1. Random Scan Display

(Also called Vector Scan or Calligraphic Display)

Definition

A **Random Scan Display** is a type of display system in which the **electron beam directly draws pictures line by line** on the screen by moving only along the required paths, instead of scanning the whole screen.

The beam moves **randomly** from one point to another to draw lines and curves.

Working Principle

- The display uses a **Cathode Ray Tube (CRT)**.
- The electron beam is controlled by **deflection circuits**.
- The beam:
    - Starts at one coordinate
    - Moves directly to another coordinate
    - Draws a straight line between them
- Only the required image parts are drawn.

📌 The beam does **not** scan the entire screen.

Key Components

1. **Display Controller** – Controls beam movement
2. **Refresh Memory (Display List)** – Stores line drawing commands
3. **CRT Monitor** – Displays the image
4. **Digital-to-Analog Converter (DAC)** – Converts digital coordinates to analog signals

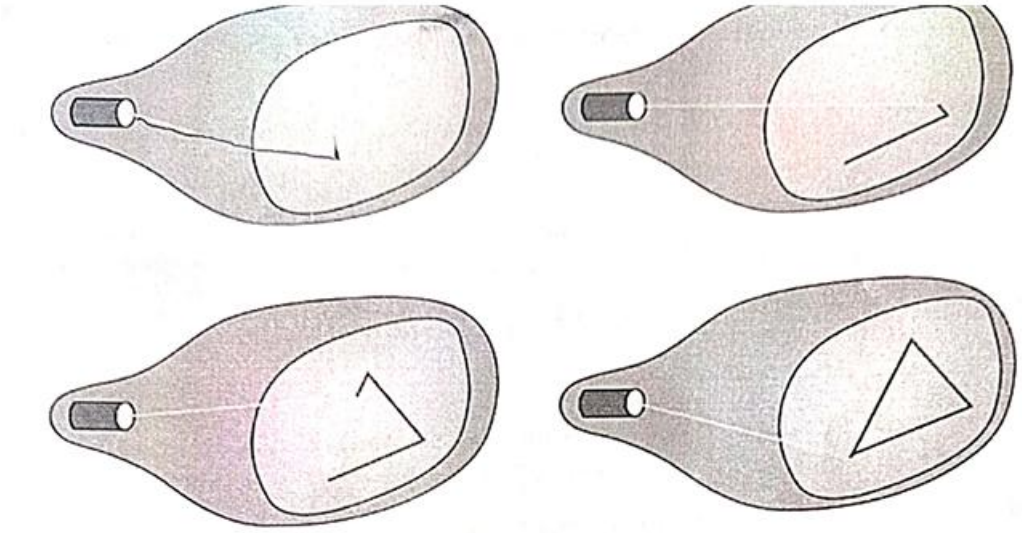
---

Image Formation

- Image is stored as a **set of line segments**
- Each line is defined by:
    - Start point (x_1, y_1)

- End point (x_2, y_2)

- The system redraws the image many times per second (refresh).



Features

✓ High resolution for line drawings
✓ Smooth and sharp lines
✓ Ideal for wireframe models
❌ Not suitable for filled images
❌ Flicker increases with complex images

Advantages

- Very **high image quality**
- **No aliasing** (jagged edges)
- Requires **less memory** than raster scan
- Fast for simple images

Disadvantages

- Not good for **complex images**
- Cannot display **realistic pictures**
- Flickering occurs if too many lines

- Expensive hardware

Applications

- CAD (Computer-Aided Design)
- CAM systems
- Engineering drawings
- Military radar displays

---

2. Raster Scan Display

(Most Common Display System)

Definition

A **Raster Scan Display** is a display system in which the **electron beam scans the entire screen line by line**, from **top to bottom**, to display the image.

This method is used in **TVs, monitors, laptops, and mobile screens**.

Working Principle

- The screen is divided into a grid of **pixels**
- The beam:
    - Starts from the **top-left corner**
    - Moves horizontally to the right
    - Returns to the next line (horizontal retrace)
    - Continues till bottom (vertical retrace)
- Entire screen is refreshed repeatedly (60–75 times/sec)

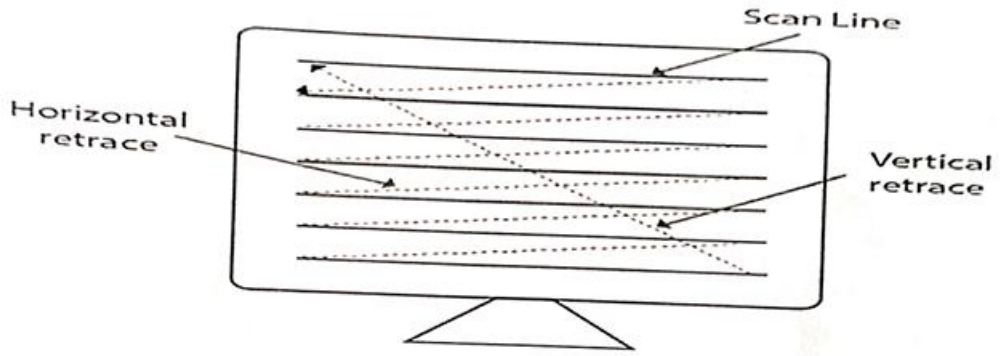
Key Components

1. **Frame Buffer (Refresh Buffer)** – Stores pixel intensity values
2. **Display Controller** – Reads frame buffer
3. **DAC** – Converts digital data to analog signals

