



GRAFIKA KOMPUTER I

PRIMITIF-PRIMITIF KELUARAN GRAFIKA RASTER

PENDAHULUAN

- Pada sistem grafika random-scan primitif-primitif grafika dibuat langsung oleh display driver pada piranti peraga.
- Bila berupa plotter, maka parameteranya garis (misalnya kedua koordinat titik ujungnya diterjemahkan menjadi kecepatan dan arah gerakan mekanis pena plotter, vertikal dan horisontal).

PENDAHULUAN

- Bila berupa *random-scan cathode ray tube* (CRT), maka akan terjadi perubahan pada deflektor.
- Pada sistem raster-scan perlu ada satu tahap lagi untuk memetakan primitif tersebut pada suatu matriks pixel.
- Tahap tersebut dikenal sebagai proses scan-conversion.
- Jadi pertanyaannya adalah Bagaimana memberi harga elemen-elemen matriks tersebut sehingga penampakkannya membentuk primitif-primitif yang diharapkan.

PRIMITIF GRAFIKA

- Pada berbagai sistem grafika tingkat primitivitas grafis berbeda-beda.
- Bagaimana kompleksitas dari konversi-konversi raster-scan sehingga terbentuk obyek primitif titik, garis, lingkaran, persegi, dll. pada matriks peragaan sistem grafika.

KRITERIA ALGORITMA

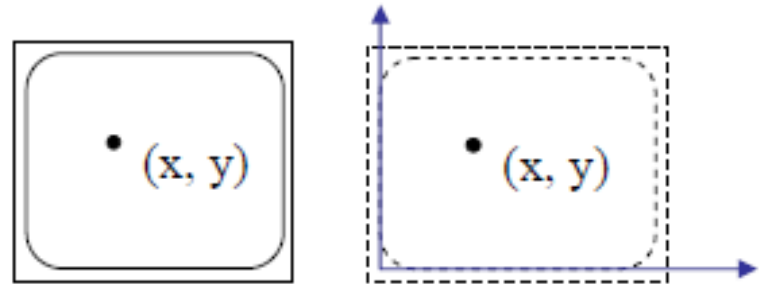
- Suatu penggambaran grafika komputer bisa jadi terdiri dari ribuan bahkan jutaan primitif-primitif grafika tsb.
- Dengan adanya peningkatan efisiensi dalam pembuatan setiap primitif tersebut maka secara proporsional akanmereduksi waktu komputasi secara keseluruhan penggambaran.

KRITERIA ALGORITMA

- Efisiensi tersebut pada umumnya dilakukan dengan sedapat mungkin :
 - Mengurangi penggunaan operasi aritmatik perkalian/penjumlahan.
 - Mengurangi penggunaan komputasi *floating point*.
 - Memanfaatkan koherensi komputasi sebelumnya secara menaik/menurun.
 - Pemodelan aljabar secara langsung.

PRIMITIF GARIS

- Garis adalah kumpulan titik-titik yang tersusun sedemiki-an rupa sehingga memiliki pangkal dan ujung.
- Suatu titik pada layar terletak pada posisi (x,y) , untuk menggambarkannya plot suatu pixel dengan posisi yang berkesesuaian.
- Contoh program :
Setpixel (x,y)



PRIMITIF GARIS

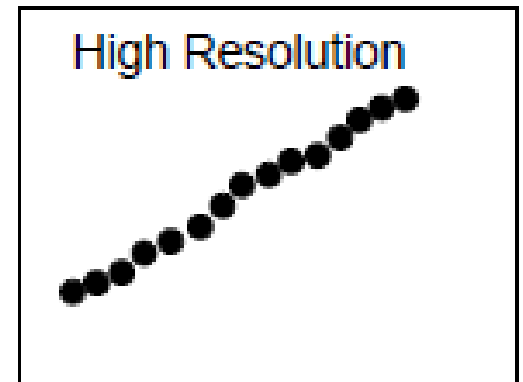
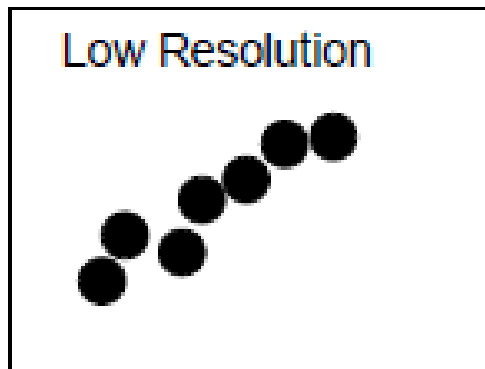
- Penampilan garis pada layar komputer dibedakan berdasarkan Resolusi-nya.
- Resolusi : keadaan pixel yang terdapat pada suatu area tertentu

