Chapter 1 : Introduction to Computer Graphics

Definition: A branch of computer science focused on computerized image creation and manipulation. Uses technology to generate images, from simple shapes to complex scenes.

- Involves computations, creation, and data manipulation.
- Acts as a rendering tool for generating and editing images.

Process & Importance:

- The process of creating computer-generated images is similar to manual image creation.
- Computational steps enhance various computer graphics applications.

Scope:

- Encompasses rendering (drawing images on machines).
- Includes photos, sketches, animations, and visual representations of unseen objects (e.g., internal body parts).

Applications:

- Common in user interfaces, TV commercials, and motion pictures.
- Used in engineering, business graphics, and mathematical modeling.
- Enables research by creating 2D and 3D images.

Technological Advances:

- Development of hardware and algorithms to improve image generation speed.
- Supports model creation, storage, and manipulation across various fields.

Interactivity & User Control:

- Modern computer graphics allow users to interact and modify object structures via input devices.

Goal of Computer Graphics:

- Enhance the realism of computer-generated images.
- Improve color accuracy and material appearance in images.
- Known as "real synthesis of the image."

Definition of Computer Graphics:

- Broadly includes all visual elements on a computer, even text and sound.
- Specifically refers to:
    - Representation and manipulation of image data.
    - Technologies for creating and modifying images.
    - A sub-field of computer science focused on visual content processing.

Transformation in Computer Graphics:

- Deals with positioning and modifying shapes like triangles.
- Uses matrices for mapping images in space.
- Transformation matrix controls location, orientation, and resizing of images.

Definition of Computer Graphics:

- Uses technology to transform and present information visually.
- Common in user interfaces, TV commercials, and motion pictures.
- Involves creating images using a computer, including business graphs, drawings, and engineering models.

Capabilities & Applications:

- Supports 2D and 3D image creation for research.
- Uses hardware and algorithms to enhance image generation speed.
- Enables storage and manipulation of object models in fields like engineering and mathematics.
- Modern computer graphics are interactive, allowing users to modify objects using input devices.

Key Aspects:

- Uses computers to create and manipulate images on a display device.
- Involves software techniques for storing, modifying, and representing images.
- Graphics are made up of pixels (smallest unit of an image on a screen).

Types of Computer Graphics:

1. **Interactive Computer Graphics:**

    - Users interact with images on the screen.
    - Two-way communication between the user and the computer.
    - Example: Video games, where users control image movement.

    Advantages of Interactive Computer Graphics:

    1. Higher image quality.
    2. More precise results or products.
    3. Increased productivity.
    4. Reduced analysis and design costs.
    5. Enhances data understanding and trend perception.

Working of Interactive Computer Graphics

Components of a Modern Graphics Display:

1. **Frame Buffer (Digital Memory)**
    - Stores the image displayed on the screen.
    - Each pixel in the raster has a corresponding memory bit (bit plane).
    - Example: A **1024 × 1024** raster requires **1,048,576** memory bits in a single bit plane.
    - A **single bit plane** results in a monochrome (black & white) display.
    - 1 bit = 1 pixel = 2 colors (black & white)

      Note: Total Colors=2^bits per pixel

2. **Monitor (Similar to a TV Set)**
    - Displays the image stored in the frame buffer.
    - Lacks tuning and receiving electronics found in home TVs.

3. **Display Controller (Video Controller)**
    - Transfers the frame buffer's contents to the monitor.
    - Converts digital frame buffer data for display on analog CRT screens.

Properties of a Video Monitor:

1. **Persistence**
    - The duration a phosphor continues to emit light after the electron beam is removed.
    - Different phosphors have varying persistence levels.

      Phosphors are materials that emit light when exposed to radiation (such as electron beams or UV light). They are commonly used in **cathode ray tube (CRT)** displays, including older **television** and **computer monitors**.

2. **Resolution**
    - The number of pixels used to form an image on the display.
    - Higher resolution provides better image clarity.

3. **Aspect Ratio**
    - The ratio of screen width to height.

- Measured in units of length or pixels.

2. Non-Interactive Computer Graphics:

- **Definition:** In non-interactive computer graphics, the user has **no control** over the image. The image is a result of a **static stored program** that follows a set of instructions.
- **Operation:** The image is produced based on pre-programmed instructions and will not change unless the program is modified. The user can only view the image, but cannot interact with it.
- **Example:** Screensavers, where the image or animation runs automatically without user intervention.
- **Communication:** It involves **one-way communication** from the computer to the user. The user can only observe the image, without the ability to modify it.

History of Computer Graphics

- **Introduction:** Computer Graphics (CG) originated as a tool for visualizing data, initially developed for scientists and engineers in government and corporate research centers, such as Bell Labs and Boeing in the 1950s. Later, academic institutions like Ohio State University, MIT, and the University of Utah became key players in advancing CG technology during the 60s and 70s.

- **Founders:** The term "Computer Graphics" was coined by Verne Hudson and William Fetter of Boeing.

- **Early Milestones in CG Development:**

    - **1940-1941:** The first digital computer-generated graphics were created at MIT.
    - **1946:** The first public presentation of computer-generated images occurred at the National Technical Conference.
    - **1950s-1960s:** John Whitney Sr. and other pioneers used computer mechanisms for creating graphic artwork and short films. IBM also began using early graphical systems, and the first light pen input device was introduced.
    - **1960s:** Early milestones include the first commercial computer graphics systems like the IBM 2250, and the development of early animation systems.
    - **1970s:** Major advancements included the invention of the Z-buffer algorithm, texture mapping, and Phong shading. These developments contributed to the rise of 3D graphics and computer animation.
    - **1980s:** The first broad use of 3D graphics occurred in Disney films, and medical imaging software using Voxel technology emerged. Video game graphics also gained popularity, with titles like Donkey Kong marking the era.
    - **1990s:** The introduction of VGA and SVGA standards, the rise of the web, and early developments in web browsers like Mosaic revolutionized computer

graphics.

