Scan Conversion Techniques

1. Introduction

Scan conversion is the process of converting the geometric primitives (lines, circles, curves, polygons) specified in a display (world or device independent) form into a set of discrete picture elements called pixels that can be displayed on a raster display device.

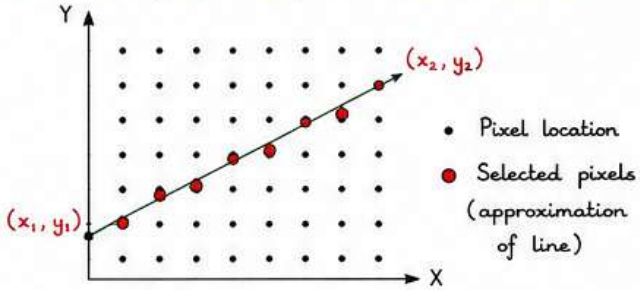
In other words, scan conversion determines which pixels of the displaydevice should be turned ON in order to approximate the desired object.

2. Need of Scan Conversion

- Computer stores object in mathematical form.
- Display devices are raster based (pixel based.
- Scan conversion converts mathematical description into pixel intensity values.
- It is necessary for displaying any object on screen.

3. Basic Idea

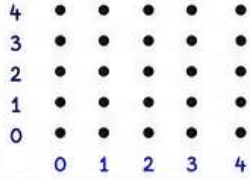
Consider a line between (x_1, y_1) and (x_2, y_2). This line is continuous in nature. But screen is made of discrete pixels. Scan conversion selects the nearest pixels to represent the line on screen.



4. Raster Display

- Screen is divided into small picture elements called pixels.
- Each pixel has a position (x, y) and intensity (color).
- Pixel intensity represents brightness or color.

Example of Pixel Grid (5 × 5)



Pixel Addressing

Top-left corner is (0,0).
X increases → (left to right)
Y increases ↓ (top to bottom)

5. Scan Conversion Process

1. **Geometric Description**
   Objects are defined using mathematical equations (lines, circles, curves, etc.).
2. **Geometric Transformation**
   Object coordinates are transformed from world coordinates to device coordinates.
3. **Clipping**
   Only the visible portion of objects is considered.
4. **Scan Conversion**
   Converted into a set of pixel positions and intensity values.
5. **Display**
   Pixels are sent to display system (frame buffer) for visualization.

6. Types of Scan Conversion Techniques

(a) **Analytical (Mathematical) Techniques**
- Based on mathematical calculations.
- Example : DDA, Bresenham's Algorithm.

(b) **Geometric (Sampling) Techniques**
- Based on geometric considerations.
- Example : Uniform sampling, Area sampling.

7. Common Primitives for Scan Conversion

Primitive	Purpose
Line	Used to draw edges of objects.
Circle	Used to draw circular shapes.
Curve	Used to draw smooth shapes.
Polygon	Used to fill areas and shapes.

Important Points

- Scan conversion converts mathematical objects into pixels.
- It is essential for raster display systems.
- Accuracy and speed are main criteria.
- Other topics (DDA, Bresenham, etc.) are based on scan conversion.

Typical Exam Questions

1. Define scan conversion.
2. Explain the need for scan conversion.
3. Describe the scan conversion process.
4. Explain analytical and geometric techniques.
5. What is raster display ? Explain.

Key Terms

- Scan Conversion
- Pixel
- Raster Display
- Frame Buffer
- Primitive
- Intensity
- Analytical Technique
- Geometric Technique
- Clipping
- Transformation

Image Representation

1. Introduction

An image is a two dimensional function $f(x,y)$ where x and y are spatial (plane) coordinates and the value of f at any point (x,y) is called the intensity or gray level of the image at that point.

To handle images in a computer, they must be represented in a digital form.

2. Need of Image Representation

* To store and process images in computer.
* Required for image processing, analysis and transmission.
* Helps in display of images on raster display devices.
* Basis for computer graphics, CAD, medical imaging, etc.

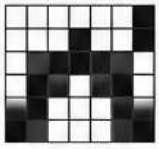
3. Types of Images

(a) Binary (Monochrome) Image

Only two possible intensity values : 0 (black) and 1 (white).

Each pixel requires 1 bit.

Example :



0 → Black
1 → White

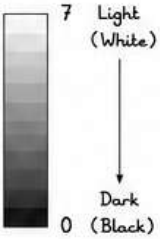
(b) Grayscale Image

Pixels have intensity values in the range 0 to (L-1), where L is the number of gray levels.

Each pixel requires $\log_2 L$ bits.

Example : L = 8 (3 bits per pixel)

0	1	2	3	4	5	6	7
1	2	3	4	5	6	7	6
2	3	4	5	6	7	6	5
3	4	5	6	7	6	5	4
4	5	6	7	6	5	4	3
5	6	7	6	5	4	3	2
6	7	6	5	4	3	2	1
7	6	5	4	3	2	1	0

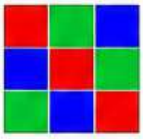


(c) Color Image

Each pixel is a combination of Red, Green and Blue components.