PRIMITIF GARIS

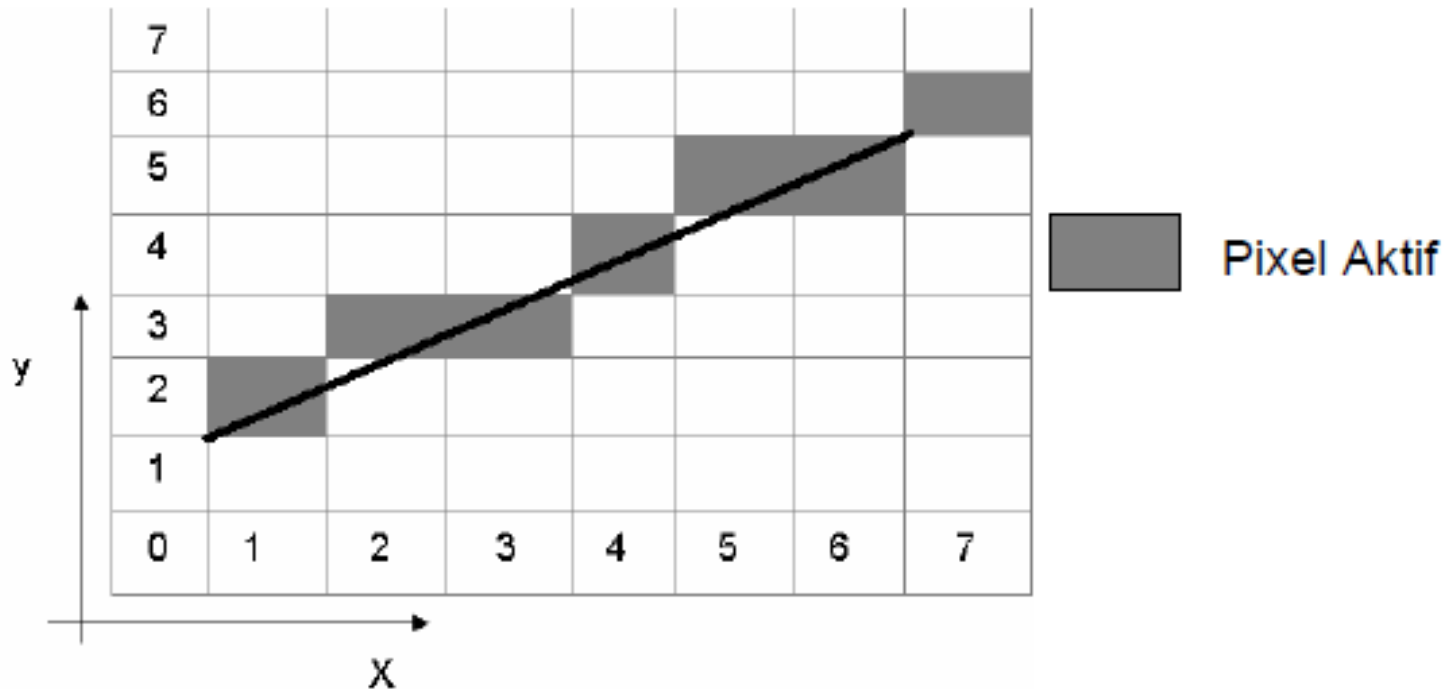
- Contoh :

Resolusi 640x480, berarti pada layar kompuer terdapat 640 pixel per-kolom dan 480 pixel per-baris.

- Resolusi dapat pula dibedakan menjadi kasar, medium dan halus.