- **Recent Developments:**
    - **2000s:** Web-based CAD systems, such as Sketchup, were launched. Google acquired Sketchup in 2006.
    - **2009-Present:** Real-time, realistic graphics became achievable on mobile devices, and big data began being used for animation.
- **Key Advances Today:** As of 2018, mobile phones can create highly realistic graphics, and CGI-based human faces can be created in real-time.

Why is Computer Graphics Used?

- **Efficient Representation of Data:**
    - Large amounts of data, like a shoe company's sales for five years, can be overwhelming. Storing and analyzing such data can be time-consuming and resource-intensive.
    - Graphics, such as charts and graphs, offer an effective way to represent complex data visually, making it easier to understand at a glance.
    - Interactive computer graphics allow two-way communication between the user and the computer, where images are modified quickly based on user input.

Applications of Computer Graphics:

1. **Education and Training:**
    - **Educational Aids:** Computer-generated models of physical, financial, and economic systems are used as learning tools.
    - **Flight Simulator:** Pilots train on flight simulators, which offer:
        - Fuel saving
        - Increased safety
        - Familiarization with a wide range of airports

2. **Use in Biology:**
    - **Molecular Biology:** Computer graphics help biologists visualize molecules, aiding in the study of their structure and behavior.

3. **Computer-Generated Maps:**
    - **Urban Planning:** Town planners and transportation engineers use computer-generated maps to aid in planning and resource management.

4. **Architecture:**
    - **Design Exploration:** Architects use interactive graphics to explore multiple design solutions efficiently, testing alternatives that may not be feasible without

computer assistance.

5. **Presentation Graphics:**
    - **Common Examples:** Bar charts, line graphs, pie charts.
    - **Used for Summarizing Reports:**
        - Financial Reports
        - Statistical Data
        - Scientific, Mathematical, and Economic Reports
        - Managerial Reports
        - Consumer Information Bulletins

6. **Computer Art:**
    - **Commercial Arts:** Used in the creation of television commercials, advertising graphics, and visual designs.

7. **Entertainment:**
    - **Movies, Music, and TV Shows:** Computer graphics are heavily used in the production of motion pictures, music videos, and television shows to create special effects and animations.

8. **Visualization:**
    - **Data Visualization:** Scientists, engineers, medical professionals, and business analysts use computer graphics to visualize and interpret large datasets.

9. **Educational Software:**
    - **Computer-Aided Instruction:** Graphics are used in the development of educational software to enhance the learning experience.

10. **Printing Technology:**
    - **Textile and Printing Design:** Used in textile design and printing technology for creating detailed and accurate designs.

Examples of Computer Graphics Packages:

➔ LOGO
➔ COREL DRAW
➔ AUTO CAD
➔ 3D STUDIO
➔ CORE
➔ GKS (Graphics Kernel System)
➔ PHIGS
➔ CAM (Computer Graphics Metafile)
➔ CGI (Computer Graphics Interface)

Display Processor

- A **Display Processor** is either an interpreter or a hardware component that converts display code into pictures.
- It is one of the four main parts of the display system.

Parts of Display Processor:

1. **Display File Memory:**
    - Used for the generation of pictures.
    - Identifies graphic entities.
2. **Display Processor:**
    - Main processor that converts data into visual output.
3. **Display Generator:**
    - Responsible for generating characters and curves.
4. **Display Console:**
    - Contains the CRT, Light Pen, Keyboard, and deflection system for user interaction.

Display Controller Functions:

1. **Handles Interrupts:** Manages interruptions in processing.
2. **Maintains Timings:** Keeps track of time for display operations.
3. **Interprets Instructions:** Decodes and executes display instructions.

Raster Scan System:

- A combination of processing units, including a **Control Processing Unit (CPU)** and a **Display Controller** (also known as a video controller).
- **Display Controller:** Controls the operation of the display device, generating horizontal and vertical drive signals to sweep the screen during raster scans.

Working of Display Processor:

1. **Registers:**
    - Two registers, X and Y, store the coordinates of screen pixels.
    - **Y register**: Increments upward from 0 (bottom) to ymax (top).
    - **X register**: Increments horizontally from 0 (leftmost) to xmax (rightmost).
    - The origin is at the bottom-left corner, following a Cartesian coordinate system.
2. **Refresh Cycle:**
    - At the start, **X register** is set to 0, and **Y register** is set to ymax.
    - This (x, y) address is translated into a memory address in the frame buffer, where the pixel color value is stored.

    - The controller retrieves this value (binary number), splits it into three parts, and sends each part to a Digital-to-Analog Converter (DAC) to control the intensity of the e-beam.
3. **Processing Each Pixel:**
    - The process is repeated for each pixel along the scan line, incrementing the X register.
    - When the first scan line is completed, the X register resets to 0, and the Y register is decremented to move to the next scan line.
    - This procedure is repeated for each scan line until the last scan line (y = 0) is processed.
4. **Color Look-up Table:**
    - For systems using a color look-up table, the frame buffer value is not directly used to control the beam intensity but as an index to fetch color values from the table.
    - This process is repeated for each pixel in the display cycle.
5. **Efficient Access to Frame Buffer:**
    - To avoid time delays, multiple adjacent pixels are fetched in a single access to the frame buffer.
    - The pixel values are stored in a register and processed in groups, controlling the beam intensity for each pixel in succession.

This process ensures that the screen is refreshed and displayed pixel by pixel, with efficient handling of color values and pixel intensities for smooth visual output.

Display Devices

- **Most Common Display Device**: Video monitors, primarily based on **Cathode Ray Tube (CRT)** technology.

Types of Display Devices:

1. **Refresh Cathode Ray Tube (CRT)**
2. **Random Scan and Raster Scan**
3. **Color CRT Monitors**
4. **Direct View Storage Tubes**
5. **Flat Panel Display**
6. **Lookup Table**

A. Cathode Ray Tube (CRT):

- **CRT Technology**: Used in traditional computer monitors and televisions.