Common color models : RGB, CMY, HSV, HSI, etc.

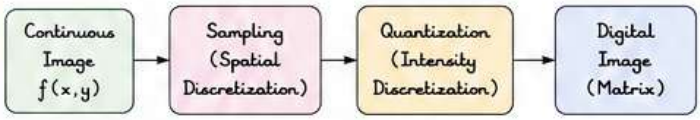
Example : (24-bit True Color Image)



RGB Model
Each pixel = (R, G, B)
Each component = 8 bits
Total = 24 bits per pixel

4. Digital Image Representation

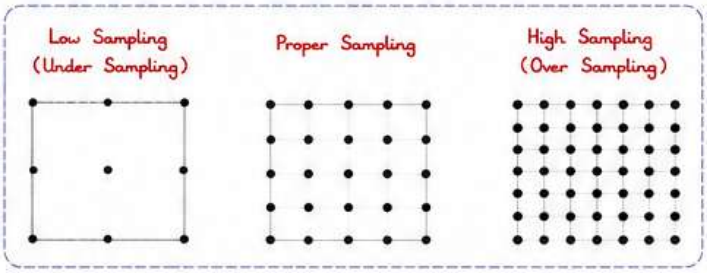
* Continuous image is sampled and quantized to convert into digital form.
* Two processes are involved : Sampling and Quantization.



5. Image Sampling

The process of converting a continuous image into a set of discrete points is called sampling.

Sampling density determines the level of detail.



6. Quantization

The process of assigning a finite set of intensity values to the sampled points is called quantization.

More the number of gray levels, better the quality but more storage is required.

Example : 3-bit Quantization (8 gray levels)

Decimal Value	Binary (3 bits)	Gray Level
0	000	Black
1	001	Very Dark Gray
2	010	Dark Gray
3	011	Gray
4	100	Light Gray
5	101	Very Light Gray
6	110	Almost White
7	111	White

7. Image Representation Parameters

1. Resolution (Spatial) : Number of pixels in rows (M) and columns (N).
2. Intensity Resolution : Number of intensity levels (L).
3. Pixel (Picture Element) : Smallest element in an image.
4. Aspect Ratio : Width / Height.
5. Storage Requirement = $M \times N \times \log_2 L$ bits.

Important Points

* Image = 2D function $f(x,y)$.
* Binary image uses 1 bit per pixel.
* Grayscale image uses $\log_2 L$ bits per pixel.
* Color image (True Color) uses 24 bits per pixel.
* Sampling converts continuous image to discrete points.
* Quantization converts infinite intensity to finite levels.

Typical Exam Questions

1. Define image and image representation.
2. Explain binary, grayscale and color images.
3. What are sampling and quantization? Explain.
4. Write the storage requirement formula for an image.
5. Explain RGB color model with example.

Key Terms

* Image
* Pixel
* Resolution
* Sampling
* Quantization
* Gray Level
* Binary Image
* Grayscale Image
* Color Image
* RGB Model

Line Drawing

1. Introduction

Line is the simplest and most fundamental object in computer graphics. A line is defined by two end points (x_1, y_1) and (x_2, y_2).

Line drawing means determining the set of pixel positions that best represent a straight line on the display screen.

2. Need of Line Drawing

* Used to draw edges of objects.
* Forms the basis for drawing shapes, polygons, characters, graphs, etc.
* Important in CAD, animations, games and scientific visualizations.

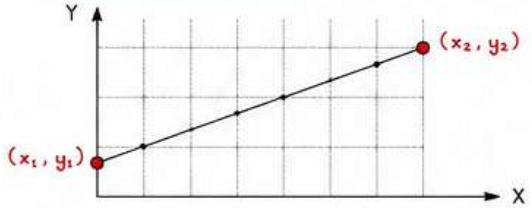
3. Basic Concept

A line can be represented in two ways :

(a) Mathematical form (b) Incremental form

Equation of line passing through (x_1, y_1) and (x_2, y_2):

$$y - y_1 = m\,(x - x_1) \qquad \text{where,} \quad m = \frac{y_2 - y_1}{x_2 - x_1}$$

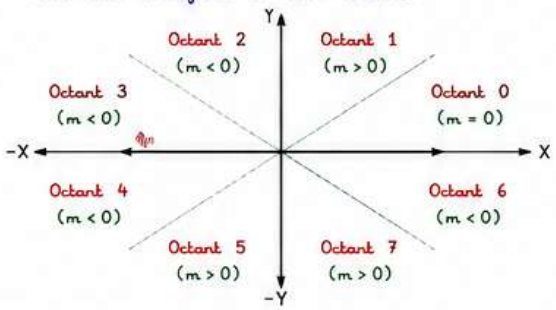


4. Slope and Octants

* Slope (m) of a line determines its direction.

