

UNIT 1

Introduction to Computer Graphics

(6 Hours) — Computer Graphics and Animation (CACS305)

1.1 Advantages of Computer Graphics and Areas of Application

Computer Graphics is the branch of computer science that deals with generating, manipulating, and displaying pictures or images with the help of a computer. It involves the use of algorithms and mathematical techniques to create, store, process, and render visual content — ranging from simple line drawings to photo-realistic 3D scenes and animations.

Advantages of Computer Graphics

- **Better understanding:** Complex data and relationships (statistical, scientific, engineering) become easier to understand when presented visually instead of as raw numbers or text.
- **Interactivity:** Users can interact with images in real time — zoom, rotate, edit — enabling design, simulation, and gaming applications.
- **Cost and time saving:** Prototypes, models, and designs can be tested virtually before physical construction, reducing material waste and development time.
- **Accuracy and precision:** Computer-generated drawings (e.g., CAD models) are far more precise than hand-drawn sketches.
- **Storage and reproducibility:** Once created, an image can be stored, copied, transmitted, and reproduced without any loss of quality.
- **Ease of modification:** Graphic objects can be edited, scaled, or transformed easily, unlike traditional media.
- **Enhanced communication:** Visual presentation of ideas is more effective and persuasive than plain text in business, education, and media.

Areas of Application

Application Area	Typical Use
Computer-Aided Design (CAD)	Designing buildings, automobiles, aircraft, electronic circuits, and mechanical parts
Presentation Graphics	Bar charts, pie charts, line graphs used in reports, summaries, and business presentations

Application Area	Typical Use
Computer Art	Fine art, commercial art, logo design, painting and drawing packages, digital animation
Entertainment	Motion pictures, music videos, and video games using computer animation and special effects
Education and Training	Simulators for flight training, medical training, and interactive instructional material
Visualization	Scientific, engineering, and medical visualization — e.g., MRI scans, weather modelling, molecular structures
Image Processing	Enhancing and modifying existing images captured by cameras or scanners
Graphical User Interfaces (GUI)	Windows, icons, menus, and pointers used in modern operating systems and applications

Note: CAD (Computer-Aided Design) is one of the largest and most influential application areas — it allows designers to build, test, and modify virtual prototypes long before manufacturing begins.

1.2 Hardware and Software for Computer Graphics

A computer graphics system requires both specialized hardware (input devices, processing units, display and output devices) and software (graphics libraries, drivers, and application programs) working together to create and display images.

Hardware for Computer Graphics

- Input devices: keyboard, mouse, joystick, digitizer/tablet, scanner, light pen, touch screen — used to specify positions, select options, and enter commands.
- Processing unit: the CPU and, in modern systems, a dedicated Graphics Processing Unit (GPU) that performs the intensive calculations (transformations, shading, rendering) required for image generation.
- Display devices: CRT monitors, LCD/LED panels, plasma displays — used to present the generated image to the user (soft copy).
- Output / hard-copy devices: printers and plotters that produce a permanent, physical record of the graphic image.

Hard Copy Devices

Hard copy devices produce a permanent record of a graphics output on paper or film. They are broadly classified by the technology used to transfer the image onto the medium.

Device	Description
Dot-matrix / Impact printer	Forms characters and images using pins striking an ink ribbon against paper
Laser printer	Uses a laser beam to charge a photosensitive drum, which attracts toner and fuses it onto paper — produces high-quality output
Ink-jet printer	Sprays tiny droplets of ink directly onto paper; supports colour output at relatively low cost
Plotter	Uses pens (or ink-jet/laser technology) to draw continuous lines; commonly used for engineering drawings and large-format prints
Electrostatic plotter	Charges paper selectively and applies toner, then fuses it — used for high-resolution large drawings

Display Technologies

Display (soft-copy) devices show images on a screen using a variety of underlying technologies:

Technology	Working Principle
CRT (Cathode Ray Tube)	An electron gun fires a beam of electrons that strikes a phosphor-coated screen, causing it to glow at the point of impact
LCD (Liquid Crystal Display)	Liquid crystals twist to control light passing through polarising filters, forming an image pixel by pixel
LED Display	Light-emitting diodes are used directly as the light source, either as a backlight (LED-LCD) or self-emissive per pixel (OLED)
Plasma Display	Tiny cells containing ionized gas emit ultraviolet light that excites phosphors to produce visible colour
DVST (Direct View Storage Tube)	An older CRT variant that retains an image on the screen without needing to be refreshed

Note: CRT technology, though largely obsolete in modern consumer displays, remains foundational to understanding raster and random (vector) scan display architectures taught in this unit.