4. **CRT / LCD / LED Monitor**

Image Formation

- Image is stored as **pixel values**
- Each pixel contains color or intensity information
- The frame buffer holds the complete image



Features

✓ Supports filled areas
✓ Displays images, text, graphics, and videos
✓ Suitable for realistic pictures
❌ Lower line quality compared to random scan
❌ Requires more memory

Advantages

- Can display **complex images**
- Supports **shading, textures, colors**
- Less flicker
- Hardware is cheaper and simpler

Disadvantages

- Requires large memory
- Aliasing effect (jagged edges)
- Lower resolution for lines

Applications

- Televisions
- Computer monitors
- Smartphones
- Multimedia systems
- Gaming and animation

---

3. Comparison Table (Very Important for Exams)

Feature	**Random Scan**	**Raster Scan**
Beam Movement	Random (line by line)	Line by line (entire screen)
Image Storage	Line segments	Pixel values
Image Quality	Very high for lines	Good for filled images
Memory Required	Low	High
Flicker	More for complex images	Less
Realistic Images	Not supported	Supported
Cost	Expensive	Cheaper
Examples	CAD, Radar	TV, Monitor

Input Devices

Input devices are hardware components used to **enter data, commands, or graphics** into a computer system.

In computer graphics, input devices are mainly used to:

- Draw shapes
- Select objects
- Enter coordinates
- Control cursor movement

1.1 Keyboard

Definition

A **keyboard** is an input device used to enter **text, numbers, and commands** into a computer.

Working

- Each key press sends a **scan code** to the CPU.
- The computer interprets the code as a character or command.

Uses

- Text entry
- Programming
- Command execution

1.2 Mouse

Definition

A **mouse** is a pointing device used to **select, move, and manipulate objects** on the screen.

Working

- Movement of mouse → cursor movement
- Buttons are used for **click, drag, drop**

Uses

- Drawing
- Selecting objects
- GUI interaction

1.3 Light Pen

Definition

A **light pen** is a pointing device that allows users to **draw or select objects directly on the screen**.

Working

- Detects light from the CRT screen
- Sends position to the system

Uses

- CAD
- Drawing applications

1.4 Trackball

Definition

A **trackball** is similar to a mouse but uses a **rotating ball**.

Uses

- Graphic design
- Limited workspace environments

1.5 Joystick

Definition

A **joystick** is an input device used to **control movement** in different directions.

Uses

- Games
- Simulations

- Virtual reality

1.6 Touch Screen

Definition

A **touch screen** allows users to **interact directly by touching the display**.

Uses

- Smartphones
- ATMs
- Kiosks

---

2. Display Devices

Display devices are output devices used to **show text, images, and graphics**.

2.1 CRT (Cathode Ray Tube)

Definition

A **CRT** is a display device that uses an **electron beam** to create images on a phosphor-coated screen.

Working Principle

1. An **electron gun** emits electrons.
2. The beam is focused using focusing electrodes.
3. Deflection coils move the beam horizontally and vertically.
4. When the beam hits the **phosphor screen**, light is produced.
5. Continuous scanning creates images.

Features

- Uses raster or random scan
- Bulky and heavy

- Consumes high power

Advantages

✓ Good color reproduction
✓ High contrast
✓ Low cost (older systems)

Disadvantages

❌ Large size and heavy
❌ High power consumption
❌ Emits radiation

Applications

- Old televisions
- Early computer monitors
- Oscilloscopes

2.2 LCD (Liquid Crystal Display)

Definition

An **LCD** is a flat-panel display that uses **liquid crystals** to control light.

Working Principle

1. A **backlight** (fluorescent or LED) produces light.
2. Light passes through liquid crystal molecules.
3. Electric voltage aligns crystals.
4. Alignment controls light passage.
5. Pixels form images.

Features

- Thin and lightweight
- Uses less power
- Flat screen technology

Advantages

✓ Low power consumption
✓ Lightweight
✓ Less eye strain

Disadvantages

❌ Limited viewing angle
❌ Lower contrast than CRT

Applications

- Laptops
- Mobile phones
- Monitors
- TVs

2.3 LED (Light Emitting Diode)

Definition

An **LED display** is an improved form of LCD that uses **LED backlighting** instead of fluorescent lamps.

📌 LED is **not a completely different technology**, it is a type of LCD.

Working Principle

- Uses **LEDs as backlight**
- Light passes through liquid crystals
- Produces brighter images

Types of LED

1. **Edge-lit LED**
2. **Full-array LED**

Advantages

✓ Very thin
✓ Better brightness and contrast
✓ Energy efficient
✓ Longer lifespan

Disadvantages

❌ More expensive than LCD

Applications

- Smart TVs
- High-resolution monitors
- Mobile devices

---

3. Comparison Table (Important for Exams)

Feature	CRT	LCD	LED
Technology	Electron beam	Liquid crystals	LCD with LED backlight
Thickness	Very thick	Thin	Very thin
Power Consumption	High	Low	Very low
Image Quality	Good	Better	Best
Radiation	Yes	No	No
Cost	Cheap (old)	Moderate	High

Graphics Software

Definition

Graphics software refers to programs and system software that are used to **create, edit, store, display, and manipulate graphical images** such as drawings, diagrams, animations, and pictures.

Graphics software acts as a **bridge between the user and graphics hardware**.

