~~ means point A is more close to actual circle than point B. Hence we select point A as a next point after (x_k, y_k)

- If midpoint is outside the circle, means point B is more close to actual circle than A. Hence we select point B after (x_k, y_k) as next point

Now, put coordinates of of midpoint in eqⁿ ①

$$d_k = (x_k + 1)^2 + \left(y_k - \frac{1}{2}\right)^2 - r^2 \quad \text{--- (III)}$$

Now, next decision parameter d_{k+1} is calculated by replacing k by $k+1$;

$$d_{k+1} = (x_{k+1} + 1)^2 + \left(y_{k+1} - \frac{1}{2}\right)^2 - r^2 \quad \text{--- (IV)}$$

Now, calculate $(d_{k+1} - d_k)$

$$d_{k+1} - d_k = \left[(x_{k+1} + 1)^2 + \left(y_{k+1} - \frac{1}{2}\right)^2 - r^2 \right] - \left[(x_k + 1)^2 + \left(y_k - \frac{1}{2}\right)^2 - r^2 \right]$$

$$d_{k+1} - d_k = (x_{k+1} + 1)^2 + \left(y_{k+1} - \frac{1}{2}\right)^2 - (x_k + 1)^2 - \left(y_k - \frac{1}{2}\right)^2 \quad \text{--- (V)}$$

① Case - 1 $d_k < 0$

So now in eqn (v) consider case - 1

i.e. $d_k < 0$ $\Rightarrow x_{k+1} = x_k + 1$

Point A
is selected

$y_{k+1} = y_k$

} Put these values in eqn (v)

$$\begin{aligned} d_{k+1} - d_k &= (x_k + 2)^2 + (y_k - \frac{1}{2})^2 - (x_k + 1)^2 - (y_k - \frac{1}{2})^2 \\ &= x_k^2 + 4x_k + 4 - x_k^2 - 2x_k - 1 \end{aligned}$$

$$d_{k+1} - d_k = 2x_k + 3$$

$d_{k+1} = d_k + 2x_k + 3$

② Case-2 $\Rightarrow d_k > 0$

for $d_k > 0 \Rightarrow x_{k+1} = x_k + 1$

Point B
is selected

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

} Put these values in eqn (v)

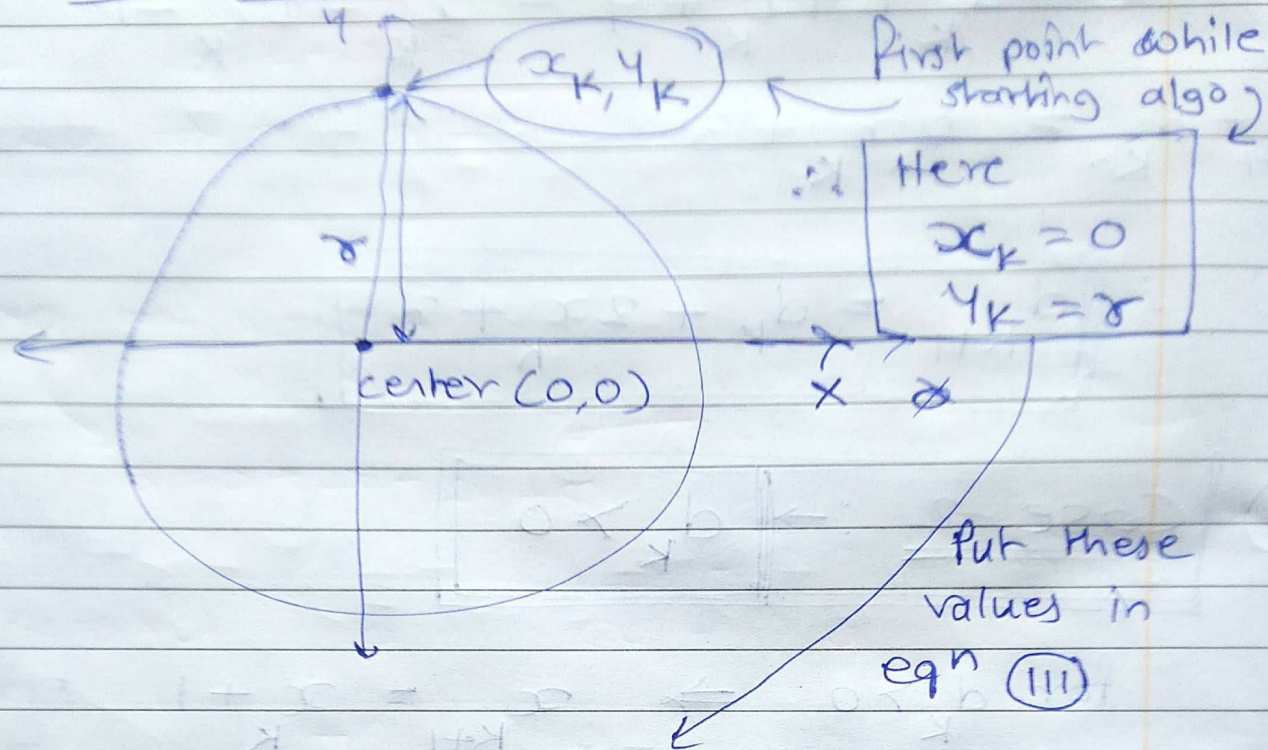
$$d_{k+1} - d_k = (x_k + 2)^2 + (y_k - \frac{3}{2})^2 - (x_k + 1)^2 - (y_k - \frac{1}{2})^2$$

