# CSE-423 Computer Graphics

Dr. Md. Haider Ali
Prof. CSE, DU

# What is Computer Graphics?

... any use of computers to create and manipulate images.

Modeling

# What is Computer Graphics?

... any use of computers to create and manipulate images.

Rendering

# Some applications of CG

| | |
|---|---|
| Movie industry: | Computer animation, Special effects |
| Games | Computer animation, AI |
| Immersive environments | Virtual reality, e.g. virtual tourism<br>Augmented reality |
| Information visualization; visual analytics | Business, Scientific |
| Computer aided design | Architectural design |
| Digital photography and cinematography | Changing lighting of a scene, Changing depth of the field, Taking a picture without closed eyes, Inferring 3D from 2D |
| Medical imagery | Surgery Simulation |
| (Digital) publishing | Architectural / landscape visualization, Art |

# Course objectives

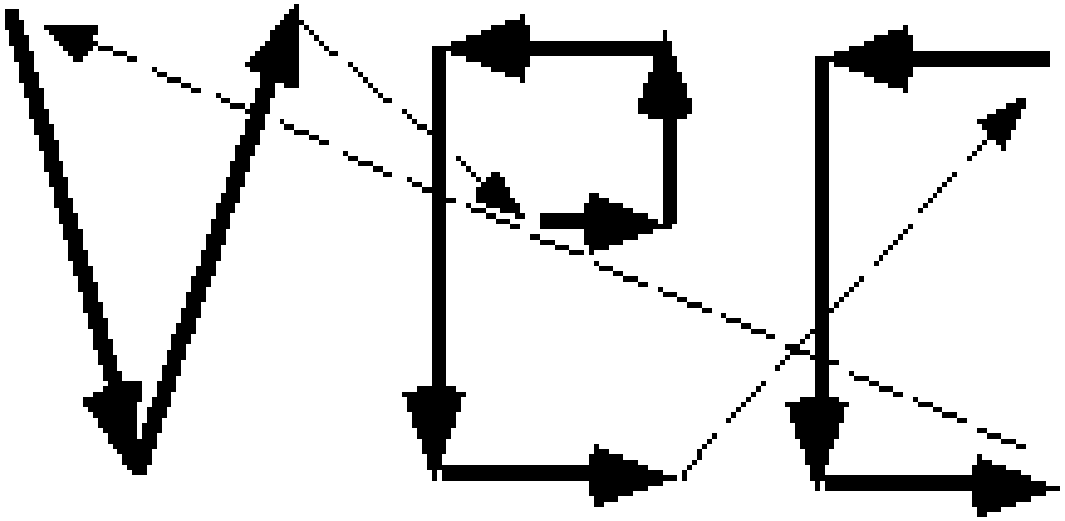
**How to make graphics software and programs:**

- Basic concepts of synthesizing images using computers
- How to write programs with graphics components
- Things behind making realistic image