- **Image Creation**: Electrons are fired from the back of the CRT towards a phosphor-coated screen, causing it to light up. Colors are created by mixing red, green, and blue light.

Components of CRT:

1. **Electron Gun**:
    - Consists of heating filament (heater) and cathode.
    - Creates a stream of electrons focused into a narrow beam directed at the screen.
2. **Control Electrode**:
    - Turns the electron beam on and off.
3. **Focusing System**:
    - Focuses the electron beam into a narrow, clear beam for a sharp image.
4. **Deflection Yoke**:
    - Controls the direction of the electron beam by generating an electric or magnetic field.
    - Linked to a **sweep or scan generator** for controlling beam movement.
5. **Phosphor-Coated Screen**:
    - Inside surface is coated with phosphors that glow when struck by the electron beam.
    - **Phosphorescence** is the light emitted by phosphors after being exposed to an electron beam.

This technology forms the foundation of older display systems, though it has been replaced by newer technologies like LCD and LED.

B. Random Scan Display and Raster Scan

I. Random Scan

Working Principle:

- Uses an **electron beam** like a pencil to create line images on a **CRT screen**.
- The image is formed by drawing **straight-line segments** by directing the beam from one point to another based on **x & y coordinates**.
- After drawing the image, the system **refreshes** the lines **30 to 60 times per second**

Other Names:

- Also called **Vector Displays**, **Stroke-Writing Displays**, or **Calligraphic Displays**.

Advantages of Random Scan Displays

1. The **electron beam** is directed only to the parts of the screen where an image needs to be drawn.

2. Produces **smooth line drawings**.
3. Offers **high resolution**.

Disadvantages of Random Scan Displays

1. Cannot display **realistic shaded scenes**.

II. Raster Scan Display

Working Principle:

    - Uses **intensity control** of pixels in a **rectangular grid (raster)** on the screen.
    - **On/Off pixel information** is stored in the **Refresh Buffer (Frame Buffer)**.
    - Used in **televisions and modern displays**.
    - Capable of **storing pixel data** for realistic object rendering.
    - Provides a **refresh rate of 60 to 80 frames per second**.
- **Frame Buffer (Raster/Bitmap)**:
    - Stores pixel positions, known as **picture elements (pixels)**.
    - Beam movement includes **horizontal retracing** and **vertical retracing**.

Types of Scanning in Raster Scan

1. **Interlaced Scanning**:
    - Traces horizontal lines **top to bottom** in two passes.
    - Can cause **fading effects**.
    - Uses a **refresh rate of 60 frames per second**.
2. **Non-Interlaced Scanning**:
    - **Odd-numbered lines** are scanned first, then **even-numbered lines**.
    - Refresh rate is **30 frames per second** but can cause **flickering**.

Advantages of Raster Scan Displays

1. **Realistic image rendering**.
2. Can generate **millions of different colors**.
3. Supports **shadow effects and shading**.

Disadvantages of Raster Scan Displays

1. **Lower resolution** compared to Random Scan.
2. **Expensive** to implement.

Random Scan	Raster Scan
1. It has high Resolution	1. Its resolution is low.
2. It is more expensive	2. It is less expensive
3. Any modification if needed is easy	3.Modification is tough
4. Solid pattern is tough to fill	4.Solid pattern is easy to fill
5. Refresh rate depends or resolution	5. Refresh rate does not depend on the picture.
6. Only screen with view on an area is displayed.	6. Whole screen is scanned.
7. Beam Penetration technology come under it.	7. Shadow mark technology came under this.
8. It does not use interlacing method.	8. It uses interlacing
9. It is restricted to line drawing applications	9. It is suitable for realistic display.

C. Color CRT Monitors

- **Color CRT Displays** use **phosphor coatings** to generate colors.
- Two main methods for producing color displays:
    1. **Beam Penetration Method**
    2. **Shadow-Mask Method**

1. Beam Penetration Method

- Used in **random-scan monitors**.
- The **CRT screen is coated with two phosphor layers**: **Red & Green**.
- The **displayed color depends on electron beam speed**:
    - **Slow electrons → Outer red layer → Red color**.
    - **Fast electrons → Inner green layer → Green color**.
- **Only 4 colors** can be produced: **Red, Green, Orange, and Yellow**.

Advantages

✔ **Inexpensive** method.

Disadvantages

✖ **Only four colors possible**.
✖ **Lower picture quality** compared to the Shadow-Mask method.

2. Shadow-Mask Method

- **Commonly used in Raster-Scan Systems**.
- Produces a **wider range of colors** than the Beam Penetration method.
- Used in **color TV sets and monitors**.

Construction

- **Each pixel contains three phosphor dots**:
    - **Red phosphor dot**
    - **Green phosphor dot**
    - **Blue phosphor dot**
- **Three electron guns** (one for each color).
- **Shadow mask grid** behind the phosphor screen with **small holes** in a **triangular pattern**.

Working Principle

- **Three beams pass through the shadow mask holes**, activating **corresponding color phosphors**.
- **Triad arrangement**: Red, Green, and Blue guns work together to form a **small color spot** on the screen.
- **Inline arrangement**:
    - Instead of a **triangular pattern**, **electron guns are aligned along a scan line**.
    - **Easier alignment**, commonly used in **high-resolution color CRTs**.

Advantages

✔ **Realistic images**.
✔ **Millions of colors can be generated**.
✔ **Supports shadow effects and shading**.

Disadvantages

✖ **More expensive than monochrome CRTs**.
✖ **Lower resolution compared to other modern display technologies**.
✖ **Convergence problems** (difficulty aligning electron beams accurately).

D. Flat Panel Display

- **Flat-Panel Displays** are video devices that offer **reduced volume, weight, and power consumption** compared to **CRT displays**.
- **Common examples**: Small TV monitors, calculators, pocket video games, laptops, and advertisement boards in elevators.

