PERTEMUAN 3

ALGORITMA GARIS DAN POLIGON

A. TUJUAN PEMAHAMAN PEMBELAJARAN

Setelah menyelesaikan materi pada pertemuan ini, mahasiswa mampu menerapkan algoritma garis dan poligon.

Pada pertemuan ini akan dijelaskan mengenai:

1. Algoritma Pembentukan Garis
2. Algoritma Garis Bresenham
3. Membuat Poligon

B. URAIAN MATERI

1. Algoritma Pembentukan Garis

Ilmu grafis merupakan salah satu bentuk dari geometri paling mendasar yang umum digunakan ketika membuat seuuah bentuk/objek yang memiliki struktuur kompleks. Titik dan juga garis lurus merupakan salah satu contoh bentuk geometri dari sebuah gambar. Kedua hal ini yaitu titik dan juga garis merupakan contoh output dalam pembentukan sebuah gambar, misalnya persegi, lingkaran, kerucut, kurva dan juga gambar-gambar permukaan yang memiliki bentuk lengkung, karakter, area warna dan hal-hal lainnya.

Keberadaan titik dan garis ddiyakini sudah ada sejak masa lampau/zaman nenek moyang, hal ini diiduukung dari kebudayaan mereka ketika mengadaka sebuah upacara/perayaan dalam sistem kepercayaan mereka yaitu dengan cara membuat goresan di dalam dinding gua. Berawal dari goresan inilah kemudian dihasilkan berbagai macam garis yang saling tersambung dan nantinya akan membentuk sebuah obyek yang biasa disebut dengan gambar. Kemudian, dari gambar ini terciptalah lukisan yang akan menjadi awal mula dari kemunculan beragam jenis huruf/alfabet yang digunakan sebagai media komunikasi oleh manusia. Selain bentuk garis yang dikenal dengan kontur, garis juga umum

digunakan untuk menjelaskan sebuah gerak maupun bentuk, dalam berbagai bentuk antara lain, bentuk dua dimensi dan tiga dimensi.

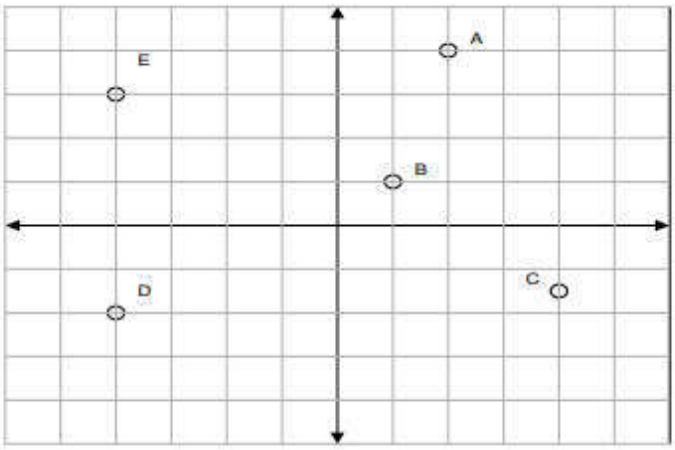
Untuk mengenal dan mengetahui ilmu tentang Algoritma Pembentukan Garis, maka kita mulai dari latar belakang dan juga sejarah dari algoritma itu sendiri. Algoritma merupakan inti utama dari ilmu informatika. Terdapat beragam ilmu pengetahuan khususnya ilmu perkomputeran mengambil dasar/mengacu ke dalam terminologi algoritma, salah satunya adalah algoritma routing di dalam sebuah jaringan komputer, kemudian ada algoritma bresenham yang digunakan dalam membuat garis lurus di dalam grafika komputer, selanjutnya terdapat algoritma Knuth-Morris-Pratt yang mempunyai kegunaan untuk mencari sebuah pola dalam sebuah teks (information retrievel), dan contoh-contoh lainnya.

Jika ditelaah dari awal mula katanya, kalimat “algoritma” memiliki sejarah tersendiri. Kita akan mendapatkan kosakata *algorsim* atau kurang lebih memiliki pengertian sebagai kegiatan perhitungan menggunakan angka – angka. Cukup banyak ilmuwan yang berusaha keras mencari darimana awal mula kosakata ini (algorism) berasal, akan tetapi hasil yang didapatkan tidaklah memuaskan. Kata algorism berasal dari nama seorang ilmuwan dari jazirah arab yang sangat terkenal akan karya-karyanya, yaitu Al-Khuwarizmi. Ia membuat sebuah buku yang diberi judul Kitab “al jabar wal-muqabala”, buku ini jika diartikan kuarng lebih memilikii arti “Buku Pemugaran dan Pengurangan”. Awal mula kata ‘aljabar’ (*algebra*) juga didapatkann dari buku tersebut.