$$= \cancel{x_k^2} + 4x_k + 4 + \cancel{y_k^2} - 3y_k + \frac{9}{4} - \cancel{x_k^2} - 2x_k - 1 - \cancel{y_k^2} + y_k - \frac{1}{4}$$

$$d_{k+1} - d_k = 2x_k - 2y_k + 5$$

$$\therefore d_{k+1} = d_k + 2x_k - 2y_k + 5$$

Now initial decision parameter (d_0):



$$d_0 = (0+1)^2 + \left(r - \frac{1}{2}\right)^2 - r^2$$

$$= \frac{5}{4} - r$$

approximate "1"

$$d_0 = 1 - r$$

DDA Circle Drawing algorithm

- 1) Read the x and y coordinates of center: (**centx**, **centy**)
- 2) Read the radius of circle: (**r**)
- 3) Initialize,

$x = 0;$

$y = r;$

We have, $2^{(n-1)} \leq r \leq 2^n$

$\epsilon = 2^{-n};$

- 4) do {

setpixel(x,y);

$x = x + \epsilon * y;$

$y = y - \epsilon * x;$

} while(x < y)

```
putpixel(centx+x, centy-y, 4);  
putpixel(centx-x, centy-y, 4);  
putpixel(centx+x, centy+y, 4);  
putpixel(centx-x, centy+y, 4);  
putpixel(centx+y, centy+x, 4);  
putpixel(centx+y, centy-x, 4);  
putpixel(centx-y, centy+x, 4);  
putpixel(centx-y, centy-x, 4);
```

Line styles: Solid, dotted, dashed and thick lines

DDA Line drawing (Solid line)

```
dx=x2-x1;   dy=y2-y1;
  if(abs(dx)>=abs(dy))
    len=abs(dx);
  else
    len=abs(dy);
  xin=dx/len;  yin=dy/len;
x=x1;  y=y1;
int gd=DETECT,gm;
initgraph(&gd,&gm,NULL);
x=x+0.5;   y=y+0.5;
putpixel(x,y,9);
int i=1;
  while(i<=len)
  {
    putpixel(x,y,9);
    x=x+xin;
    y=y+yin;
    i++;
  }
```

Dotted line

```
while(i<=len)
{
    if(i%5==0)
    {
        putpixel(x,y,9);
    }
    x=x+xin;
    y=y+yin;
    i++;
}
```

Dashed line

```
while(i<=len)
{
    if(i%8<4)
        {
            putpixel(x,y,9);
        }
    x=x+xin;
    y=y+yin;
    i++;
}
```


Thick line

```
while(i<=len)
{
    for(j=0;j<t;j++)    // where t is the thickness of line
    {
        putpixel(x,y+j,9);
    }
    x=x+xin;
    y=y+yin;
    i++
}
```

Dashed-Dotted line

```
while(i<=len)
{
    if(i%9<4 || i%9==6)
    {
        putpixel(x,y,9);
    }
    x=x+xin;
    y=y+yin;
    i++;
}
```