# Primitif Grafis

- Primitif Grafis dapat diartikan sebagai sebuah bentuk dasar dari objek grafis yang dapat dibentuk
- Primitif grafis yang menjadi dasar bagi semua objek grafis adalah titik (DOT)

# PRIMITIF GRAFIS

- Pixel (Dot) ➔ Posisi (x,y), Warna
- Garis (Line) ➔ Posisi (x1,y1,x2,y2), Warna, Thickness, Pattern
- Lingkaran (Circle) ➔ Pusat(x,y), Radius, Warna, Thickness, Pattern
- Ellipse ➔ Pusat(x,y), Radius Horisonal/Vertikal, Warna, Thickness, Pattern
- Kurva ➔ Teratur/Tidak teratur (Bezier)
- Character ➔ Type, Slanted, Thickness, Color, dll

# Titik (Pixel/Dot)

- Pustaka Grafis (Library)
  BGI ➔ putpixel (x,y,color)
  OPENGL ➔ GL_Drawdot(x,y,color)

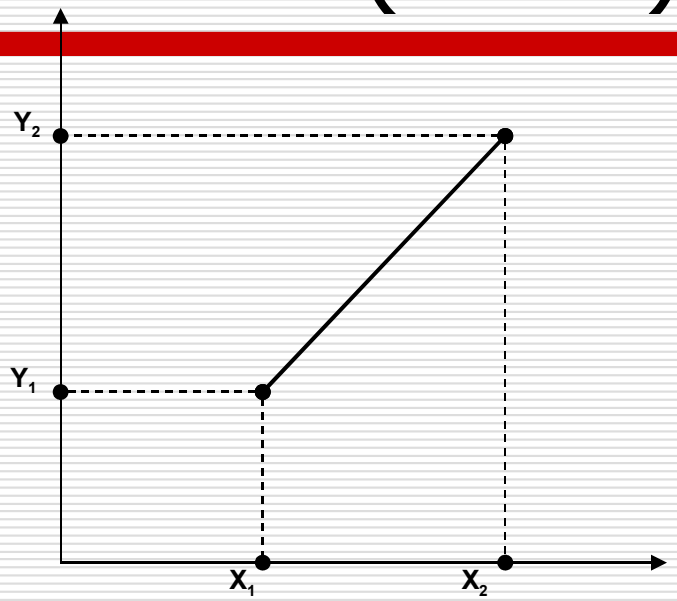
- Low Level Programming
  - Interupt ➔ H/W, S/W
  - DMA (Direct Memory Access)

    Modus Text : B800:0000h
    Modus Grafis : A000:0000h

# Titik (Pixel/Dot)

```
void setpixel_DMA (x,y,color)
{
    char *Segmen = 0xA0000000;
    (Segmen+(800*y+x)) = color;
}
procedure setpixel_DMA(x,y,color);
begin
    mem[A000:(800*y+x)] := color;
end;
```

# GARIS (LINE)




$$\frac{y - y_1}{y_2 - y_1} = \frac{x - x_1}{x_2 - x_1}$$

$$y = m.x + c$$

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

# Algoritma Pembuatan Garis

- Algoritma pembuatan garis lurus vertikal dan horisontal relatif mudah, tetapi bila garis tersebut miring, maka algoritma menjadi sulit.
- Misal : Line (1,3,8,5)

