Pertemuan 4

ALGORITMA RASTER DASAR UNTUK GRAFIK 2D

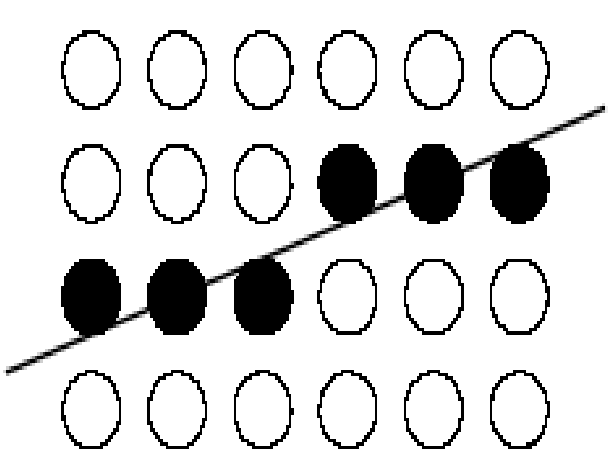
- Suatu paket pendekatan grafik raster dari grafik primitif (titik, garis, dan poligon) dideskripsikan dalam suatu sistem koordinat 2D, melalui letak pixel-pixel dalam pixmap ke pendekatan intensitas atau warna.
- Konversi *(conversion)* dari primitif ke pixmap secara umum dikenal dengan nama scan conversion

Scan converting lines (konversi scan garis).

- Scan conversi biasa akan menghitung pixel-pixel yang berada pada sekitar garis lurus pada suatu pixmap (suatu raster grid).
- Algoritma scan conversion akan menghasilkan suatu garis dengan brightness konstan (tidak tergantung pada panjang dan orientasi) dan hal ini bekerja dengan cepat.

- **Persamaan** garis menurut koordinat Cartesian adalah y = m.x + c, dimana m adalah slope (kemiringan) dari garis yang dibentuk dari dua titik, (x0, y0)(xe, ye).untuk penambahan x sepanjang garis yaitu dx akan mendapatkan penambahan sebesar dy = m.dx.

- **Misalkan** pixel-pixel diatas merupakan suatu lingkaran disjoin satu sama lain (merepresentasikan suatu titik phospor pada layar) pada suatu grid integer (x,y), serta awal dan akhir garis tersebut berada pada koordinat-koordinat integer (x0, y0) dan (xe, ye).



- Algoritma sederhana untuk scan convert garis adalah dengan increment.
- Pertama hitung slope $m = \Delta y / \Delta x$, dimana $\Delta y = y_1 - y_e$ Sekarang dimulai pada titik paling kiri dan increment x dengan 1 tiap waktu penghitungan $y_i = mx_i + c$, dimana c adalah offset (nilai y pada saat x = 0). Set intensitas pada nilai pixel (xi, Round(yi)), dimana Round(yi)= Floor(yi + 5).

- Di sini memilih pixel yang paling dekat dengan garis sesungguhnya, tetapi sangat tidak efisien karena tiap pixel membutuhkan suatu pengali folating point , suatu penambahan dan suatu call untuk Floor.

$$y_{i+1} = mx_{i+1} + c = m(x_i + \Delta x) + c = y_i + m\Delta x$$

dan Δx dalam kasus tersebut akan menjadi 1, dan diperoleh :

$$x_{i+1} = x_i + 1$$

$$y_{i+1} = y_i + m$$

ini merupakan algoritma yang lebih efisien, bekerja efisien pada |m| < 1.

- Berikut diberikan pseucode sederhana untuk algoritma tersebut diatas.

```
 void Line ( int x0,  int xe,  int y0,  int ye,  int
{           /* Assumes -1 <= m <= 1, x0 < xe */
    int x;
    float y,dx,dy,m;

   dx = xe - x0;
   dy = ye - y0;
   m = dy / dx;
   y = y0;

    for (x = x0; x <= xe; x++){
      WritePixel(x,(int) floor(y + 0.5),value);
      y += m;
   }
}
```

Algoritma garis DDA

- Digital Differential Analyzer (DDA) merupakan suatu algoritma pembentukan garis berdasarkan perhitungan dx atau dy, dengan menggunakan rumus dy = m.dx.
- Garis dibuat dengan menggunakan dua endpoint yakni berupa titik awal dan titik akhir.
- Setiap koordinat titik yang membentuk garis diperoleh dari perhitungan, selanjutnya dikonversikan menjadi nilai integer.

Algoritma garis DDA

- Berikut ini adalah langkah-langkah pembentukan garis dengan menggunakan algoritma DDA :

1. Tentukan dua titik yang akan dihubungkan dalam pembentukan garis.
2. Tentukan salah satu titik sebagai titik awal (x0, y0) dan titik akhir (xe, ye).
3. Hitung dx = xe – x0 dan dy = ye – y0
4. Tentukan step, yaitu jarak maksimum jumlah penambahan nilai x maupun nilai y, melalui
   - Jika nilai absolute dari dx lebih besar dari absolute dy, maka step = absolute dx
   - Jika tidak, maka step = absolut dy.
5. Hitung penambahan koordinat pixel, yaitu x_increment = dx/step dan y_increment = dy/step.

