Computer Graphics

Short notes for B.Tech CSE / IT · BCA · MCA

Uncodemy Handwritten Notes · free for students

I. Introduction

- **Computer Graphics** = creating, storing and changing pictures using a computer.
- Applications: CAD, games, animation, simulation, medical imaging, GUIs, data visualisation.
- **Pixel** = smallest addressable point on the screen. **Resolution** = pixels across × down (e.g. 1920 × 1080).
- **Aspect ratio** = width : height (e.g. 16 : 9).

➤ Raster vs random (vector) scan

Point	Raster scan	Random scan
Drawing	Beam sweeps row by row	Beam draws lines directly
Picture stored as	Pixels in frame buffer	Line commands in display file
Realistic images	Yes (shading, colour)	No, line drawings only
Lines	Jagged (aliasing)	Smooth
Example	TVs, monitors	Old plotters, vector displays

➤ Frame buffer

FORMULA Frame buffer size (bits) = horizontal pixels × vertical pixels × bits per pixel.

EXAMPLE 1024 × 768 with 24 bits per pixel = 786432 × 24 = 18874368 bits = 2359296 bytes = 2.25 MB.

FORMULA Number of colours = 2^n for n bits per pixel.

➤ Display devices

- **CRT**: electron gun, deflection system, phosphor screen. Refresh needed because phosphor fades.

- **LCD**: liquid crystals twist light; needs a backlight. **LED / OLED**: OLED pixels emit their own light.
- **Interlaced** scan: odd lines first, then even lines.

2. Line Drawing Algorithms

➤ DDA (Digital Differential Analyser)

1. dx = x2 − x1, dy = y2 − y1.
2. steps = max(abs(dx), abs(dy)).
3. x_inc = dx / steps, y_inc = dy / steps.
4. Start at (x1, y1); add increments each step and round.

EXAMPLE Line (2, 2) to (6, 4). dx = 4, dy = 2, steps = 4, x_inc = 1, y_inc = 0.5. Points: (2,2), (3,2.5→3), (4,3), (5,3.5→4), (6,4).

- Uses floating point and rounding, so slower than Bresenham.

➤ Bresenham's line algorithm (0 < m < 1)

- Uses only integer addition and subtraction.

FORMULA p0 = 2dy − dx. If p < 0: next point (x+1, y), p = p + 2dy. Else: next point (x+1, y+1), p = p + 2dy − 2dx.

EXAMPLE (20, 10) to (30, 18). dx = 10, dy = 8. p0 = 16 − 10 = 6. 2dy = 16, 2dy − 2dx = −4. p0 = 6 ≥ 0 → (21, 11), p = 2. p ≥ 0 → (22, 12), p = −2. p < 0 → (23, 12), p = 14. p ≥ 0 → (24, 13), p = 10.

EXAM POINT Bresenham is preferred over DDA because it avoids floating point and rounding.

3. Circle and Ellipse Drawing

- Circles have 8-way symmetry: compute one octant, mirror to the other seven: (±x, ±y) and (±y, ±x).

➤ Midpoint circle algorithm

FORMULA $p0 = 1 - r$. If $p < 0$: next $(x+1, y)$, $p = p + 2x + 3$. Else: next $(x+1, y-1)$, $p = p + 2(x - y) + 5$. Here x, y are values before the step. Stop when $x \geq y$.

EXAMPLE $r = 10$, start $(0, 10)$, $p0 = -9$. $p < 0 \rightarrow (1, 10)$, $p = -9 + 3 = -6$. $p < 0 \rightarrow (2, 10)$, $p = -6 + 5 = -1$. $p < 0 \rightarrow (3, 10)$, $p = -1 + 7 = 6$. $p \geq 0 \rightarrow (4, 9)$, $p = 6 + 2(3 - 10) + 5 = -3$.

- **Bresenham's circle**: $d0 = 3 - 2r$, similar integer steps.
- **Midpoint ellipse**: divides the first quadrant into two regions (slope magnitude < 1 and > 1); ellipse has 4-way symmetry.

4. Filling and Anti-aliasing

- **Scan-line polygon fill**: for each scan line, find intersections with edges, sort them, fill between pairs.
- **Boundary fill**: start at a seed point, fill until the boundary colour is reached.
- **Flood fill**: replace all connected pixels of the old interior colour.
- **4-connected** (up, down, left, right) vs **8-connected** (also diagonals).
- **Inside-outside test**: even-odd rule, non-zero winding rule.
- **Aliasing** = jagged edges (staircase effect). **Anti-aliasing**: supersampling, area sampling, pixel phasing.

5. 2D Transformations

- Use **homogeneous coordinates** $(x, y, 1)$ so all transformations become 3×3 matrix multiplications.

Transform	Matrix rows
Translation (tx, ty)	[1 0 tx], [0 1 ty], [0 0 1]
Scaling (sx, sy)	[sx 0 0], [0 sy 0], [0 0 1]
Rotation θ (anticlockwise)	[cosθ −sinθ 0], [sinθ cosθ 0], [0 0 1]
Reflection about x-axis	[1 0 0], [0 −1 0], [0 0 1]
Shear in x (shx)	[1 shx 0], [0 1 0], [0 0 1]

FORMULA Rotation about origin: $x' = x\cos\theta - y\sin\theta$, $y' = x\sin\theta + y\cos\theta$.

