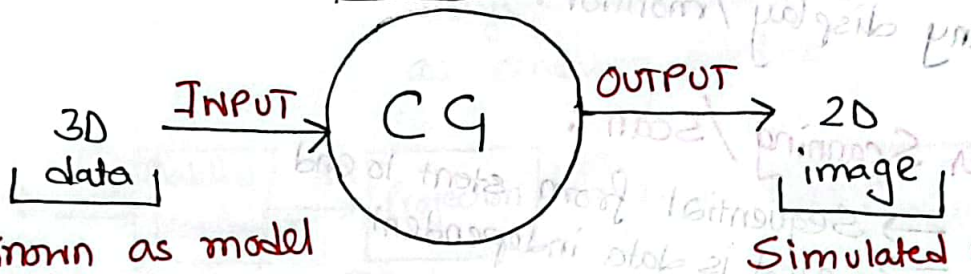
INTRODUCTION TO COMPUTER GRAPHICS

LECTURE 1

1) 3D

Computer Graphics



Known as model

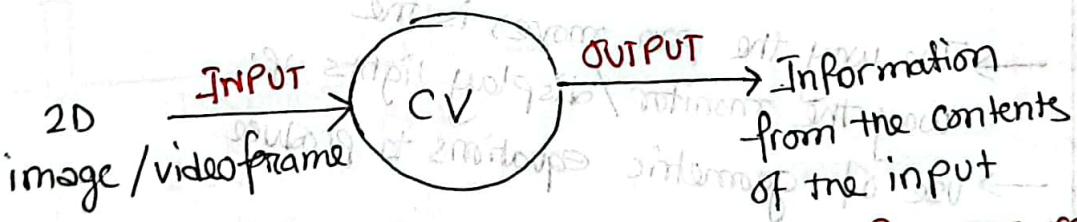
Simulated output

The equation of the object to be simulated is called the model / 3D data.

Example : Animated movies

2) if the input is 2D

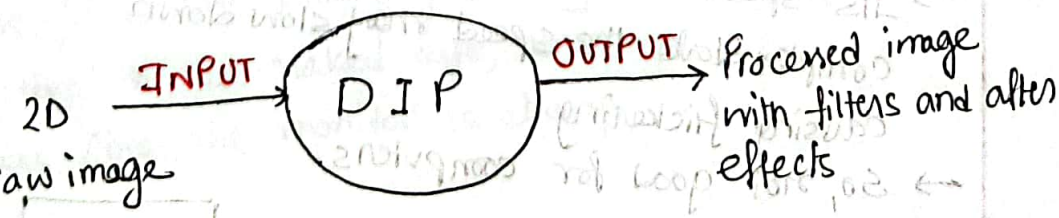
Computer Vision



Example: face recognition

3) 2D

Digital image processing



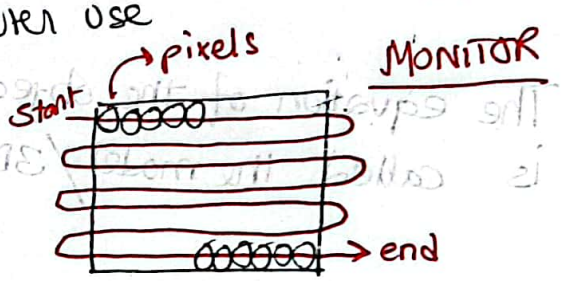
Example: Adobe photoshop
Adobe after effects.

Ø Scanning

The process in which each pixel is being generated : on any display / monitor.

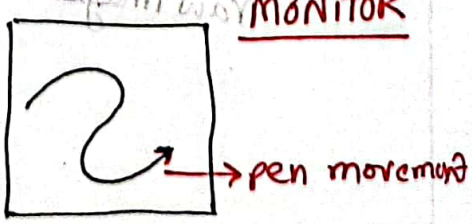
1) Raster Scanning / Scan :

→ Sequential from start to end
→ Speed is data independent
→ is better for computer use

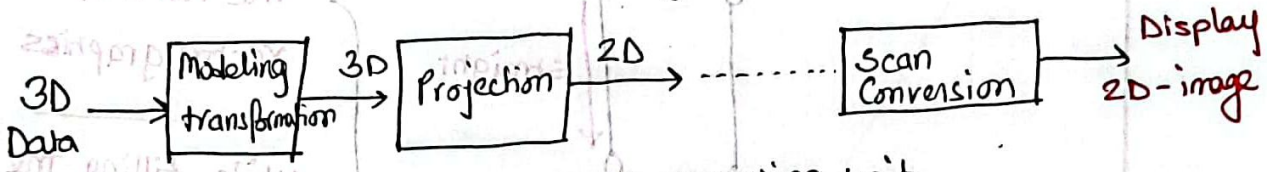


2) Vector / Random Scan / Scanning :

→ The way the pen moves is the way the monitor / display lights up.
→ use of geometric equations to produce image.
→ Vector scanning for line drawing, but not for field object.
→ It's speed is data dependant, for more complex data the speed may slow down causing flickering
→ So, not good for computers.