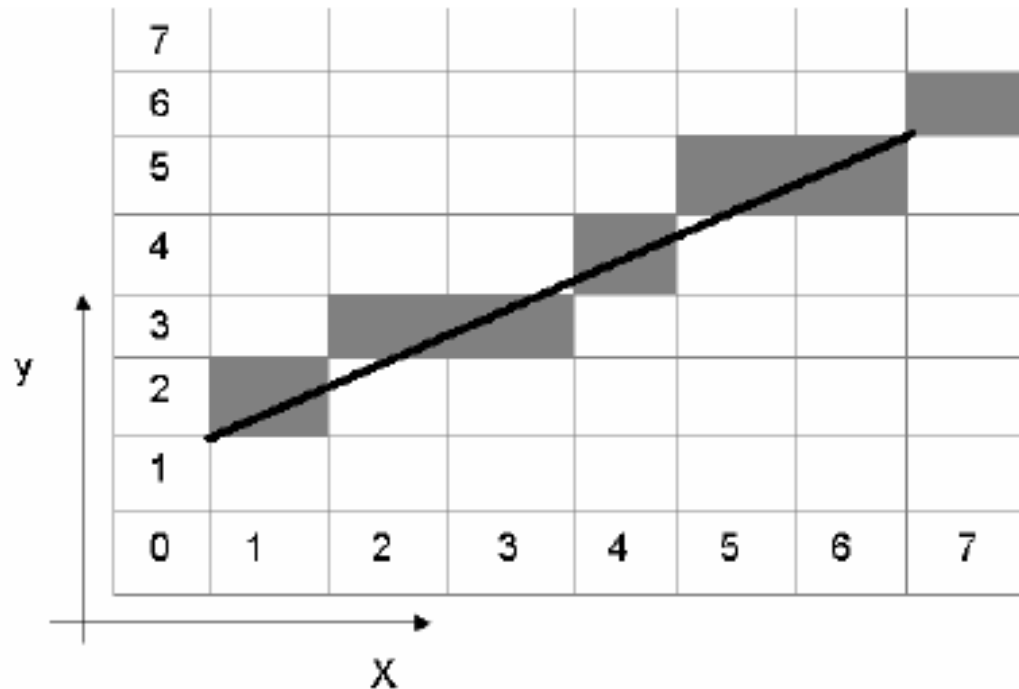



PRIMITIF GARIS



- Untuk menggambarkan garis seperti gambar di atas, diperlukan pixel aktif.
- Parameter pixel address yang membentuk garis pada layar adalah :

PRIMITIF GARIS



 Pixel Aktif

Pixel	X	Y
1	1	2
2	2	3
3	3	3
4	4	4
5	5	5
6	6	5
7	7	6

PRIMITIF GARIS

- Untuk menampilkan atau menggambarkan garis pada layar dibutuhkan minimal 2 titik (endpoint), yaitu titik awal dan akhir.
 - Awal garis dimulai dengan titik atau pixel pertama, P_1 diikuti titik kedua, P_2 .
 - Untuk mendapatkan titik-titik selanjutnya sampai ke P_n perlu dilakukan inkrementasi atas nilai koordinat sumbu X dan Y pada titik sebelumnya.
 - Perhitungan inkrementasi untuk masing-masing sumbu adalah berbeda :

PRIMITIF GARIS

Jenis	Sumbu-X	Sumbu-Y
Horizontal	Gerak ($X=X+1$)	Konstan
Vertikal	Konstan	Gerak ($Y=Y+1$)
Diagonal	Gerak ($X=X+1$)	Gerak ($Y=Y+1$)
Bebas	Gerak ($X=X+n$)	Gerak ($Y=Y+n$)

n dan m adalah nilai inkrementasi

PRIMITIF GARIS

- Persamaan Umum Garis : $y = mx + c$



Garis Horizontal

- Garis yang membentang secara paralel dengan sumbu X dengan asumsi titik P1 pada koordinat X1 lebih kecil daripada X2 dari P2, sedangkan Y1 dan Y2 konstant
- Algoritma:
 1. Menentukan titik awal (P1) dan titik akhir (P2)
 2. Periksa posisi sumbu (koordinat)
 - Jika titik akhir < titik awal, Lakukan inkrementasi sumbu X dari titik awal sampai titik akhir
 - Jika tidak, maka Lakukan dekrementasi sumbu X dari titik awal sampai titik akhir
 3. Tampilkan garis menggunakan parameter koordinat yang telah dihitung.

Garis Vertikal

- Garis yang membentang secara paralel dengan sumbu Y dengan asumsi titik P1 pada koordinat Y1 lebih kecil daripada Y2 dari P2, sedangkan X1 dan X2 konstant
- Algoritma:
 1. Menentukan titik awal (P1) dan titik akhir (P2)
 2. Periksa posisi sumbu (koordinat)
 - Jika titik akhir $<$ titik awal, Lakukan inkrementasi sumbu Y dari titik awal sampai titik akhir
 - Jika tidak, maka Lakukan dekrementasi sumbu Y dari titik awal sampai titik akhir
 3. Tampilkan garis menggunakan parameter koordinat yang telah dihitung.

Garis Diagonal

- Garis yang membentang secara paralel 45 derajat dari sumbu X atau sumbu Y dengan asumsi titik awal P1 dengan koordinat X1 dan Y1 lebih kecil daripada X2 dan Y2 atau sebaliknya.
- Algoritma :
 1. Menentukan titik awal (P1) dan titik akhir (P2)
 2. Periksa posisi sumbu (koordinat)
 - Jika titik akhir $<$ titik awal, Lakukan inkrementasi sumbu X dan sumbu Y dari titik awal sampai titik akhir
 - Jika tidak, maka Lakukan dekrementasi sumbu X dan sumbu Y dari titik awal sampai titik akhir
 3. Tampilkan garis menggunakan parameter koordinat yang telah dihitung.