a. Titik

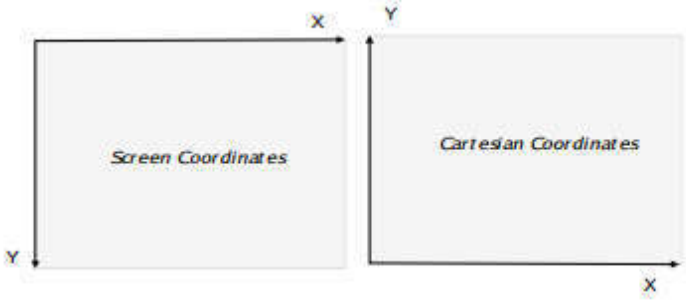
Sebuah titik dalam cabang ilmu grafika komputer dapat didefinisikan sebagai letak/posisi objek didalam sebuah *Coordinates System*. Pola titik kordinat yang pada umumnya digunakan adalah kordinat polar ataupun kordinat kartesius. Ketika kita menggunakan ilmu grafika komputer, cara paling sering yaitu adalah dengan menggunakan kordinat kartesius. Dalam penggunaannya, posisi titik difungsikan untuk mengetahui letak/posisi dari gabungan dari angka-angka untuk selanjutnya menetapkan letak titiknya dengan menggunakan kordinat posisi x dan juga y.

Sebagai contohnya adalah, Ketika kita meletakan posisi titik sebagai berikut : titik A berada di kordinat (2,4), titik B (1,1), titik C (4,-1.5), titik D (-4,-2), dan titik E (–4,3) dalam koordinat cartesius, maka kita dapat menggambarkannya seperti dibawah ini:



Gambar 3. 1 Letak titik dalam koordinat cartesius

Terdapat 2 (dua) koordinat didalam komputer khusunya yang umum digunakan oleh Operation Systems (OS) dalam Windows, yang pertama adalah kordinat layar (*Screen Cordinates*), kedua adalah kordinat kartesius (*Cartesian Cordinates*), perbedaan diantara dua sistem kordinat ini seringkali membuat kita bingung. Maka dari itu kita bisa mencoba memperhatikan contoh dibawah ini :

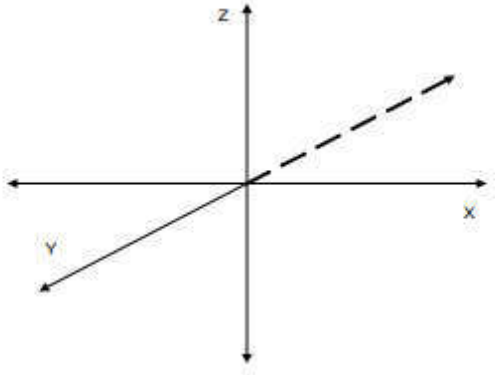


Gambar 3. 2 Perbandingan antara Cartesian dan Screen Cordinates

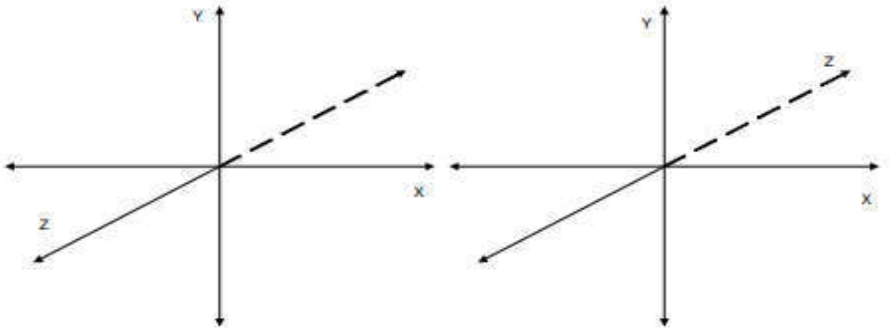
Secara umum prinsip monitor memang didesain untuk membuat garis dari bagian atas menuju bagian bawah, maka dari itu letak titik y pada kordinat kartesius dan kordinat layar memiliki titik sumbu yang berbeda, dalam kordinat kartesius posisi sumbu titik Y mengarah ke atas, sebaliknya *untuk kordinat* layar, posisi sumbu titik Y memiliki arah ke bawahh. Pada umumnya ketika kita akan menggunakan teknik *rendering pipeline*, hal

utama yang perlu kita lakukan yaitu mengubah posisi *Cartesian Coordinates* menjadi *Screen Coordinates*.