Types of Flat Panel Displays

1. Emissive Display

- Converts **electrical energy into light**.
- Examples:
    - **Plasma Panel**
    - **Thin Film Electroluminescent Display**
    - **LED (Light Emitting Diodes)**

2. Non-Emissive Display

- Uses **optical effects** to convert **external light sources (e.g., sunlight) into graphical patterns**.
- Example: **LCD (Liquid Crystal Display)**

E. Plasma Panel Display Summary

- Also called **Gas-Discharge Display**.
- Consists of an **array of small fluorescent lights**.

Components

1. **Cathode**
    - Made of fine wires.
    - Delivers **negative voltage** to gas cells along the **negative axis**.
2. **Anode**
    - Made of line wires.
    - Delivers **positive voltage** along the **positive axis**.
3. **Fluorescent Cells**
    - Small **gas-filled pockets** (neon gas).
    - Emits light when voltage is applied.
4. **Glass Plates**
    - Act as **capacitors**.
    - Maintain a **continuous glow** when voltage is applied.

Working Principle

- **Gas glows** when there is a significant **voltage difference** between horizontal and vertical wires (**90V - 120V**).
- Plasma panels **do not require refreshing**.
- **Erasing is done** by reducing the voltage to **90V**.
- Each plasma cell has **two states (ON/OFF)**, making it **stable**.
- **Resolution**: Up to **512 × 512 pixels**.

Advantages

✔ **High resolution**
✔ **Large screen sizes** possible

✔ **Less volume & weight**
✔ **Flicker-free display**

Disadvantages

✖ **Poor resolution** compared to modern displays
✖ **Complex wiring** for anode & cathode
✖ **Difficult addressing mechanism**

F. LED (Light Emitting Diode) Display

- **Uses a matrix of diodes** to form pixels.
- **Picture data is stored** in a refresh buffer.
- Data is converted to **voltage levels**, which **control the diodes** to produce a light pattern.

G. LCD (Liquid Crystal Display)

- Produces images by **passing polarized light** through a **liquid-crystal material**.
- Uses **two glass plates** at **right angles**, filled with liquid crystal.
- **Row conductors (vertical) & column conductors (horizontal)** define pixel positions.
- LCDs are **temperature-dependent** (0-70°C).
- Requires **very little power** and is **flat & lightweight**.

Advantages of LCD

✔ **Low power consumption**
✔ **Small size**
✔ **Low cost**

Disadvantages of LCD

✖ **Temperature-sensitive** (0-70°C)
✖ **Does not emit light** → Poor contrast
✖ **No color capability** (for basic LCDs)
✖ **Lower resolution** compared to CRT

Input Devices

Input devices are hardware used to transfer data to a computer. The data can be in the form of **text, graphics, sound, or commands**.

Types of Input Devices:

1. **Keyboard** – Used for typing text and commands.

2. **Mouse** – A pointing device for navigation and selection.
3. **Trackball** – A stationary device with a rotating ball for navigation.
4. **Spaceball** – Similar to a trackball but allows 3D movement.
5. **Joystick** – Used for gaming and 3D navigation.
6. **Light Pen** – A stylus-like device for direct screen interaction.
7. **Digitizer** – Converts drawings or designs into digital form.
8. **Touch Panels** – Allow direct input through touch.
9. **Voice Recognition** – Converts spoken words into digital input.
10. **Image Scanner** – Converts physical images into digital form.

Output Devices

Output devices translate processed data from a computer into a **form that humans can understand**, such as text, graphics, or sound.

Types of Output Devices:

1. **Printers** – Convert digital documents into physical printouts.
2. **Plotters** – Used for printing large-scale graphics and technical drawings.

Chapter 2 : Graphics Primitives

Scan Conversion (Rasterization)

- **Definition:** Representing regular objects as discrete pixels is called **Scan Conversion** (also known as **Rasterization**).
- Every graphics system must transform primitives into a collection of pixels.
- Each pixel in the graphical system has two states: **on or off**.

Scan Converting Rate

- A technique used in **video processing** to change **horizontal and vertical frequency** for various purposes.
- A **scan converter** is used to perform scan conversions.
- **Lines** are defined by their **starting and ending points**.
- **Circles** are defined by their **radius, equation, and center point**.

Methods of Scan Conversion

1. **Analog Method**
    - Best for **analog videos**.
    - Uses a **large number of delay cells**.
    - Also known as **non-retentive, memory-less, or real-time method**.
2. **Digital Method**
    - Also known as the retentive **or buffered method**.

    - Uses two speeds:
        - **n_1 speed** – Stores the image in a **line or frame buffer**.
        - **n_2 speed** – Reads the image.

Pixel (Picture Element - Pel)

- **Definition:** A **rectangular dot** centered at an **integral position on an integral grid**.
- **Addressable real point** in a graphics system.

Pixel Attributes

1. **Intensity** – Defines the **brightness** of the image.
2. **Name/Address** – Defines the **position** of the pixel.

Image Size & Aspect Ratio

- **Image size** is measured by **total horizontal × total vertical pixels** (e.g., **512×512, 640×480, 1024×768**).
- **Aspect Ratio:** The **width-to-height ratio** of an image, measured in unit length or number of pixels.

Advantages of Scan Conversion

- Used in various applications such as **video projectors, TV, HDTV, video capture cards, LCD monitors, etc.**
- Has **wide applications** in display technologies.
- Can be performed **efficiently** using **high-speed integrated circuits**.

Disadvantages of Scan Conversion

- Can only be applied with **LSI (Large Scale Integration) and VLSI (Very Large Scale Integration) integrated circuits**.
- In **digital scan conversion**, the **analog video signal** is converted into **digital data**.

DDA (Digital Differential Analyzer) Line Drawing Algorithm

- **Purpose:**
    - Interpolates variables between two points.
    - Used for **rasterization** of polygons, lines, and triangles.
- **Key Features:**
    - Also called an **incremental scan conversion method**.
    - Performs calculations **step-by-step**, using the **previous step's result** for the next step.