Software for Computer Graphics

- Graphics packages / libraries: OpenGL, DirectX, GD library — provide ready-made functions for drawing primitives, transformations, and rendering.
- General-purpose application software: CAD tools (AutoCAD), paint and drawing packages (Adobe Photoshop, CorelDRAW), 3D modelling and animation suites (Blender, Maya).

- Special-purpose software: written for a specific application, e.g. an architectural design tool or a medical imaging package, often built on top of a general-purpose graphics library.

1.3 Random Scan Display System, Video Controller, Random Scan Display Processor

Random Scan (Vector Scan) Display System

In a random scan (vector scan) system, the electron beam is directed only to the parts of the screen where a picture is to be drawn, moving from one specified endpoint to another in any order — much like a pen drawing lines on paper. The beam does not sweep the entire screen; it "jumps" directly between the endpoints of the lines that make up the picture.

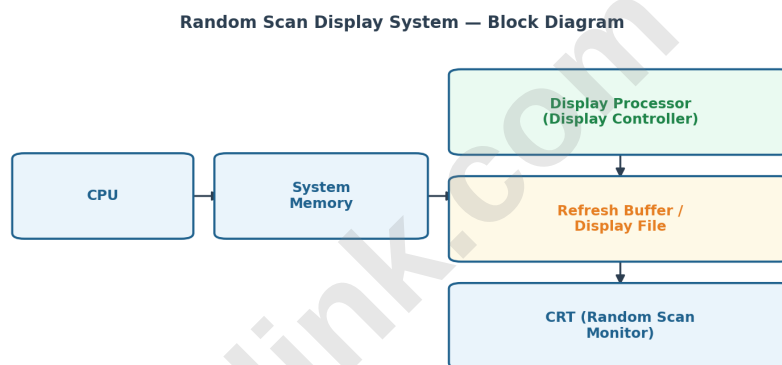


Fig 1.1 — Block diagram of a random scan display system: CPU and memory send display commands to the display processor, which repeatedly refreshes the CRT from the refresh buffer / display file.

Picture definitions in a random scan system are stored as a set of line-drawing commands in a region of memory called the refresh display file (also called the display list or display program). To display a picture, the system cycles through the set of commands, drawing each component line in turn. After all commands have been processed, the system loops back to the first line command and repeats the cycle — typically 30 to 60 times per second — to produce a flicker-free image.

Video Controller

The video controller is the hardware component responsible for controlling the operation of the display device. In a raster system, it accesses the frame buffer to retrieve intensity/colour values for each pixel and generates the corresponding video signals to drive the electron beam (or pixel drivers) in a fixed raster-scan pattern, refreshing the entire screen line-by-line at a fixed rate.

- Contains a fixed area of the frame buffer memory it scans out on every refresh cycle.

- Uses raster-scan generator circuitry to synchronise horizontal and vertical retrace signals with the pixel data being read.
- May include a look-up table (LUT) to translate stored pixel codes into actual colour/intensity values before sending them to the screen.

Random Scan (Vector) Display Processor

The random scan display processor (also called a display processing unit or graphics controller in vector systems) interprets the commands stored in the display file and converts them into the analog deflection voltages needed to steer the electron beam directly between the endpoints of each line or character to be drawn.

- Fetches each instruction from the display file in sequence.
- Converts endpoint coordinates into deflection voltages for the beam-positioning circuitry.
- Repeats the entire command sequence at a high refresh rate to keep the image visible and flicker-free.
- Supports commands for lines, characters, and short vector sequences, but cannot efficiently fill large solid areas — a key limitation compared to raster systems.

