UNIT 1

PART A

pixel - smallest addressable unit (on digital raster or display screen)

video display devices - Raster Scan (more memory; entire pixel grid/frame buffer) and Random Scan (Vector Scan; drawing commands, less memory)

CRT - Cathode Ray Tube (fig. 1)

area filling primitives - (1) Polygon fill areas and (2) Curved boundary fill areas (or seed-filled regions)

Polygon inside testing or Polygon scan conversion identifies **polygon's interior pixels**

types of refresh buffers - RAM (VRAM or DRAM)

Object geometry - continuous; **raster representation** - discrete approximation

DDA

1. Calculate Δx, Δy and steps

$\Delta x=x_2-x_1$; $\Delta y=y_2-y_1$; steps=max(|Δx,Δy|)

increments: dx=Δx/steps; dy=Δy/steps

start: x_1,y_1; iterations x=x+dx, y=y+dy

Disadvantage:
Floating-point operations are computationally slower and hardware-intensive.

Bresenham's Line Drawing Algorithm

find Δx and Δy

$p_0=2\Delta y-\Delta x$

if $p_k<0$; $p_{k+1}=p_k+2\Delta y$; $x_{k+1}=x_k+1$; $y_{k+1}=y_k$

if $p_k>0$; $p_{k+1}=p_k+2\Delta y-2\Delta x$; $x_{k+1}=x_k+1$; $y_{k+1}=y_k+1$

Mid-Point Circle Drawing

starting point : (0, r)

$p_0=1-r$

$p_k<0 : x_{k+1}=x_k+1; y_{k+1}=y_k;$
$p_{k+1}=p_k+2x_{k+1}+1;$

$p_k>0 : x_{k+1}=x_k+1; y_{k+1}=y_k-1;$
$p_{k+1}=p_k+2x_{k+1}-2y_{k+1}+1;$

Definitions

Boundary fill colors inward until a specific boundary color is reached.
Flood fill replaces a target color within a bounded area, regardless of the boundary color.
Scan-line fill is more efficient, using horizontal lines to fill polygon interiors.

Flood Fill vs Boundary Fill

Flood Fill	Boundary Fill
Fills all connected pixels having the same old color.	Fills pixels until a boundary color is reached.
Stops when a different color is encountered.	Region is defined by a closed border.
Checking condition	
if (pixel_color == old_color)	if (pixel_color != boundary_color && pixel_color != fill_color)

Disadvantage
A 4-connected Flood Fill may leak when the boundary is not closed in the 4-connected sense, even if it appears visually closed.

Filled area vs. line drawing

For the Statement	Against the Statement
Fill algorithms handle large pixel regions (efficiency matters)	very frequently used per frame (constant memory)
Boundary Fill can be slow and may leak if the boundary is not properly closed (recursive stack)	Bresenham are highly optimized and essential
Scan-line Fill is fast, iterative, and reliable (minimal memory)	poor choice impacts overall rendering

Application context:
In real-time systems (games), scan-line-based rasterization is preferred for speed and stability.
In paint programs, Boundary Fill is acceptable for smaller regions.

Raster vs. Random Scan

Architectural Trade-offs

Raster Scan	Random Scan
Pixel-based, uses frame buffer (LCD/OLED displays)	Vector-based, draws only required lines
Scans entire screen sequentially	Stores line endpoints (display list)
Supports shading, textures, complex images	Produces smooth, high-quality lines
May have aliasing (jagged edges)	Limited to wireframe images
Requires continuous refresh	Limited by number of vectors per refresh

Raster vs. Random Scan (cont)

Real-time 3D / Gaming, Medical Imaging	It cannot efficiently render complex, filled, or continuous-tone images (only simple line drawings)

Rendering Complexity: Raster: Highly suitable; Random: Not suitable (only wirefr-ames)

Interactivity: Raster: Good (because of GPU acceleration); Random: Poor for complex models

Cost-effectiveness: Raster: Low cost; Random: High cost (requires specialized hardware)

Jagged lines

Causes

Discrete Sampling - Continuous lines mapped to pixel grid → stair-step effect

Nyquist Violation - Sampling rate too low → high-frequency details lost

Binary Pixel Coverage - Pixel is either fully ON/OFF, no partial coverage → sharp edges

Solution - Supersampling:
Render at higher resolution, then scaled down after sampling.
The system takes multiple samples within each pixel, rather than just one in the center, to calculate the accurate color of that pixel (Cause 3)

fig. 1

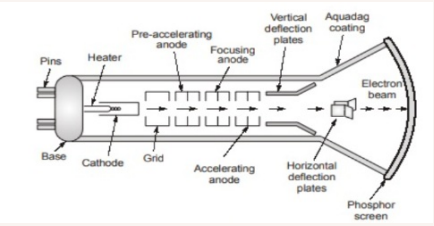

fig. 2