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

* The coordinate plane is divided into 8 octants.
* By using symmetry, we can draw lines in one octant and then transform to other octants.



5. Methods of Line Drawing

Different algorithms are used to determine the pixels closest to the ideal line.

① Simple DDA Algorithm

② Bresenham's Line Drawing Algorithm

6. Cases for Line Drawing

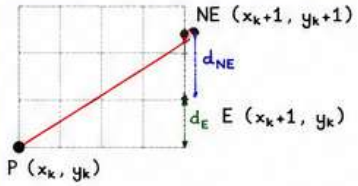
Case	Condition	Increment	Major Step	Minor Step
Case 1	$0 \le m \le 1$	$x = x + 1$	Along x	y may change
Case 2	$m > 1$	$y = y + 1$	Along y	x may change
Case 3	$-1 \le m \le 0$	$x = x + 1$	Along x	y may change
Case 4	$m < -1$	$y = y + 1$	Along y	x may change

We generally draw lines in first octant $(0 \le m \le 1)$ and then apply symmetry.

7. Pixel Selection Principle

The ideal line may not pass through the center of pixels. So, we choose the pixel whose center is nearest to the actual line.

The distance can be measured vertically (for $0 \le m \le 1$) or horizontally (for $m > 1$).



Choose E if $d_E < d_{NE}$

Otherwise choose NE

8. Example

Draw a line between (2, 2) and (8, 5) using basic concept.

Step 1 : Find slope

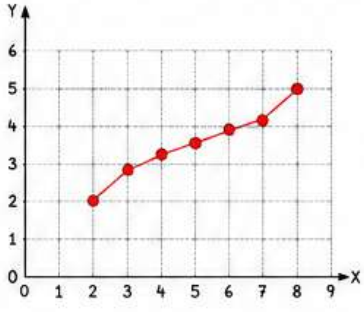
$$m = \frac{5-2}{8-2} = \frac{3}{6} = 0.5 \quad (0 \le m \le 1)$$

So, x will be the major step.

Step 2 : Calculate y for each x

$$y = 2 + 0.5\,(x - 2)$$

x	y (Calculated)	y (Rounded)	Pixel (x, y)
2	2.0	2	(2, 2)
3	2.5	3	(3, 3)
4	3.0	3	(4, 3)
5	3.5	4	(5, 4)
6	4.0	4	(6, 4)
7	4.5	5	(7, 5)
8	5.0	5	(8, 5)



9. Summary

- ✓ Line drawing is the process of determining the pixels that best represent a straight line.
- ✓ Based on the slope, we decide the direction of major and minor steps.
- ✓ Algorithms like DDA and Bresenham's are used for efficient line drawing.
- ✓ Symmetry can be used to draw lines in all octants.

Important Points

* A line is defined by two end points.
* When $|m| \le 1$, we step in x; when $|m| > 1$, we step in y.
* DDA uses floating point calculations.
* Bresenham's algorithm uses only integer calculations (faster and more accurate).

Typical Exam Questions

1. Define line. Explain need of line drawing.
2. Explain different cases of line drawing.
3. Describe DDA algorithm for line drawing.
4. Explain Bresenham's line drawing algorithm.
5. Draw line between two points using DDA or Bresenham's algorithm.

Key Terms

* Line
* Slope
* Octant
* Major Step
* Minor Step
* DDA Algorithm
* Bresenham's Algorithm
* Pixel
* Nearest Pixel
* Symmetry

Simple DDA Algorithm

1. Introduction

DDA stands for Digital Differential Analyzer.

It is a line drawing algorithm used to determine the set of pixels that best represents a straight line between two given end points on the display screen.

2. Basic Idea

* The line is divided into equal steps.
* The values of x and y are calculated at each step using incremental values.
* The calculated (x, y) values are then rounded to the nearest integer to get the pixel positions.
* This method uses floating point arithmetic.

3. Mathematical Formulation

Let the line be drawn between two end points (x_1, y_1) and (x_2, y_2).

$\Delta x = x_2 - x_1$, $\Delta y = y_2 - y_1$

Steps are taken along the major axis.

$$m = \text{slope} = \frac{\Delta y}{\Delta x}$$

(a) If $|m| \le 1$ (i.e., $|\Delta y| \le |\Delta x|$) → x is major axis

$$x_{k+1} = x_k + 1$$
$$y_{k+1} = y_k + m$$

(b) If $|m| > 1$ (i.e., $|\Delta y| > |\Delta x|$) → y is major axis

$$y_{k+1} = y_k + 1$$
$$x_{k+1} = x_k + \frac{1}{m}$$

At each step, the calculated values are rounded to nearest integer to obtain the pixel position.