Garis Bebas

- Garis yang membentang antara 2 titik, P1 dan P2, selalu membentuk sudut yang besarnya sangat bervariasi.
- Sudut yang terbentuk menentukan kemiringan suatu garis atau disebut *gradient/slop* atau disimbolkan dengan parameter *m*.
- Jika titik-titik yang membentuk garis adalah : $(x1, y1)$ dan $(x2, y2)$ maka $m = \Delta Y / \Delta X$ atau $m = (y2 - y1) / (x2 - x1)$

Algoritma Bresenham

Algoritma Bresenham untuk $|m| < 1$:

1. Input 2 endpoints, simpan endpoints kiri sebagai (x_0, y_0) .
2. Panggil frame buffer (plot titik pertama)
3. Hitung konstanta Δx , Δy , $2\Delta y$, $2\Delta y - 2\Delta x$ dan nilai awal parameter keputusan $p_0 = 2\Delta y - \Delta x$
4. Pada setiap x_k di garis, dimulai dari $k = 0$, ujilah :
 - Jika $p_k < 0$, maka $\text{plot}(x_{k+1}, y_k)$ dan $p_{k+1} = p_k + 2\Delta y$
selain itu maka $\text{plot}(x_{k+1}, y_{k+1})$ dan $p_{k+1} = p_k + 2\Delta y - 2\Delta x$
5. Ulangi tahap 4 sebanyak Δx kali

Algoritma Bresenhem

- Contoh soal :

Hitunglah posisi piksel hingga membentuk sebuah garis yang menghubungkan titik (12,10) dan (17,14) !

- Jawab :