Algorithm Steps:

1. **Start.**
2. Define **starting point (x_1, y_1)** and **ending point (x_2, y_2)**
3. Compute **Δx, Δy, and slope (m)**:

    - $\Delta x = x_2 - x_1$
    - $\Delta y = y_2 - y_1$
    - $m = \Delta y / \Delta x$

4. Handle three cases based on the slope mm:
    - **If m < 1:**
        - **Increment x by 1** and compute y.
        - Formula: $(x_{k+1}, y_{k+1}) = (x_k + 1, y_k + m)$
    - **If m>1:**
        - **Increment y by 1** and compute x.
        - Formula: $(x_{k+1}, y_{k+1}) = (x_k + 1/m, y_k + 1)$
    - **If m=1:**
        - **Increment both x and y by 1**.
        - Formula: $(x_{k+1}, y_{k+1}) = (x_k + 1, y_k + 1)$
5. **Repeat Step 4** until the ending point (x2,y2)(x_2, y_2) is reached.
6. **Stop.**

Advantages of Digital Differential Analyzer (DDA):

- **Simple to implement:** The algorithm is easy to understand and apply.
- **Faster than direct line equation:** DDA performs line drawing more efficiently compared to using direct mathematical equations.
- **No multiplication required:** Unlike some other algorithms, DDA doesn't rely on multiplication, making it simpler.
- **Tracks point overflow:** It helps in identifying when the point moves beyond its expected position.

Disadvantages of Digital Differential Analyzer (DDA):

- **Floating-point arithmetic is time-consuming:** DDA involves floating-point operations, which can slow down the computation.
- **Round-off errors are time-consuming:** The algorithm's rounding of values may add complexity and delay.
- **Inaccurate point positions:** Sometimes the positions of points might not be perfectly accurate, especially with large lines.

Bresenham's Line Drawing Algorithm

- **Introduction:**
  Bresenham's Line Drawing Algorithm, introduced by Jack Elton Bresenham in 1962, is a highly efficient method for drawing lines by using incremental integer calculations (addition, subtraction, multiplication). It is especially useful in scan conversion of lines in computer graphics.

Steps of the Algorithm:

1. **Start.**
2. **Initialize the starting and ending points:**
   - Let the starting point be **(x_1, y_1)** and the ending point be **(x_2, y_2)**.
3. **Calculate Δx and Δy:**
   - **$\Delta x = x_2 - x_1$**
   - **$\Delta y = y_2 - y_1$**
   - **Slope (m) = Δy / Δx**
4. **Calculate the decision parameter (pk):**
   - **$pk = 2\Delta y - \Delta x$**
5. **Decision Making Based on pk:**
   - **Case 1:** If **$pk < 0$**,
     - **$pk_{-1} = pk + 2\Delta y$**
     - **$xk_{-1} = xk + 1$**
     - **$yk_{-1} = yk$**
   - **Case 2:** If **$pk \geq 0$**,
     - **$pk_{-1} = pk + 2\Delta y - 2\Delta x$**
     - **$xk_{-1} = xk + 1$**
     - **$yk_{-1} = yk + 1$**
6. **Repeat step 5:**
   - Continue until you reach the ending point.
   - The total number of iterations is **Δx - 1**.
7. **Stop.**

Key Characteristics:

- **Efficient:** Uses only integer operations (addition, subtraction, and multiplication), making it faster than other methods that require floating-point operations.
- **Decision Parameter (pk):** The decision parameter is used to decide whether to move the y-coordinate or keep it the same at each step.

Advantages of Bresenham's Line Drawing Algorithm:

- **Simple to Implement:** The algorithm only involves integer operations (addition, subtraction, multiplication), making it easy to implement.
- **Quick and Incremental:** The algorithm works in an incremental manner, making it efficient.

- **Faster than other methods:** While not as fast as the Digital Differential Analyzer (DDA) algorithm, it is still faster than many other line-drawing algorithms.
- **Higher Pointing Accuracy:** Bresenham's algorithm provides better pointing accuracy compared to the DDA algorithm.

Disadvantages of Bresenham's Line Drawing Algorithm:

- **Basic Line Drawing Only:** The algorithm is limited to drawing basic straight lines and does not support curves or other complex shapes.
- **Rough Line Appearance:** The lines drawn using this algorithm may not appear smooth due to the pixelated nature of the raster display.

Mid-Point Line Drawing Algorithm in Computer Graphics

- **Introduction:**
  - The **Mid-point Subdivision Algorithm** is an extension of the **Cyrus-Beck algorithm**.
  - Introduced by **Pitway and Van Aken**.
  - It is an **incremental line drawing algorithm**, using previous calculations to determine the next point.
  - The algorithm subdivides the line at its midpoints continuously for accurate line plotting.
- **Key Purpose:**
  - Used for drawing a line with close approximation between two points.
  - Helps compute visible areas of lines in a window.
  - Follows the **bisection method** to divide the line into equal partitions.

Algorithm of Mid-Point Subdivision Line Drawing:

1. **Start.**
2. **Consider the starting point** as (x1, y1) and ending point as (x2, y2).
3. **Calculate Δd** using the formula:
   - **Δd = 2(Δy - Δx)**
4. **Calculate the decision parameter (di)**:
   - **di = 2Δy - Δx**
5. **Increment x or y coordinates** based on the decision parameter (di):
   - **Case 1**: If **di < 0**:
     - **xk+1 = xk + 1**
     - **yk+1 = yk**
     - Update **dn = di + 2Δy**
   - **Case 2**: If **di ≥ 0**:
     - **xk+1 = xk + 1**
     - **yk+1 = yk + 1**
     - Update **dn = di + 2(Δy - Δx)**

6. **Repeat Step 5** until the ending point of the line is reached.
7. **Stop.**

Circle Basics

- A circle is an **eight-way symmetric** shape (8 symmetrical points).
- Each quadrant has **2 octants**, and knowing one point allows calculation of 7 others.
- Circle's shape: no corners or sides; all radii from the center to the circumference are equal.