Random Scan vs Raster Scan — Quick Comparison

- ✓ Random scan draws only the outline of a picture; raster scan paints every pixel on the screen.
- ✓ Random scan uses a refresh display file; raster scan uses a frame buffer.
- ✓ Random scan gives smoother, higher-resolution lines; raster scan supports realistic, filled, shaded images.
- ✓ Random scan struggles with complex, solid-area scenes; raster scan handles them naturally.

1.4 Raster Graphics

In a raster scan system, the electron beam (or, in modern flat panels, the pixel-driving circuitry) sweeps across the screen one row at a time, from top to bottom, turning on or off — or varying the intensity of — each pixel along the way. This sweeping pattern is called a raster.

Raster Scan Pattern

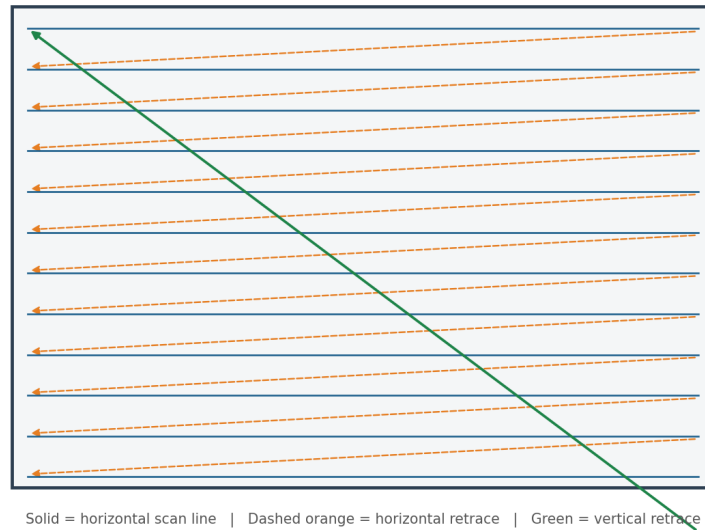


Fig 1.2 — Raster scan pattern: the beam sweeps left-to-right along each scan line (solid), performs a horizontal retrace to the start of the next line (dashed), and a vertical retrace back to the top after the last line (green).

As the beam moves across each scan line, its intensity is switched on and off (or modulated) according to the pixel values stored in the frame buffer (also called the refresh buffer) — a dedicated block of memory holding the colour/intensity value for every pixel on the screen. After completing a scan line, the beam undergoes a horizontal retrace to return to the left edge of the screen to begin the next line; after the bottom line, it performs a vertical retrace back to the top-left corner to begin the next frame.

Key Terms

Term	Meaning
Pixel (picture element)	The smallest addressable point on the screen; stores one colour/intensity value
Resolution	The number of pixels that can be displayed, typically expressed as width × height (e.g. 1920 × 1080)
Frame buffer	Memory area that stores the intensity value for every pixel on the screen for one complete frame
Refresh rate	The number of times per second the entire screen image is redrawn (e.g. 60 Hz)
Bit planes / bit depth	The number of bits used per pixel; determines the number of colours or grey levels available
Aspect ratio	The ratio of vertical points to horizontal points needed to produce equal-length lines in both directions

Because every pixel on the screen is refreshed on every pass — regardless of whether it is part of a picture or blank background — raster systems can easily display filled, shaded, and photo-realistic areas, which is why virtually all modern displays (LCD, LED, plasma) use raster-scan technology, even though many no longer use an electron beam.

Key Takeaways — Raster Graphics

- ✓ A raster is the pattern formed by horizontal scan lines that sweep the full screen.
- ✓ The frame buffer stores one value per pixel and is read out on every refresh cycle.
- ✓ Raster systems naturally support solid-area filling, shading, and photo-realistic images.
- ✓ Screen resolution and bit depth (bits per pixel) together determine image quality and colour range.

1.5 Scan Conversion Algorithms (Line, Circle, Ellipse)

Scan conversion (also called rasterization) is the process of converting a geometric primitive — defined mathematically by continuous coordinates — into a discrete set of pixels on a raster display that best approximates its shape. Because a pixel grid has finite resolution, scan conversion algorithms must select which pixels to illuminate quickly and accurately, ideally using only integer arithmetic for speed.