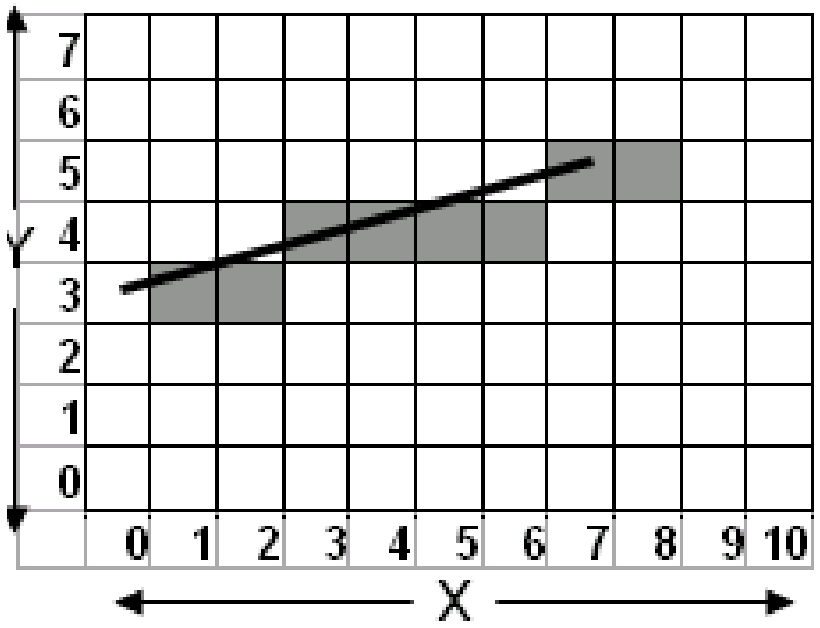
$$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{5 - 3}{8 - 1} = \frac{2}{7}$$

$$c = y_1 - m.x_1 = 3 - \frac{2}{7}.1 = \frac{19}{7}$$

$$y = \frac{2}{7}.x + \frac{19}{7}$$

# Algoritma Pembuatan Garis

| Xi | Yi | Yi (dibulatkan) |
|---|---|---|
| 1 | 3.00 | 3 |
| 2 | 3.29 | 3 |
| 3 | 3.57 | 4 |
| 4 | 3.86 | 4 |
| 5 | 4.14 | 4 |
| 6 | 4.43 | 4 |
| 7 | 4.71 | 5 |
| 8 | 5.00 | 5 |



# Analisa Algoritma

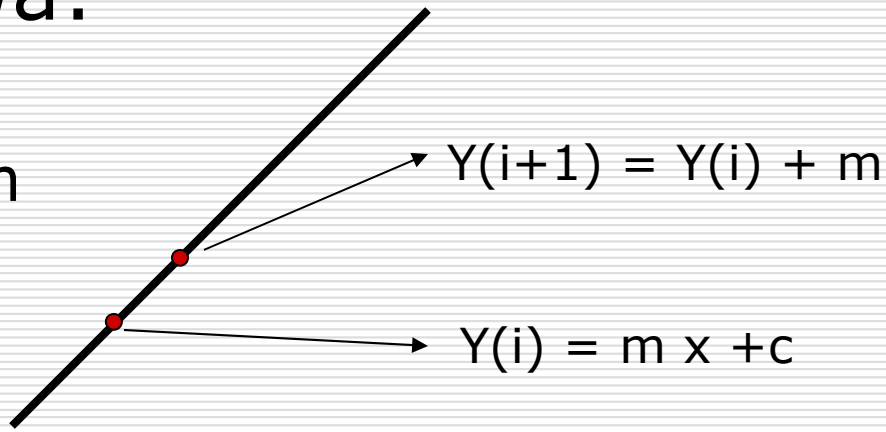
- Bentuk umum persamaan garis : y=mx+c digunakan apabila nilai m berada pada rentang 0 s/d 1 {artinya bahwa delta x lebih besar daripada delta y} sedangkan apabila m > 1 maka digunakan persamaan x=(y-c)/m. Mengapa???

# DDA (Digital Differential Analyser) Algorithm

- DDA merupakan pengembangan dari algoritma persamaan garis sebelumnya (persamaan matematis)
- Ide dasar dari DDA adalah menggunakan nilai pemetaan yang telah didapat (bisa x atau y) untuk menghitung nilai selanjutnya untuk mempercepat proses komputasi

# DDA (Digital Differential Analyser) Algorithm

- Dari persamaan y=mx+c maka bisa disimpulkan bahwa:
  - Y(i) = m x +c
  - Y(i+1) = Y(i) + m




