ALGORITMA MIDPOINT UNTUK PENGGAMBARAN GRAFIK BERKECEPATANG TINGGI

Kartika Gunadi
Fakultas Teknik, Jurusan Teknik Informatika - Universitas Kristen Petra
e-mail: kgunadi@petra.ac.id

ABSTRAK : Penggambaran grafik garis lurus dan kurva memerlukan waktu komputasi yang tinggi, untuk mereduksi waktu komputasi yang tinggi tersebut dapat dilakukan dengan peningkatan kemampuan komputasi prosesor dan peningkatan efisiensi algoritma. Algoritma Midpoint merupakan Algoritma dengan dasar operasi bilangan integer, sehingga memerlukan waktu operasi yang lebih sedikit dibandingkan dengan algoritma yang menggunakan operasi bilangan riel. Implementasi ke dalam bahasa pemrograman C dari kedua macam algoritma diatas, menunjukkan bahwa waktu komputasi algoritma midpoint lebih cepat sebesar 8 kali pada pembuatan garis lurus, dan lebih cepat sebesar 15 kali pada penggambaran lingkaran, dibandingkan dengan waktu komputasi algoritma yang menggunakan dasar operasi bilangan riel. Dan waktu komputasi algoritma midpoint lebih cepat sebesar 6 kali pada pembuatan garis lurus, dibandingkan dengan waktu komputasi algoritma yang Breserham telah menggunakan dasar operasi bilangan integer juga.

Kata kunci: Penggambaran garis, penggambaran kurva, Algoritma Bresenham, Algoritma midpoint, Algoritma DDA.

ABSTRACT : *Line and curve drawing are time consuming, reducing computation time can be done by improving performance of processor and algorithm. Midpoint algorithm is based on integer operation, so it is less time-consuming than algorithm based on float operation. Implementation of both type of algorithms in C language, shows that midpoint algorithm is 8 times faster for line drawing, and 15 times faster for curve drawing than algorithm based on float operation. For line drawing, Midpoint algorithm is 6 times faster than Bresenham algorithm which is based on integer operation*

Keywords: line drawing, curve drawing, Bresenham algorithm, midpoint algorithm, DDA algorithm.

1. PENDAHULUAN

Perkembangan kemampuan komputasi prosesor yang pesat telah membuat komputer desktop mempunyai kemampuan komputasi yang besar. Hal ini mendorong perkembangan program aplikasi yang memerlukan komputasi yang besar seperti program aplikasi yang menggunakan grafik 3 dimensi. Peningkatan kemampuan komputasi prosesor untuk aplikasi grafik yang sarat komputasi, perlu dibarengi peningkatan efisiensi algoritma, sehingga pembuatan grafik garis dan kurva yang merupakan dasar pembuatan grafik dapat memberikan hasil yang optimal.

Algoritma midpoint merupakan algoritma pembuatan garis dan kurva dengan dasar operasi bilangan integer yang menonjolkan ciri kecepatan. Sehingga algoritma ini dapat dipakai sebagai algoritma pembuatan grafik yang menuntut kecepatan sebagai hal yang diutamakan. Pembahasan algoritma Midpoint dilakukan dengan mengasumsikan garis lurus dari kiri ke kanan, dan gadient antara 0 dan 1, sedangkan untuk lingkaran dengan mengasumsikan hanya sebagian lingkaran dengan sudut sebesar 45◦, hal ini dilakukan untuk mempermudah penjelasan, sedangkan untuk kondisi yang lain dapat diderivasi dengan cara yang serupa. Untuk mendapatkan kinerja algoritma midpoint, dilakukan uji kecepatan komputasi dengan cara mengimplementasikan kedalam bahasa pemrograman C, dan melakukan perbandingan waktu komputasi

dengan algoritma yang menggunakan dasar komputasi bilangan riel, maupun algoritma lain yang telah banyak dikenal seperti algoritma dda dan algoritma bressenham.