Rendering Pipe-Line: From input 3D data (model) through simulation to output "simulated 2D image". The whole process is known as rendering pipeline.

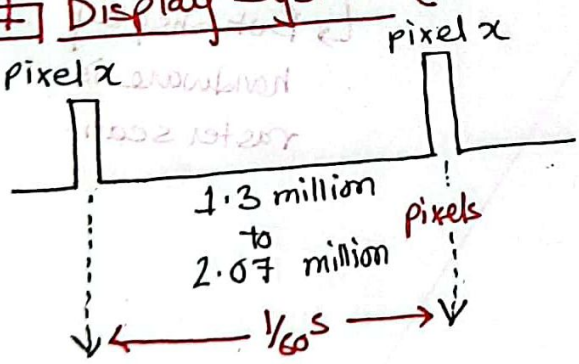


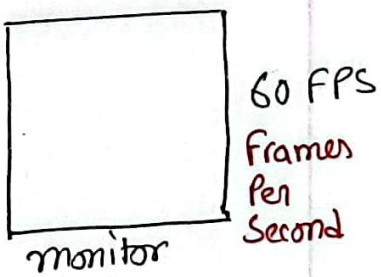
Hardware used: GPU → Graphical processing unit

Display Hardware

- It is part of computer graphics which takes in 2D input

Display System (How it displays)





pixel x will light up again after 1/60 seconds.

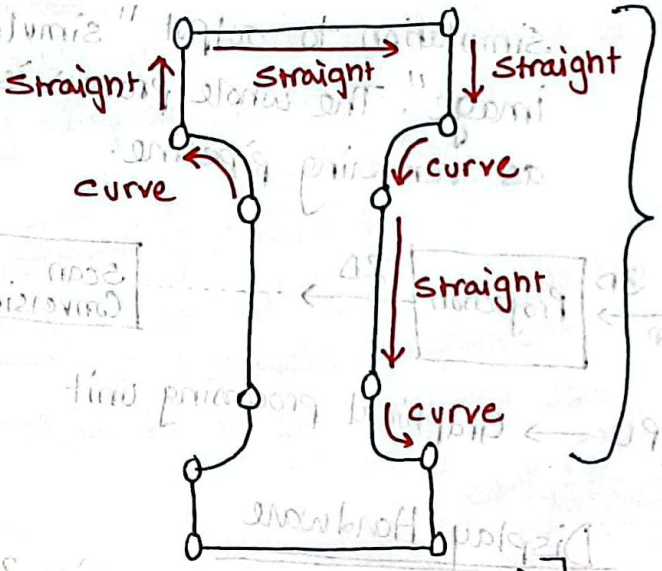
- Since the pixels turn on & off so fast, it is not phisically/visually perceptive to the naked eye, hence it feels like the monitor is always "on".
- when all pixels light up, it is known as a single frame.

Scanning:

→ takes place at the time of capturing and at the time of displaying

- A process that streams a 1D (dimensional) dot, from a 2D image.
- Can use set of dots to create a 2D image.

Example of scanning :

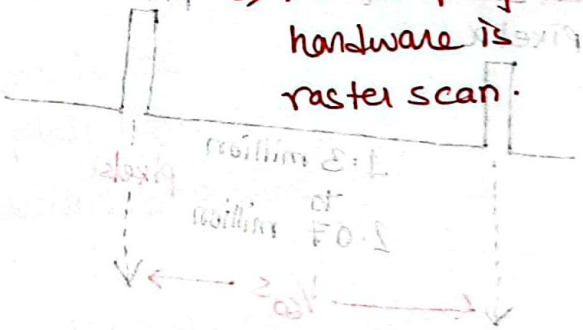


Creating the boundary is the work of vector graphics

While filling the letter boundary with pixels is work of raster scanning

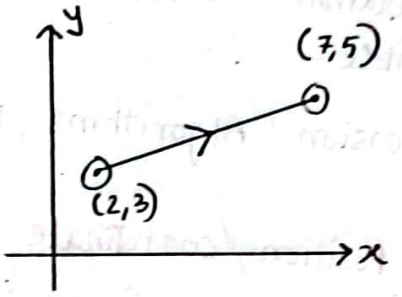
Recent hardware is a hybrid of both.