Line Scan Conversion

A straight line is defined by two endpoints, (x_0, y_0) and (x_1, y_1) . The goal of a line-drawing algorithm is to determine which intermediate pixels most closely approximate the path of the ideal mathematical line between them.

1. DDA (Digital Differential Analyzer) Algorithm

The DDA algorithm is an incremental scan-conversion method that calculates pixel positions along a line by evaluating either Δy or Δx at each unit step along the axis with the greater difference. For a line with slope $m = \Delta y / \Delta x$, if $|\Delta x| \geq |\Delta y|$ the algorithm steps x by 1 and increments y by m at each step (and vice versa when $|\Delta y| > |\Delta x|$), rounding each computed coordinate to the nearest pixel.

Example 1.1 — DDA Algorithm Steps

1. Compute $\Delta x = x_1 - x_0$ and $\Delta y = y_1 - y_0$.
2. Let $\text{steps} = \max(|\Delta x|, |\Delta y|)$.
3. Compute increments: $x_{\text{inc}} = \Delta x / \text{steps}$, $y_{\text{inc}} = \Delta y / \text{steps}$.
4. Set $(x, y) = (x_0, y_0)$; plot $\text{round}(x)$, $\text{round}(y)$.
5. Repeat " $x += x_{\text{inc}}$; $y += y_{\text{inc}}$; plot $\text{round}(x)$, $\text{round}(y)$ " for steps iterations.

Note: DDA is simple to implement but relies on floating-point addition and rounding at every step, making it slower and less accurate over long lines than integer-based methods such as Bresenham's algorithm.

2. Bresenham's Line Algorithm

Bresenham's algorithm is an efficient, integer-only method for line rasterization. Assuming the line has a slope $0 < m < 1$, it starts at (x_0, y_0) and steps x by 1 at each iteration, using a decision parameter p to decide whether y should also increment, based on which of the two candidate pixels lies closer to the true line.

Example 1.2 — Bresenham's Line Algorithm ($0 < m < 1$)

1. Compute $\Delta x = x_1 - x_0$, $\Delta y = y_1 - y_0$.
2. Initial decision parameter: $p_0 = 2\Delta y - \Delta x$.
3. At each step k , if $p_k < 0$: plot (x_k+1, y_k) ; $p_{k+1} = p_k + 2\Delta y$.
4. Otherwise: plot (x_k+1, y_k+1) ; $p_{k+1} = p_k + 2\Delta y - 2\Delta x$.
5. Repeat for $k = 0, 1, \dots, \Delta x - 1$.

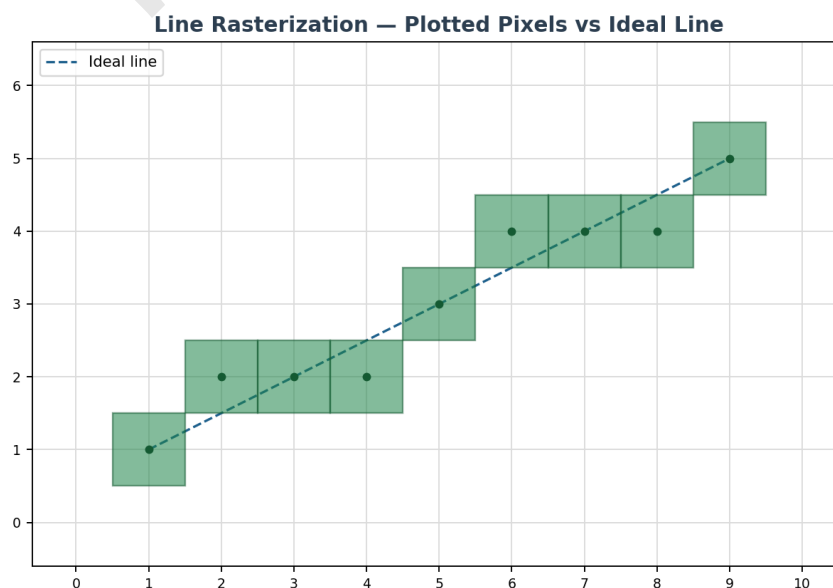


Fig 1.3 — Rasterized pixels (green squares) selected by an integer line algorithm compared against the ideal mathematical line (dashed blue).