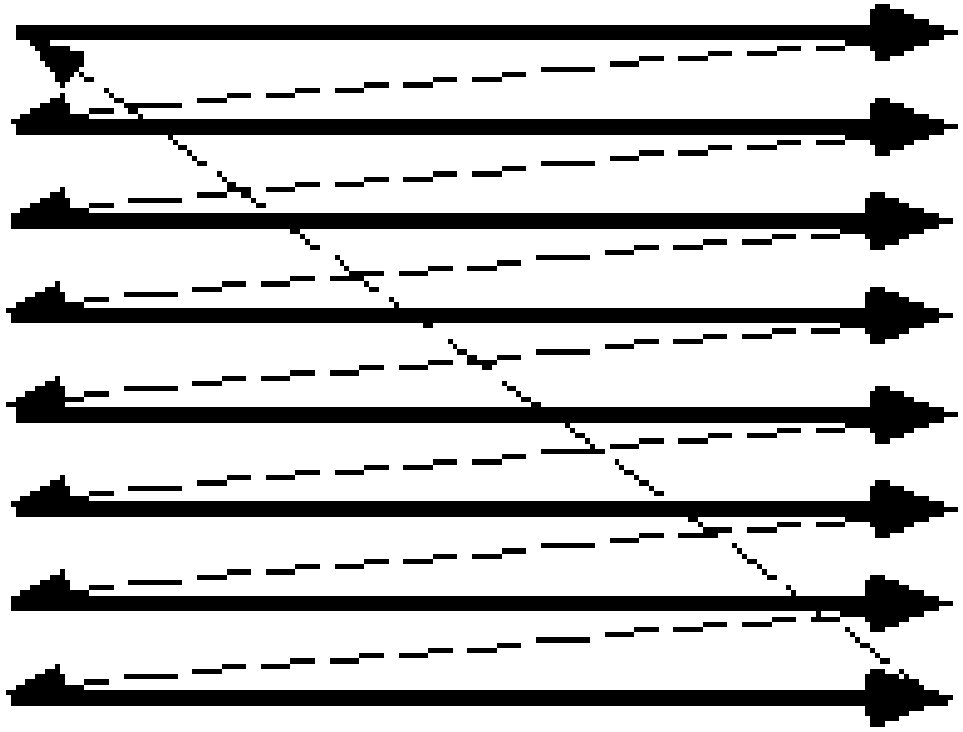
❖ **Not !**

- How to use Paint and Imaging packages (Adobe Photoshop)
- How to use CAD packages (AutoCAD)
- How to use Graphics packages (3D Studio MAX, MAYA)
- Graphics Hardware Design
- Artistic aspects of image creation

