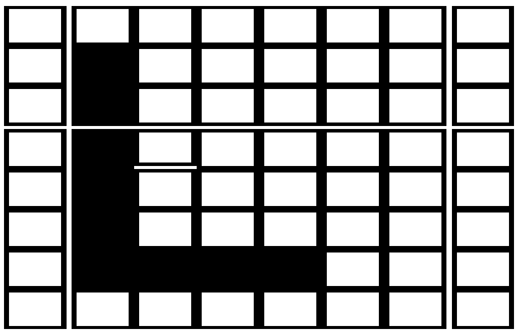
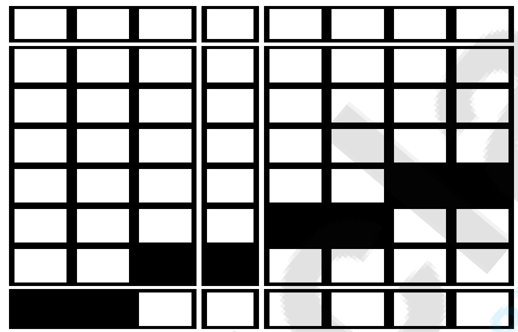
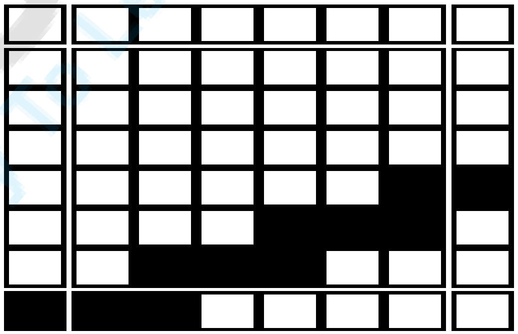
LINE DRAWING ALGORITHMS

Basic Concepts of Line Drawing

The line should appear as a straight line.





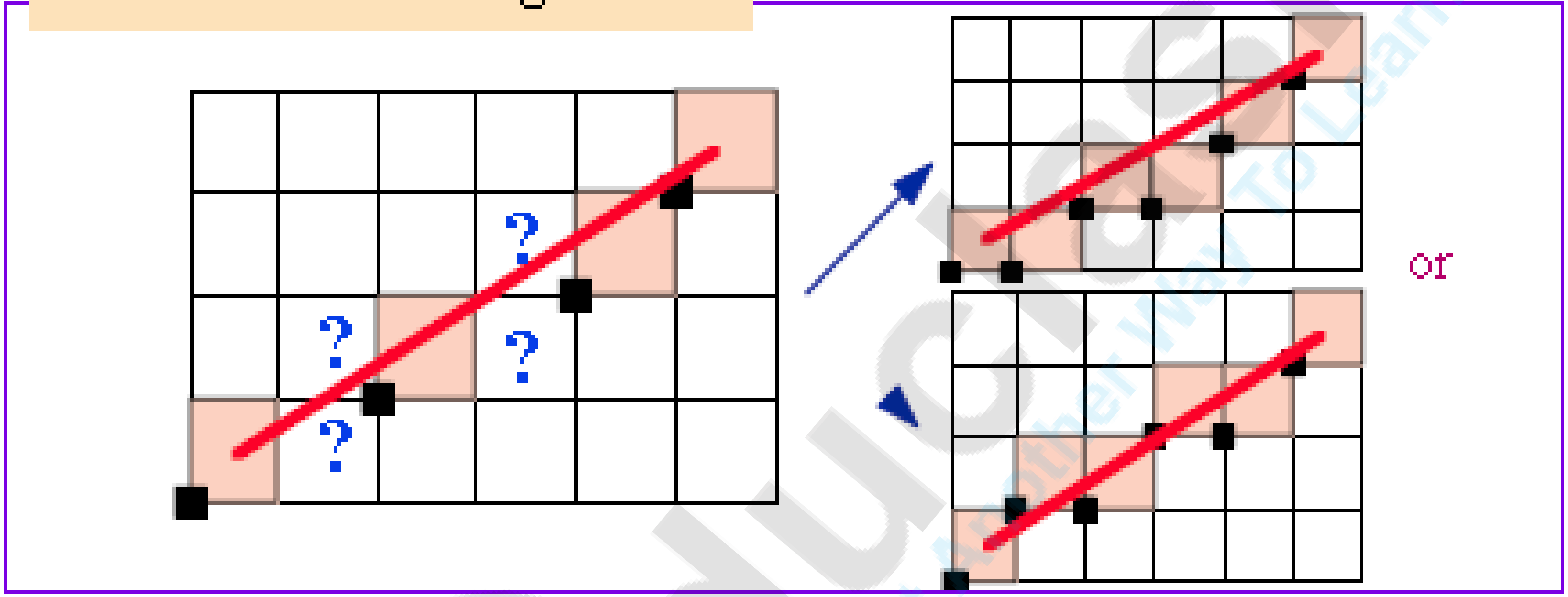


Vertical and horizontal lines   Line with other orientation

Horizontal and vertical lines are straight and have same width.
The line with any other orientation is neither straight nor has same width.
In this case we have to accept approximate pixels in such situations.
A slanted line becomes like a staircase.