2. GARIS LURUS

Garis lurus dinyatakan dinyatakan dalam persamaan :

$$y = mx + c \quad (1)$$

dimana :
m : gradient dan
c : konstanta.

Untuk menggambarkan piksel-piksel dalam garis lurus, parameter yang digunakan tergantung dari gradient, jika besarnya gradient diantara 0 dan 1, maka digunakan sumbu x sebagai parameter dan sumbu y sebagai hasil dari fungsi, sebaliknya, bila gradient melebihi 1, maka sumbu y digunakan sebagai parameter dan sumbu x sebagai hasil dari fungsi, hal ini bertujuan untuk menghindari terjadinya gaps karena adanya piksel yang terlewatkan. Hasil dari fungsi biasanya merupakan bilangan riel, sedangkan koordinat pixel dinyatakan dalam bilangan integer (x,y), maka diperlukan operasi pembulatan kedalam bentuk integer terdekat.

Penggambaran garis lurus dengan metode diatas dimulai dengan operasi bilangan riel untuk menghitung gradient m dan konstanta c.

$$m = (y2 - y1) / (x2-x1) \quad (2)$$

$$c = y1 – m* x1 \quad (3)$$

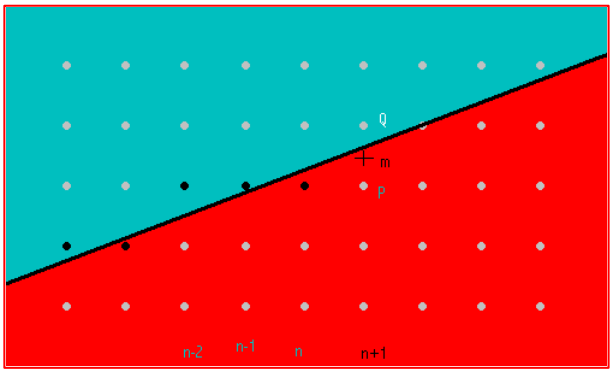
Operasi bilangan riel berikutnya adalah menghitung nilai y dengan persamaan (1) untuk mendapatkan koordinat piksel (x,y), untuk setiap nilai x, dari =x1 sampai x=x2, operasi inilah yang perlu dihindari, karena operasi ini memerlukan waktu operasi yang besar.

2.1 Algoritma Bresenham

Bresenham pada tahun 1965, melakukan perbaikan dari algoritma perhitungan koordinat piksel yang menggunakan persamaan (1), dengan cara menggantikan operasi bilangan riel perkalian dengan operasi penjumlahan, yang kemudian dikenal dengan Algoritma Bresenham. Pada algoritma bresenham, nilai y kedua dan seterusnya, dihitung dari nilai y sebelumnya, sehingga hanya titik y pertama yang perlu dilakukan operasi secara lengkap. Perbaikan algoritma ini ternyata tidak menghasilkan perbaikan yang cukup siginifikan. Perbaikan berikutnya dilakukan dengan cara menghilang-kan operasi bilangan riel dengan operasi bilangan integer. Operasi bilangan integer jauh lebih cepat dibandingkan dengan operasi bilangan riel, terutama pada penambahan dan pengurangan.

2.2 Algoritma Midpoint untuk Penggam-baran Garis

Algoritma midpoint dikembangkan oleh Pitteway pada tahun 1967. Pada gambar 1, titik abu-abu menyatakan posisi piksel, titik hitam menyatakan posisi piksel yang telah digambar. Berdasarkan piksel ke n yang telah digambar, diperlukan metode untuk menentukan piksel berikut yang akan digambar, karena penggambaran dilakukan dari kiri ke kanan, maka piksel berikutnya harus pada kolom n+1. Karena gradien diantara 0 dan 1, maka piksel berikutnya adalah pada posisi titik p atau titik q.