# Display System



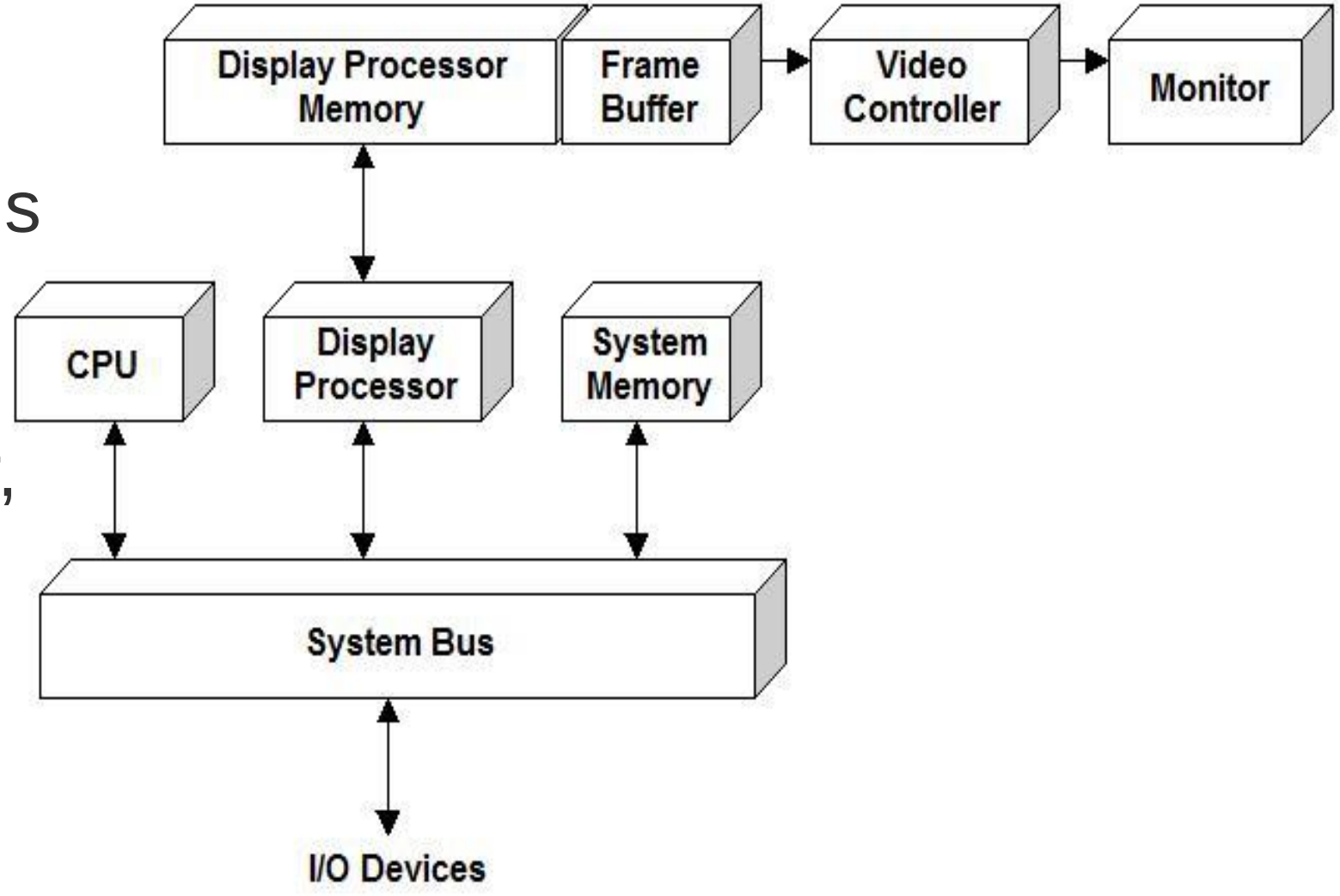
Vector



Raster

# Architecture of a Display System

- The components of a raster system contains display processor, display-processor memory, frame buffer, video controller, and input/output devices.

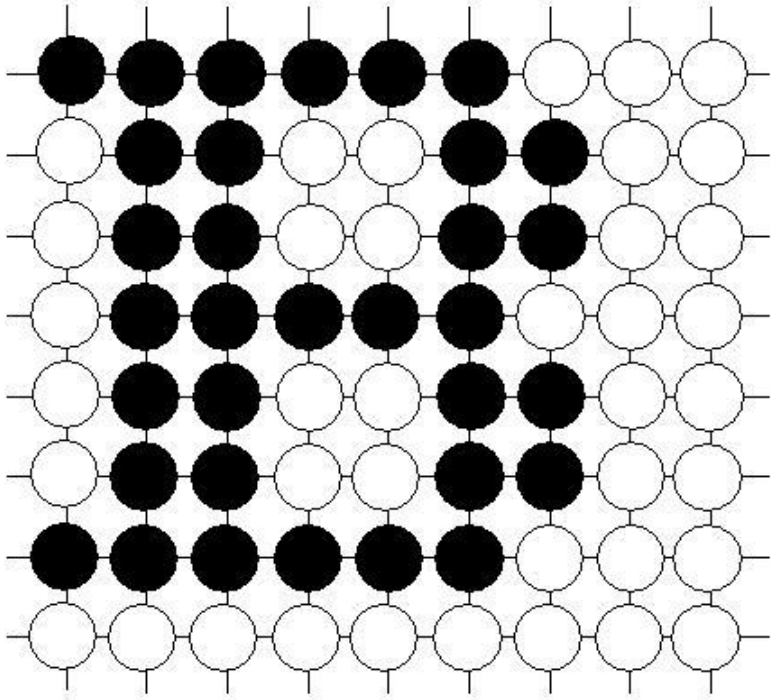


# Display Processor

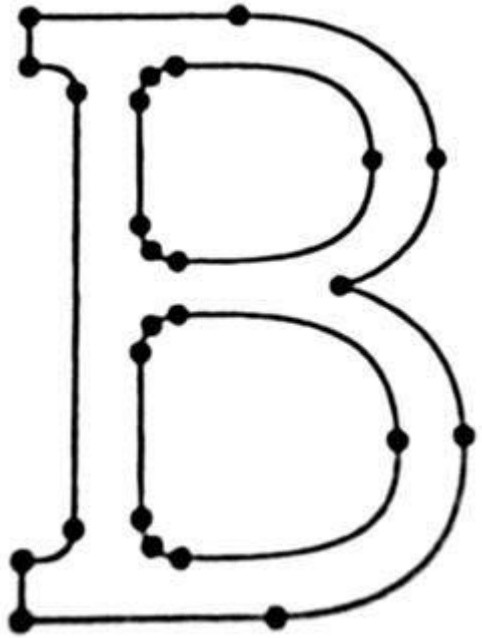
- Purpose: frees the CPU from the graphics routine task.
- Major task: digitizes a picture definition given in an application program into a set of pixel values for storage in the frame buffer.
- This digitization process is called scan conversion.
- Straight lines and other geometric objects are scan converted into a set of discrete points, corresponding to screen pixel locations.