Types of Graphics Software

1.1 System Graphics Software

This software controls **graphics hardware** and provides basic graphics operations.

(a) Graphics Drivers

- Interface between **graphics hardware and OS**
- Converts high-level graphics commands into hardware commands

(b) Graphics Libraries / APIs

- Provide predefined functions to draw graphics
- Hide hardware complexity

Examples:

- OpenGL
- DirectX
- Vulkan

1.2 Application Graphics Software

Used by end users to **create graphics and multimedia content**.

(a) Raster Graphics Software

Works on **pixels**.

Examples:

- Adobe Photoshop
- GIMP

- MS Paint

Uses:

- Photo editing
- Image manipulation

(b) Vector Graphics Software

Works on **mathematical shapes (lines, curves)**.

Examples:

- Adobe Illustrator
- CorelDRAW
- Inkscape

Uses:

- Logos
- Diagrams
- CAD drawings

(c) 3D Graphics Software

Used to create **3D models, animation, and visual effects**.

Examples:

- Blender
- Maya
- 3ds Max

Functions of Graphics Software

- Drawing lines and shapes
- Color filling and shading
- Image transformation (scaling, rotation)
- Animation

2. Graphics Standards

Definition

Graphics standards are **rules, specifications, and formats** that ensure **compatibility, portability, and consistency** of graphics across different hardware and software platforms.

Need for Graphics Standards

- Hardware independence
- Portability of graphics programs
- Easy data exchange
- Consistent output on different systems

Types of Graphics Standards

2.1 Graphics Programming Standards

(a) GKS (Graphical Kernel System)

Definition:
GKS is the **first ISO-approved graphics standard** used for **2D graphics**.

Features:

- Device independent
- Supports basic 2D graphics
- Uses primitives like lines, circles, text

Limitations:

- No 3D support
- Limited interactivity

(b) GKS-3D

- Extension of GKS

- Supports **3D graphics**
- Includes transformations and viewing

(c) PHIGS (Programmer's Hierarchical Interactive Graphics System)

Features:

- Supports **hierarchical structures**
- Interactive graphics
- Better for CAD applications

(d) OpenGL

Definition:
OpenGL is a **cross-platform graphics API** used for **2D and 3D graphics rendering**.

Features:

- Hardware independent
- High performance
- Widely used in games and simulations

(e) DirectX

- Developed by Microsoft
- Used mainly for **Windows-based graphics**
- High-speed rendering

2.2 Graphics File Format Standards

These standards define **how graphics are stored and exchanged**.

(a) Raster Image Formats

Format	Description
BMP	Uncompressed Windows bitmap

Format	Description
JPEG	Compressed photographic images
PNG	Lossless compression
GIF	Animation support

(b) Vector Image Formats

Format	Description
SVG	Scalable Vector Graphics
EPS	Encapsulated PostScript
PDF	Portable document graphics

2.3 Multimedia & Animation Standards

(a) MPEG

- Video and audio compression standard
- Used in multimedia and streaming

(b) JPEG

- Image compression standard

(c) VRML / X3D

- Standards for **3D web graphics**

---

3. Comparison: Graphics Software vs Graphics Standards

Feature	Graphics Software	Graphics Standards
Purpose	Create & edit graphics	Ensure compatibility
Nature	Programs/tools	Rules & specifications
Examples	Photoshop, Blender	GKS, OpenGL, JPEG
Dependency	User-based	System & platform-based

Light Source in Computer Graphics

Definition

A **light source** in computer graphics is an object that **emits light** to illuminate surfaces so that they become **visible and realistic**.

Lighting helps to show:

- Shape
- Depth
- Texture
- Realism

Types of Light Sources

1.1 Point Light Source

- Emits light from a **single point**
- Light spreads in all directions

Example: Bulb, candle

1.2 Directional Light Source

- Light rays are **parallel**
- Light source assumed at infinity

Example: Sunlight

1.3 Spotlight

- Emits light in a **cone shape**
- Has position and direction

Example: Torch, stage lights

1.4 Ambient Light

- Uniform light present everywhere

- No direction or position

Purpose: Prevents complete darkness

---

2. Illumination Model

Definition

An **illumination model** is a **mathematical model** that describes **how light interacts with surfaces** to produce the final color seen by the viewer.

It calculates **intensity of light** at each surface point.

---

3. Local Illumination Model

📌 Considers only:

- Light source
- Surface properties
- Viewer position

📌 Does **not** consider:

- Reflection from other objects
- Shadows between objects

3.1 Components of Illumination

(a) Ambient Reflection

Concept:

- Background light scattered equally
- Independent of light position

Formula:

$$I_a = K_a \cdot I_{ambient}$$

Where:

- K_a= Ambient reflection coefficient

- $I_{ambient}$= Ambient light intensity

(b) Diffuse Reflection (Lambert's Law)

Concept:

- Light reflected equally in all directions
- Depends on **angle** between light and surface normal

Formula:

$$I_d = K_d \cdot I_l \cdot \cos\ \theta$$

Where:

- K_d= Diffuse coefficient
- I_l= Light intensity
- θ= Angle between light direction and surface normal

📌 Maximum when light hits surface directly.

