Computer Graphics

Aviskar Poudel

July 22, 2025

1 IMAGE CAPTURE FROM PHONE IN COMPUTER GRAPHICS

When a phone camera captures an image, it can be modeled in computer graphics using the **pinhole camera model** and **rendering pipeline** concepts. The key idea is that the 3D world is projected onto a 2D plane (the image sensor or screen).

Steps (Graphics Analogy)

- **3D World / Scene:** Real-world environment with objects and light.
- **Camera View:** The virtual viewpoint or lens capturing the scene.
- **Projection Transformation:** Converts 3D to 2D using a perspective projection matrix.
- **Viewport Mapping:** Maps normalized coordinates to actual pixel coordinates.
- **Rasterization & Sampling:** Approximates lines and shapes into discrete pixels.
- **Shading and Lighting:** Determines final color at each pixel.
- **Image Output:** Stored as a bitmap or JPEG image.

Block Diagram Representation:

3D Scene → Camera View → Projection Transformation → Viewport Mapping → Rasterization → Shading → Image Output

2 DDA (DIGITAL DIFFERENTIAL ANALYZER)

Advantages

- Simple to understand and implement.

- Can handle all line slopes.
- Works well for short lines or where floating point operations are cheap.
- Can be generalized to other geometric primitives.

Disadvantages

- Uses floating-point arithmetic.
- Slower than integer-based algorithms.
- Accumulates rounding errors.
- Not suitable for real-time applications or embedded systems.

Is DDA a Vector Display Algorithm?

No. DDA is designed for **raster displays**, where lines are approximated by lighting up pixels. Vector displays use direct beam control and do not rely on pixel grids.

3 BRESENHAM'S LINE ALGORITHM (BLA)

Advantages

- Uses only integer arithmetic.
- Very efficient and suitable for low-resource devices.
- Produces highly accurate, consistent lines.
- Adaptable to all slopes.

Disadvantages

- More complex implementation than DDA.
- Not easily generalizable to curves or anti-aliased lines.
- Requires case handling for different slope octants.

4 DDA VS BLA COMPARISON

Feature-wise Comparison

Feature	**DDA**	**Bresenham's Line Algorithm**
Arithmetic Used	Floating-point operations	Integer-only arithmetic
Speed	Slower	Faster
Accuracy	Rounding errors possible	Very accurate
Ease of Implementation	Simple	More complex
Slope Handling	Handles all slopes easily	Needs separate cases
Hardware Suitability	Less suitable for low-power devices	Ideal for real-time and embedded systems
Anti-Aliasing	Not supported	Not supported
Generalization	Easier to generalize	Limited to lines

Summary Table

Category	**DDA**	**BLA**
Advantages	• Simple • Flexible • General purpose	• Fast • Accurate • Efficient
Disadvantages	• Slow • Rounding error • Floating-point dependent	• More logic • No anti-aliasing • Hard to extend

5 VECTOR VS RASTER GRAPHICS

Vector Graphics

Vector graphics use mathematical equations to define paths (lines, curves, shapes). These paths are resolution-independent and scale without losing quality. Common formats include `.svg`, `.pdf`, and `.ai`.

Advantages

- Scalable without quality loss.
- Smaller file sizes for simple images.
- Easy to edit shapes and layers.
- Resolution-independent (sharp on any screen).

Disadvantages

- Poor at handling photo-like details.
- Requires rasterization for display.
- Limited support for rich color and textures.
- Complex to create detailed illustrations.

Raster Graphics

Raster graphics are composed of a grid of pixels, where each pixel has a color value. These are resolution-dependent and commonly used for photographs. Common formats include `.jpg`, `.png`, and `.bmp`.

Advantages

- Excellent for detailed and complex images.
- Rich in color, gradients, and textures.
- Direct display on screens (no rasterization needed).
- Pixel-level editing possible.

Disadvantages

- Quality loss when scaled (pixelation).
- Larger file sizes at high resolutions.
- Harder to edit individual elements.
- Resolution-dependent (may look blurry on high-DPI screens).