→ But display hardware is raster scan.

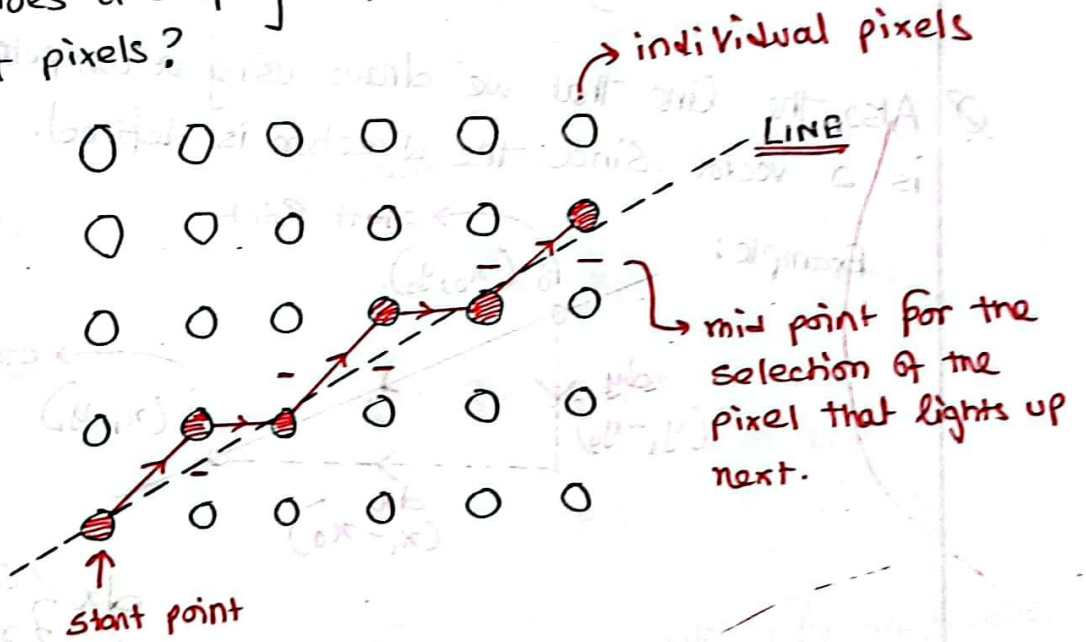


LINE DRAWING ALGORITHMS → Scan Conversion Algorithms LECTURE 2

- In the case of drawing a line, the line segment is defined by the starting point (the coordinate) an the ending point (the coordinate).



- But how does a display outputs this information through the use of pixels?



Ø But how do we select which pixel to turn on?

Ø Few considerations:

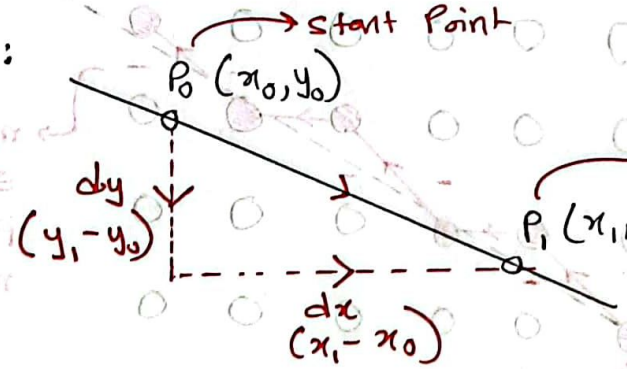
1) The line has to be as smooth looking as possible.
2) The calculations should be as fast as possible.

Ø In scan conversion Algorithms, there's generally two outputs

1) Position / coordinate of a pixel (x, y)
2) RGB value of a pixel (The color)

Ø Also, the line that we draw using a computer is a vector. Since the direction is defined.