Untuk Operation Systems (OS) Linux, posisi koordinat Y yang dipakai baik *Cartesian* maupun *Screen Coordinates*, memiliki kesamaan yaitu Y memiliki arah positif kearah atas. Sedangkan dalam koordinat 3D, tidak berbeda seperti dalam 2D, akan tetapi terdapat tambahan 1 (satu) titik lagi yang menjadi sumbu yaitu titik z atau biasa disebut sebagai axis-z. Setidaknya terdapat 2 teknik dalam membuat gambar sumbu di titik X, Y kemudian Z. Seperti kita lihat pada gambar, yaitu sumbu titik z memiliki arah menuju atas (gambar 3.3). Kedua adalah dengan koordinat sumbu y mengarah ke atas (gambar 3.4).



Gambar 3. 3 Posisi koordinat



Z mengarah keatas

Gambar 3. 4 Posisi koordinat Y mengarah keatas

b. Garis

Hal pertama yang akan kita pelajari pada pembahasan ini adalah apa itu garis ?. Garis adalah sebuah kumpulan titik, Contohnya adalah saat kita akan membuat sebuah garis lurus, maka hal utama yang perlu diketahui adalah posisi titik awal dan titik akhir dari garis tersebut. Setelah kita mengetahuinya, maka kita bisa menyambungkan titik awal dengan akhir sehingga membentuk sebuah garis lurus.

Persamaan garis lurus umumya dikenal dan dinotasikan dengan persamaan y = mx + c. Simbol *m* disini merupakan gradien yang merupakan proses dari pembagian antara titik delta Y dan juga delta X. Sedangkan simbol *c* merupakan konstanta. Hal ini menjadi fondasi dasar dalam algoritma yang digunakan dalam pembuatan garis lurus. Menggunakan rumus ini juga kita akan memulai mempelajari algortima yang secara umum dipakai untuk pembuatan sebuah garis lurus.

Saat ini, performa komputasi prosesor komputer telah sangat cepat. Sehingga hal ini juga berpengaruh terhadap kemampuan komputasi yang semakin handal. Oleh karena itu saat ini semakin banyak program aplikasi yang memakai grafik 3 dimensi yang cukup kompleks. Keadaan seperti ini tentunya perlu diimbangi dengan efisiensi algoritma yang ada, jadi pembuatan garis dan juga kurva dalam pembuatan grafik dapat menghasilkan gambar dengan kualitas maksimal.

Dalam menggambarkan sebuah pembuatan garis, baik dari Langkah awal sampai kepada Langkah akhir terdapat bermacam algorima yang dapat digunakan. Beberapa algoritma yang umum digunakan antara lain adalah algoritma Brute Force, DDA (Digital Differential Analyzer) dan Bressenham.

c. Algoritma Brute Force

Penggunaan algoritma ini dalam pembuatan sebuah gambar garis memiliki dasar dengan kedudukan titik (2-6), berikut penjelasannya :

Tentukan titik akhir dari ($x1$,$y1$) dan ($x2$,$y2$)

1) Jika $x_1 = x_2$ (garis vertikal), maka

- $y = y+ 1$ dan x tetap
- gambar titik (x,y) dilayar

2) Jika $y1 = y2$ (garis horisontal), maka

- $x = x + 1$ dan y tetap
- gambar titik (x,y) dilayar

{ anggap dari $x2 > x1$, (jika kebalikannya, maka $x2$ diganti $x1$)}

3) Hitunglah kemiringan dari titik $m = (y2 - y1)/(x2 - x1)$

4) $N = x2 - x1 + 1$

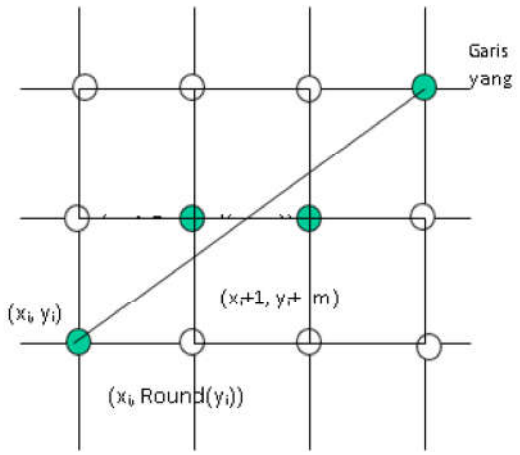
5) $x = x1$

6) aturan

- $y = m\,(x - x_1)$
- $y_a = Round\,(y)$,
- Buat gambar sesuai posisi (x, y_a)
- X = x + 1

7) Selesai

Karena posisi piksel berupa bilangan bulat (integer), Kemudian nilai m umumnya adalah pecahan, oleh karena itu y merupakan bilangan pecahan dan perlu dilakukan pembulatan nilai y.