6. Koordinat selanjutnya (x + x_increment, y + y_increment).
7. Posisi pixel pada layar ditentukan melalui pembulatan nilai koordinat tersebut.
8. Ulangi langkah 5 dan 6 untuk menentukan posisi pixel selanjutnya, sampai x = xe dan y = ye

Contoh DDA

Berikut ini adalah contoh pembentukan suatu garis dengan menggunakan algoritma DDA suatu garis yang menghubungkan titik (10, 10) dan (17, 16). Pertama-tama ditentukan dx dan dy, selanjutnya dicari step untuk mendapatkan x_increment dan y_increment.

- dx = xe – x0 = 17 – 10 = 7
- dy = ye – y0 = 16 – 10 = 6
- selanjutnya dihitung dan dibandingkan nilai absolutnya : abs(dx) = 7 dan abs(dy) = 6.
- Karena abs(dx) > abs(dy), maka step = abs(dx) = 7, sehingga :
- x_increment = 7/7 =1 dan y_increment = 6/7

Tabel : Contoh DDA

k	x	y	(x,, y) bulat
	10	10	(10, 10)
0	11	10.86	(11, 11)
1	12	11.72	(12, 12)
2	13	12.57	(13, 13)
3	14	13.43	(14, 13)
4	15	14.28	(15, 14)
5	16	15.14	(16, 15)
6	17	16	(17, 16)

Algoritma garis titik tengah

Algoritma garis titik tengah(Midpoint Line Algorithm)

- **A**lgoritma garis titik tengah (midpoint line algorithm) disebut juga dengan algoritma garis Bressenham, tidak menggunakan penghitungan floating point.
- Algoritma ini merupakan suatu generalisasi dari Bresenham, yang merupakan pengembangan dari algortima klasik yang lebih menarik, karena hanya menggunakan perhitungan matematik dengan bilangan integer.
- Algoritma ini merupakan algoritma konversi penambahan nilai integer yang dapat juga dipakai untuk lingkaran.

- Untuk menggambarkan pendekatan Bressenham, pertama kali diperlukan proses pembentukan garis dengan slope lebih kecil dari 1.
- Posisi pixel sepanjang garis ditentukan pada interval x. Dimulai dari titik sebelah kiri (x0, y0), kemudian ditentukan posisi kolom x dengan posisi scan-line y yang mendekati path (jalur) dari garis tersebut. Posisi pixel berikutnya adalah (xk+1, yk) atau (xk+1, yk+1). Posisi ini akan tergantung pada nilai parameter p.
- Jika p lebih kecil dari 0, maka posisi selanjutnya adalah (xk+1, yk), dan bila tidak maka posisi selanjutnya adalah (xk+1, yk+1). Sedangkan nilai p berikutnya adalah pk+1 = pk + 2 dy atau pk+1 = pk + 2 dy – 2 dx.

Algoritma Bressenham

Algoritma Bressenham untuk membentuk garis adalah sebagai berikut :

1. Tentukan dua titik yang akan dihubungkan dalam pembentukan garis.
2. Tentukan salah satu titik disebelah kiri sebagai titik awal, yaitu (x0, y0) dan titik lainnya sebagai titik akhir (xe, ye).
3. Hitung dx, dy, 2 dx dan 2 dy – 2 dx.
4. Hitung parameter : p0 = 2 dy – dx.
5. Untuk setiap xk sepanjang jalur garis, diawali dengan k = 0, dan
   - Jika pk < 0, maka titik selanjutnya adalah (xk +1, yk) dan pk+1 = pk + 2 dy
   - Jika tidak, maka titik selanjutnya adalah (xk +1, yk + 1) dan pk+1 = pk + 2 dy – 2 dx.
6. Ulangi langkah 5. untuk menentukan posisi pixel selanjutnya, sampai x = xe dan y= ye.

Berikut ini diberikan contoh algoritma garis Bressenham dalam pembentukan suatu garis yang menghubungkan titik (10, 10) dan (17, 16), maka pertama-tama ditentukan bahwa titik (10, 10) berada di sebelah kiri dan ini merupakan titik awal, sedangkan titik (17, 16) merupakan titik akhir. Posisi pixel yang membentuk garis dapat ditentukan dengan melalui perhitungan sebagai berikut :

- dx = xe – x0 dan dy = ye – y0
- parameter p0 = 2 dy – dx.

- Melalui perhitungan dengan menggunakan algoritma Bressenham, maka posisi pixel yang membentuk garis dapat diperoleh dalam tabel sebagai berikut :

k	p_k	$(x_k+1, y_k+1$
	(10, 10)	
0	3	(11, 11)
1	1	(12, 12)
3	-1	(13, 13)
4	11	(14, 13)
5	9	(15, 14)
6	7	(16, 15)
7	5	(17, 16)

- Berikut ini diberikan Pseucode untuk untuk konversi algoritma ***midpoint line scan*** :

```
 void MidpointLine ( int x0,  int xe,  int y0,  int ye,  int valu
{           /* Assumes 0 <= m <= 1, x0 < xe, y0 < ye */
    int x,y,dx,dy,d,incE,incNE;

   dx = xe - x0;
   dy = ye - y0;
   d = 2*dy - dx;
   incE = 2*dy;
   incNE = 2*(dy-dx);
   x = x0;
   y = y0;
   WritePixel(x,y,value);
    while (x < xe) {
        if (d <= 0) {
           d += incE;
           x++;
       }  else {
           d += incNE;
           x++;
           y++;
       }
       WritePixel(x,y,value);
   }
}
```

Di ketahui dua titik (10,8) dan (18,17)

- Buatlah tabel perhitungannya.

  dengan menggunakan algoritma pembentukan garis DDA dan Bresenham (MID Point)