4. Algorithm ($|m| \le 1$)

Input : Two end points (x_1, y_1) and (x_2, y_2)
Output : Set of pixels representing the line

1. $\Delta x = x_2 - x_1$
2. $\Delta y = y_2 - y_1$
3. $m = \frac{\Delta y}{\Delta x}$
4. $x = x_1$, $y = y_1$
5. Plot the first point (round(x), round(y))
6. For i = 1 to Δx
   - $x = x + 1$
   - $y = y + m$
   - Plot (round(x), round(y))
7. Stop

Note : For $|m| > 1$, interchange the role of x and y. Use y as major axis.

5. Example

Draw a line between $P_1(2, 2)$ and $P_2(10, 6)$ using Simple DDA Algorithm.

Solution :

$\Delta x = 10 - 2 = 8$

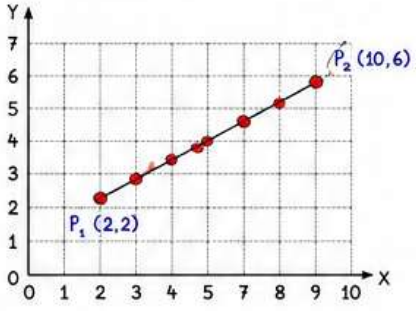
$\Delta y = 6 - 2 = 4$

$m = \frac{\Delta y}{\Delta x} = \frac{4}{8} = 0.5$ ($|m| < 1$)

So, x is the major axis.

$x_{k+1} = x_k + 1$

$y_{k+1} = y_k + 0.5$



6. Step by Step Calculation

k	x_k (Calculated)	y_k (Calculated)	round(x_k)	round(y_k)	Pixel (x, y)
0	2.000	2.000	2	2	(2, 2)
1	3.000	2.500	3	3	(3, 3)
2	4.000	3.000	4	3	(4, 3)
3	5.000	3.500	5	4	(5, 4)
4	6.000	4.000	6	4	(6, 4)
5	7.000	4.500	7	5	(7, 5)
6	8.000	5.000	8	5	(8, 5)
7	9.000	5.500	9	6	(9, 6)
8	10.000	6.000	10	6	(10, 6)

7. Step by Step ($|m| > 1$) [y is major axis]

1. $\Delta x = x_2 - x_1$
2. $\Delta y = y_2 - y_1$
3. $m = \frac{\Delta x}{\Delta y}$
4. $x = x_1$, $y = y_1$
5. Plot first point (round(x), round(y))
6. For i = 1 to Δy
   - $y = y + 1$
   - $x = x + m$
   - Plot (round(x), round(y))
7. Stop

Example :

Line between (2, 2) and (5, 10)

$\Delta x = 3$, $\Delta y = 8$

$m = \frac{3}{8} = 0.375$

y is major axis.

8. Advantages

- ✓ Simple and easy to understand.
- ✓ Requires less memory.
- ✓ Works for all types of lines.
- ✓ Uses incremental calculations.

9. Limitations

- × Uses floating point arithmetic (slower than integer methods).
- × Rounding causes accumulated errors (less accuracy).
- × Produces jagged lines than Bresenham's algorithm.

Important Points

* DDA is a direct approach based on slope.
* Major axis determines the number of steps.
* $|m| \le 1$ → x is major axis.
* $|m| > 1$ → y is major axis.
* Pixel positions are obtained by rounding.
* Uses floating point calculations.

Typical Exam Questions

1. What is DDA Algorithm ?
2. Explain Simple DDA Algorithm with steps.
3. Draw a line between (2, 2) and (10, 6) using DDA Algorithm.
4. Write the DDA Algorithm for $|m| > 1$.
5. Compare DDA and Bresenham's line algorithm.

Key Terms

* DDA : Digital Differential Analyzer
* Δx : Change in x ($x_2 - x_1$)
* Δy : Change in y ($y_2 - y_1$)
* m : Slope of the line
* Major Axis : Direction with more steps
* Rounding : Conversion to nearest integer

RGPV Notes Hub

IT601 – Computer Graphics & Multimedia

Unit – II

Topic – 5

Date : __/__/__

Bresenham's Line Drawing Algorithm

1. Introduction

Bresenham's Line Drawing Algorithm is an efficient scan conversion algorithm used to draw a straight line on a raster display device.

It uses only integer addition and subtraction, hence it is fast and accurate.

2. Basic Idea

- We draw the line from left to right.
- At each step, we decide whether to choose the pixel E (East) or NE (North-East).
- A decision parameter (p_k) is used to choose the nearest pixel to the ideal line.

3. Assumptions

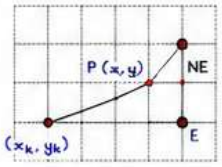
- Line is drawn from (x_0, y_0) to (x_1, y_1).
- Slope (m) of the line is : $0 \le m \le 1$ (i.e., $|\Delta y| \le |\Delta x|$)
- If $m > 1$, interchange x and y.
- If the line is drawn from right to left, interchange the end points.