(c) Specular Reflection

Concept:

- Produces **shiny highlights**
- Depends on viewer position

Formula:

$$I_s = K_s \cdot I_l \cdot (\cos\ \alpha)^n$$

Where:

- K_s= Specular coefficient
- α= Angle between reflected ray and viewer direction
- n= Shininess exponent

---

4. Phong Illumination Model (Most Important ⭐)

Definition

The **Phong illumination model** is a **local illumination model** that combines:

- Ambient
- Diffuse
- Specular components

Complete Phong Equation

$$I = I_a + I_d + I_s$$
$$I = K_a I_a + K_d I_l(\vec{N} \cdot \vec{L}) + K_s I_l(\vec{R} \cdot \vec{V})^n$$

Where:

- $\vec{N}$= Surface normal
- $\vec{L}$= Light direction
- $\vec{R}$= Reflection vector
- $\vec{V}$= Viewer direction

Features of Phong Model

✓ Simple and efficient
✓ Produces realistic highlights
✓ Widely used in real-time graphics

Limitations

❌ Not physically accurate
❌ No global light interaction
❌ No shadows or color bleeding

---

5. Attenuation (Distance Effect)

Light intensity decreases with distance.

Formula:

$$I = \frac{I_0}{a + bd + cd^2}$$

Where:

- d= Distance from light source
- a, b, c= Attenuation constants

---

6. Global Illumination Model

📌 Considers:

- Reflection from other surfaces
- Shadows
- Transparency

Examples:

- Ray tracing
- Radiosity

📌 More realistic but computationally expensive.

---

7. Comparison: Local vs Global Illumination

Feature	Local	Global
Light Interaction	Direct only	Multiple reflections
Shadows	No	Yes
Realism	Medium	High
Speed	Fast	Slow

Introduction to Color Models

What is Color?

Color is a **visual perception** created when light of different wavelengths reaches the human eye.
In computer graphics, colors must be represented **numerically** so that computers can store, process, and display them.

Definition of Color Model

A **color model** is a **mathematical model** that represents colors using a set of numbers or components.
Each color is expressed as a **combination of values** in a specific color space.

Purpose of Color Models

Color models are used to:

- Represent colors digitally
- Create realistic images
- Convert colors between devices
- Support image processing and multimedia
- Match human color perception

---

2. RGB Color Model (Red, Green, Blue)

Definition

The **RGB color model** is an **additive color model** in which colors are created by **adding different intensities of red, green, and blue light**.

It is the **most widely used color model** in computer graphics.

Components

- **Red (R)**
- **Green (G)**
- **Blue (B)**

Each component typically ranges from:

- **0 to 255** (8 bits per channel)

Working Principle (Additive Mixing)

- Colors are formed by **adding light**
- Black → absence of light

$$(0, 0, 0)$$

- White → full intensity of all colors

$$(255, 255, 255)$$

Examples

- Red → (255, 0, 0)
- Green → (0, 255, 0)
- Blue → (0, 0, 255)
- Yellow → (255, 255, 0)

Geometric Representation

- Represented as a **3D color cube**
- Each axis represents R, G, and B
- Origin = Black
- Opposite corner = White

Advantages

- Simple and hardware-friendly
- Directly supported by displays
- Fast processing

Disadvantages

- Not intuitive for humans
- Difficult to choose colors manually
- Device dependent

Applications

- Computer monitors
- Television screens
- Digital cameras
- Mobile displays

---

3. CMY Color Model (Cyan, Magenta, Yellow)

Definition

The **CMY color model** is a **subtractive color model** used mainly in **printing systems**.

It works by **subtracting (absorbing) light** rather than adding it.

Components

- **Cyan (C)**
- **Magenta (M)**
- **Yellow (Y)**

Working Principle (Subtractive Mixing)

- White paper reflects all light
- Inks absorb certain wavelengths
- Remaining light determines the color seen

Relationship with RGB

CMY is the **complement of RGB**:

$$C = 1 - R$$

$$M = 1 - G$$
$$Y = 1 - B$$

(Values normalized between 0 and 1)

Why CMY is Used in Printing

- Printers work with **ink**, not light
- Subtractive process matches physical printing
- More accurate color reproduction on paper

Advantages

- Suitable for hard-copy output
- Accurate color mixing for printing

Disadvantages

- Difficult to get pure black
- Often extended to **CMYK** (adds Black)

Applications

- Printers
- Plotters
- Publishing industry

---

4. HSV Color Model (Hue, Saturation, Value)

Definition

The **HSV color model** represents colors in a way that closely matches **human perception**.

It is designed to be **user-friendly**.

Components

Hue (H)

- Represents **color type**
- Measured in degrees (0°–360°)
- Example:
    - 0° → Red
    - 120° → Green
    - 240° → Blue

Saturation (S)

- Represents **purity of color**
- 0% → Gray
- 100% → Pure color

Value (V)

- Represents **brightness**
- 0% → Black
- 100% → Bright color

Representation

- Represented as a **hexcone** or **cylinder**
- Hue around the circle
- Value on vertical axis

Advantages

- Easy color selection
- Intuitive for humans
- Separates color from brightness

Disadvantages

- Not suitable for hardware processing
- Requires conversion to RGB for display

Applications

- Image editing tools
- Color pickers
- Photo enhancement software

---

5. HLS (HSL) Color Model (Hue, Lightness, Saturation)

Definition

The **HLS (or HSL)** color model is similar to HSV but uses **Lightness** instead of Value.