Note: Bresenham's algorithm uses only integer addition, subtraction, and multiplication by 2 (a bit-shift), making it significantly faster than DDA and the standard choice in most graphics hardware and libraries.

3. Midpoint Circle Algorithm

A circle is symmetric about its centre in eight octants, so it is enough to compute pixel positions for one octant (typically from angle 0° to 45°) and obtain the remaining seven octants by reflecting the coordinates.

Midpoint Circle Algorithm — 8-Way Symmetry

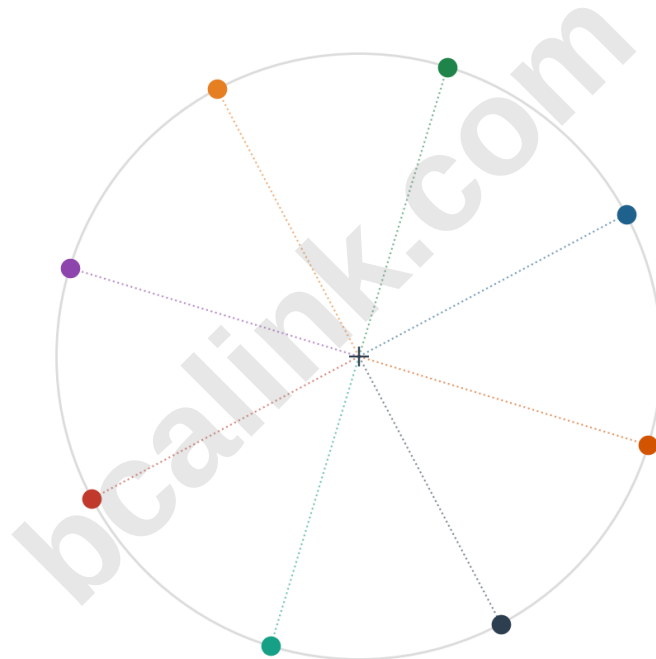


Fig 1.4 — Eight-way symmetry of a circle: a single computed point (x, y) in one octant yields seven corresponding points in the other octants.

The midpoint circle algorithm starts at $(0, r)$ and, at each step, tests whether the midpoint between the two candidate next pixels lies inside or outside the true circle boundary, using a decision parameter:

Example 1.3 — Midpoint Circle Algorithm

1. Initial decision parameter: $p_0 = 1 - r$.
2. Start at $(x, y) = (0, r)$.
3. At each step, if $p_k < 0$: choose (x_{k+1}, y_k) ; $p_{k+1} = p_k + 2x_{k+1} + 1$.
4. Otherwise: choose (x_{k+1}, y_{k+1}) ; $p_{k+1} = p_k + 2x_{k+1} + 1 - 2y_{k+1}$.
5. Continue while $x < y$; plot all eight symmetric points at each step.

4. Midpoint Ellipse Algorithm

An ellipse with semi-major axis a and semi-minor axis b has only four-way symmetry (not eight-way, since $a \neq b$ in general). The midpoint ellipse algorithm therefore divides the curve into two regions at the point where the slope equals -1 , applying a separate decision parameter in each region.

Example 1.4 — Midpoint Ellipse Algorithm (Outline)

1. Region 1 (slope magnitude < 1 , near the major axis): start at $(0, b)$; step x by 1 each iteration; use decision parameter $p_1 = b^2 - a^2b + a^2/4$ to decide whether y also decreases.
2. Region 2 (slope magnitude > 1 , near the minor axis): switch to stepping y by 1 each iteration; use a second decision parameter p_2 , computed from the last point of Region 1.
3. Plot all four symmetric points — $(\pm x, \pm y)$ — at every computed position.

Note: Splitting the ellipse into two slope regions ensures that, in both regions, consecutive pixels never skip more than one row or column — preserving a smooth, continuous curve.