$$1. (x_0, y_0) = (12, 10)$$

$$2. \Delta x = 5, \Delta y = 4, 2\Delta y = 8, 2\Delta y - 2\Delta x = -2$$

$$3. p_0 = 2\Delta y - \Delta x = 3$$

$$\Delta x = 5,$$

$$\Delta y = 4,$$

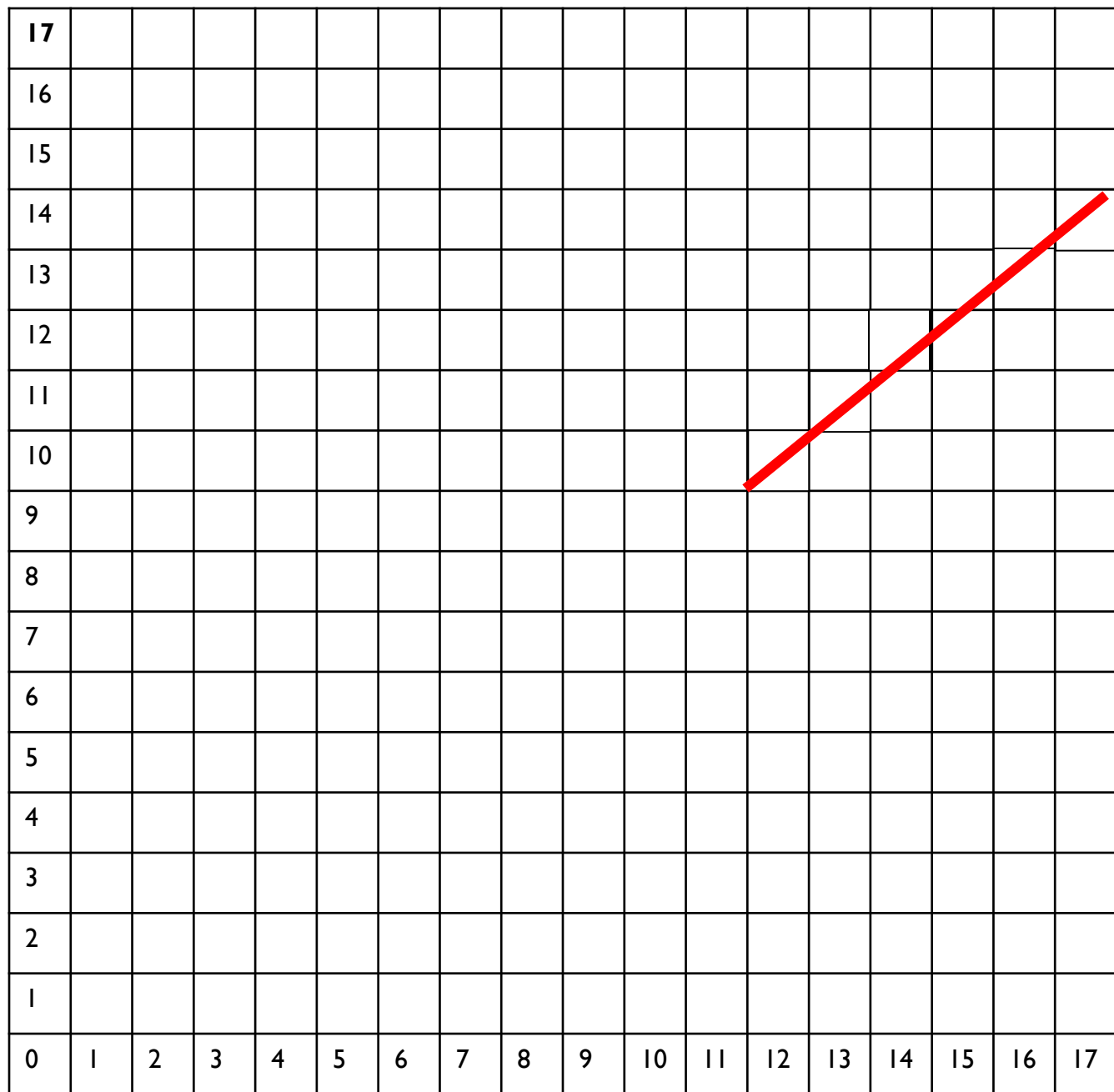
$$2\Delta y - 2\Delta x = -2$$

$$p_0 = 2\Delta y - \Delta x = 3$$

- **Jika** $p_k < 0$, maka $\text{plot}(x_{k+1}, y_k)$ dan $p_{k+1} = p_k + 2\Delta y$
selain itu maka $\text{plot}(x_{k+1}, y_{k+1})$ dan

$$p_{k+1} = p_k + 2\Delta y - 2\Delta x$$

k	p_k	(x_{k+1}, y_{k+1})



Algoritma Bresenhem

- **Soal Tugas 1:**

Hitunglah posisi piksel hingga membentuk sebuah garis yang menghubungkan titik (14,11) dan (20,15) !

- Jawab :

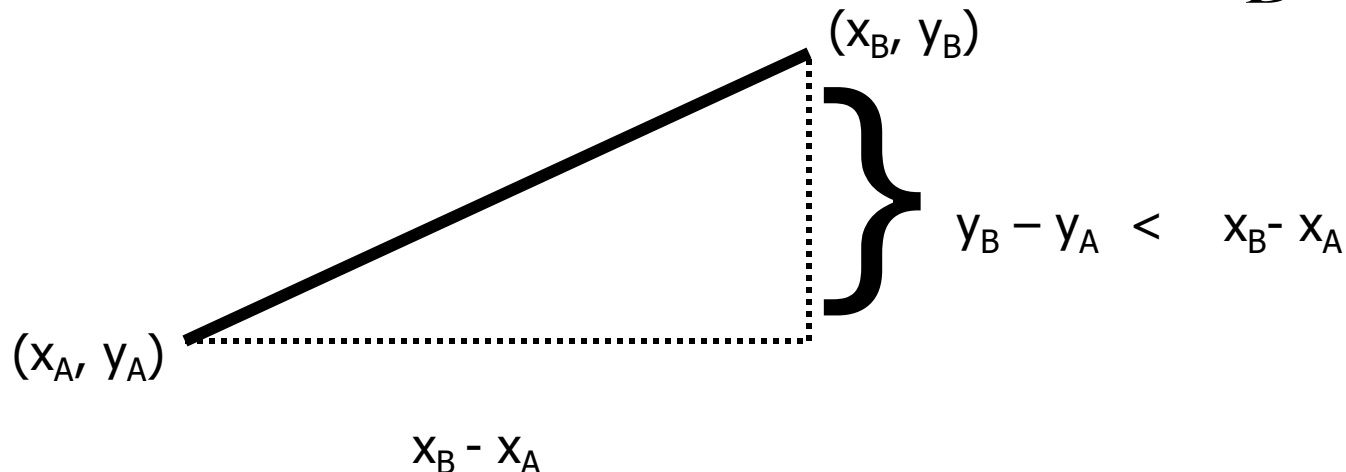
Tugas perorangan (pribadi).

**Dikumpulkan paling lambat sabtu
12 Oktober 2024 saat awal kuliah**

PRIMITIF GARIS

- Domain pembahasan pada ruang sub-kuadran $(0, \pi/4)$
- $x_A < x_B$ dgn gradien
- $0 < m < 1$