Components

Hue (H)

- Same as HSV

Lightness (L)

- 0 → Black
- 0.5 → Normal color
- 1 → White

Saturation (S)

- Color purity

Difference Between HSV and HLS

- HSV emphasizes brightness
- HLS emphasizes **balance between black and white**
- HLS gives better control over shades

Advantages

- More natural color control
- Useful in graphic design

Applications

- Color design systems
- Graphic user interfaces

---

6. YIQ Color Model

Definition

The **YIQ color model** is used in **NTSC television broadcasting systems**.

Components

Y (Luminance)

- Brightness information
- Black-and-white image

I (In-phase)

- Orange–cyan color information

Q (Quadrature)

- Purple–green color information

Working Principle

- Separates brightness from color
- Allows compatibility with black-and-white TVs
- Human eye is more sensitive to luminance

Advantages

- Efficient transmission
- Backward compatibility
- Reduced bandwidth usage

Disadvantages

- Limited use today
- NTSC-specific

Applications

- Television broadcasting
- Video compression systems

---

7. Comparison Table (Very Important ⭐)

Model	Type	Nature	Used In
RGB	Additive	Device-based	Displays
CMY	Subtractive	Ink-based	Printing
HSV	Perceptual	User-friendly	Image editing
HLS	Perceptual	Design-friendly	UI & design
YIQ	Video	Broadcast-based	TV systems

Properties of Light

Light is a form of **electromagnetic energy** that produces the sensation of vision when it enters the human eye.
In computer graphics, understanding light is essential for **color representation and display systems**.

Important Properties of Light

1.1 Intensity

- Intensity refers to the **amount of energy** carried by light.
- It determines how **bright or dim** a color appears.
- Higher intensity → brighter light.

1.2 Wavelength

- Wavelength is the **distance between two successive peaks** of a light wave.
- Measured in **nanometers (nm)**.
- Visible light range: **400 nm (violet) to 700 nm (red)**.

1.3 Frequency

- Frequency is the **number of light waves passing a point per second**.
- Related to wavelength by:

$$c = \lambda f$$

where
c= speed of light,
λ= wavelength,
f= frequency.

1.4 Color

- Color depends mainly on the **wavelength** of light.
- Different wavelengths produce different colors.

- Example:
    - 450 nm → Blue
    - 550 nm → Green
    - 650 nm → Red

1.5 Spectral Distribution

- It describes how light energy is distributed across different wavelengths.
- Important for color mixing and display devices.

---

2. Standard Primaries

Definition

Standard primaries are a fixed set of **three primary colors** that can be combined in different proportions to produce a wide range of colors.

They form the basis of **color specification in computer graphics**.

Why Three Primaries?

- The human eye has **three types of color receptors (cones)**.
- Any color sensation can be matched by mixing **three suitable primary colors**.

Additive and Subtractive Primaries

Additive Primaries (RGB)

Used when **light is added**.

Primary Color	
R	Red
G	Green
B	Blue

- Used in **monitors, TVs, projectors**
- Adding all three → White

Subtractive Primaries (CMY)

Used when **light is absorbed**.

Primary Color	
C	Cyan
M	Magenta
Y	Yellow

- Used in **printing**
- Combining all → Black (approximately)

CIE Standard Primaries

To standardize color representation, the **CIE (Commission Internationale de l'Éclairage)** defined **standard primaries**.

- These are **imaginary primaries**
- Chosen so that all visible colors can be represented
- Used in **color measurement and specification**

Tristimulus Values

Any color can be represented by three values:

$$X, Y, Z$$

These are called **tristimulus values**, corresponding to CIE standard primaries.

---

3. Chromaticity

Definition

Chromaticity represents the **quality of color independent of brightness**.

It depends only on:

- Hue

- Saturation

Not on:

- Intensity

Chromaticity Coordinates

From tristimulus values X, Y, Z, chromaticity coordinates are defined as:

$$x = \frac{X}{X + Y + Z}$$
$$y = \frac{Y}{X + Y + Z}$$
$$z = \frac{Z}{X + Y + Z}$$

Since:

$$x + y + z = 1$$

Only **x and y** are needed.

---

4. Chromaticity Diagram

Definition

A **chromaticity diagram** is a **2D graphical representation** that shows **all visible colors** independent of brightness.

It is based on **CIE color specification system**.

Features of Chromaticity Diagram

4.1 Shape

- Horseshoe-shaped boundary
- Boundary represents **pure spectral colors**
- Straight line at bottom represents **purple colors**

4.2 Spectral Locus

- Outer curved boundary
- Each point corresponds to a **single wavelength**

4.3 White Point

- Located near the center
- Represents **equal energy white**

4.4 Gamut

- The region inside the triangle formed by three primaries
- Represents colors that a device can display

Color Mixing on Chromaticity Diagram

- Mixing two colors → point on straight line joining them
- Mixing three colors → point inside the triangle

---

5. Importance of Chromaticity Diagram

- Helps in choosing primaries
- Used to compare color gamuts
- Device-independent color representation
- Used in color calibration

---

6. Advantages of Using Standard Primaries and Chromaticity Diagram

✓ Device independence
✓ Accurate color specification
✓ Universal color communication
✓ Useful in graphics, printing, and video

2D TRANSFORMATION

1.1 Introduction

In computer graphics, objects are created in a **model coordinate system**. To display these objects correctly on the screen, we often need to **move, resize, rotate, or flip** them.
This change in position, size, or orientation of an object is called **transformation**.

Definition

A **2D transformation** is the process of changing the **position, size, shape, or orientation** of a two-dimensional object using **mathematical operations**.