# Display Processor

Characters can be defined with rectangular pixel grids, or they can be defined with outline shapes. The array size for character grids can vary from about 5x7 to 9x12 or more for higher quality displays.



pixel grid



outline shapes

# Display Processor

- A character grid is displayed by superimposing the rectangular grid pattern into the frame buffer at a specified coordinate position (Bit-map font, Raster Graphics)

- For characters that are defined as outlines, the shapes are scanned converted into the frame buffer by locating the pixels positions closest to the outline (True-type font, Vector and Raster Graphics)

# Frame-Buffer

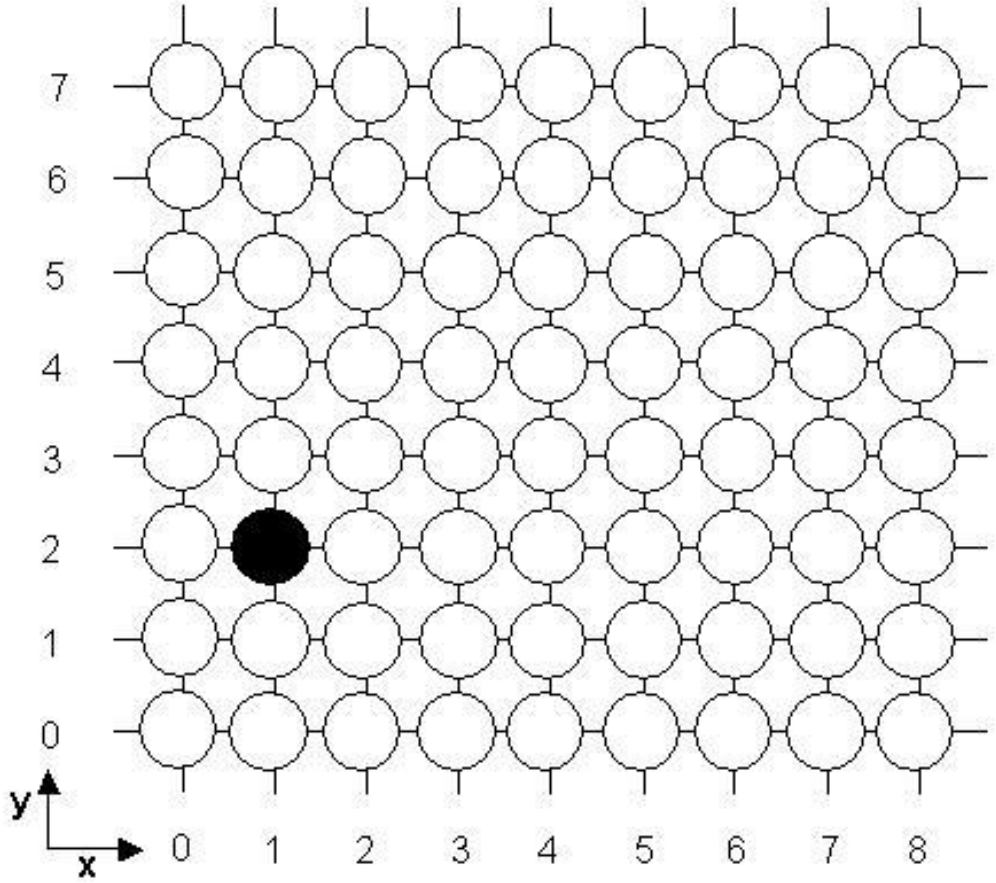
- Each screen pixel corresponds to a particular entry in a 2D array residing in memory. This memory is called a frame buffer or a bit map.
- The number of rows in the frame buffer equals to the number of raster lines on the display screen.
- The number of columns in this array equals to the number of pixels on each raster line.
- The term pixel is also used to describe the row and the column location in the frame buffer array that corresponds to the screen location. A 1920x1080 display screen requires 2,073,600 pixel memory locations.