Symmetry Points (Reflection Method)

If one point is (R, S), the 7 others are:

- (R, -S), (-R, -S), (-R, S), (S, R), (S, -R), (-S, -R), (-S, R)

Example:

- Given point (4,6)
- Reflected points: (4, -6), (-4, -6), (-4, 6), (6, 4), (6, -4), (-6, -4), (-6, 4)

Circle Drawing Algorithms

1. **Bresenham's Circle Drawing Algorithm**
2. **Mid-Point Circle Drawing Algorithm**

Bresenham's Circle Drawing Algorithm

🔸 **Features:**

- Uses **integer arithmetic only** (no floating point).
- Faster and suitable for **raster displays**.
- Uses decision parameters to determine closest pixel.
- Based on **circle equation**: $x^2+y^2=r^2$

🔹 **Algorithm Steps:**

1. **Start**
2. Assign start coordinates:
    - x=0
    - y=r
3. Compute initial decision parameter:
    - $d0=3-2r$
4. Use loop to determine next points:
    - **Case 1:** If dk<0

        - x(k+1) = xk+1
        - yk+1=yk
        - dk+1=dk+4x(k+1)+6
    - **Case 2:** If dk≥0d_k \geq 0
        - xk+1=xk+1x_{k+1} = x_k + 1
        - yk+1=yk−1y_{k+1} = y_k - 1
        - dk+1=dk+4(xk+1−yk+1)+10d_{k+1} = d_k + 4(x_{k+1} - y_{k+1}) + 10
5. If circle **center ≠ (0,0)**:
    - Adjust each point:
        - X = Xc+x
        - Y=Yc+y
6. **Repeat until** x≥y
7. **Stop**

Example: Circle of radius 8, center at (0, 0)

- d0=3−2×8=−13d_0 = 3 - 2 × 8 = -13
- Start from (0, 8) and follow steps
- Stop when x≥yx \geq y

Generated points for 1st octant:
(0,8), (1,8), (2,8), (3,7), (4,6), (5,5)

Mirrored across other octants and quadrants using symmetry

Advantages

- Easy to implement.
- Faster due to use of **only integer arithmetic**.
- Based on the standard circle equation.

Disadvantages

- **Less accurate** compared to Mid-point Circle Drawing.
- **Not suitable for complex/high-resolution graphics**.

Midpoint Circle Drawing Algorithm

Algorithm Steps:

1. **Start.**
2. **Initialize center coordinates** as (p_0, q_0):
    - $p_0 = 0$
    - q_0 = r (radius)
3. **Calculate initial decision parameter**:

- $d_0 = 1 - r$

4. **Let current point be (p_k, q_k)**.
   - Determine next point (p_{k+1}, q_{k+1}) using the value of d_k.
5. **Two cases based on d_k**:
   - **Case 1** ($d_k < 0$):
     - $p_{k+1} = p_k + 1$
     - $q_{k+1} = q_k$
     - $d_{k+1} = d_k + 2 \cdot p_{k+1} + 1$
   - **Case 2** ($d_k \geq 0$):
     - $p_{k+1} = p_k + 1$
     - $q_{k+1} = q_k - 1$
     - $d_{k+1} = d_k - 2 \cdot (q_{k+1}) + 2 \cdot p_{k+1} + 1$
6. **If center is not at origin**, draw points as:
   - x = x_c + p
   - y = y_c + q
7. **Repeat steps 5 and 6 until** $p \geq q$.
8. **Stop.**

Example:

- **Given:** Center (0, 0), radius r = 10
- **Start point:** $p_0 = 0$, $q_0 = 10$
- **Initial decision parameter:**
  $d_0 = 1 - 10 = -9$
- **Calculate successive points:**

d	**d_{k+1}**	**(p, q)**
-9	-6	(1, 10)
-6	-1	(2, 10)
-1	6	(3, 10)
6	-3	(4, 9)

-3 8 (5, 9)

8 5 (6, 8)

Coordinates for Octants:

- **Octant 1:** Original calculated points.
- **Octant 2:** Swap p and q.

Octant 1 (p, q)	Octant 2 (q, p)
(0, 10)	(10, 0)
(1, 10)	(10, 1)
(2, 10)	(10, 2)
(3, 10)	(10, 3)
(4, 9)	(9, 4)
(5, 9)	(9, 5)
(6, 8)	(8, 6)

Complete Circle Coordinates by Quadrant:

For each (p, q) from octants, use 8-way symmetry:

Quadrant 1	Quadrant 2	Quadrant 3	Quadrant 4
(p, q)	(-p, q)	(-p, -q)	(p, -q)

All 8 symmetric points are generated this way.

Advantages:

- Efficient and powerful.
- Easy to implement.
- Based on simple circle equation ($x^2 + y^2 = r^2$).
- Good for drawing smooth curves on raster displays.

Disadvantages:

- Can be time-consuming.
- May produce some inaccuracies in circle points.

Scan Conversion of an Ellipse

- An **ellipse** is a closed plane curve formed by the intersection of a plane with a cone.
- It is **symmetrical** in four directions, similar to a circle but elongated.
- An ellipse is defined as the set of points where the **sum of distances** from two fixed points (foci) is constant.

Midpoint Ellipse Drawing Algorithm

This algorithm is used to plot ellipses on raster displays using decision parameters, divided into **two regions**.