Comparison Table

Feature	Vector Graphics	Raster Graphics
Composition	Mathematical paths	Pixel grid (bitmap)
Scalability	Infinite, no quality loss	Pixelates when scaled
File Size	Small (for simple graphics)	Large (for high-res images)
Editing	Easy shape-level editing	Pixel-level editing only
Detail Handling	Poor at realistic detail	Excellent for complex images
Display Rendering	Needs rasterization	Directly displayable
Best For	Logos, icons, diagrams	Photos, textures, realistic art
Formats	SVG, AI, EPS, PDF	JPG, PNG, BMP, GIF

6 ALIASING IN COMPUTER GRAPHICS

Aliasing refers to visual artifacts that occur when high-resolution or continuous signals (like edges or curves) are represented on a low-resolution pixel grid, resulting in jagged or "stair-step" lines, often called "jaggies."

Cause

Aliasing happens due to **undersampling**, where the pixel resolution is insufficient to capture the signal's details, violating the Nyquist Theorem (sampling at twice the highest frequency).

Examples

- **Diagonal Lines:** Appear as jagged "stair-steps."
- **Curved Shapes:** Look blocky or uneven.
- **Moving Objects:** Cause flickering or "crawling" artifacts.
- **Texture Mapping:** Produce Moiré patterns or distortions.

Solution: Anti-Aliasing

Anti-aliasing techniques smooth jagged edges:

- **Supersampling (SSAA):** Renders at higher resolution and downsamples.
- **Multisample Anti-Aliasing (MSAA):** Samples multiple points per pixel edge.
- **Post-processing Filters:** E.g., Fast Approximate Anti-Aliasing (FXAA).
- **Alpha Blending:** Blends edge pixels with the background.

Summary Table

Term	Description
Aliasing	Jagged or distorted lines/curves due to low resolution
Cause	Undersampling (low resolution for high-frequency signals)
Fix	Anti-aliasing (SSAA, MSAA, FXAA, alpha blending)

7 ASPECT RATIO

Aspect ratio in computer graphics is defined as the ratio of width to height of a display or image.

Definition

$$\text{Aspect Ratio} = \frac{\text{Width}}{\text{Height}}$$

Examples

Format	Width	Height	Aspect Ratio
Full HD	1920	1080	16:9
Square	1000	1000	1:1
Vertical Reel	1080	1920	9:16

8 HORIZONTAL SCAN RATE

Horizontal Scan Rate (or Horizontal Frequency) is the number of times per second a display draws a single horizontal line across the screen, measured in kilohertz (kHz).

Definition

$$\text{Horizontal Scan Rate (kHz)} = \frac{\text{Number of Horizontal Lines per Second}}{1000}$$

Calculation

$$\text{Horizontal Scan Rate} = \text{Vertical Refresh Rate} \times \text{Number of Scan Lines}$$

Example: For a 1080p display at 60 Hz:

$$60 \times 1080 = 64,800\,\text{lines/sec} = 64.8\,\text{kHz}$$

Summary Table

Term	Description
Horizontal Scan Rate	Lines scanned horizontally per second (kHz)
Affects	Speed of frame rendering from top to bottom
Depends on	Resolution and refresh rate
Typical Range	15 kHz – 100+ kHz

9 FRAME BUFFER, REFRESH BUFFER, BITMAP, AND PIXMAP

Frame Buffer / Refresh Buffer

A region of memory storing pixel data for display, typically RGB values. The term "refresh buffer" is often synonymous, emphasizing its role in refreshing CRT displays.

Bitmap

A 1-bit-per-pixel image (black or white), stored as a grid of bits. Broadly, it may refer to any pixel-based image.

Pixmap

A multi-bit-per-pixel image (e.g., 24-bit RGB), storing color information for each pixel, used in modern graphics.

Summary Table

Term	Description	Memory Content	Used For
Frame Buffer	Holds pixel data for display	RGB values	Real-time screen drawing
Refresh Buffer	Same as frame buffer	RGB values	Refreshing CRT displays
Bitmap	1-bit-per-pixel grid	0 (black), 1 (white)	Icons, fonts
Pixmap	Multi-bit-per-pixel grid	Color info (8–32 bits)	Full-color images, GUIs

10 DEPTH BUFFER (Z-BUFFER)

A depth buffer (or Z-buffer) stores the depth (Z-value) of each pixel to determine which surfaces are visible in a 3D scene.

How It Works

- Each pixel has a Z-value (distance from camera).
- When rendering, the incoming Z-value is compared to the stored Z-value.
- If closer, the pixel is drawn, and the depth buffer is updated.
- If farther, the pixel is discarded.

Advantages

- Automatically handles overlapping objects.
- Efficient for real-time rendering (e.g., OpenGL, DirectX).
- No manual object sorting required.