Gambar 1. Garis Lurus

Persamaan garis lurus yang telah dinyatakan dalam persamaan (1) dapat dinyatakan dalam fungsi x,y berikut.

$$f(x, y) = ax + by + c = 0 \quad (4)$$

Fungsi f(x,y) dalam persamaan (4), akan memberikan nilai 0 pada setiap titik yang

terletak pada garis, dan bernilai positip pada setiap titik yang terletak dibawah garis, dan bernilai negatif pada setiap titik yang terletak diatas garis.

Maka untuk menentukan apakah titik P atau Q sebagai koordinat piksel berikutnya, maka dilakukan dengan cara menghitung nilai fungsi f(x,y) dalam persamaan (4) pada titik P dan titik Q . Jika fungsi f(x,y) tersebut memberikan nilai positif, maka piksel berikutnya adalah Q, sebaliknya piksel berikutnya adalah P.

$$g(x, y) = f(xn + 1, yn + 1/2) \quad (5)$$

Fungsi g(x,y) persamaan (5) merupakan variabel penentu, dengan mengevaluasi g (x, y) dapat ditentukan piksel berikutnya yang mana berdasarkan tanda plus atau minus dari hasil fungsi g(x,y).

Untuk mempercepat komputasi fungsi g(x,y), dilakukan dengan cara incremental berdasarkan nilai sebelumnya. Untuk setiap piksel, increment sederhana (DeltaG) dipakai sebagai variabel penentu. Karena hanya ada 2 pilihan piksel pada setiap tahap, maka hanya ada 2 increment yang dapat digunakan. Hal ini dilakukan dengan cara pengurangan nilai g(x,y) yang berurutan dengan menggunakan persamaan 4 dan persamaan 5.

$$DeltaG = a * DeltaX + b * DeltaY \quad (6)$$

Dimana DeltaX dan DeltaY adalah increment yang dipakai pada x dan y, yang bernilai 0 atau 1. Bila bergeser 1 piksel ke kanan :

$$DeltaG1 = a \quad (7)$$

Bila bergeser 1 piksel ke kanan dan 1 piksel ke atas.

$$DeltaG2 = a + b \quad (8)$$

Jadi nilai dari variable penentu dapat dihitung dari nilai sebelumnya dengan cara menambah dengan (a) atau (a+b). Algoritma untuk menggambar garis lurus dari (x1, y1) sampai (x2, y2) dilakukan dengan langkah-langkah sebagai-berikut:

1. Gambar piksel pertama (x1,y1). Hitung variabel penentu dengan persamaan (3).
2. Tentukan tanda variabel penentu. Jika variabel penentu bernilai positif, increment x dan y dan tambahkan (a+b) pada vaiabel penentu, sebaliknya increment x dan y dan tambahkan (a) pada variabel penentu.
3. Plot piksel pada posisi (x, y).
4. Ulangi langkah mulai langkah kedua, sampai piksel terakhir (x2,y2).

3. LINGKARAN

Kurva lingkaran dinyatakan dinyatakan dalam persamaan :

$$(x-xc)^2 + (y-yc)^2 = r^2 \quad (9)$$

dimana :
(xc,yc) : koordinat titik pusat lingkaran
r : jari-jari lingkaran

Untuk menggambarkan piksel-piksel dalam kurva lingkaran, dapat digunakan sumbu x dari x = (xc-r) sampai x = (xc+r) sebagai parameter dan sumbu y sebagai hasil dari persamaan (10)

$$y = yc +- sqrt(r^2 - (x-xc)^2 \quad (10)$$

Algoritma ini memerlukan waktu operasi yang besar, karena mengandung operasi bilangan riel perkalian maupun eksponential, dan menghasilkan posisi koordinat piksel yang tidak merata, karena terjadinya gaps yang disebabkan adanya perubahan gradient.

