Output Primitives

Deep Prakash Singh

Points and Lines

Line Drawing Algorithms

DDA Algorithm

Bresenham's Line Algorithm

Midpoint Circle Algorithm

Primitives

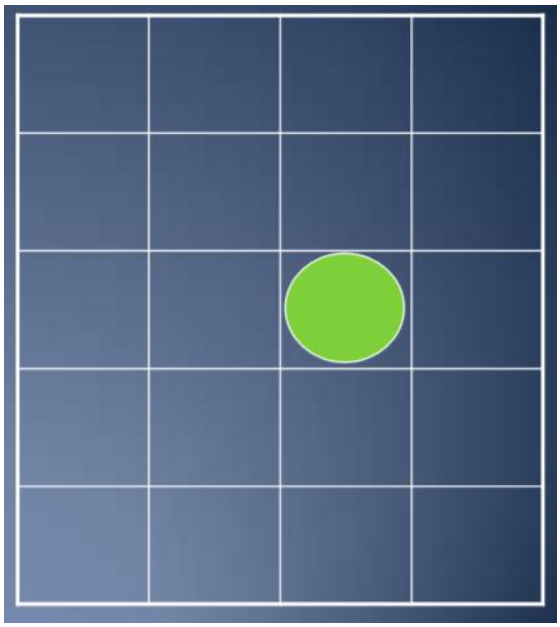
- 2-D Drawing or a 3-D Object consist of Graphical Primitives such as Points, Lines, Circles & Filled Polygons.
- Graphics System or the Application Program convert each primitive from its geometric definition into a set of Pixels that make up the primitive in the Image Space.
- This Conversion is referred to as SCAN CONVERSION or RASTERIZATION.

RASTERIZATION: Process of determining which pixels provide the best approximation to a desired line on the screen.

SCAN CONVERSION: Combination of rasterization and generating the picture in scan line order.

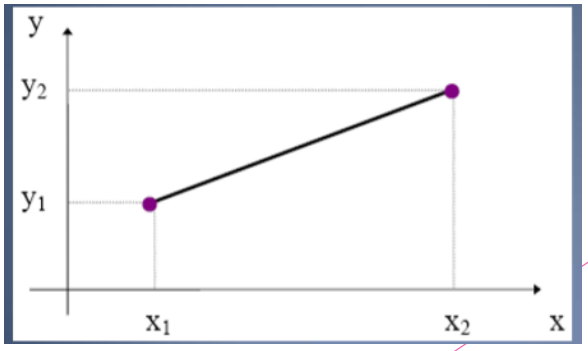
Point

- A point is shown by illuminating a pixel on the screen

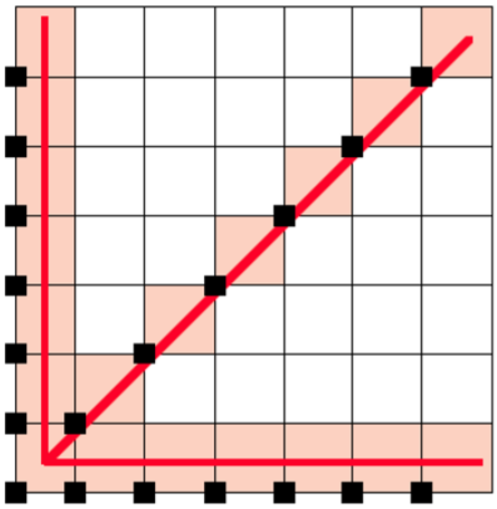


Lines

- A line segment is completely defined in terms of its two endpoints.
- A line segment is thus defined as: Line_Seg = { (x1, y1), (x2, y2)
- They must start and end accurately.
- Lines should have constant brightness along their length
- Lines should drawn rapidly.



For horizontal, vertical and 45°lines, the choice of raster elements is obvious. This lines exhibit constant brightness along the length:

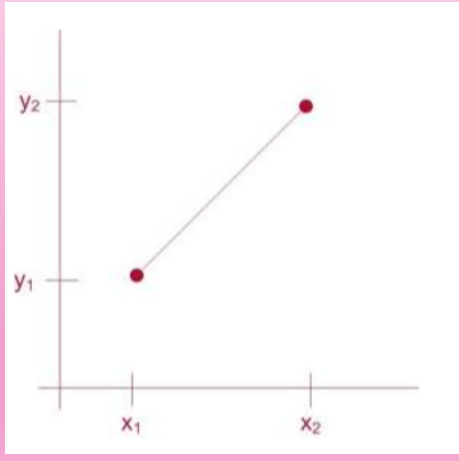


Line Drawing Algorithms

- Direct Use of Line Equation
- Digital Differential Analyzer (DDA) Algorithm
- Bresenham's Line Algorithm

Direct Use of Line Equation

A simple approach to scan-converting a line is to first scan – convert P_1 and P_2 to pixel coordinates (x_1, y_1) and (x_2, y_2) respectively by using simple line equation. Then set slope $m = (y_2, y_1)/ (x_2, x_1)$ and $b = y_1 - mx_1$



- Given Points (x_1, y_1) and (x_2, y_2)
- All line drawing algorithm make use of the fundamental equations :
- Line Eqn. $y = m. x + b$
- Slope $m = y_2 - y_1 / x_2 - x_1$
- Y-intercept $b = y_1 - m x_1$
- X-interval $\rightarrow \Delta x = \Delta y/m$
- Y- interval $\rightarrow \Delta y = m. \Delta x$
- If $m > 1$, increment y and find x
- If $m \leq 1$, increment x and find y

Steps are as follow:

Start at the pixel for the left-hand end point x_1

Step along the pixels horizontally until we reach right-hand end of the line , x_2

For each pixel compute the corresponding y value.

Round this value to the nearest integer to select the nearest pixel.