$$m = \frac{y_B - y_A}{x_B - x_A}$$



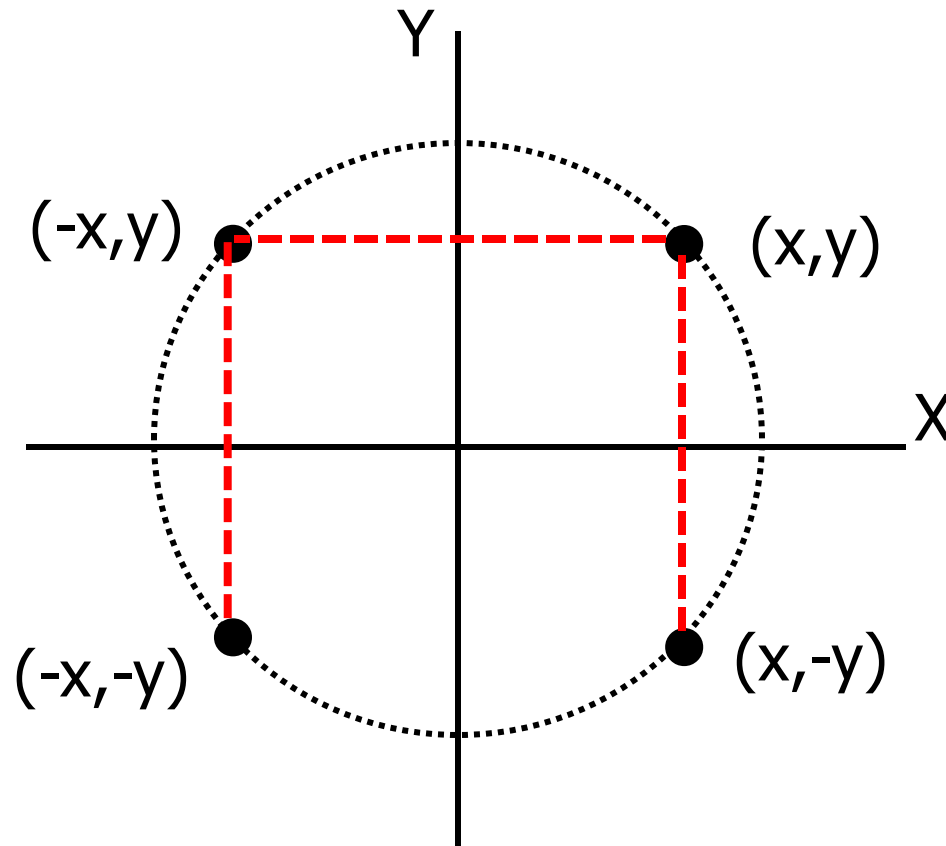
Algoritma Midpoint untuk Garis

```
MidpointLine(int xA, int yA, int xB, int yB, v)
{ Int Dx, Dy, d, incE, incNE, xi, yi;
  Dx=xB-xA; Dy=yB-yA; d=2*Dy-Dx;
  incE=2*Dy; incNE=2*(Dy-Dx); Xi=xA; yi=yA;
  writepixel(xi, yi, v)
  while (xi<=xB)
  { if (d <= 0)
      { d=d+incE;}
    else
      { d=d+incNE; yi++; }
    xi++;
    writepixel(xi, yi, v)
  }
}
```