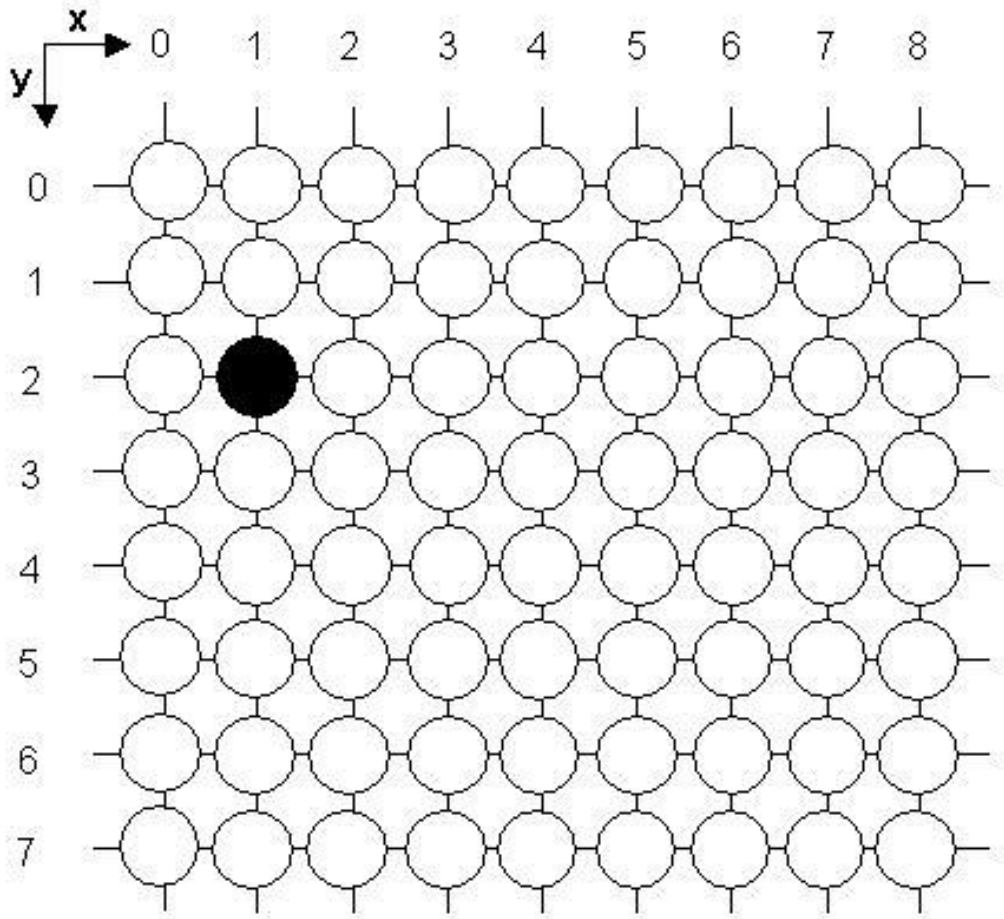
# Frame-Buffer

- Whenever we wish to display a pixel on the screen, a specific value is placed into the corresponding memory location in the frame buffer array.
- Each screen pixel’s location and corresponding memory’s location in the frame buffer is accessed by nonnegative integer coordinate pair (x, y).
- The x value refers to the column, the y value to the row position.