Example:

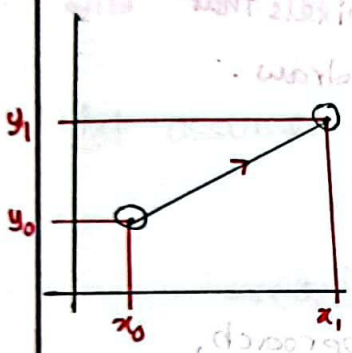


dx, dy { carries the magnitude & direction

+ve/-ve defines the direction.

Equations of a line:

$$y = mx + c, \quad \text{---} (i)$$

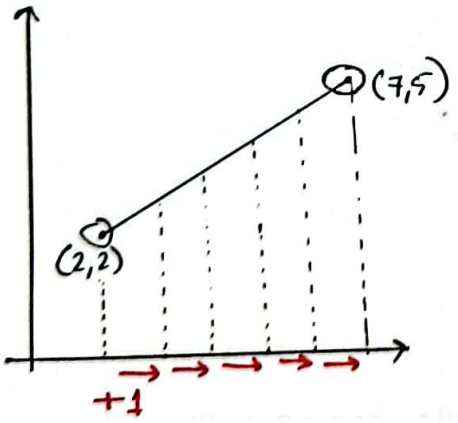


where, $$m = \frac{y_1 - y_0}{x_1 - x_0} \quad \text{---} (ii)$$

the constant c, can be figured out by plugging in the value of the start/end coordinates.

$$\Rightarrow c = y_0 - m \cdot x_0$$

Example of a simpler approach to calculate each pixel in a line →



↳ we will increase the step size of x by one (+1) and calculate the value of y across it.

(1) $$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{5-2}{7-2} = 3/5$$

(2) The value of c:

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

$$\Rightarrow c = 2 - 3/5 \cdot 2 = 4/5$$

(3) Now calculate y for each value of x:

(i) $y(2) = \underline{2}$

(ii) $y(3) = (3/5 \cdot 3) + 4/5 = 13/5 = \underline{2.6}$

(iii) $y(4) = ((3/5 \cdot 4) + 4/5) = 16/5 = \underline{3.2}$

(iv) $y(5) = (3/5 \cdot 5) + 4/5 = 19/5 = \underline{3.8}$

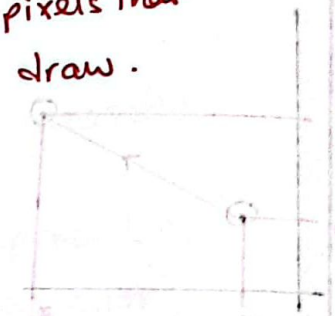
(v) $y(6) = (3/5 \cdot 6) + 4/5 = 22/5 = \underline{4.4}$

(vi) $y(7) = (3/5 \cdot 7) + 4/5 = 5 = \underline{5}$

So,

	Pixel (x, y)
→ $y(2) = 2$	$(2, 2)$
→ $y(3) = 2.6 \approx 3$	$(3, 3)$
→ $y(4) = 3.2 \approx 3$	$(4, 3)$
→ $y(5) = 3.8 \approx 4$	$(5, 4)$
→ $y(6) = 4.4 \approx 4$	$(6, 4)$
→ $y(7) = 5$	$(7, 5)$

These are the pixels that we draw.

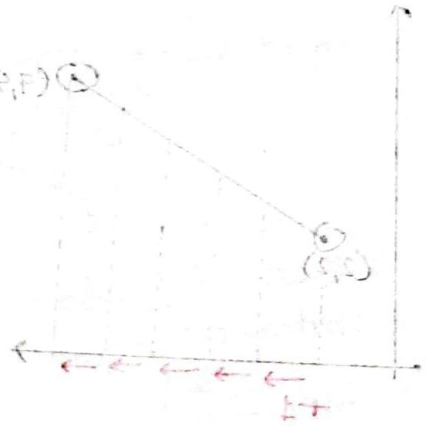


While this might be a very simple approach, the algorithm is way too slow since:

→ The equation $y = mx + b$ / $y = mx + c$, requires the multiplication of m and x in every step

→ we also need to Round off the resulting y coordinates.

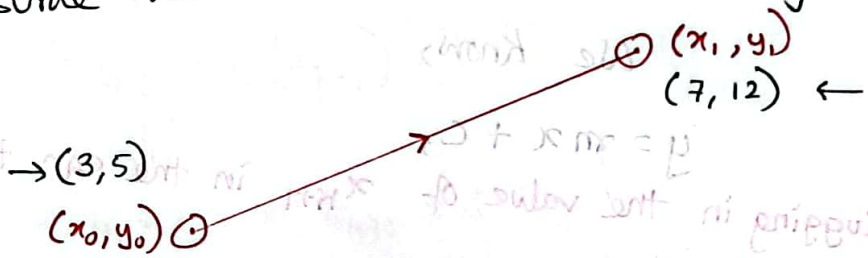
∅ We'll need a faster approach.