- Nilai pemetaan selanjutnya setara dengan kenaikan nilai m

# DDA (Digital Differential Analyser) Algorithm

$$y = m.x + c$$

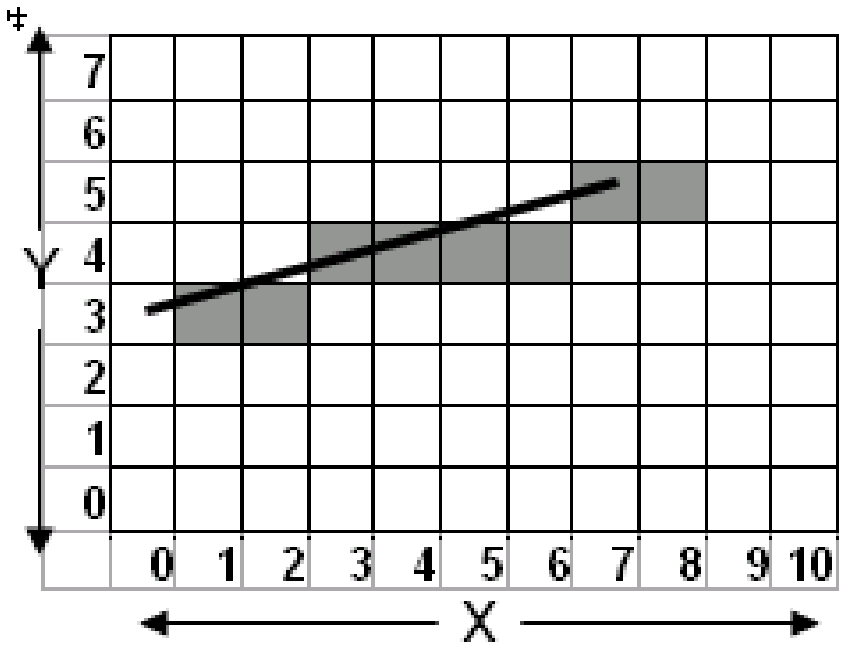
$$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{\Delta y}{\Delta x}$$

- Jika $0<m<1$ ➔ $y_{k+1} = y_k + m$
  $x_{k+1} = x_k + 1$
- Jika $m>1$ ➔ $x_{k+1} = x_k + 1/m$
  $y_{k+1} = y_k + 1$

# Proses DDA

Dengan mengambil contoh soal yang sama pada persamaan garis secara Matematis, maka proses DDA dapat digambarkan sebagai berikut

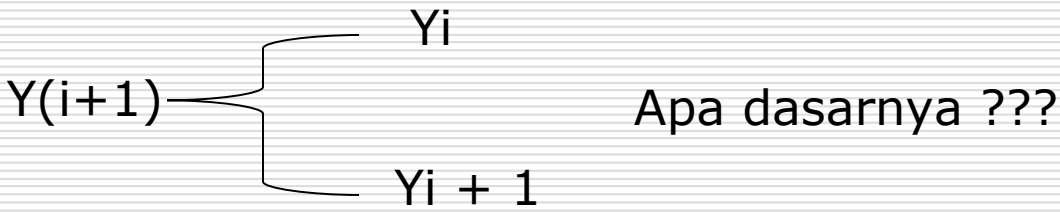
| Xi | Yi | Yi (dibulatkan) |
|---|---|---|
| 1 | 3.00 | 3 |
| 2 | 3.29 | 3 |
| 3 | 3.57 | 4 |
| 4 | 3.86 | 4 |
| 5 | 4.14 | 4 |
| 6 | 4.43 | 4 |
| 7 | 4.71 | 5 |
| 8 | 5.00 | 5 |




Perubahan nilai Yi setara dengan gradien m

# Bresenham's Algorithm

- Dengan mengamati algoritma pembuatan garis sebelumnya maka dapat kita katakan bahwa :
  - proses perhitungan yang terjadi seolah-olah menyatakan bahwa nilai pemetaan berikutnya sama dengan pemetaan sebelumya atau berubah (Ada pedoman perubahan)



# Bresenham's Algorithm

- Setelah melakukan penelitian yang serius, maka seorang pakar menemukan suatu metode dalam pembuatan garis yang memiliki kecepatan komputasi yang lebih cepat dibandingkan dengan metode sebelumnya => Bresenham's Algorithm (Mengapa lebih cepat ??)
- Algoritma Bresenham's melakukan proses pemetaan berdasarkan sebuah decision parameter *p*