4. Mathematical Formulation

Let the current pixel be (x_k, y_k).

Two possible pixels in next step are :

E $(x_k + 1, y_k)$ and NE $(x_k + 1, y_k + 1)$



If P is below the line, choose E.

If P is above the line, choose NE.

If P is on the line, either E or NE.

The decision parameter p_k is defined as :

$$p_k = 2\Delta y\, x_k - 2\Delta x\, x_k + C$$

where $C = 2\Delta y\, y_0 - 2\Delta x\, x_0 + \Delta x$

$\Delta x = x_1 - x_0$, $\Delta y = y_1 - y_0$

5. Initial Decision Parameter

At starting point (x_0, y_0),

$$p_0 = 2\Delta y\, y_0 - 2\Delta x\, x_0 + \Delta x$$

$$= 2\Delta y \left(y_0 + \tfrac{1}{2}\right) - 2\Delta x\,(x_0 + 1)$$

6. Algorithm ($0 \le m \le 1$)

Input : (x_0, y_0) , (x_1, y_1)

Output : Set of pixels representing the line

$\Delta x = x_1 - x_0$, $\Delta y = y_1 - y_0$

Initial decision parameter : $p_0 = 2\Delta y \left(y_0 + \frac{1}{2}\right) - 2\Delta x\,(x_0 + 1)$

$x = x_0$, $y = y_0$

Plot (x, y)

Repeat for $k = 0$ to $\Delta x - 1$

  If $p_k < 0$

    Choose E pixel $(x = x + 1,\ y = y)$

    $p_{k+1} = p_k + 2\Delta y$

  Else

    Choose NE pixel $(x = x + 1,\ y = y + 1)$

    $p_{k+1} = p_k + 2\,(\Delta y - \Delta x)$

  End of If

  Plot (x, y)

End of Repeat

7. Example

Draw a line between (2, 2) and (10, 6) using Bresenham's Line Drawing Algorithm.

Solution :-

$\Delta x = 10 - 2 = 8$, $\Delta y = 6 - 2 = 4$ $(m = 0.5)$

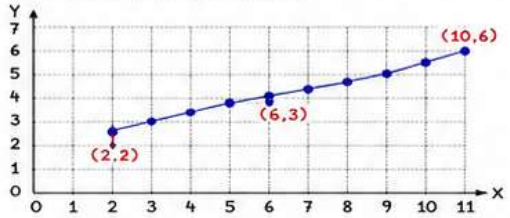
$p_0 = 2\Delta y \left(y_0 + \frac{1}{2}\right) - 2\Delta x\,(x_0 + 1)$

$= 2 \times 4 \times (2.5) - 2 \times 8 \times 3$

$= 20 - 48 = -28$

k	p_k	Decision	Pixel Plotted (x, y)	Next p_{k+1}
0	−28	E	(2, 2)	−20
1	−20	E	(3, 2)	−12
2	−12	E	(4, 2)	−4
3	−4	E	(5, 2)	4
4	4	NE	(6, 3)	−20
5	−20	E	(7, 3)	−12
6	−12	E	(8, 3)	−4
7	−4	E	(9, 3)	4
8	4	NE	(10, 4)	−20

(For last step we plot (10, 6))



8. Handling Other Cases

- If $m < 0$ (negative slope)
  → Use the above algorithm with $\Delta y = |y_1 - y_0|$ and change y by −1 when NE is chosen.
- If $m > 1$
  → Interchange x and y and apply algorithm.
- If line is drawn from right to left
  → Interchange end points so that $x_0 < x_1$.

9. Advantages

- ✓ Uses only integer addition and subtraction.
- ✓ Fast and efficient.
- ✓ Less memory requirement.
- ✓ Suitable for real-time applications.
- ✓ Produces accurate and visually pleasing lines.

10. Limitations

- × Works for straight lines only.
- × More complex logic needed for steep slopes ($m > 1$).
- × Not suitable for sub-pixel accuracy.

11. Applications

- Computer graphics systems.
- CAD / CAM software.
- Game development.
- Image drawing and editing tools.
- Scientific and engineering visualization.

Important Points

- Bresenham's algorithm is based on decision parameter.
- It avoids floating point calculations.
- Works efficiently for $0 \le m \le 1$.
- Symmetry can be used to handle all octants.
- Only addition and subtraction are required.

Typical Exam Questions

1. Explain Bresenham's Line Drawing Algorithm.
2. Derive the decision parameter for Bresenham's algorithm.
3. Draw a line between (2, 2) and (10, 6) using Bresenham's algorithm.
4. How other octants are handled in Bresenham's algorithm?
5. Compare DDA and Bresenham's line drawing algorithms.