Why 2D Transformations are Needed

- To move objects on the screen
- To resize objects
- To rotate objects
- To create animation
- To change object orientation

Types of 2D Transformations

1. Translation
2. Scaling
3. Rotation
4. Reflection
5. Shearing

All transformations are represented using **matrix multiplication**.

1.2 Translation

Definition

Translation is the process of **moving an object from one position to another** without changing its shape or size.

Translation Formula

If a point **P(x, y)** is translated by **(tx, ty)**, the new point **P'(x', y')** is:

$$x' = x + tx$$
$$y' = y + ty$$

Translation Matrix (Homogeneous Coordinates)

$$\begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix}$$

Applications

- Moving objects in animation
- Scrolling
- Positioning objects on screen

1.3 Scaling

Definition

Scaling changes the **size of an object** by expanding or shrinking it.

Scaling Formula

$$x' = x \cdot S_x$$
$$y' = y \cdot S_y$$

Where:

- S_x→ Scaling factor in x-direction
- S_y→ Scaling factor in y-direction

Scaling Matrix

$$\begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Types of Scaling

- **Uniform Scaling** → $S_x = S_y$
- **Non-uniform Scaling** → $S_x \neq S_y$

Applications

- Zoom in / zoom out
- Resizing objects

1.4 Rotation

Definition

Rotation is the process of turning an object around a **fixed point (pivot)**, usually the origin.

Rotation Formula

For rotation by angle **θ** (anticlockwise):

$$x' = x\cos\ \theta - y\sin\ \theta$$
$$y' = x\sin\ \theta + y\cos\ \theta$$

Rotation Matrix

$$\begin{bmatrix} \cos\theta & -\sin\ \theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Applications

- Turning objects
- Games
- Animations

1.5 Reflection

Definition

Reflection creates a **mirror image** of an object about a line or axis.

Common Reflections

Reflection Axis	Result
X-axis	y becomes −y
Y-axis	x becomes −x
Origin	x and y both change sign
Line y = x	x and y interchange

Reflection Matrix (about X-axis)

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

1.6 Shearing

Definition

Shearing distorts an object by **slanting its shape**.

Shearing Types

X-direction shearing

$$x' = x + Sh_x \cdot y$$

Y-direction shearing

$$y' = y + Sh_y \cdot x$$

Applications

- Creating italic text
- Special effects

1.7 Homogeneous Coordinates

To combine multiple transformations into one, we use **homogeneous coordinates**.

A 2D point (x, y) is written as:

$$(x, y, 1)$$

1.8 Composite Transformation

Multiple transformations can be applied together by **multiplying matrices** in sequence.

📌 **Order matters** in transformations.

---

2. 2D VIEWING

2.1 Introduction

In real life, we do not see everything—only a **portion of the world**.
Similarly, in computer graphics, we select a **part of the scene** to display on the screen.
This process is called **2D viewing**.

Definition

2D viewing is the process of selecting a **window** in world coordinates and mapping it onto a **viewport** on the display device.

2.2 Window and Viewport

Window

- Defined in **world coordinate system**
- Represents the portion of the scene to be viewed

Viewport

- Defined in **device coordinate system**
- Area on the screen where the window is displayed

2.3 Window-to-Viewport Mapping

This mapping ensures:

- Correct positioning
- Proper scaling
- Aspect ratio preservation

Mapping Equations

$$x_v = x_{vmin} + \frac{(x_w - x_{wmin})(x_{vmax} - x_{vmin})}{(x_{wmax} - x_{wmin})}$$

$$y_v = y_{vmin} + \frac{(y_w - y_{wmin})(y_{vmax} - y_{vmin})}{(y_{wmax} - y_{wmin})}$$

2.4 Viewing Pipeline in 2D

1. Object creation
2. Modeling transformation
3. Window selection
4. Clipping
5. Viewport mapping
6. Display

2.5 Clipping

Definition

Clipping removes parts of objects that lie **outside the window**.

Types of Clipping

- Point clipping
- Line clipping
- Polygon clipping

📌 Common algorithm: **Cohen-Sutherland Line Clipping**

2.6 Aspect Ratio

Aspect ratio is the **ratio of width to height**.

If aspect ratio is not maintained:

- Image appears stretched or compressed

---

3. Difference Between 2D Transformation and 2D Viewing

Feature	2D Transformation	2D Viewing
Purpose	Modify object	Display selected region
Applied On	Objects	Entire scene
Operations	Translate, scale, rotate	Window-viewport mapping
Output	Modified object	Displayed scene

What is Clipping? (Computer Graphics)

Definition

Clipping is the process of **removing the portions of graphical objects that lie outside a specified viewing area** (called the *clipping window* or *viewing window*).
Only the **visible part inside the window** is displayed on the screen.

Why Clipping is Needed

In computer graphics, scenes are often larger than the display area. Clipping is used to:

- Display only the required portion of a scene
- Improve rendering efficiency
- Avoid drawing invisible objects
- Maintain correct viewing boundaries

Clipping Window

- A **clipping window** defines the visible region in **world coordinates**.
- Any object **inside** the window → displayed
- Any object **outside** the window → removed
- Objects **partially inside** → only the visible part is displayed

---

Types of Clipping

1. Point Clipping

- Checks whether a point lies **inside or outside** the window.
- A point is visible if:

$$x_{min} \leq x \leq x_{max}, y_{min} \leq y \leq y_{max}$$

---

2. Line Clipping

- Removes portions of a line that lie outside the window.
- Only the segment inside the window is drawn.