Basic Concepts of Line Drawing

Rasterization of straight lines.



Rasterization yields uneven brightness: Horizontal and vertical lines appear brighter than the 45° lines.

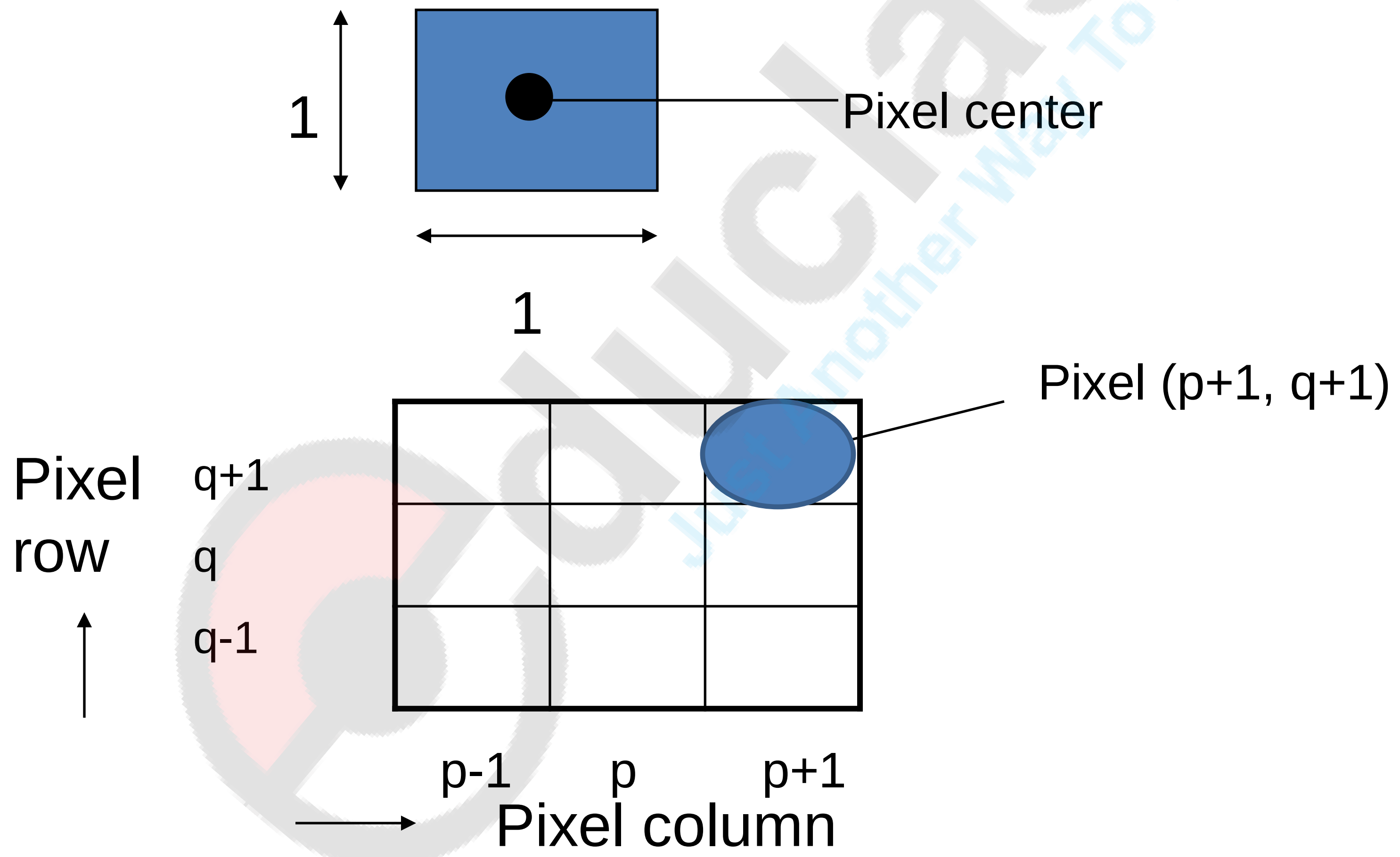
- The special procedure determine which pixel will provide the best approximation to the desired picture or graphics object.
- The process of determining the appropriate pixels for representing picture or graphics object is known as Rasterization.

Basic Concepts of Line Drawing

- Line should have constant density.
  - To maintain constant density dots should be equally spaced.