# Frame-Buffer

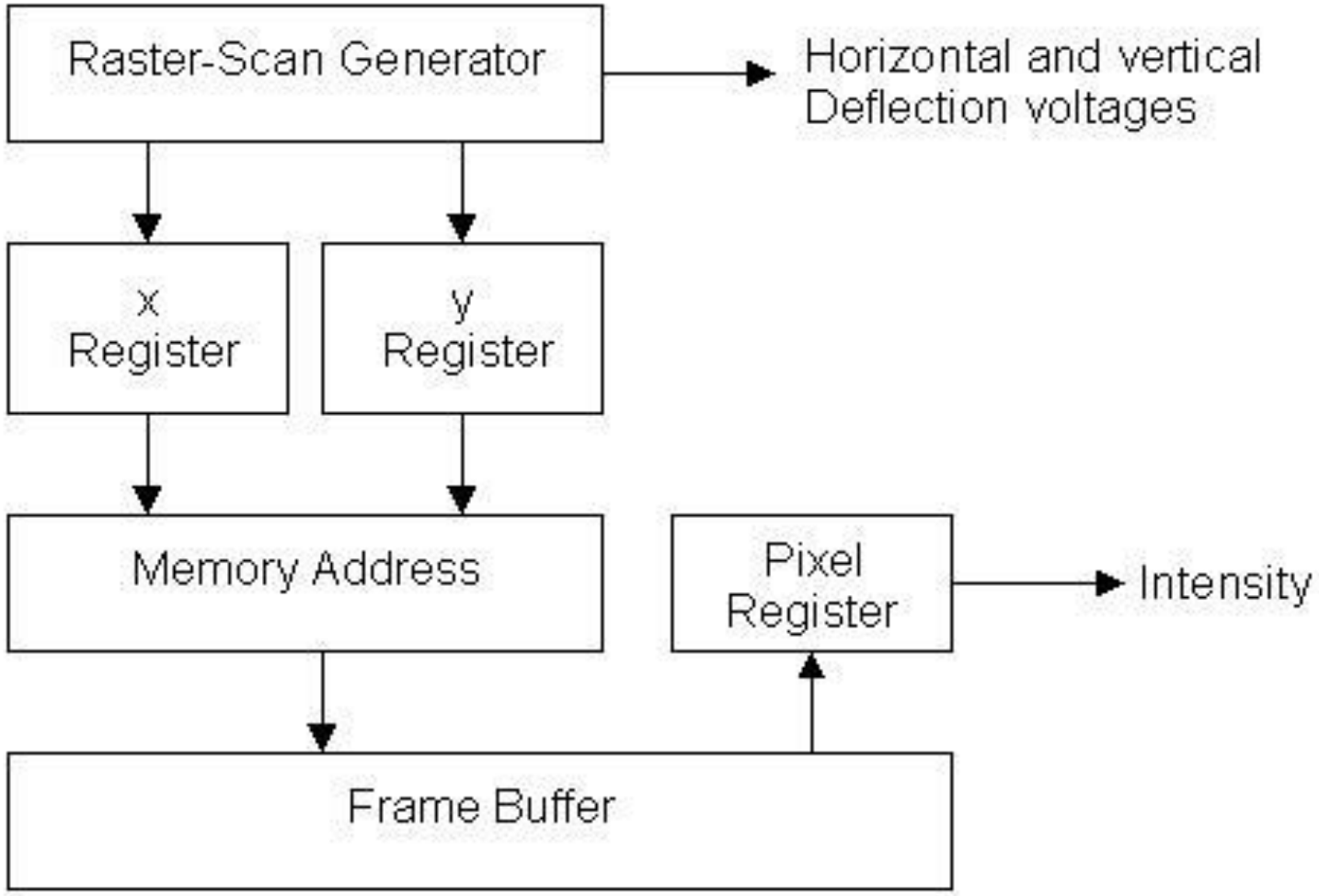
- The origin of this coordinate system is positioned at the bottom-left corner of the screen or it is positioned at the upper-left corner of the screen.



# Video controller

- Video controller is used to control the operation of the display device (Monitor/Screen).
- Video controller accesses the frame buffer to refresh the screen.
- In figure, the basic refresh operations of the video-controller are shown.



# Video controller

- Two registers are used to store the coordinates of the screen pixels.
- Initially, the x register is set to 0 and the y register is set to the value for the top scan line.
- The contents of the frame buffer at this pixel position are then retrieved and used to set the intensity of the CRT beam.
- Then the x register is incremented by 1, and the process is repeated for the next pixel on the top scan line.

# Video controller

- This procedure is continued for each pixel along the top scan line.
- After the last pixel on the top scan line has been processed, the x register is reset to 0 and the y register is set to the value for the next scan line down from the top of the screen.
- Pixels along this scan line are then processed in turn, and the procedure is repeated for each successive scan line.
- After cycling through all pixels along the bottom scan line (y=0), the video controller resets the registers to the first pixels position on the top scan line and the refresh process starts over.
- The screen must be refreshed at a rate of at least 60 frames per second.

# Line Drawing Algorithm

# Digital Differential Analyzer (DDA)

- The line (vector) generation algorithms which determine the pixels that should be turned ON are called as digital differential analyzer (DDA).
- It is one of the techniques for obtaining a rasterized straight line.
- This algorithm can be used to draw the line in all the directions/quadrants.