Common Algorithms:

- Cohen–Sutherland Line Clipping
- Liang–Barsky Line Clipping

---

3. Polygon Clipping

- Clips polygons by removing outside edges and generating new vertices.
- Ensures the polygon remains closed.

Common Algorithm:

- Sutherland–Hodgman Polygon Clipping

---

4. Curve Clipping

- Used for circles, ellipses, and splines.
- Curve is broken into visible segments.

---

5. Text Clipping

- Determines how text behaves when it crosses window boundaries.
- Can be:
    - All-or-none
    - Character clipping
    - String clipping

Clipping in Graphics Pipeline

Clipping is performed **after viewing transformation** and **before rasterization**.

Advantages of Clipping

✓ Improves performance
✓ Reduces memory usage
✓ Ensures correct display
✓ Essential for window–viewport mapping

Cohen–Sutherland Line Clipping Algorithm

1. Introduction

In computer graphics, lines may lie **completely inside**, **completely outside**, or **partially inside** a viewing window.
The **Cohen–Sutherland algorithm** is used to **clip a line segment** against a **rectangular clipping window**.

---

2. Definition

The **Cohen–Sutherland line clipping algorithm** is an **efficient and simple line clipping algorithm** that uses **region codes (outcodes)** to determine whether a line is visible, invisible, or partially visible with respect to a rectangular window.

---

3. Basic Idea

- The 2D space is divided into **9 regions**:
    - 1 inside the window
    - 8 outside regions
- Each endpoint of a line is assigned a **4-bit region code**
- Logical operations are used to decide:
    - **Trivially accept**
    - **Trivially reject**
    - **Partially accept (needs clipping)**

---

4. Clipping Window

Let the rectangular window be defined by:

- x_{min}, x_{max}
- y_{min}, y_{max}

---

5. Region Code (Outcode)

Each endpoint is assigned a **4-bit binary code**:

Bit Position	Meaning	Condition
Bit 1	Top	$y > y_{max}$
Bit 2	Bottom	$y < y_{min}$
Bit 3	Right	$x > x_{max}$
Bit 4	Left	$x < x_{min}$

📌 Inside the window → **0000**

---

6. Region Layout

1001 | 1000 | 1010

-----|------|-----

0001 | 0000 | 0010

-----|------|-----

0101 | 0100 | 0110

---

7. Algorithm Steps

Step 1:

Compute the **outcodes** for both endpoints of the line.

Step 2: Trivial Acceptance

If:

$$\text{Outcode}_1 = 0000 \text{AND} \text{Outcode}_2 = 0000$$

➞ The line lies **completely inside** the window.

✓ **Accept the line**

Step 3: Trivial Rejection

If:

$$\text{Outcode}_1 \ \& \ \text{Outcode}_2 \neq 0$$

➞ Line lies **completely outside** the window in the same region.

❌ **Reject the line**

Step 4: Partial Visibility

If neither accepted nor rejected:

- The line crosses the window boundary
- Calculate the intersection point
- Replace the outside endpoint with intersection
- Recalculate outcode
- Repeat the process

---

8. Intersection Calculation

If the line intersects:

Left boundary

$$x = x_{min}$$
$$y = y_1 + m(x_{min} - x_1)$$

Right boundary

$$x = x_{max}$$
$$y = y_1 + m(x_{max} - x_1)$$

Bottom boundary

$$y = y_{min}$$
$$x = x_1 + \frac{(y_{min} - y_1)}{m}$$

Top boundary

$$y = y_{max}$$
$$x = x_1 + \frac{(y_{max} - y_1)}{m}$$

Where:

$$m = \frac{y_2 - y_1}{x_2 - x_1}$$

---

9. Example (Conceptual)

- One endpoint inside (0000)
- Other endpoint outside (e.g., 1000 – above window)
- Line intersects top boundary
- Intersection point is calculated
- Outside point replaced
- Final clipped line is drawn

---

10. Advantages

✓ Simple and easy to implement
✓ Fast for trivial cases
✓ Uses bitwise operations

---

11. Disadvantages

❌ Not efficient for lines with many intersections
❌ Repeated calculations may be required
❌ Limited to rectangular windows

---

12. Applications

- Window systems
- CAD software
- Graphical user interfaces
- 2D graphics rendering

Liang–Barsky Line Clipping Algorithm

1. Introduction

In computer graphics, line clipping is used to display only the **visible part of a line segment** inside a clipping window.
The **Liang–Barsky algorithm** is a **parametric line clipping algorithm** that is **more efficient** than Cohen–Sutherland.

---

2. Definition

The **Liang–Barsky line clipping algorithm** is a line clipping technique that uses the **parametric equation of a line** and **inequality testing** to determine the visible portion of a line segment within a rectangular clipping window.