EXAMPLE Rotate (4, 0) by 90° anticlockwise: $x' = 4(0) - 0(1) = 0$, $y' = 4(1) + 0(0) = 4$. New point (0, 4).

➤ Composite transformation

- Rotate about a fixed point (xf, yf): translate to origin → rotate → translate back.
- Matrix: $T(xf, yf) \cdot R(\theta) \cdot T(-xf, -yf)$. Applied right to left.
- Matrix multiplication is not commutative, so the order matters.

EXAMPLE Scale (2, 3) by sx = 2, sy = 2 about fixed point (1, 1): translate → (1, 2), scale → (2, 4), translate back → (3, 5).

6. Windowing and Clipping

- **Window** = area of the world we want to see. **Viewport** = area on the screen where it is shown.

FORMULA Window to viewport: xv = xvmin + (xw − xwmin) × sx, where sx = (xvmax − xvmin) / (xwmax − xwmin). Same for y.

➤ Cohen–Sutherland line clipping

- Each endpoint gets a 4-bit **region code** (TBRL: top, bottom, right, left).

```
 1001 | 1000 | 1010
------+------+------
 0001 | 0000 | 0010
------+------+------
 0101 | 0100 | 0110
```

- Both codes 0000 → fully inside, accept.
- Logical AND of codes ≠ 0000 → fully outside, reject.
- Otherwise find the intersection with a window edge and repeat.

➤ Other clipping

- **Liang–Barsky**: parametric line clipping, fewer intersection calculations than Cohen–Sutherland.
- **Midpoint subdivision**: keep splitting the line at its midpoint.
- **Sutherland–Hodgman**: polygon clipping, clips against one window edge at a time.

- **Weiler–Atherton**: polygon clipping that also handles concave polygons correctly.

7. 3D Concepts and Projections

- 3D transformations use 4×4 matrices with (x, y, z, 1).
- 3D rotation about the z-axis: $x' = x\cos\theta - y\sin\theta$, $y' = x\sin\theta + y\cos\theta$, $z' = z$. Similar for x and y axes.

➤ Projections

Type	Projectors	Size	Use
Parallel	Parallel to each other	True size kept	Engineering drawings
Perspective	Meet at centre of projection	Far objects look smaller	Realistic views

- Parallel: **orthographic** (projectors ⊥ to plane: top, front, side; also isometric) and **oblique** (cavalier, cabinet).
- Perspective: one-point, two-point, three-point (number of **vanishing points**).

FORMULA Simple perspective (viewer at origin, plane at $z = d$): $x' = x \cdot d / z$, $y' = y \cdot d / z$.

➤ 3D representation

- Polygon meshes, curved surfaces, sweep representations, **CSG** (constructive solid geometry), octrees.

8. Curves, Hidden Surfaces and Shading

➤ Curves

- **Bezier curve**: shaped by control points; passes through first and last point only. Degree = number of control points − 1.

FORMULA Cubic Bezier: $P(t) = (1-t)^3P0 + 3t(1-t)^2P1 + 3t^2(1-t)P2 + t^3P3$, for $0 \le t \le 1$.

- **B-spline**: local control (moving one point changes only part of the curve); degree independent of number of points.
- Continuity: C0 (joined), C1 (same slope), C2 (same curvature).

Hidden surface removal

Method	Idea
Back-face removal	Drop faces with normal pointing away from the viewer
Z-buffer (depth buffer)	Keep the nearest z value per pixel; image space
Painter's algorithm	Sort by depth, draw farthest first
Scan-line	Depth check along each scan line
Area subdivision (Warnock)	Split screen areas until simple

EXAM POINT Z-buffer is simple and works for any scene, but needs extra memory for one depth value per pixel.

Illumination and shading

- Light types: **ambient**, **diffuse** (Lambert: $I = k_d \cdot I_l \cdot \cos\theta$), **specular** (Phong: shiny highlight).
- **Flat shading**: one colour per polygon. **Gouraud**: interpolate intensities at vertices. **Phong**: interpolate normals, better highlights.
- Colour models: **RGB** (additive, screens), **CMY / CMYK** (subtractive, printers), **HSV** (hue, saturation, value). $C = 1 - R$, $M = 1 - G$, $Y = 1 - B$.

9. Animation Basics

- **Keyframe** animation: artist draws key poses; **in-betweening** (tweening) fills frames between them.
- **Morphing**: one shape changes smoothly into another.
- Frame rate: frames per second (fps); about 24 fps or more looks smooth.

10. Most Asked Questions (practice)

1. Raster scan vs random scan. Find frame buffer size for a given resolution.
2. Explain DDA and plot a line with it.
3. Bresenham's line algorithm with a worked example.
4. Midpoint circle algorithm with a worked example.
5. Boundary fill vs flood fill. 4-connected vs 8-connected.
6. Rotate a triangle about a fixed point using composite matrices.
7. Explain Cohen–Sutherland line clipping with region codes.

8. Parallel vs perspective projection.
9. Explain Bezier curves and their properties.
10. Explain Z-buffer and Gouraud vs Phong shading.

TIP Every paper has numericals on DDA, Bresenham, midpoint circle, 2D transformation and clipping. Practise each with a full table of points.