Gambar 3. 5 Dibutuhkan pembulatan dari nilai **y** yang disepertemuankan letak titik piksel berupa bilangan bulat

Contoh 1:

Diketahui terdapat dua titik yaitu titik A yang terletak di kordinat (2,1) dan juga titik B (8,5), bila titik A dijadikan posisi awal kemudian titik B menjadi letak titik akhir, Dengan memakai algoritma brute force, coba gambarkan garis diantara kedua titik A dan B sehingga menghubungkan kedua titik tersebut.

Cara Menjawab :Posisi titik di awal (x_1 , y_1) = (2,1), kemudian (x_2 , y_2) = (8,5)

a. tidak terpenuhi

b. tidak terpenuhi

c. $m = (5 – 1)/(8 – 2) = 0,67$

d. N = 8 – 2 + 1 = 7

e. Kemudian terakhir adalah $x = 2$ $y = 0,67\ (x – 2) + 1$

f. Ulangi langkah ini sampai 7 kali

Pengulangan ke-1:

$x = 2$; $y = 0,67.\ (2 – 2) + 1 = 1$

penggabungan ke nilai : $y = 1$, gambarlah pada posisi (2,1) dalam layar $x = 2 + 1 = 3$

Pengulangan ke-2:

$x = 3$; $y = 0,67.\ (3 – 2) + 1 = 1,67$

penggabungan ke nilai : $y = 2$ gambarlah pada posisi (3,2) dalam layar $x = 3 + 1 = 4$

Pengulangan ke-3:

$x = 4$; $y = 0,67.\ (4 – 2) + 1 = 2,34$

penggabungan ke nilai : $y = 2$ gambarlah pada posisi (4,2) dalam layar $x = 4 + 1 = 5$

Pengulangan ke-4:

$x = 5$; $y = 0,67.\ (5 – 2) + 1 = 3,01$

penggabungan ke nilai : $y = 3$ gambarlah pada posisi (5,3) dalam layar

$x = 5 + 1 = 6$

Pengulangan ke-5:

$x = 6$; $y = 0{,}67. (3 – 2) + 1 = 3{,}68$

penggabungan ke nilai : $y = 4$ gambarlah pada posisi (6,4) dalam layar $x = 6 + 1 = 7$

Pengulangan ke-6:

$x = 7$; $y = 0{,}67. (7 – 2) + 1 = 4{,}35$

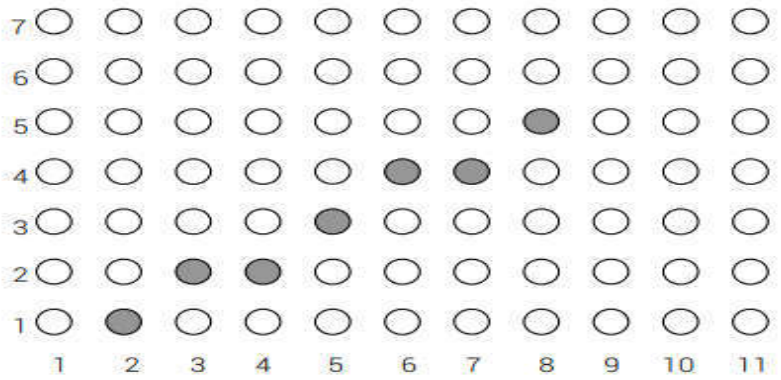
penggabungan ke nilai $y = 4$ gambarlah pada posisi (7,4) dalam layar $x = 7 + 1 = 8$

Pengulangan ke-7:

$x = 8$; $y = 0{,}67. (7 – 2) + 1 = 5{,}02$

penggabungan ke nilai : $y = 5$ gambarlah pada posisi (8,5) dalam layar , maka

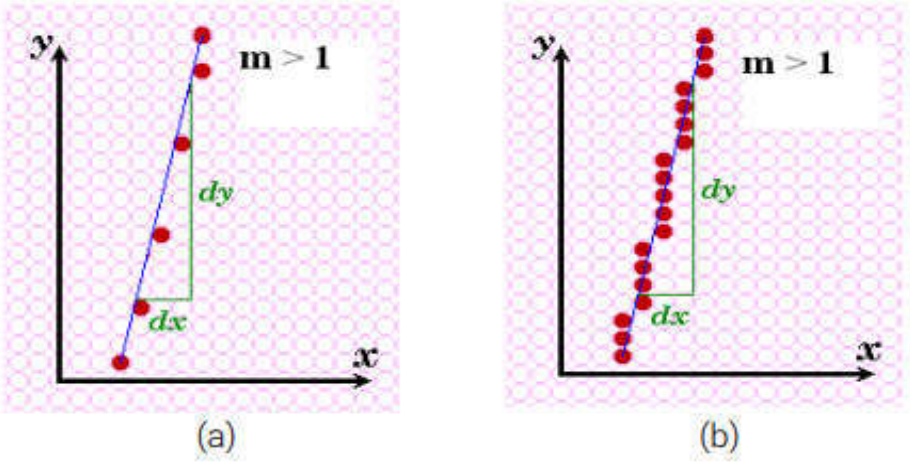
Akan didapatkan posisi titik-titik yang akan membentuk sebuah garis yaitu : (2,1), (3,2), (4,2), (5,3),(6,4), (7,4), (8,5).



Gambar 3. 6 Letak titik kordinat menggunakann perhitungan brute force algorithm dalam grafik raster.

Permasalahan :

Dalam hal gradien *m* > *1*, letak titik pembentuk dari garis terlihat tidak tersambung (gambar 3.7.a) hal ini menyepertemuankan adanya *gap*/jarak diantara piksel, sehingga perlu dibuat interpolasi diantara piksel.



Gambar 3. 7 (a) Gambar garis dengan kemiringan m > 1, terdapat jarak diantara titik (b) posisi setelah dilakukan interpolasi

Selain memakai teknik interpolasi, dalam hal kemiringgan m > 1, pindahkan x dengan y untuk menghilangkan jarak/gap yang ada diantara titik-titik tersebut. Selanjutnya rumus algoritma untuk garis m > 1 dapat dijelaskan seperti di bawah ini :

1) Tentukan kedua titik yang akan menjadi ujung masing-masing sumbu ($x1$,$y1$) dan ($x2$,$y2$)
2) Apabila $x1 = x2$ (garis vertikal), selanjutnya
   a) $y = y + 1$, kemudian x bernilai sama
   b) gambarlah letak posisi (x,y) dalam layar
   c) Langkah sudah selesai
3) Jika $y1 = y2$ (garis horisontal), selanjutnya
   a) $x = x + 1$, kemudian y bernilai sama
   b) gambarlah letak posisi (x,y) dalam layar
   c) Langkah sudah selesai

Misalkan nilai $y2 > y1$, (bila keballikannya, maka ganti $y2$ sehingga menjadi $y1$)

4) Hitunglah gradien dari garis $m = (x2 - x1) / (y2 - y1)$

5) $N = y2 - y1 + 1$

6) $y = y1$

7) Ulangi sejumlah N pengulangan :

   a) $x = m\,(y - y_1) + x_1$

   b) Gabungkan nilai sampai mendekati xa = round (x),

   c) gambarlah letak posisi (xa,y) dalam layar

   d) $y = y + 1$

8) Semua langkah telah selesai.

Langkah tersebut merupakan salah satu contoh dalam mengggunakan Algoritma Brute Force, namun bukan berarrti algoritma ini tidak memiliki kelemahan. Salah satu kelemahan dari penggunaan akgoritma adalah prosesnya yang memakan waktu cukup lama. Hal ini dikarenakan langkah iterasi/ pengulangan memiliki perkalian antar angka pecahan (floating point), pembulatan dann lainnya sampai didapatkan angka yang berbentuk bilangan integer (bilangan bulat).

Contoh 2:

Terdapat 2 (dua) titik yaitu simbol A (4,3) dan juga simbol B (7,8). Diketahui A menjadi titik awal sedangkan letak posisi akhirnya adalah simbol B, dengan algoritma *brute force* buatlah gambar garis yang menghubungkan antara kedua titik tersebut.