Algorithm Steps

Step 1: Read the ellipse radii:

- `rp` = radius along x-axis
- `rq` = radius along y-axis

Step 2: Initialize starting coordinates:

- `p = 0, q = r`

Step 3: Calculate **initial decision parameter for Region 1**:

- `dk1 = r²q - r²p * rq + ¼ * r²p`

Step 4:

Initialize dp and dq as:

- `dp = 2 * r²q * p`
- `dq = 2 * r²p * q`

Step 5: Region 1 Drawing (dp < dq)

Use a **do-while loop**:

Loop condition: while `dp < dq`
Inside the loop:

- Plot (p, q)
- **If `dk1 < 0`:**
  - `p = p + 1, q = q`
  - `dp = dp + 2 * r²q`
  - `dk1 = dk1 + 2 * r²q * p + 2 * r²q + r²q`
- **Else:**
  - `p = p + 1, q = q - 1`
  - `dp = dp + 2 * r²q`
  - `dq = dq - 2 * r²p`
  - `dk1 = dk1 + 2 * r²q * p + 2 * r²q - (2 * r²p * q - 2 * r²p) + r²q`

Step 6: Region 2 Initialization

Calculate **initial decision parameter for Region 2**:

- `dk2 = r²q(p + ½)² + r²p(q - 1)² - r²p * r²q`

Step 7: Region 2 Drawing (q > 0)

Use a **do-while loop**:

Loop condition: while `q > 0`
Inside the loop:

- Plot (p, q)
- **If `dk2 < 0`:**
    - `p = p, q = q - 1`
    - `dq = dq - 2 * r²p`
    - `dk2 = dk2 - (2 * r²p * q - 2 * r²p) + r²p`
- **Else:**
    - `p = p + 1, q = q - 1`
    - `dq = dq - 2 * r²p`
    - `dp = dp + 2 * r²q`
    - `dk2 = dk2 + 2 * r²q * p + 2 * r²q - (2 * r²p * q - 2 * r²p) + r²q`

Step 8: Plot points in **other three quadrants** using symmetry.

Step 9: Stop.

Chapter Three: Transformations

Introduction

Transformation in computer graphics refers to modifying an image's appearance by applying specific rules or algorithms. An image, made up of shapes like lines, rectangles, circles, or triangles, can undergo transformations to:

- Change its **position**
- Adjust its **size**
- Alter its **angle**

The resulting image after applying these operations is called a **transformed image**.

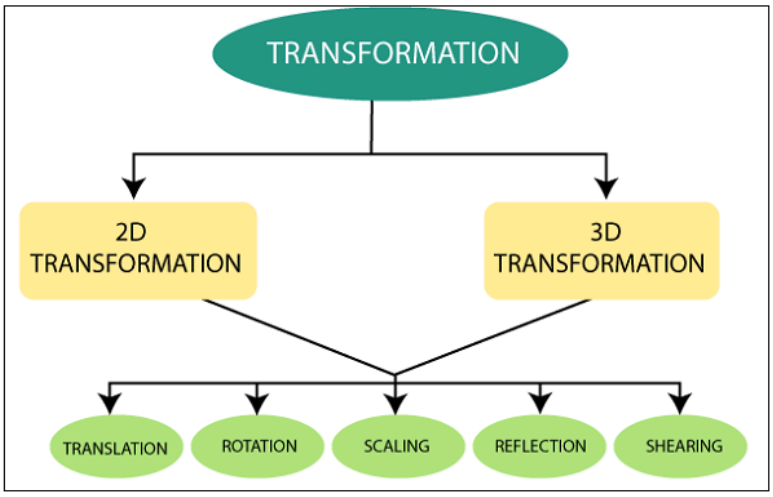
Types of Object Transformation

1. Geometric Transformation

- The **object moves**, but the **background remains fixed**

2. Coordinate Transformation

- The **background moves**, while the **object stays fixed**.



Categories of Transformation

1. Two-Dimensional (2D) Transformation

Involves movement and changes in a 2D plane (x- and y-axis). Types include:

- **Translation**: Moves the object to a new position.
- **Rotation**: Rotates the object around the origin by a specified angle.
- **Scaling**: Resizes the object.
- **Reflection**: Flips the object over a line, typically 180°.
- **Shearing (Skewing)**: Slants the object in the x or y direction.

2. Three-Dimensional (3D) Transformation

Involves movement and changes in 3D space (x-, y-, and z-axis). Types include:

- **Translation**: Moves the object in 3D space.
- **Rotation**: Rotates the object around the origin or an axis.
- **Scaling**: Resizes the object in all three dimensions.
- **Reflection**: Flips the object at an angle (commonly 180°) in 3D space.
- **Shearing (Skewing)**: Slants the object including the z-axis.

2D Translation

2D Translation is a geometric transformation that moves an object from one position to another along a straight path in the 2D plane. It involves adding a constant distance (translation vector) to the coordinates of each point of the object.

Basics

- **Original Point:** P(x, y)

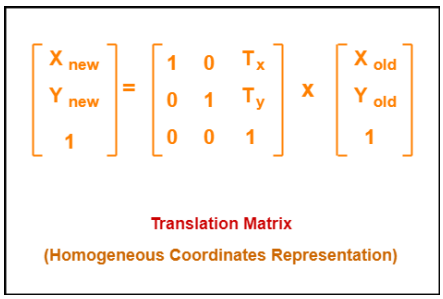
- **Translation Vector (Tx, Ty):** the distance to shift the object.
- **Translated Point:** P'(x', y') where:
    - **x' = x + Tx**
    - **y' = y + Ty**

This operation does not alter the size or shape of the object; it simply moves the object without rotation or distortion.

3. Matrix Representation

In homogeneous coordinates, a 2D point (x, y) becomes (x, y, 1).