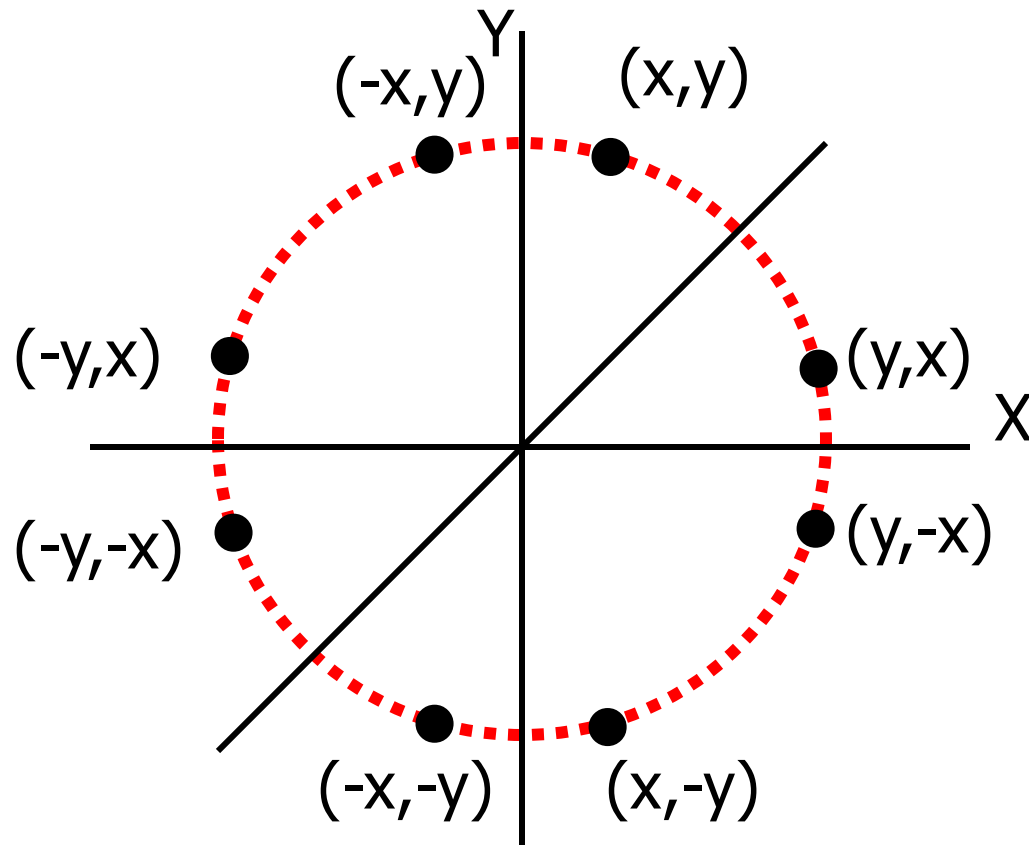
PRIMITIF LINGKARAN

- Sementara diasumsikan penggambaran lingkaran berpusat di titik $O(0,0)$.
- Karena sifat simetri 8 arah, penghitungannya cukup dilakukan pada $1/8$ lingkaran, lalu diaplikasikan pada ketujuh titik lainnya.
- Selanjutnya, bergerak searah jarum jam (*clock-wise*) dari $\pi/2$ ke $\pi/4$.

PRIMITIF LINGKARAN



PRIMITIF LINGKARAN



PRIMITIF LINGKARAN

- Algoritma *Circle Points* :

```
CirclePoints(int x, int y, int v) {  
    writePixel (x,y,v); writePixel (y,x,v);  
    writePixel (-x,y,v); writePixel (-y,x,v);  
    writePixel (x,-y,v); writePixel (y,-x,v);  
    writePixel (-x,-y,v); writePixel (-y,-x,v);  
}
```

PRIMITIF LINGKARAN

- Secara sederhana kita menghitung (x,y) dari persamaan $y^2 = r^2 - x^2$, mulai dari $\pi/2$ ke $\pi/4$ maka nilai x menaik satu demi satu, sementara nilai y berubah antara 0 dan -1.
- Secara sederhana pula, kita menghitung nilai y dari fungsi $y = (r^2 - x^2)^{1/2}$.

ALGORITMA MIDPOINT UNTUK LINGKARAN

```
MidpointCircle(int r, int v) {
    int Dx, Dy, d, incNE, incE, xi, yi;
    d = 1-r; incE = 3; incNE = -2*r+5
    Circlepoints(xi,yi,v);
    while (y>x) {
        If (d<=0) {
            d = d +incE; incE += 2; incNE += 2; }
        Else {
            d=d+incNE; incE +=2; incNE +=4; yi--; }
        Xi++;
        Circlepoints(xi,yi,v);
    }
}
```

ALGORITMA MIDPOINT UNTUK LINGKARAN

- Bagaimana jika lingkaran berpusat di suatu titik bukan $O(0,0)$, misalnya di titik $P(x_1, y_1)$?
- Dapat dilakukan dengan melakukan modifikasi algoritma Circle Points dan pemanggilnya

ALGORITMA MIDPOINT UNTUK LINGKARAN

- Modifikasi Algoritma *Circle Points* :

```
CirclePoints(int x, int y, int x1, int y1,  
            int v) {  
    writePixel (x1+x,y1+y,v); writePixel (x1+y,y1+x,v);  
    writePixel (x1-x,y1+y,v); writePixel (x1-y,y1+x,v);  
    writePixel (x1+x,y1-y,v); writePixel (x1+y,y1-x,v);  
    writePixel (x1-x,y1-y,v); writePixel (x1-y,y1-x,v);  
}
```

PRIMITIF ELLIPS

- Suatu ellips dengan sumbu mayor $2a$ (arah sumbu X) dan sumbu minor $2b$ (arah sumbu Y) adalah yang memenuhi persamaan : $x^2b^2 + y^2a^2 = a^2b^2$.
- Tidak seperti lingkaran, ellips hanya dapat dibagi ke dalam empat ruang simetris saja.

PRIMITIF ELLIPS

- Selain itu dalam satu kuadran, subkuadran tidak terbagi dengan sudut yang sama dan harus dihitung masing-masing.

Algoritma *Flood-Fill*

```
void FloodFill(int x, int y) {  
    if (pixel[x,y] != fillcolor &&  
        pixel[x,y] != boundarycolor) {  
        SetPixel(x,y,fillcolor);  
        FloodFill(x+1,y);  
        FloodFill(x-1,y);  
        FloodFill(x+1,y+1);  
        FloodFill(x,y+1);  
        FloodFill(x-1,y+1);  
        FloodFill(x+1,y-1);  
        FloodFill(x,y-1);  
        FloodFill(x-1,y-1);  
    }  
}
```

Algoritma *Flood-Fill*

- Algoritma tersebut dapat digunakan jika diketahui suatu titik di dalam poligon sebagai titik awal *Flood-Fill*.
- Dalam grafika interaktif (misalkan PaintBrush), algoritma tersebut sering digunakan.
- Algoritma rekursif pada algoritma tersebut dapat menghabiskan sistem memori (*stack*) komputer → kinerja rendah.