Cara Menjawab :

a. titik awalan A (4,3), kemudian simbol B (7,8)

b. Belum terpenuhi

c. Belum terpenuhi

d. $m = (7 - 4)/(8 - 3) = 0{,}6$

e. N = 8 – 3 + 1 = 6

f. nilai y = 3, x = 0,6. (y – 3) + 4

g. Ulangi sejumlah 6 (enam) pengulangan

Pengulangan ke- 1:

y= 3; x = 0,6. (3 – 3) + 4 =4

bulatkan nilai mendekati : x = 4

buat gambar di posisi (4,3) dalam layar y = 3 + 1 = 4

Pengulangan ke- 2:

y= 4; x = 0,6. (4 – 3) + 4 =4,6

bulatkan nilai mendekati : x = 5

buat gambar di posisi (5,4) dalam layar y = 4 + 1 = 5

Pengulangan ke- 3:

y= 5; x = 0,6. (5 – 3) + 4 =5,2

bulatkan nilai mendekati : x = 5

buat gambar di posisi (5,5) dalam layar y = 5 + 1 = 6

Pengulangan ke- 4:

y= 6; x = 0,6. (6 – 3) + 4 =5,8

bulatkan nilai mendekati : x = 6

buat gambar di posisi (6,6) dalam layar y = 6 + 1 = 7

Pengulangan ke- 5:

y= 7; x = 0,6. (7 – 3) + 4 =6,4

bulatkan nilai mendekati : x = 6

buat gambar di posisi (6,7) dalam layar y = 7 + 1 = 8

Pengulangan ke- 6:

y= 8; x = 0,6. (8 – 3) + 4 =7

bulatkan nilai mendekati : x = 7

buat gambar di posisi (7,8) dalam layar $y = 8 + 1 = 9$

Maka akan didapatkan posisi pembentukan garis : (4,3), (5,4), (5,5), (6,6), (6,7), (7,8).

d. Algorithm DDA (Digital Differential Analyzer)

Algoritma DDA memiliki dasar persamaan yaitu (2 - 8). Setiap garis memakai persamaan awal (x1,y1) dan akhiran (x2,y2). Selanjutnya titik koordinat dilakukan penghitungan dan diubah sampai hasilnya menjadi bilangan bulat (integer). Salah satu kelebihan algoritma DDA yaitu dappat dipakai dalam menghitung dengan semua kemiringan garis ($0 < m < 1$; $m>1$; $-1 < m < 0$; $m < -1$). Dibawah ini merupakan cara-cara membuat garis dengan menggunakan algoritma Digital Differential Analyzer (DDA):

a. Pilihlah titik yang ingin dihubungkan dengan membentuk garis lurus.
b. Pilih salah satu titik yang akan menjadi awalan ($x1$, $y1$) sedangkan titik lainnya akan diposisikan menjadi letak posisi di akhir ($x2$, $y2$).
c. Hitunglah : $dx = x2 - x1$, selanjutnya $dy = y2 - y1$
d. Lanjutkan langkah, lakukan perhitungan menggunakan aturan sebagaimana dibawah ini :
   - jika $|dx| > |dy|$ artinya $step = |dx|$
   - jika bukan, artinya $step = |dy|$
e. Hitunglah penjumlahan piksel kordinat menggunakan perhitungan:
   - $x\_inc = dx / step$
   - $y\_inc = dy / step$
f. Kemudian lanjutkan : $x = x + x\_inc$ $y = y + y\_inc$
g. Hitung penambahan angka hingga mendekati $u = \text{Round}(x)$, $v = \text{Round}(x)$, selanjutnya letakan piksel (u, v) pada layar
h. Lakukan lagi langkah ke 6 & 7 dalam membantu mencari letak piksel lainnya hingga didapatkan $x = x2$ dan $y = y2$.

Contoh

Terdapat 2 letak titik yaitu A (2,1) selanjutnya yaitu B(8,5), Apabila simbol posisi A diketahui menjadi letak titik awal, kemudian letak posisi akhirnya adalah titik B, Cobalah untuk membuat garis yang

menghubungkan kedua titik dengan mmemakai algoritma Digital Differential Analyzer.

Cara Menjawab :

Titik awal (x_1, y_1) = A(2,1) dan Titik akhir (x_2, y_2) = B(8,5)