---

3. Basic Idea

- A line is represented in **parametric form**
- Clipping is done by checking **parameter values**
- Instead of repeated intersection calculations, it finds the **entering and leaving points** directly

---

4. Clipping Window

The rectangular clipping window is defined by:

- x_{min}, x_{max}
- y_{min}, y_{max}

---

5. Parametric Representation of a Line

A line segment between points
$(x_1, y_1)$and $(x_2, y_2)$is written as:

$$x(t) = x_1 + t(x_2 - x_1)$$
$$y(t) = y_1 + t(y_2 - y_1)$$

Where:

$$0 \leq t \leq 1$$

---

6. Clipping Conditions

The line must satisfy the following inequalities:

$$x_{min} \leq x(t) \leq x_{max}$$
$$y_{min} \leq y(t) \leq y_{max}$$

These are rewritten in the form:

$$p_i \cdot t \leq q_i$$

---

7. Values of $p$and q

Boundary	p	q
Left	$-(x_2 - x_1)$	$x_1 - x_{min}$
Right	$x_2 - x_1$	$x_{max} - x_1$
Bottom	$-(y_2 - y_1)$	$y_1 - y_{min}$
Top	$y_2 - y_1$	$y_{max} - y_1$

---

8. Parameter Values t

$$t = \frac{q}{p}$$

Two values are maintained:

- t_{enter}→ maximum of entering values
- t_{leave}→ minimum of leaving values

Initial values:

$$t_{enter} = 0, t_{leave} = 1$$

---

9. Algorithm Steps

Step 1

Initialize:

$$t_{enter} = 0, t_{leave} = 1$$

Step 2

For each boundary:

- If $p = 0$and $q < 0$:
  ❌ Line is completely outside → Reject
- Else:
  - Compute $t = q/p$

Step 3

- If $p < 0$:

$$t_{enter} = \max(t_{enter}, t)$$

- If $p > 0$:

$$t_{leave} = \min(t_{leave}, t)$$

Step 4

If:

$$t_{enter} > t_{leave}$$

❌ Reject the line

Else:
✓ Accept and compute clipped endpoints

---

10. Final Clipped Line

The visible portion is between:

$$t = t_{enter} \text{and} t = t_{leave}$$

Coordinates:

$$x = x_1 + t(x_2 - x_1)$$
$$y = y_1 + t(y_2 - y_1)$$

---

11. Advantages

✓ More efficient than Cohen–Sutherland
✓ Fewer intersection calculations
✓ Directly computes clipped line
✓ Suitable for real-time graphics

---

12. Disadvantages

❌ More complex to understand
❌ Mathematical calculations involved

---

13. Comparison with Cohen–Sutherland

Feature	Cohen–Sutherland	Liang–Barsky
Approach	Region codes	Parametric
Speed	Moderate	Faster
Intersections	Repeated	Minimal
Complexity	Simple	Slightly complex

Polygon Clipping

1. Introduction

In computer graphics, objects such as windows, icons, and shapes are often represented as **polygons**.
When a polygon lies partially outside the viewing window, the **invisible parts must be removed**, while keeping the visible shape intact.
This process is called **polygon clipping**.

---

2. Definition

Polygon clipping is the process of **removing the portions of a polygon that lie outside the clipping window** and generating a new polygon that lies completely inside the window.

---

3. Why Polygon Clipping is Needed

- To display only visible parts of objects
- To maintain correct shapes inside the window
- To improve rendering efficiency
- Used in 2D viewing pipeline

---

4. Clipping Window

- Usually a **rectangular window**
- Defined by:
    - x_{min}, x_{max}
    - y_{min}, y_{max}

---

5. Sutherland–Hodgman Polygon Clipping Algorithm

Definition

The **Sutherland–Hodgman algorithm** is a widely used polygon clipping algorithm that clips a polygon **edge by edge** against each boundary of the clipping window.

---

6. Basic Idea

- The polygon is clipped **one boundary at a time**
- Boundaries are processed in order:
    1. Left
    2. Right
    3. Bottom
    4. Top
- Output of one boundary becomes input for the next

---

7. Algorithm Steps

Step 1

Take the list of polygon vertices as input.

Step 2

Clip the polygon against the **left boundary**.

Step 3

Use the output vertices and clip against the **right boundary**.

Step 4

Repeat the process for **bottom** and **top boundaries**.

Step 5

The final output is the **clipped polygon** inside the window.

---

8. Four Cases in Polygon Clipping

For each edge of the polygon, consider the current vertex (**P**) and next vertex (**Q**).

Case 1: Inside → Inside

- Both P and Q are inside the window
  ✓ Output Q

Case 2: Inside → Outside

- P is inside, Q is outside
  ✓ Output intersection point only

Case 3: Outside → Inside

- P is outside, Q is inside
  ✓ Output intersection point
  ✓ Output Q

Case 4: Outside → Outside

- Both P and Q are outside
  ❌ Output nothing

---

9. Intersection Calculation

Intersection points are calculated using **line equations** with window boundaries.

Example (Left boundary $x = x_{min}$):

$$x = x_{min}$$
$$y = y_1 + m(x_{min} - x_1)$$

---

10. Example (Conceptual)

- Given a polygon partially outside the window
- After clipping against all four boundaries
- A new polygon is generated
- Shape remains closed and valid

---

11. Advantages

✓ Simple and systematic
✓ Preserves polygon shape
✓ Easy to implement
✓ Works well for convex polygons

---

12. Disadvantages

❌ Not suitable for concave polygons without modification
❌ Intersection calculations required
❌ Works only for rectangular windows

---

13. Applications

- Window systems
- CAD applications
- Computer graphics rendering
- Game graphics

---

14. Comparison with Line Clipping

Feature	Line Clipping	Polygon Clipping
Object	Line segment	Polygon
Output	Line segment	New polygon
Algorithms	Cohen–Sutherland, Liang–Barsky	Sutherland–Hodgman