DDA (Digital Differential Algorithm)

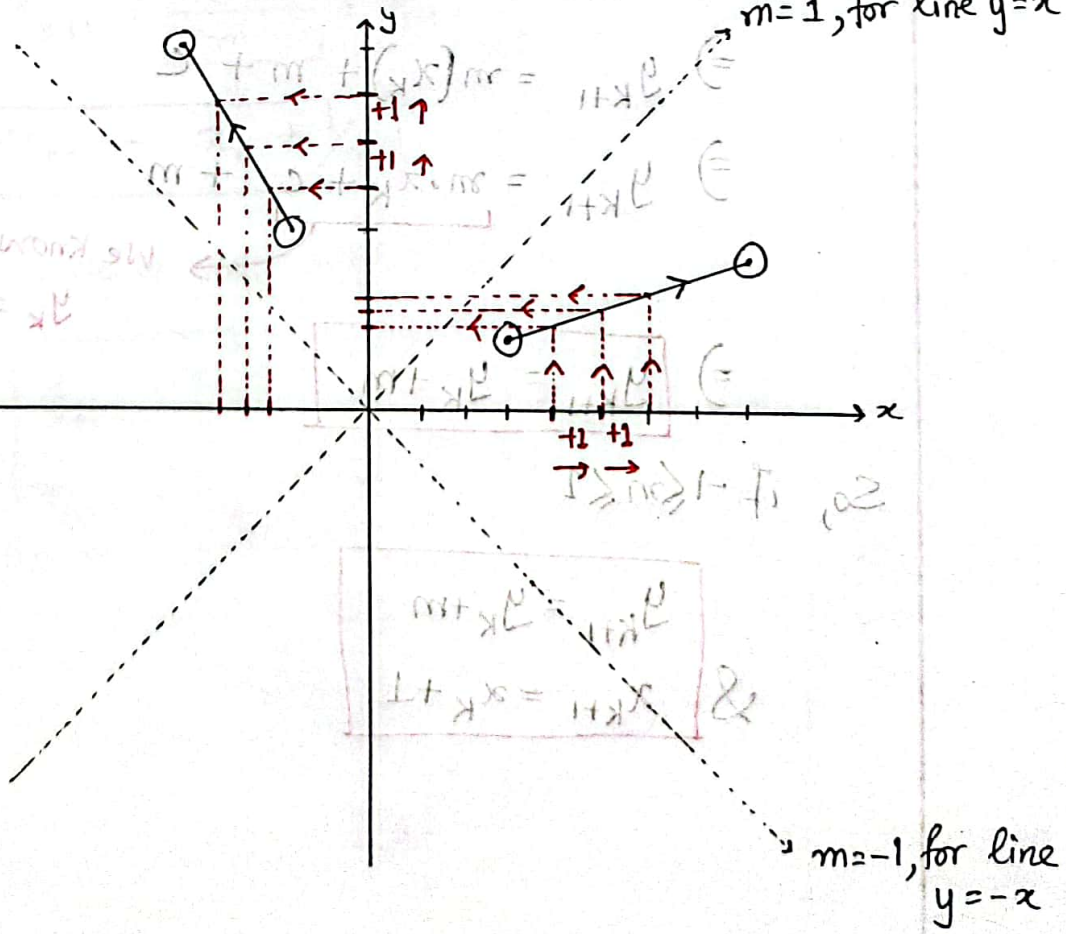
The DDA Algorithm is an incremental approach in order to speed up scan conversion. Simply calculate y_{k+1} based y_k and the deciding factor here is the gradient (m).