Translation matrix (3x3):



Matrix multiplication:

| x' | | 1 0 Tx | | x |

| y' | = | 0 1 Ty | * | y |

| 1 | | 0 0 1 | | 1 |

Resulting in:

x' = x + Tx

y' = y + Ty

Example 1: Translate a Single Point

- **Initial Point:** (2, 1)
- **Translation Vector:** (Tx = 2, Ty = 3)
- **New Point:**
    - x' = 2 + 2 = 4
    - y' = 1 + 3 = 4
    - **Result:** (4, 4)

2D Rotation

- **2D Rotation** is the movement of an object around a fixed point (usually the origin) by a certain angle in the XY-plane.
- **Key Terms:**
    1. **Rotation Point (Pivot):** The fixed point the object rotates around.
    2. **Rotation Angle (θ):**
        - Positive (+): Anti-clockwise rotation
        - Negative (−): Clockwise rotation
- **Rotation Formula (around origin):**
  $x' = x\cos\theta - y\sin\theta$ and $y' = x\sin\theta + y\cos\theta$
- **Matrix Form (2x2):(-sin tetha)**

$$\begin{pmatrix} P_1 \\ Q_1 \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

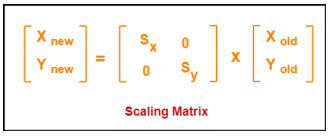
- **Homogeneous Matrix (3x3):**
  Used for transformations including rotation, translation, and scaling.

$$\begin{pmatrix} P_1 \\ Q_1 \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} x \begin{pmatrix} P_0 \\ Q_0 \\ 1 \end{pmatrix}$$

- **Applications:** Rotation can be applied to lines, curves, polygons, circles, and any 2D shapes.
- **Example:** Rotating point (5,5) by 30° anticlockwise gives approx. (1.83, 6.83).

2D Scaling

- **2D Scaling** is used to **resize an object** by stretching or shrinking it along the X and Y axes.
- **Scaling Factors:**
    - **Sx**: Scaling along the X-axis
    - **Sy**: Scaling along the Y-axis
    - If **Sx or Sy > 1** → object is **enlarged**
    - If **Sx or Sy < 1** → object is **shrunk**
- **Scaling Equations:**
  $x' = x \cdot Sx$ and $y' = y \cdot Sy$
- **Matrix Form (2x2):**



- **Homogeneous Matrix (3x3):** Used for combining scaling with translation and rotation in graphics.

Example: Given Square Coordinates:

- P(1, 4), Q(4, 4), R(4, 1), S(1, 1)
  Scaling Factors: Sx = 3, Sy = 4

New Coordinates after Scaling:

- P → (3, 16)
- Q → (12, 16)
- R → (12, 4)
- S → (3, 4)

2D Reflection

- **Reflection** creates a **mirror image** of an object across a line (axis or diagonal).
- The **size and shape remain unchanged**—only the position changes.
- Reflections are commonly performed across:
  - **X-axis**
  - **Y-axis**
  - **Origin**
  - **Lines like Y = X or Y = -X**

Types of Reflections and Their Equations:

1. **Reflection along the X-axis**
   - $X' = X$
   - $Y' = -Y$
2. **Reflection along the Y-axis**
   - $X' = -X$
   - $Y' = YY' = Y$
3. **Reflection through the Origin (XY-plane)**
   - $X' = -X$
   - $Y' = -Y$
4. **Reflection along Y = X**
   - $X' = Y$
   - $Y' = X$
5. **Reflection along Y = -X**
   - $X' = -Y$
   - $Y' = -X$

Matrix Representation (Homogeneous Coordinates – 3x3 Matrix):

Each reflection type has a corresponding 3x3 matrix, used to perform transformations in computer graphics.

Example 1 – Reflection on the X-axis:

Given Triangle:

- A(3, 4), B(6, 4), C(5, 6)

Reflected across X-axis:

- A → (3, -4)
- B → (6, -4)
- C → (5, -6)

Example 2 – Reflection on the Y-axis:

Given Triangle:

- A(3, 4), B(6, 4), C(5, 6)

Reflected across Y-axis:

- A → (-3, 4)
- B → (-6, 4)
- C → (-5, 6)

2D Shearing

Shearing is a transformation that alters the shape of a 2D object by shifting its points along the x-axis, y-axis, or both. It effectively "slides" layers in one direction, changing the object's appearance without rotating or scaling it.

Shearing is defined using **shearing factors**:

- **SHx** → Shearing along the x-axis (Horizontal Shearing)
- **SHy** → Shearing along the y-axis (Vertical Shearing)

A. Horizontal Shearing (along X-axis)

- **Effect**: Changes the x-coordinate; y remains constant.
- **Formula**:
    - $X1=X0+SHx \cdot Y0$
    - $Y1=Y0$
- **Matrix Representation** (Homogeneous Coordinates):

$$\begin{pmatrix} X_1 \\ Y_1 \end{pmatrix} = \begin{pmatrix} 1 & SH_x \\ 0 & 1 \end{pmatrix} x \begin{pmatrix} X_0 \\ Y_0 \end{pmatrix}$$

$$\begin{pmatrix} X_1 \\ Y_1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ SH_x & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} x \begin{pmatrix} X_0 \\ Y_0 \\ 1 \end{pmatrix}$$

B. Vertical Shearing (along Y-axis)

- **Effect**: Changes the y-coordinate; x remains constant.
- **Formula**:
    - X1=X0
    - Y1=Y0+SHy⋅X0
- **Matrix Representation** (Homogeneous Coordinates):

$$\begin{pmatrix} X_1 \\ Y_1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ SH_y & 1 \end{pmatrix} x \begin{pmatrix} X_0 \\ Y_0 \end{pmatrix}$$

$$\begin{pmatrix} X_1 \\ Y_1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & SH_y & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} x \begin{pmatrix} X_0 \\ Y_0 \\ 1 \end{pmatrix}$$

Example: Shearing a Triangle

Given

- Triangle with points: P(2, 2), Q(0, 0), R(2, 0)
- Shearing factors: SHx = 2, SHy = 2

1. Horizontal Shearing (SHx = 2)

→ **New Coordinates after Horizontal Shear**:
(6, 2), (0, 0), (2, 0)

2. Vertical Shearing (SHy = 2)

→ **New Coordinates after Vertical Shear**:
(2, 6), (0, 0), (2, 4)