$dx = x_2 - x_1 = 8 - 2 = 6$ dan $dy = y_2 - y_1 = 5 - 1 = 4$

Karena: $|dx| > |dy|$, maka $step = |dx| = 6$

$x\_inc = dx / step = 6/6 = 1$

$y\_inc = dy / step = 4/6 = 0{,}67$

Pengulangan ke-1: $(x,y) = (2,1)$

$x+x\_inc = 2 + 1 = 3$

$y+y\_inc = 1 + 0{,}67 = 1{,}67$

Nilai kordinat berikutnya : $(x,y) = (3; 1{,}67)$

Angka yang mendekati $(3; 1{,}67) \approx (3,2)$. Gambarlah (3,2) dalam layar

Pengulangan ke-2: $(x,y) = (3; 1{,}67)$

$x+x\_inc = 3 + 1 = 4$

$y+y\_inc = 1{,}67 + 0{,}67 = 2{,}34$

Nilai kordinat berikutnya: $(x,y) = (4; 2{,}34)$

Angka yang mendekati $(4; 2{,}34) \approx (4,2)$. Gambarlah (4,2) dalam layar

Pengulangan ke-3: $(x,y) = (4; 2{,}34)$

$xA+x\_inc = 4 + 1 = 5$

$yA+y\_inc = 2{,}34 + 0{,}67 = 3{,}01$

Nilai kordinat berikutnya: $(x,y) = (5; 3{,}01)$

Angka yang mendekati $(5; 3{,}01) \approx (5,3)$. Gambarlah (5,3) dalam layar

Pengulangan ke-4: $(x,y) = (5; 3{,}01)$

$xA+x\_inc = 5 + 1 = 6$

yA+y_inc = 3,01 + 0,67 = 3,68

Nilai kordinat berikutnya: (*x,y*) = (6; 3,68)

Angka yang mendekati (6; 3,68) ≈ (6,4). Gambarlah (6,4) dalam layar

Pengulangan ke-5: (*x,y*) = (6; 3,68)

xA+x_inc = 6 + 1 = 7

yA+y_inc = 3,68 + 0,67 = 4,35

Nilai kordinat berikutnya: (*x,y*) = (7; 4,35)

Angka yang mendekati (7; 4,35) ≈ (7,4). Gambarlah (7,4) dalam layar

Pengulangan ke-6: (*x,y*) = (7; 4,35)

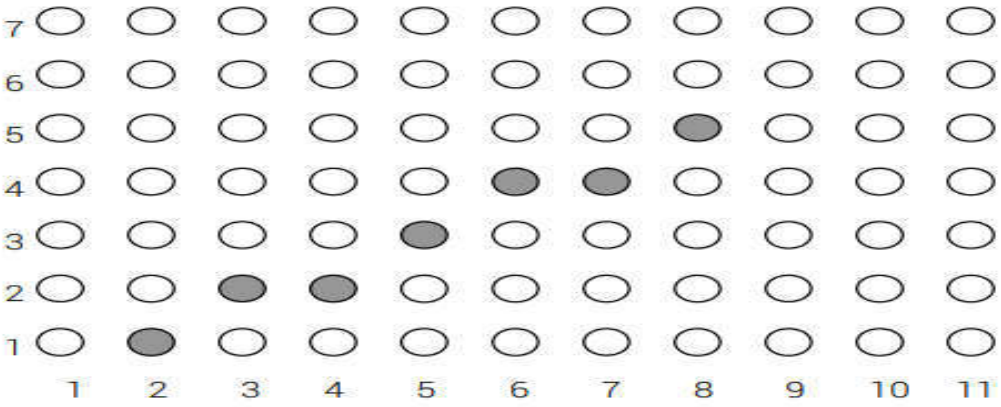
xA+x_inc = 7 + 1 = 8

yA+y_inc = 4,35 + 0,67 = 5,02

Nilai kordinat berikutnya: (*x,y*) = (8; 5,02)]

Angka yang mendekati (8; 5,02) ≈ (8,5). Gambarlah (8,5) dalam layar Setelah didapatkan $x = x_2 = 8$, maka pengulangan tidak diperlukan lagi, Maka didapatkan hasil akhir posisi titik yaitu: (2,1), (3,2), (4,2), (5,3),

(6,4), (7,4) dan (8,5).

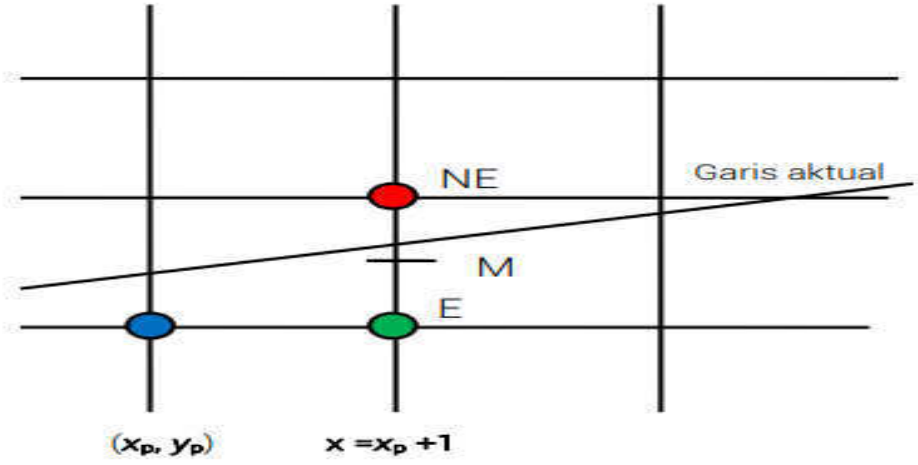


Gambar 3. 8 Gambar raster graphics pemmbentukan garis memakai hasil perhitungan algoritma Digital Differential Analyzer.