1.6 Area Filling and Clipping

Area Filling

Area filling is the process of colouring or shading all pixels that lie within the boundary of a closed figure, such as a rectangle, ellipse, or polygon.

Filling a Rectangle

Because a rectangle's boundary is aligned to two pairs of horizontal and vertical edges, filling it is straightforward: for every scan line y between the top and bottom edges, set every pixel between the left edge x -coordinate and the right edge x -coordinate to the fill colour.

Example 1.5 — Rectangle Fill (Scan-Line Method)

For a rectangle with corners (x_0, y_0) and (x_1, y_1) :

for $y = y_0$ to y_1 :

for $x = x_0$ to x_1 : setPixel(x, y , fillColor)

Filling an Ellipse

An ellipse can be filled using the same symmetry exploited by the midpoint ellipse algorithm: instead of merely plotting the boundary point at each computed (x, y) , a horizontal span is drawn from $-x$ to $+x$ at each value of y (and its mirror at $-y$), filling the interior scan line by scan line as the boundary is traced.

- At each step of the midpoint ellipse algorithm, draw a horizontal line from $(-x, y)$ to (x, y) instead of plotting only the two boundary points.
- Repeat symmetrically for the $(-x, -y)$ to $(x, -y)$ span.
- This combines boundary computation and area filling into a single efficient pass.

Clipping

Clipping is the process of identifying and retaining only the portion of a picture that lies within a specified region — the clip window — and discarding everything outside it. Clipping is essential because a scene is often modelled at a larger scale than what fits on the visible screen or viewport.

Line Clipping — Cohen–Sutherland Algorithm

The Cohen–Sutherland algorithm assigns each line endpoint a 4-bit region code based on its position relative to the clip window's four boundary lines (top, bottom, right, left). Each bit is set to 1 if the point lies beyond the corresponding boundary.

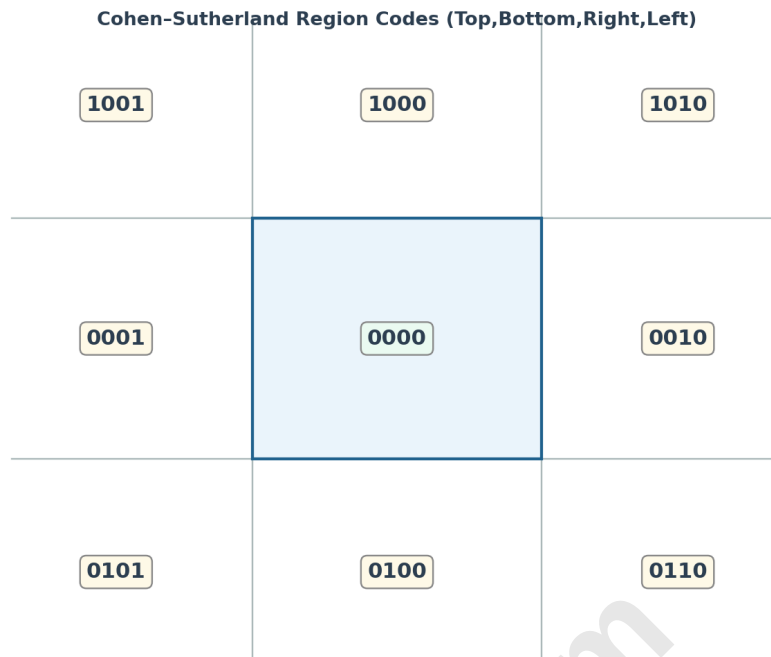


Fig 1.5 — Region codes (Top Bottom Right Left) assigned to the nine zones formed by extending the clip window's edges; the code 0000 marks points inside the window.

Example 1.6 — Cohen-Sutherland Clipping Steps

1. Compute the 4-bit region code for both endpoints of the line.
2. Trivial accept: if both codes are 0000, the entire line is inside — keep it as is.
3. Trivial reject: if the bitwise AND of both codes is non-zero, both points lie outside the same boundary — discard the line entirely.
4. Otherwise, pick an endpoint with a non-zero code, find its intersection with the corresponding clip boundary, and replace that endpoint with the intersection point.
5. Recompute the region code for the new point and repeat from step 2 until the line is trivially accepted or rejected.