- The line should be drawn rapidly.
  - This implies that we have to draw line using minimum of computation.

Pixel plotting



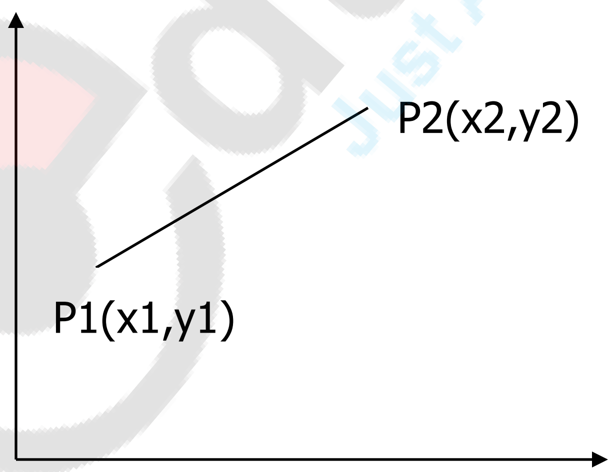
Line drawing Algorithm

- A line is defined by its two endpoints & the equation of line $y=mx + c$, where

  m= slope

  c = 'y' intercept of the line

  (x_1,y_1) & (x_2,y_2) are the end points



- Given that the two endpoints of a line segment are specified at positions (x_1 , y_1) and (x_2 , y_2). We can determine values for the slope m and y intercept c with the following calculations:

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

$$c = y_1 - m \cdot x_1$$

For any given x interval Δx along a line, we can compute the corresponding y interval Δy as

$$\Delta y = m \, \Delta x$$

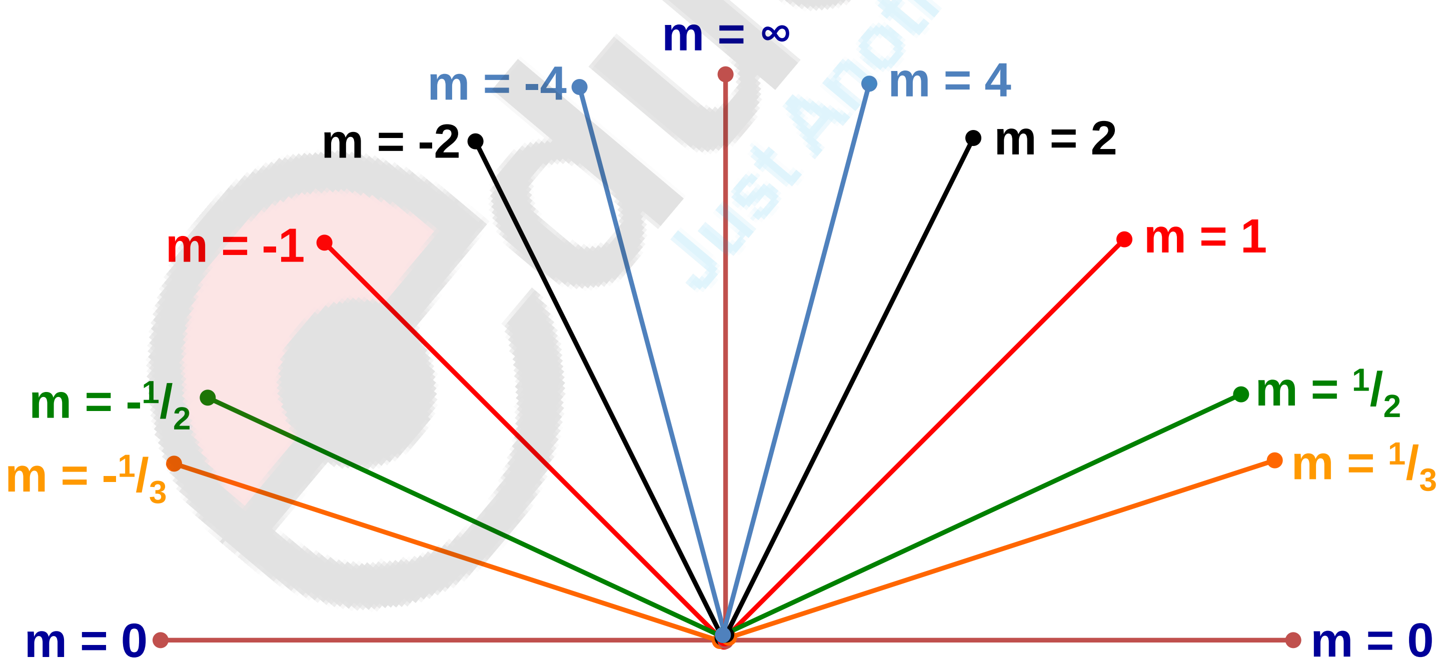
Similarly, we can obtain the x interval Δx corresponding to a specified Δy as

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

- Once the intervals are known the values for next x and next y on the straight line can be obtained as follows.
- $X_{i+1} = X_i + \Delta x$
- $Y_{i+1} = Y_i + \Delta y$

Lines & Slopes

- The slope of a line (m) is defined by its start and end coordinates
- The diagram below shows some examples of lines and their slopes



Digital Differential Analyser (DDA) Algorithm

- The digital differential analyzer (DDA) is a scan – conversion line algorithm based on calculating either Δy or Δx.