# Bresenham's Algorithm

$$P_0 = 2\Delta y - \Delta x$$

Plot (Xo,Yo)
Repeat
    If Pk<0 then
        Plot ($X_{k+1}$,$Y_k$)
        $P_{k+1} = P_k + 2\Delta y$
    Otherwise
        Plot ($X_{k+1}$,$Y_{k+1}$)
        $P_{k+1} = P_k + 2\Delta y - 2\Delta x$
Until X=end

# Bresenham's Algorithm

- Contoh : Line (20,10,30,18)

  $\Delta X = 10, \Delta Y = 8$

  $Po = 2\Delta Y - \Delta X = 6$

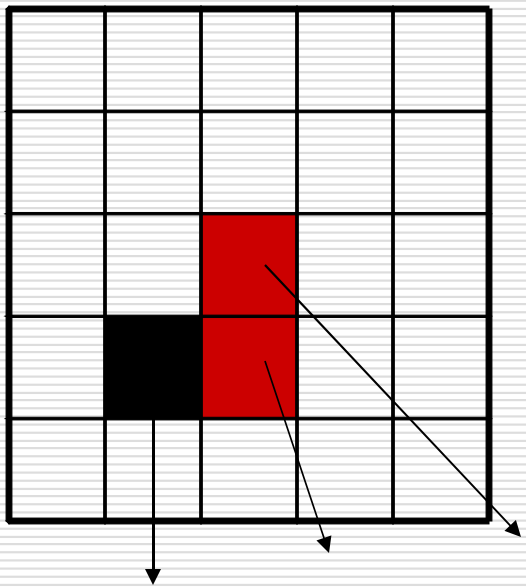
  $2.\Delta Y = 16$

  $2\Delta Y - 2\Delta X = -4$

| K | $P_k$ | $(X_{k+1},Y_{k+1})$ | K | $P_k$ | $(X_{k+1},Y_{k+1})$ |
|---|---|---|---|---|---|
| 0 | 6 | (21,11) | 5 | 6 | (26,15) |
| 1 | 2 | (22,12) | 6 | 2 | (27,16) |
| 2 | -2 | (23,12) | 7 | -2 | (28,16) |
| 3 | 14 | (24,13) | 8 | 14 | (29,17) |
| 4 | 10 | (25,14) | 9 | 10 | (30,18) |

# Bresenham's Algorithm

- Ilustrasi penentuan parameter



Kita hitung jarak (d) antar titik y rumus dengan y yang mungkin:

$y = m ( X_{k+1}) + c$

$d1 = y - y_k = m (x_{k+1}) + c - y_k$

$d2 = y_{k+1} - y = y_{k+1} - m (x_{k+1}) - c$

$d1 - d2 = 2m(x_{k+1}) - 2y_k + 2c - 1$

Penyederhanaan:

$pk = \Delta x(d1-d2) = 2\Delta y.x_k - 2\Delta x.y_k + Z$

Bila pk <0 maka pilih pemetaan d1

Bila pk >=0 maka pilih pemetaan d2

$x_{k+1}, y_k$ $x_{k+1}, y_{k+1}$

$x_k$,yk setelah ini ada 2 kemungkinan yaitu xk+1,yk+1 atau xk+1,yk

$2\Delta y + \Delta x(2c-1)$

# Bresenham's Algorithm

- Nilai pk berikutnya dapat dihitung dengan cara:

  $p_k = 2\Delta y.x_k - 2\Delta x.y_k + Z$

  $p_{k+1} = 2\Delta y.(x_{k+1}) - 2\Delta x.y_{k+1} + Z$

Disederhanakan

$p_{k+1} - p_k = 2\Delta y.(x_{k+1} - x_k) - 2\Delta x.(y_{k+1} - y_k)$

Kita tahu bahwa $x_{k+1} = x_k + 1$

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

Dimana nilai $(y_{k+1} - y_k)$ bisa 0 atau 1tergantung pada nilai $p_k$