Let assume we want to draw the following line:



First, $m = \frac{(y_2 - y_1)}{(x_2 - x_1)}$ OR $\frac{y_1 - y_0}{x_1 - x_0}$

So, $m = \frac{12 - 5}{7 - 3} = 7/4$



Ø There are two conditions to DDA

1) if gradient m is $(-1 \leq m \leq 1)$ → increment of x will be by 1

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

x_{k+1} → the next value of x

x_k → the current value of x

We know,

$$y = mx + c,$$

□ plugging in the value of x_{k+1} in the eqn to get y_{k+1}

$$y_{k+1} = m(x_{k+1}) + c$$

$$\Rightarrow y_{k+1} = m(x_k + 1) + c$$

$$\Rightarrow y_{k+1} = m(x_k) + m + c$$

$$\Rightarrow y_{k+1} = m \cdot x_k + c + m$$

We know, $y_k = mx_k + c$

$$\Rightarrow \boxed{y_{k+1} = y_k + m}$$

So, if $-1 \leq m \leq 1$

$$\boxed{\begin{aligned} y_{k+1} &= y_k + m \\ x_{k+1} &= x_k + 1 \end{aligned}}$$ &

2) If m lies outside the range $\rightarrow$ increment of y will be by 1.

So, $y_{K+1} = y_K + 1$

&

$y = mx + c \longrightarrow$ also, $x = \frac{y-c}{m}$

$\Rightarrow y_{K+1} = m(x_{K+1}) + c$

$\Rightarrow y_K + 1 = m \cdot x_{K+1} + c$

$\Rightarrow y_K - c + 1 = m \cdot x_{K+1}$

$\Rightarrow x_{K+1} = \frac{y_K - c + 1}{m}$

$\Rightarrow x_{K+1} = \frac{y_K - c}{m} + 1/m$

$\Rightarrow \boxed{x_{K+1} = x_K + 1/m}$

So, for m outside the range:

$$\boxed{y_{K+1} = y_K + 1}$$

&

$$\boxed{x_{K+1} = x_K + 1/m}$$

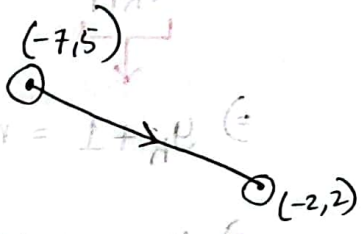
So,

$-1 \leq m \leq 1$	otherwise
$x_{k+1} = x_k + 1$	$x_{k+1} = x_k + 1/m$
$y_{k+1} = y_k + m$	$y_{k+1} = y_k + 1$

X

Q1 Draw a line using DDA for:

$P_1(-7,5)$ $P_2(-2,2)$



$$m = \frac{2-5}{-2+7} = \frac{-3}{5} = -0.6$$

Since m is in the range: $(-1 \leq m \leq 1)$

the value of
- x increases
- y decreases

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

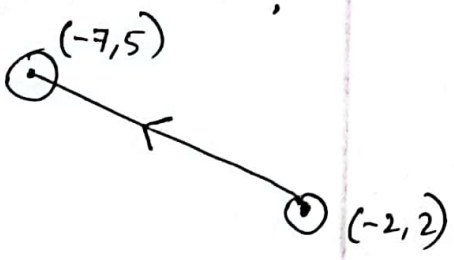
→ since m is inherently negative we don't need to use $(y_k - m)$

x(+1)	y(+m)	y(round off)	PIXEL
-7	5	—	(-7,5)
-6	4.4	≈ 4	(-6,4)
-5	3.8	4	(-5,4)
-4	3.2	3	(-4,3)
-3	2.6	3	(-3,3)
-2	2	2	(-2,2)

(Q2) What if the line was in the opposite direction?

$$m = \frac{5-2}{-7+2} = \frac{3}{-5} = \boxed{-0.6}$$

in range $(-1 \leq m \leq 1)$



x_k (-1)	y_k (-m)	y_k (round off)	PIXEL
-2	2	—	(-2, 2)
-3	2.6	3	(-3, 3)
-4	3.2	3	(-4, 3)
-5	3.8	4	(-5, 4)
-6	4.4	4	(-6, 4)
-7	5	5	(-7, 5)

x decreases
y increases

since x decreases we will use $x_{K+1} = x_K - 1$ instead

since y increase and the value of m is negative we will use $y_{K+1} = y_K - m$ instead.

Pseudo Code for DDA

```
DDA (x0, y0, x1, y1) {
    m = (y1 - y0) / (x1 - x0) ;
    if ( m <= 1 && m >= -1 ) {
        while ( x0 <= x1 ) {
            x0 = x0 + 1
            y0 = y0 + m
            draw (x0, y0)
        }
    }
(*) else {
        while ( y0 <= y1 ) {
            x0 = x0 + (1/m)
            y0 = y0 + 1
            draw (x0, y0)
        }
    }
```