Untuk menghindari posisi koordinat piksel yang tidak merata, koordinat piksel (x,y) dinyatakan dengan menggunakan koordinat polar dalama persamaan (11)

$$x = xc + r \cos q \quad (11a)$$

$$y = yc + r \sin q \quad (11b)$$

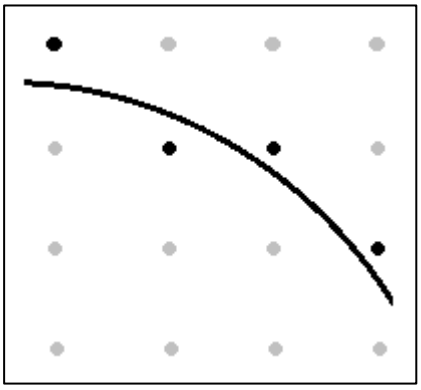
Persamaan (11) diatas mengandung operasi bilangan riel perkalian untuk mendapatkan koordinat piksel (x,y), untuk setiap nilai x, dari x = (xc-r) sampai x = (xc+r), operasi inilah yang perlu dihindari, karena operasi ini memerlukan waktu operasi yang besar.

3.1 Algoritma Midpoint

Komputasi untuk membuat kurva lingkaran dimulai dengan mengidentifikasi bagian-bagian dari lingkaran yang dapat ditentukan dengan menggunakan sifat simetri, hal ini dilakukan dengan cara membagai lingkaran dengan masing-masing mempunyai sudut sebesar 45°, sehingga dalam sebuah lingkaran dibagi menjadi

8 bagian. Sebagai contoh, digambarkan bagian dari lingkaran dari sudut 90◦ sampai 45◦.

Seperti pada algoritma midpoint untuk garis lurus, maka pada setiap tahapan, terdapat 2 koordinat piksel yang harus dipilih yaitu (x+1, y) atau (x+1, y-1).



Gambar 2. Lingkaran

Langkah berikutnya, adalah menyatakan persamaan lingkaran dan fungsi untuk menentukan variabel penentu. Persamaan lingkaran dengan pusat (0,0) dinyatakan dalam persamaan (12).

$$f(x, y) = x*x + y+y - r*r = 0 \quad (12)$$

Fungsi f(x, y) persamaan (12) akan bernilai positif jika titik (x,y) diluar lingkaran, dan bernilai negatif jika titik (x,y) didalam lingkaran. Fungsi untuk variabel penentu dan increment dinyyatakan dalam persamaan (13), (14), dan (15).

$$g(x, y) = (x + 1)*(x + 1) + (y - 1/2)*(y - 1/2) - r*r \quad (13)$$

$$DeltaG1 = 2x + 3 \quad (14)$$

$$DeltaG2 = 2x - 2y + 5 \quad (15)$$

Berbeda dengan nilai dari increment pada algoritma garis lurus yang bernilai konstan, pada kurva lingkaran, increment tidak konstan. Terlihat bahwa komputasi increment memerlukan operasi perkalian, tetapi operasi perkalian dapat diubah dengan cara komputasi nilai increment secara increment pula, sehingga diperlukan 2 komputasi increment dalam setiap piksel yang diproses. Secara umum, kurva polinomial orde n memerlukan n level increment. Pada titik awal (x1,y1), komputasi variabel penentu mempunyai bagian bilangan riel, sehingga operasi bilangan integer tidak dapat digunakan secara langsung. Dalam praktek hal ini diselesaikan dengan cara menambahkan nilai 1/4 pada variable penentu. Hal ini tidak mempengaruhi perubahan tanda bilangan, karena operasi yang dilakukan adalah operasi bilangan integer, sehingga menghasilkan operasi yang lebih cepat.

4. IMPLEMENTASI PERANGKAT LUNAK

Untuk mendapatkan hasil kerja dari suatu algoritma, maka algoritma pembuatan grafik garis lurus dan kurva lingkaran perlu diimplementasikan kedalam bahasa pemrograman dan dilakukan pengukuran waktu komputasi. Pada penulisan ini algoritma pembuatan grafik diimplementasikan kedalam bahasa pemrograman C, dengan menggunakan compiler Turbo C++. Listing lengkap (source code) dari program ini dapat didownload lewat homepage Internet penulis dengan alamat http://faculty.petra.ac.id/ kgunadi

5. HASIL PENGUKURAN

Pengukuran kecepatan komputasi pada semua algoritma menggunakan basis dibawah ini, yang selengkapnya dapat dilihat pada program1.cpp. Perulangan sebanyak 10 kali dilakukan dengan tujuan untuk mendapatkan ketelitian dari satuan pengukuran terkecil yaitu sebesar 1/100 detik, satuan yang mampu diberikan oleh bahasa pemrograman C. Hasil ditunjukan pada tabel 1 dan tabel 2.