Key Terms

- Scan Conversion
- Raster Display
- Decision Parameter (p_k)
- E (East) Pixel
- NE (North-East) Pixel
- Octant
- Slope (m)
- Integer Arithmetic

CIRCLE DRAWING

1. INTRODUCTION

- A circle is the set of all points in a plane that are at a fixed distance (radius r) from a fixed point (center).
- The equation of a circle with center (x_c, y_c) and radius r is :

$$(x - x_c)^2 + (y - y_c)^2 = r^2$$

For a circle centered at origin (0, 0) :

$$x^2 + y^2 = r^2$$

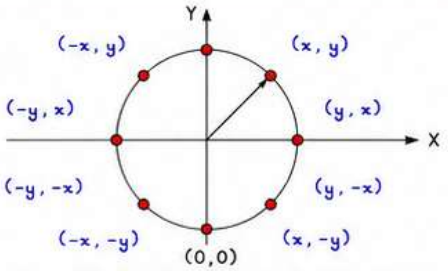
2. BASIC IDEA

- We calculate the points of the circle in one octant (0° to 45°) and plot the remaining points using symmetry.
- This reduces the calculations to $\frac{1}{8}$th of the circle.

3. 8-WAY SYMMETRY OF CIRCLE

If (x, y) is a point on the circle centered at (x_c, y_c), then the other 7 symmetric points are :

1. $(x_c + x, y_c + y)$	5. $(x_c + y, y_c + x)$
2. $(x_c - x, y_c + y)$	6. $(x_c - y, y_c + x)$
3. $(x_c + x, y_c - y)$	7. $(x_c + y, y_c - x)$
4. $(x_c - x, y_c - y)$	8. $(x_c - y, y_c - x)$



Thus, we calculate points in one octant and use symmetry to get the full circle.

4. GENERAL METHOD (FLOATING POINT)

1. Start at the point (0, r).
2. Choose E (x + 1, y) or SE (x + 1, y - 1) as the next point.
3. Use the circle equation to decide.

Decision Parameter (p) :

$$p = x^2 + y^2 - r^2$$

Initial value :

At (0, r), $p_0 = 0^2 + r^2 - r^2 = 0$

At Point (x, y)	Choose	New Point	Update p
p < 0	E	(x + 1, y)	p + 2x + 1
p ≥ 0	SE	(x + 1, y - 1)	p + 2x - 2y + 1

5. ALGORITHM (GENERAL METHOD)

Input : Center (x_c, y_c) and radius r
Output : Set of pixel positions of circle

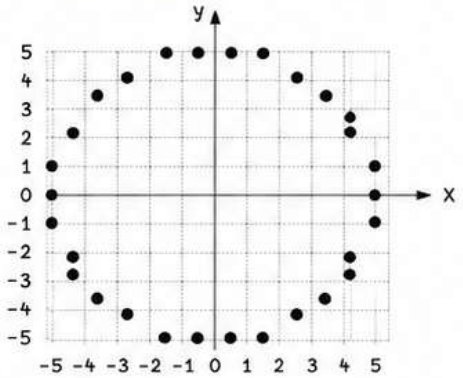
1. x = 0, y = r
2. p = 0
3. Plot 8 symmetric points of (x, y)
4. While (x < y)
   If (p < 0)
   x = x + 1
   p = p + 2x + 1
   Else
   x = x + 1
   y = y - 1
   p = p + 2x - 2y + 1
5. End While
6. Stop

6. EXAMPLE

Draw a circle of radius r = 5 centered at (0, 0) using general method.

Step (k)	x	y	p	Decision	Next (x, y)
0	0	5	0	p ≥ 0 → SE	(1, 4)
1	1	4	-4	p < 0 → E	(2, 4)
2	2	4	1	p ≥ 0 → SE	(3, 3)
3	3	3	0	p ≥ 0 → SE	(4, 2)
4	4	2	3	p ≥ 0 → SE	(5, 1)
5	5	1	8	Stop (x ≥ y)	

Points in one octant :
(0,5) (1,4) (2,4) (3,3) (4,2) (5,1)
Other points are obtained by 8-way symmetry.



7. ADVANTAGES

- ✓ Simple and easy to understand.
- ✓ Uses 8-way symmetry to reduce computations.
- ✓ Suitable for small radius and educational purposes.

8. LIMITATIONS

- × Uses floating point arithmetic (slower).
- × Not suitable for hardware with integer only.
- × Accumulation of rounding errors.

IMPORTANT POINTS

- Circle equation : $x^2 + y^2 = r^2$ (center at origin).
- Use 8-way symmetry.
- Only $\frac{1}{8}$th points are calculated.
- Start from (0, r) and move towards $(\frac{r}{\sqrt{2}}, r\frac{1}{\sqrt{2}})$.
- General method uses floating point arithmetic.