- We sample the line at unit intervals in one coordinate and determine corresponding integer values nearest the line path for the other coordinate.

- Calculations at each step are performed using results from the preceding step.

- Suppose at step i we have calculated (x_i, y_i) to be a point on the line.
- The next point (x_{i+1}, y_{i+1}) should satisfy $\Delta y/\Delta x = m$ where

  we have $y_{i+1} = y_i + m\Delta x$

  or $x_{i+1} = x_i + \Delta y/m$

DDA LINE ALGORITHM

- **If the slope is less than or equal to 1 ($|m| \leq 1$)**

- Start with $x=x_1$ and $y=y_1$ and set $\Delta x = 1$(unit increment in x-direction)
- Compute each successive y value by $y_{i+1} = y_i + m$
- Subscript i takes integer values starting from 1, for the first point, and increases by 1 until final end point is reached. Since m can take any real number between 0 to1, the calculated y values must be rounded to the nearest integer.

DDA LINE ALGORITHM

1. Read the line end points (x_1, y_1) and (x_2, y_2) such that they are not equal. If they equal then plot that point and exit.
2. $\Delta x = | x_2 - x_1 |$ and $\Delta y = | y_2 - y_1 |$
3. If $(\Delta x \geq \Delta y)$ then length $= \Delta x$ else length $= \Delta y$

   end if
4. $dx = (x_2 - x_1) / \text{length}$

   $dy = (y_2 - y_1) / \text{length}$
5. $x = x_1 + 0.5 \times \text{sign}(dx)$

   $y = y_1 + 0.5 \times \text{sign}(dy)$

DDA LINE ALGORITHM

6. i = 0

```
while (i ≤ length)
{
 plot (integer(x), integer(y))
x = x + dx
y = y + dy
i = i + 1
}
```

7. Stop.

DDA LINE ALGORITHM

- **Advantages of DDA Algorithm**
  1. It is the simplest algorithm and it does not require special skills for implementation.
  2. It is faster method for calculating pixel positions than the direct use of equation.

- **Disadvantages of DDA Algorithm**
  1. Floating point addition in DDA algorithm is still time-consuming.

Example

1) Consider the line from (0, 0) to (4, 6). Use the DDA algorithm to rasterize this line.
2) Consider the line from (0, 0) to (-6,-6). Use the DDA algorithm to rasterize this line.
3) Consider the line from (0, 0) to (8, 4). Use the DDA algorithm to rasterize this line.
4) Consider the line from (-3, 3) to (4, -4). Use the DDA algorithm to rasterize this line.

Example 1

i	Plot	x	Y
		0.5	0.5
0	(0, 0)		
		1.167	1.5
1	(1,1)		
		1.833	2.5
2	(1,2)		
		2.5	3.5
3	(2,3)		
		3.167	4.5
4	(3,4)		
		3.833	5.5
5	(3,5)		
		4.5	6.5
6	(4,6)		
		5.167	7.5

Example 2

i	Plot	x	Y
		-0.5	-0.5
0	(0,0)		
		-1.5	-1.5
1	(-1, -1)		
		-2.5	-2.5
2	(-2,-2)		
		-3.5	-3.5
3	(-3,-3)		
		-4.5	-4.5
4	(-4,-4)		
		-5.5	-5.5
5	(-5,-5)		
		-6.5	-6.5
6	(-6,-6)		
		-7.5	-7.5

```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
#include<dos.h>
#include<math.h>
void dda(int, int, int, int, int, int);
int sign(int);
void main()
{
        float x1, y1, x2, y2;
        clrscr();
        cout << " Enter the cordinates of P :";
cin >> x1 >> y1;
        cout << "\n Enter the cordinates of Q :";
        cin >> x2 >> y2;
        int gd = DETECT, gmode;
        initgraph(&gd, &gmode,"c:\\tc\\bgi");
        int maxx = getmaxx();
        int maxy = getmaxy();
        line(maxx/2,0,maxx/2,maxy);
        line(0,maxy/2, maxx, maxy/2);
```

"Success is not final, failure is not fatal: it is the courage to continue that counts."

Reference: Computer Graphics , Schaum's Series

Computer Graphics , A.P.Godse,D.A.Godse