Algoritma *scan-line*

- Bila diketahui ada suatu piksel dalam area poligon maka piksel-piksel lain dapat dengan mudah dicapai dari piksel tersebut dengan suatu algoritma rekursif.
- Namun secara umum, hal itu tidak selalu bisa dilakukan.
- Untuk mengatasinya digunakan algoritma *scan-line*.

Algoritma *scan-line*

- Ide dasarnya :
 1. Scan secara horisontal dari kiri ke kanan (menurut nilai absis),
 2. Beri warna piksel-piksel di antara dua pasang urutan ganjil-genap titik potong tersebut.
 3. Ulangi 1-2, mulai dari nilai ordinat titik terkecil s/d ordinat titik terbesar.

Algoritma *scan-line*

- **Kerangka Algoritmanya :**

```
Mencari ordinat min dan max (ymin dan ymax)
//inisialisasi ymin dan ymax
.....
For (yi=ymin; yi<=ymax; yi++) {
// mencari titik potong
.....
// urutkan absis dari kiri-kanan p1, p2, ..., pn
.....
//  $\forall$  pi & pi+1 (i ganjil) plot di antaranya.
.....
}
```


Algoritma *scan-line*

- Masalah garis tepi horisontal.
 - Jika suatu garis tepi diketahui horisontal (kedua ordinatnya sama) maka garis tersebut bisa diabaikan.
- Kasus titik ujung garis tepi.
 - Jika suatu garis scan melintasi titik pertemuan dua garis tepi maka akan ada dua titik potong.
 - Perlu dibedakan kasus dengan menganggap satu titik potong (2 ruas garis berarah sama) atau tetap dua titik potong (2 garis berlawanan arah).

Algoritma *scan-line*

- Efisiensi algoritma (I)
 - Pencarian titik potong merupakan proses yang bisa diefisienkan dengan memanfaatkan sifat koherensi inkremental. Jika x_i absis titik potong tepi antara (x_A, y_A) dan (x_B, y_B) pada garis scan ke- i , maka pada garis scan ke- $i+1$ absisnya $x_{i+1} = x_i + dx$, dengan $dx = (x_B - x_A) / (y_B - y_A)$. Asumsinya : $y_A < y_B \rightarrow x = x_A$

Algoritma *scan-line*

- Efisiensi algoritma (2)
 - Titik potong baru akan ada jika garis scan $y_i \in (y_A, y_B)$. Jika $y_A < y_B$ dan $y_i < y_A$ maka tidak ada titik potong. Setelah $y_i \geq y_A$ suatu counter mundur digunakan untuk memeriksa apakah masih terdapat titik potong atau tidak. Sebelumnya inisialisasi counter = $(y_B - y_A + 1)$.

Algoritma *scan-line*

- Tabel garis tepi
 - Perlu suatu tabel yang menyimpan informasi masing-masing garis tepi, sebelum iterasi diinisialisasi dengan algoritma InfoTepi.

Algoritma InfoTepi

```
Private InfoTepi[] tabelTepi=new InfoTepi[nTepi];
Private class InfoTepi  {
    int counter; double x, dx;
    int yawal;
    void InfoTepi(int x1, int y1; int x2; int y2)  {
        dx = (y2-y1)/(x2-x1);
        if (y1 < y2)  {
            x = x1; yawal = y1;
            counter = y2-y1+1;
        } else  {
            x = x2; yawal = y2;
            counter = y2-y1+1;
        }
    }
    void updateTepi()  {
        counter--; x += dx;
    }
}
```

Algoritma *scan-line*

- Pengurutan horisontal dan plot piksel
 - Suatu array diperlukan untuk mencatat dan mengurutkan absis dari titik-titik potong yang terjadi pada garis scan yang sedang berlangsung berdasarkan tabel tepi.
 - Tahapannya :
 - Periksa setiap tepi pada tabel, apakah berpotongan dengan garis scan ?
 - Jika ya, maka ambil harga x . lalu plot piksel-piksel antara setiap pasangannya.

Algoritma *scan-line*

- Langkahnya yaitu :

```
// periksa perpotongan
double []cross = new double[nTepi];
int p = 0;
for (int i=0; i < nTepi; i++) {
    if (tabelTepi[i].yawal >= yi && tabelTepi[i].counter > 0) {
        // jika berpotongan maka ambil harga x
        cross[p] = tabelTepi[i].x; p++;
    }
}
//urutkan p data terkiri
cross = Sort(cross, p);
// plot antara setiap pasangan
for (int i=0; i < p; i += 2) {
    drawLine(round(cross [i]), yi, round(cross [i+1]), yi);
}
```



Notes :