TYPICAL EXAM QUESTIONS

1. What is the equation of a circle?
2. Explain 8-way symmetry of a circle with diagram.
3. Explain general circle drawing algorithm.
4. Draw a circle of radius 6 using general method.
5. Compare general method and Bresenham's algorithm for circle drawing.

KEY TERMS

- Circle
- Radius (r)
- Center (x_c, y_c)
- Octant
- 8-Way Symmetry
- Decision Parameter (p)
- General Method

Topic – 7 : General Method of Circle Drawing (14 Marks)

1. Introduction :

The general method of circle drawing is based on the circle equation.

2. Circle Equation :

$$(x - x_c)^2 + (y - y_c)^2 = r^2$$

where (x_c, y_c) = center, r = radius

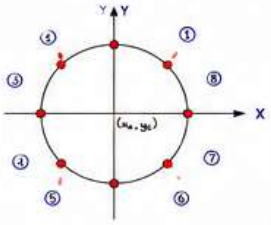
3. Algorithm (General Method) :

(i) Input center (x_c, y_c) and radius r.
(ii) Initialize : $x = 0$, $y = r$
(iii) Compute decision parameter : $p = 1 - r$
(iv) Plot the 8 symmetric points of (x, y).
(v) While $(x < y)$
  If $p < 0$
    $x = x + 1$
    $p = p + 2x + 1$
  Else
    $x = x + 1$
    $y = y - 1$
    $p = p + 2x - 2y + 1$
(vi) End While

4. 8-Way Symmetry :

If (x, y) is a point on the circle, other 7 symmetric points are :

$(x + x_c, y + y_c)$
$(x + x_c, y_3 - y_c)$
$(x_c - y_c, y_c + y_c)$
$(x_c - x_c, y_3 - y_c)$
$(x + y_c, y_c + x_c)$
$(x_c - y_c, y_c + x_c)$
$(x + y_c, y_c - x_c)$
$(x_c - y_c, y_c - x_c)$



Important Points :

- Uses floating point arithmetic.
- Only 1/8th points are calculated.
- 8-way symmetry is used.
- Start from (0,r) and move towards $\left(\frac{r}{\sqrt{2}}, \frac{r}{\sqrt{2}}\right)$.

Topic – 8 : Symmetric DDA Algorithm (14 Marks)

1. Introduction :

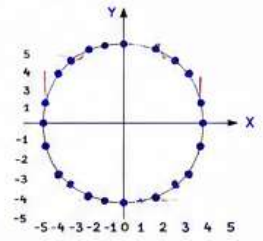
Symmetric DDA algorithm uses incremental approach (floating point) and 8-way symmetry to draw a circle.

2. Algorithm :

(i) Input center (x_c, y_c) and radius r.
(ii) Initialize : $x = 0$, $y = r$
(iii) Compute : $dx = 1$, $dy = 1$
(iv) Plot the 8 symmetric points.
(v) While $(x < y)$
  $x = x + dx$
  $y = y - dy$
  Plot the 8 symmetric points (round x, y)
(vi) End While
(vii) Stop

3. 8-Way Symmetry :

Same as general method (Topic – 7).



4. Example :

Draw circle of radius r = 5 centered at (0,0).

k	x (float)	y (float)	round(x)	round(y)	Points (one octant)
0	0.000	5.000	0	5	(0,5)
1	0.200	4.000	0	4	(1,4)
2	0.400	3.000	0	3	(2,4)
3	0.600	2.000	1	2	(3,2)
4	0.800	1.000	1	1	(4,1)
(Then x > y, stop)					

Important Points :

- Simple and easy.
- Uses floating point.
- 8-way symmetry.
- More computation than Bresenham.

Topic – 9 : Bresenham's Circle Drawing Algorithm (14 Marks)

1. Introduction :

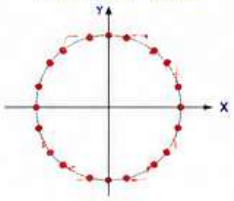
Bresenham's circle algorithm uses only integer arithmetic and is more efficient than general method.

2. Algorithm :

(i) Input center (x_c, y_c) and radius r.
(ii) Initialize : $x = 0$, $y = r$
(iii) Decision parameter : $p = 1 - r$
(iv) Plot the 8 symmetric points.
(v) While $(x < y)$
  If $p < 0$
    $x = x + 1$
    $p = p + 2x + 1$
  Else
    $x = x + 1$
    $y = y - 1$
    $p = p + 2x - 2y + 1$
(vi) End While
(vii) Stop

3. 8-Way Symmetry :

Same 8 symmetric points as in general method.



4. Example : Draw circle of radius r = 5 at (0,0)

k	x	y	p	Plot Point (one octant)
0	0	5	-4	(0,5)
1	1	5	-1	(1,5)
2	2	4	3	(2,4)
3	3	4	-2	(3,4)
4	4	3	5	(4,3)
(Then x > y, stop)				

Important Points :

- Uses only integer arithmetic.
- Fast and efficient.
- Less memory.
- Suitable for real-time applications.