Note: The Cohen-Sutherland algorithm is efficient because most lines in a typical scene are either entirely inside or entirely outside the window, allowing them to be resolved instantly by the trivial-accept or trivial-reject tests without computing any intersection.

Clipping a Circle

Circle clipping compares the distance of each boundary point from the circle's centre against the clip window's edges, or, more generally, tests whether each point on the circle (computed via the midpoint circle algorithm) falls inside the rectangular or polygonal clip region — retaining only the arcs that lie within the window and computing new boundary intersection points where the circle crosses a clip edge.

Clipping an Ellipse

Ellipse clipping follows the same principle: each point generated by the midpoint ellipse algorithm is tested against the clip window's boundaries, and only the portions of the ellipse's boundary (and any filled spans) lying inside the window are retained, with new edge points computed at each crossing.

Polygon Clipping — Sutherland–Hodgman Algorithm

Clipping a polygon against a rectangular window cannot simply reuse the line-clipping algorithm on each edge in isolation, because doing so can produce disconnected line segments instead of a single closed polygon. The Sutherland–Hodgman algorithm solves this by clipping the polygon successively against each of the four window boundaries in turn, each time producing a new, still-closed polygon that becomes the input to the next boundary's clip.

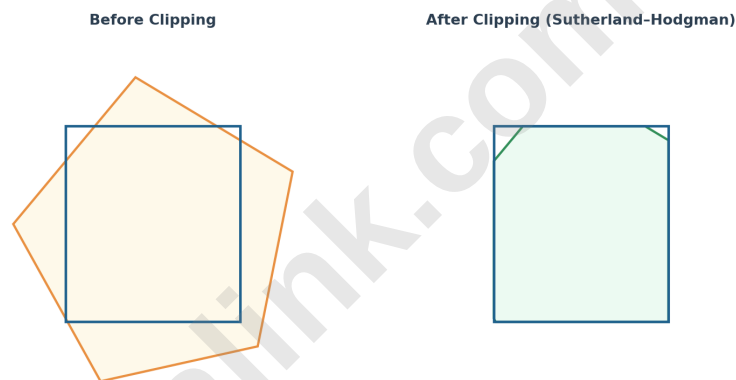


Fig 1.6 — A polygon before clipping (left) and the resulting closed polygon after clipping against a rectangular window using the Sutherland–Hodgman method (right).

Example 1.7 — Sutherland–Hodgman: Clipping Against One Boundary

For each edge (previous vertex → current vertex) of the input polygon, compared against a single clip boundary:

- If both vertices are inside → output the current vertex.
- If the previous vertex is inside and current is outside → output only the intersection point.
- If the previous vertex is outside and current is inside → output the intersection point, then the current vertex.
- If both vertices are outside → output nothing.

Note: The four-boundary process (left, right, bottom, top) is applied sequentially, with the output polygon of one stage becoming the input to the next — guaranteeing a single, correctly closed clipped polygon at the end, even for concave shapes (though self-intersecting artifacts can occur for highly concave polygons, which more advanced algorithms such as Weiler–Atherton are designed to handle).

Key Takeaways — Unit 1

- ✓ Computer graphics improves understanding, accuracy, and communication across CAD, entertainment, education, and visualization.
- ✓ Graphics hardware spans input devices, processing units (CPU/GPU), display devices, and hard-copy output devices.
- ✓ Random scan systems trace only picture outlines using a display file; raster scan systems paint every pixel using a frame buffer.
- ✓ DDA and Bresenham's algorithm rasterize straight lines; the midpoint algorithm extends the same idea to circles and ellipses.
- ✓ Area filling extends boundary algorithms to colour the interior of shapes scan-line by scan-line.
- ✓ Cohen–Sutherland clips lines using region codes; Sutherland–Hodgman clips polygons boundary-by-boundary to preserve a closed shape.

— End of Unit 1 —