# DDA Algorithm

**Assumption:**

1. $(x_0, y_0)$ and $(x_1, y_1)$ are the two end-points of a line and are not equal.
2. (Symmetric round-off) integer function is used to convert the real values into integer value. *e.g.,* if value is <8.5 then it is converted to 8 and if value is >8.5 then it is converted to 9.
3. Sign function is used which returns +1, -1 for the arguments ≥0, <0. *e.g.*, sign(5) = +1,
   sign(-3) = -1,

# Contd….

**Step 1:** Approximate the line length

$$\begin{aligned}
&\textit{if } abs(x_1 - x_0) >= abs(y_1 - y_0) \textit{ then} \\
&\quad length = abs(x_1 - x_0) \\
&\textit{else} \\
&\quad length = abs(y_1 - y_0) \\
&\textit{end if}
\end{aligned}$$

**Step 2:** Decide increment

$$\Delta x = (x_1 - x_0) / length$$

$$\Delta y = (y_1 - y_0) / length$$

# Contd….

**Step 3:** Decide flow

$x = x_0$

$y = y_0$

**Step 4:** Begin main loop

*i = 0*

*while (i <= length)*

*drawpixel (intger x+ 0.5 * sign(x), integer y+ 0.5 * sign(y))*

$x = x + \Delta x$

$y = y + \Delta y$

*i = i + 1*

*end while*

**finish**

# Examples Q1

Rasterize the line from (–3, 5) to (3, -5) using DDA algorithm.

Solution: Given, $x_0 = -3.0, y_0 = 5.0, x_1 = 3.0, y_1 = -5.0$

**Step-1:** $abs(x_1 - x_0) = abs(3.0 + 3.0) = 6.0,$

$abs(y_1 - y_0) = abs(-5.0 - 5.0) = 10.0$

since. $abs(y_1 - y_0) > abs(x_1 - x_0),\ length = 10.0$

**Step-2:** $\Delta x = (x_1 - x_0)/length = 0.6$

$\Delta y = (y_1 - y_0)/length = -1.0$

**Step-3:** $x = x_0 + 0.5*sign(x) = -3.0 + 0.5*(-1) = -3.5$

$y = y_0 + 0.5*sign(y) = 5.0 + 0.5*(1) = 5.5$

# Step-4: Begin main-loop

| i | Pixel coordinate | X actual | Y actual |
|---|---|---|---|
| 0 | (-3, 5) | -3.0 | 5.0 |
| 1 | (-2, 4) | -2.4 | 4.0 |
| 2 | (-2, 3) | -1.8 | 3.0 |
| 3 | (-1, 2) | -1.2 | 2.0 |
| 4 | (-1, 1) | -0.6 | 1.0 |
| 5 | (0, 0) | 0.0 | 0.0 |
| 6 | (1, -1) | 0.6 | -1.0 |
| 7 | (1, -2) | 1.2 | -2.0 |
| 8 | (2, -3) | 1.8 | -3.0 |
| 9 | (2, -4) | 2.4 | -4.0 |
| 10 | (3, -5) | 3.0 | -5.0 |

# Code for the previous problem

```
import math
def sign(a):
        if a >= 0:
                return 1.0
        else:
                return -1.0
def dda():
        x0, y0, x1, y1 = -3.0, 5.0, 3.0, -5.0
        dx = x1 – x0
        dy = y1 – y0
        length = abs(dx) if abs(dx) >= abs(dy) else abs(dy)
        dx /= length
        dy /= length
        print("dx =", dx, "and dy =", dy)
        i = 0
        while i <= length:
                print("(", int(x + sign(x) * 0.5), ",", int(y + sign(y) * 0.5), ")", "x = ", x, "y = ", y)
                # or simply print("(", round(x), ",", round(y), ")", "x = ", x, "y = ", y)
                x += dx
                y += dy
                i += 1
dda()
```

Q.2 Rasterize the line for the equation x/2 + y/10 = 1 using DDA line algorithm.

**Solution:** Given is $x/2 + y/10 = 1$ --------eq. (1)

Put $x = 0$ in eq. 1 , then $y = 10$

⁂ the 1st end point $(x_0, y_0)$ is $(0, 10)$

Put $y = 0$ in eq. 1 , then $x = 2$

⁂ the 2nd end point $(x_1, y_1)$ is $(2, 0)$

Here, $x_0=0$, $y_0=10$, $x_1=2$, $y_1=0$

Do yourself

# Advantages and Disadvantages of DDA

- Advantages :
  1. Simplest line drawing algorithm
  2. No special skills required for its implementation
  3. DDA draws the line faster than drawing the line by directly using the line equation.
- Disadvantages :
  1. It dependents on orientation which makes the end point accuracy poor.
  2. It requires floating point addition to determine each successive point which is time consuming.
  3. Error due to limited precision in floating point representation may cause calculated points to shift away from their actual position when the line is relatively long.