The Bresenham's Line Algorithm

The Bresenham's Line Algorithm is a simple and efficient algorithm used for drawing a line on a pixel grid. The algorithm is based on an incremental error approach, making it faster than other methods.

—Derivation of the Algorithm—

The objective is derive the algorithm to determine which pixel to turn on to approximate a stright line between the two given points. $(x_0, y_0), (x_1, y_1)$

- Calculate the differences between the two points:

$$\Delta x = x_1 - x_0$$

$$\Delta y = y_1 - y_0$$

- The decision parameter (P) is used to determine which pixel to chose at each step of the algorithm

$$P = 2 \cdot \Delta_y - \Delta_x$$

→ the initial P is → $P_0 = 2 \cdot \Delta_y - \Delta_x$

- we set the initial point to (x_0, y_0) and ploting it.
- for each x from $x_0 + 1$ to x_1:

if $P \geq 0$, increment y and update P as fallows: -5 4

$$y = y + 1$$ 2,3 9,7

$$P = P + 2 \cdot \Delta y - 2 \cdot \Delta x$$ 3,4

8 - 10

if $P < 0$, update P as fallows:

$$P = P + 2 \cdot \Delta y$$

- for each step, plot the pixel at the current cordinates (x, y)

⟹ after the iteration is complete, you will have a series of plotted points forming a line between (x_0, y_0) and (x_1, y_1)

Example

$\Rightarrow (2,3)$ and $(7,7)$

$\Delta x = x_1 - x_0 = 7 - 2 = \underline{\underline{5}}$

$\Delta y = y_1 - y_0 = 7 - 3 = \underline{\underline{4}}$

$P_0 = 2\Delta_y - \Delta_x = 2 \cdot 4 - 5 = \underline{\underline{3}}$

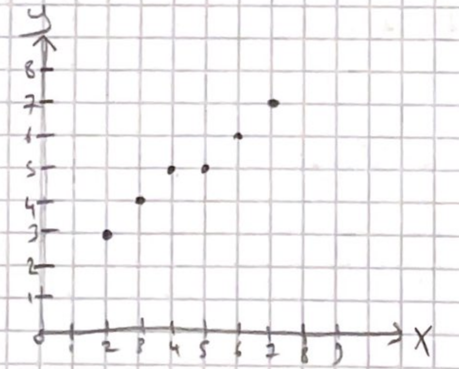
1) $(3,4) \rightarrow P = P_0(3) + \underbrace{2 \cdot \Delta y(4) - 2 \cdot \Delta x(5)}_{-2} = \underline{\underline{1}}$

2) $(4,5) \rightarrow P = 1 + (-2) = \underline{-1}$

3) $(5,5) \rightarrow P = -1 + 2\Delta y(4) = 7$

4) $(6,6)$

$(7,7)$



The Bresenhan's Midpoint Circle Algorithm

The algorithm is an efficient method for drawing a circle on a grid-based display. The algorithm avoids using floating-point arithmetic and takes advantage of symmetry to reduce the computation effort.

- Set the initial decision parameter P as $P = 1 - r$, where r is the radius of the circle.

- Initialize the variables x and y as the starting point of the circle, typically $x = 0$ and $y = r$.

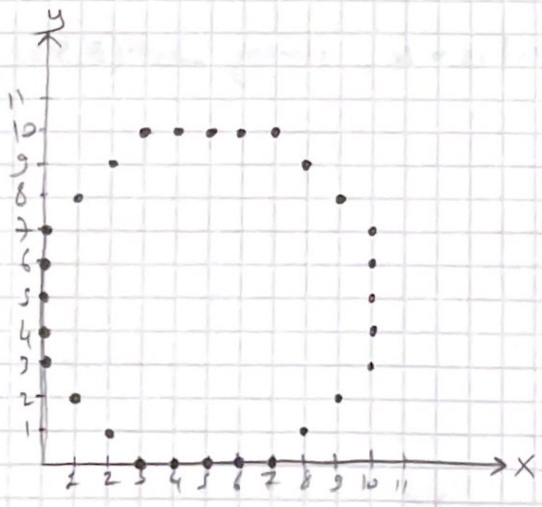
- Repeat the following steps until $x \leq y$:
- Plot the pixel at coordinates (x, y) in the first octant.

- If $(P < 0)$, update it as $P = P + 2x + 1$
- If $(P \geq 0)$, update it as $P = P + 2(x - y) + 1$ and $y - 1$

Increment x by 1. $(x+1)$

- Repeat the above steps for octans 2, 3 and 4 by symmetry. The points in those octans can be obtained by reflection and negation of coordinates.

example radius $= 5$ starting point $= (5, 5)$



The Midpoint Circle Drawing Algorithm

The algorithm is also efficient method of drawing a circle on a grid-based display. It is an adaptation of Breshenam's Circle Algorithm with a slight modification to the decision parameter.

- Set the initial decision parameter P as $P = 5/4 - r$, where r is the radius of circle

- Initialize the variables x and y as the starting point of the circle.

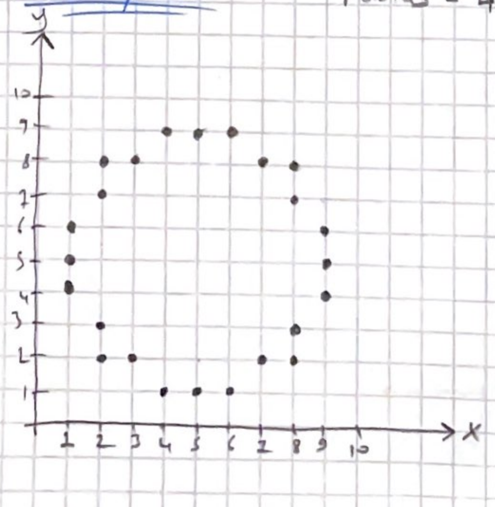
- Repeat the following steps until $x \leq y$
- Plot the pixel at coordinates (x, y) in the first octant.

- If $(P < 0)$, update it as $P = P + 2x + 1$
- If $(P \geq 0)$, update it as $P = P + 2(x - y) + 1$ and $y - 1$

- Increment x by 1. $(x+1)$

- Repeat the above steps for octans 2,3 and 4 by symmetry. The points in those octans can be obtained by reflection and negation of coordinates.

example radius = 4 starting point=(5,5)


y
10
9
8
7
6
5
4
3
2
1
1
2
3
4
5
6
7
8
9
10
x