Disadvantages

- Requires extra memory.
- Limited precision may cause Z-fighting.
- Challenges with transparent objects.

Applications

- **3D Game Rendering:** Ensures correct depth ordering (e.g., cars in front obscure those behind).
- **VR/AR:** Maintains spatial accuracy.
- **CAD:** Displays visible parts of complex models.
- **3D Animation:** Handles layering in tools like Blender.
- **Shadow Mapping:** Determines shadowed areas.
- **Scientific Visualization:** Displays layered 3D data.
- **Depth of Field (Blurring):** Uses depth to apply blur to out-of-focus areas.

Depth-based Blurring

Depth of Field (DoF) mimics camera optics by blurring objects outside a focal depth:

- Depth buffer provides Z-values.
- Pixels farther from focal depth are blurred more.
- Used in games, simulations, and cinematic effects.

Summary Table

Property	Description
Stores	Depth (Z-values) per pixel
Used For	Visibility testing in 3D rendering
Memory Type	Separate from frame buffer
Typical Use	Real-time graphics, games
Common Problem	Z-fighting due to limited precision

11 RASTER SCAN VS VECTOR SCAN DISPLAYS

Comparison

Feature	Raster Scan	Vector Scan
Display Method	Scans line by line, pixel by pixel	Draws lines point-to-point
Image Storage	Frame buffer (pixel grid)	Display list (geometric primitives)
Image Type	Filled images, photos, text	Line drawings, CAD
Refresh Rate	Constant (e.g., 60 Hz)	Varies with complexity
Hardware Complexity	Simpler beam control	Complex beam control
Brightness	Uniform across screen	Varies with line length
Used In	TVs, monitors, games	Oscilloscopes, radar, simulators
Aliasing	Prone to jaggies	No aliasing (smooth lines)

Advantages and Disadvantages

Raster Scan:

- *Advantages:* Supports rich images, compatible with modern displays, easier to implement.
- *Disadvantages:* Aliasing, high memory use, not ideal for precise drawings.

Vector Scan:

- *Advantages:* Precise lines, no aliasing, low memory use.
- *Disadvantages:* Limited to line drawings, complex hardware, flicker with many vectors.

12 COORDINATE SYSTEMS IN GRAPHICS

World Coordinates

Global coordinate system for the entire scene, using arbitrary units (e.g., meters).

Viewing Coordinates

Scene transformed relative to the camera's perspective.

Device Coordinates

Final pixel positions on the screen (integers).

Transformation Pipeline

$$\boxed{\text{World Coordinates} \rightarrow \text{Viewing Coordinates} \rightarrow \text{Normalized Device Coordinates} \rightarrow \text{Device Coordinates}}$$

Example: Map Drawing

Consider a city map in world coordinates (meters):

- City Center: (0, 0)
- Park: (300, 150)
- Bridge: (-200, 100)
- Hospital: (100, -250)

Viewing Coordinates: Center on Hospital (100, -250) → (0, 0):

- City Center: (-100, 250)
- Park: (200, 400)
- Bridge: (-300, 350)

Device Coordinates: Map to 800×600 pixel screen, 1 meter = 1 pixel, center at (400, 300):

- Hospital: (400, 300)
- Park: (600, 700)
- City Center: (300, 550)
- Bridge: (100, 650)

Summary Table

Coordinates	**Purpose**	**Example**
World	Defines scene layout	Tree at (130, 230)
Viewing	Camera's perspective	Tree at (30, 30)
Device	Pixel location on screen	Tree at (300, 250)

13 Z-BUFFER IN 3D GRAPHICS PIPELINE

In the 3D graphics pipeline, the Z-buffer is used during **rasterization** and **fragment processing** to handle visibility.

Pipeline Overview

- **Model to World:** Objects in global scene.
- **World to View:** Scene from camera's perspective.
- **View to Projection:** 3D to 2D (perspective/orthographic), Z retained.
- **Rasterization:** Geometry to fragments, Z per fragment.
- **Fragment Processing:** Depth testing (Z-buffer), shading.
- **Frame Buffer:** Final pixels stored.

Z-buffer Role

- Stores Z-value (depth) for each pixel.
- Compares incoming fragment Z with stored Z.
- Closer fragments update pixel and Z-buffer; farther ones are discarded.

Summary Table

Stage	**Z-buffer Involvement**
Projection	Z transformed
Rasterization	Z per fragment
Fragment Processing	Depth testing (Z-buffer used)