Pengukuran dilakukan dengan berbagai ukuran panjang garis, untuk mengetahui pengaruh waktu komputasi perintah-perintah yang ada di dalam perulangan sepanjang grafik, maupun pengarus waktu komputasi perintah-perintah yang ada sebelumnya.

```
void main()
{
  gettime(t1)
  For (counter=1;counter<=10;counter++)
    TestCode(); // Algoritma yang diuji
```

```
   gettime(t2)
}
```

Eksekusi program dilakukan pada komputer dengan prosesor Intel 80286-12 dengan clock 12 Mhz, tanpa Math Co-processor dan pada sistem operasi single-tasking DOS (Disk Operating System), hal ini dilakukan untuk menghindari efek time-slice dari sistem operasi multi-tasking.

Tabel 1. Waktu Dan Rasio Penggambaran Garis Lurus (1/100 detik)

Algoritma \ Panjang garis	**640x 100**		**640x 10**		**640x 1**	
Algoritma Persamaan (1)	4647	19	473	17	49	8
Algoritma DDA	3862	16	390	14	43	7
Algoritma Bresenham	2702	11	274	10	33	6
Algoritma MidPoint	242	1	28	1	6	1

Tabel 2. Waktu Dan Rasio Penggambaran Lingkaran (1/100 detik)

Algoritma \ Jari-jari	**480x 100**		**480x 10**		**480x 1**	
Algoritma Persamaan (10)	2265	15	253	15	29	14
Algoritma Bresenham	146	1	17	1	2	1
Algoritma MidPoint	148	1	16	1	2	1

6. KESIMPULAN

Panjang garis atau banyak piksel dalam garis lurus sangat berpengaruh terhadap perbandingan performance antara sebuah algoritma dengan algoritma yang lain, hal ini disebabkan adanya perbedaan waktu operasi yang berada didalam perulangan sepanjang pembuatan piksel, dan waktu operasi yang berada pada sebelumnya.

Panjang jari-jari dalam lingkaran tidak berpengaruh terhadap perbandingan performance antara sebuah algoritma dengan algoritma yang lain, hal ini menunjukkan perbandingan waktu operasi yang berada didalam perulangan sepanjang pembuatan piksel, dan waktu operasi yang berada pada sebelumnya berimbang.

Algoritma dengan dasar operasi bilangan integer memberikan waktu operasi yang lebih cepat dibandingkan dengan algoritma dengan dasar operasi bilangan riel, hal ini ditunjukkan dengan waktu komputasi algoritma DDA, algoritma Bresenham dan algoritma Midpoint yang lebih cepat, baik pada pembuatan garis lurus maupun lingkaran dibandingan waktu komputasi dengan algoritma yang menggunakan dasar operasi bilangan riel.

Algoritma midpoint memberikan waktu operasi tercepat diantara algoritma penggambaran garis lurus yang telah menggunakan dasar operasi bilangan integer, seperti algoritma DDA, algoritma Bresenham. Jadi algoritma Midpoint merupakan algoritma yang cocok untuk penggambaran grafik yang menuntut kecepatan sebagai hal yang diutamakan.

DAFTAR PUSTAKA

1. Donald Hearn and M.Pauline Baker, *Computer graphics*, NJ : Prentice-Hall, 1992.

2. Borland International, Inc., *Turbo C++ Reference Guide*, 1990.

3. http://www.geocities.com/Broadway/4574/midpoint/index.htm

4. http://hops.cs.jhu.edu/~stevec/graphics/midpoint.html