Topic – 10 : Curves

1. Introduction :

A curve is a smooth continuous path in a plane formed by movement of a point.

2. Types of Curves :

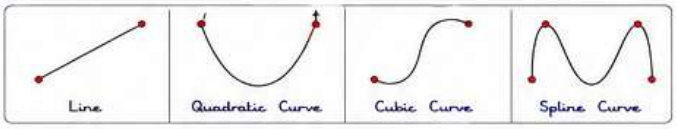
(i) Algebraic Curves
(ii) Parametric Curves
(iii) Explicit Curves
(iv) Implicit Curves
(v) Spline Curves
(vi) Bezier Curves

3. Properties of Curves :

- Continuity
- Smoothness
- Tangent
- Curvature

4. Classification by Continuity :

- C^0 : Position Continuity
- C^1 : Tangent Continuity
- C^2 : Curvature Continuity



Applications :

Curves are widely used in Animation, CAD/CAM, Font design, Scientific visualization, Games etc.

Topic – 11 : Parametric Function (14 Marks)

1. Introduction :

In parametric representation, a curve is defined by two functions x(t) and y(t) in terms of a parameter t.

2. Parametric Equations :

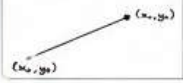
$$x = x(t)$$
$$y = y(t)$$
$$t_1 \le t \le t_2$$

3. Advantages :

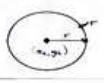
- Represents complex curves easily.
- Useful for animation.
- Independent of coordinate system.

4. Examples :

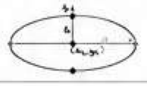
(i) Line
$x = x_0 + t(x_1 - x_0)$
$y = y_0 + t(y_1 - y_0)$
$0 \le t \le 1$



(ii) Circle
$x = x_c + r \cos t$
$y = y_c + r \sin t$
$0 \le t \le 2\pi$



(iii) Ellipse
$x = x_c + a \cos t$
$y = y_c + b \sin t$
$0 \le t \le 2\pi$



(iv) Parabola
$x = t$
$y = t^2$
$-\infty < t < \infty$

Important Points :

- Parametric form gives smooth curves.
- t is independent variable.

Topic – 12 : Bezier Method (14 Marks)

1. Introduction :

Bezier curve is defined by control points. The curve always lies within the convex hull of control points.

2. Bezier Curve Equation :

$$B(t) = \sum_{i=0}^{n} B_{i,n}(t)\, P_i \quad , \quad 0 \le t \le 1$$

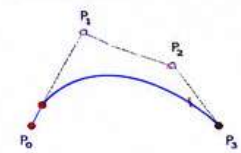
where $B_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-i}$ (Bernstein Basis)

P_i = control points

3. Example : Cubic Bezier Curve (4 control points)

$$B(t) = (1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t^2(1-t) P_2 + t^3 P_3$$

4. Diagram (Cubic)



5. Advantages :

- Easy to implement.
- Produces smooth curves.
- End points lie on the curve.
- Used in CAD, animation.

Important Points :

- Curve always inside convex hull.
- Moving control points changes the shape.

Topic – 13 : B-Spline Method (14 Marks)

1. Introduction :

B-Spline (Basis Spline) is a generalization of Bezier curve. It provides more local control.

2. Basis Function (Cox-de Boor Formula) :

(i) $N_{i,0}(t) = \begin{cases} 1, & t_i \le t < t_{i+1} \\ 0, & \text{otherwise} \end{cases}$

(ii) $N_{i,p}(t) = \frac{t - t_i}{t_{i+p} - t_i} N_{i,p-1}(t) + \frac{t_{i+p+1} - t}{t_{i+p+1} - t_{i+1}} N_{i+1,p-1}(t)$

3. General Equation :

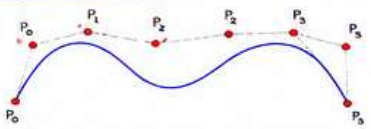
$$C(t) = \sum_{i=0}^{n} N_{i,p}(t)\, P_i$$

where,
P_i = control points
p = degree of curve
$N_{i,p}(t)$ = basis function

4. Terminology :

- Control Points : $P_0, P_1, \ldots, P_n$
- Knot Vector : $U = \{u_0, u_1, \ldots, u_m\}$
- Degree : p
- Knot Condition : $m = n + p + 1$

5. Example (Cubic B-Spline)



Comparison : Bezier vs B-Spline

Feature	Bezier Curve	B-Spline Curve
Control	Global (all control points affect curve)	Local (only nearby points affect)
Curve Form	Passes through first & last control points	Does not pass through control points
Knot Vector	Not required	Required
Flexibility	Less	More