Algoritma Digital Differential Analyzer (DDA) memiliki kelebihan jika dibandingkan dengan Algoritma Brute Force. Kelebihan tersebut terletak pada prosesnya yang menghabiskan waktu raltif lebih cepat dan juga lebih baik saat digunakan dengan kemiringan garis m>1. Akan tetapi algoritma DDA juga memiliki kelemahan yaitu prosedur dalam membuat garis tetap diharuskan melakukan pembulatan. Kekurangan lainnya adalah karena variabel *x, y* dan juga *m* tetap mengharuskan angka bilangan real dan bukan bilangan pecahan.

2. Algoritma Bresenham

Algoritma Bresenham pertama kali dicetuskan di tahun 1965 oleh J.E. Bresenham. Temuannya berhasil mengoptimalkan kualitas gambar garis khususnya di dalam *raster graphics*. Algoritma ini sudah tidak memakai *floating point arithmethic*. Dalam algoritma Bresenham, angka pembulatan pada posisi piksel tidak lagi diperlukan. Sehingga iterasi (pengulangan) yang dilakukan cukup memakai sistem *incremental* saja. Kelebihan lainnya adalah karena algoritma ini cukup kompatibel dengan sotware maupun hardware yang digunakan untuk menggambar garis. Kekurangannya adalah karena algoritma ini hanya bisa digunakan untuk posisi kemiringan (0 < m < 1) saja.



Gambar 3. 9 Gambar garis yang terletak antara titik NE dan E dan memiliki garis tengah M.

Perhatikan Gambar 3.9 apabila gradien pada garis *m,* yang yang memiliki nilai 0 < *m* < 1, kemudiaan memiliki posisi awalan pada (*x*0*, y*0) sedangkan posisi

akhirnya berada pada (*x1,y1*). Kita ibaratkan letak simbol/titik saat ini berada di titik (x_p, y_p) sehingga harus ditentukan titik mana yang akan menjadi posisi simbol *East* (E) dan juga *Northeast* (NE). Jika simbol Q merupakan perpotongan antara garis yang memiliki nilia garis $x = x_p+1$, kemudian simbol M merupakan posisi tengah yang terletak antara simbol NE dan jug E. Apabila simbol M yang menjadi letak titik tengah berada di atas dari garis, maka simbol E yang akan dipilih, sebaliknya yaitu apabila simbol M berada di bawah garis, berarti simbol NE yang akan dipilih. Untuk kasus seperti ini sebaiknya ditentukan dimana letak garis, apakah akan terletak diatas titik M atau berada dibawahnya dari titik M tersebut. Dalam menggunakan algoritma ini, hal yang perlu diingat adalah setiap garis dapat didefinisikan seperti penjelasan dibawah ini:

F(x, y) = ax + by + c = 0, Terbentuklah nilai 3 – 1

Hitunglah: $dy = y_1 - y_0$ dan $dx = x_1 - x_0$, berarti dapat kita tulis seperti berikut :

$$y = mx + B = (dy/dx)\, x + B,\ (dy/dx)\, x - y + B = 0$$

Lakukan penghitungan perkalian menggunakan d*x* sehingga (d*y*) – (d*x*)*y* + (d*x*)B = 0 Dan didapatkan $a = dy$, $b = -dx$ dan $c = (dx)B$.

Apabila F (*x, y*) = 0, maka (*x, y*) memiliki letak tepat dia garis

Apabila F (*x, y*) > 0, maka (*x, y*) memiliki letak di bawah dari garis

Apabila F (*x, y*) < 0, maka (*x, y*) memiliki letak di atas dari garis

Dalam menggunakan karakteristik midpoint, maka coba hitung:

F(M) =F (x_p + 1, y_p + ½) = a (x_p + 1) + b (y_p + ½) + c Kemudian ujilah simbol dari d = F(M).

Jika d >0, M memiliki posisi di bawah dari garis, artinya kita pilih NE. Jika d = 0, ambil E (walaupun kita dapat mengambil pilihan NE)

Jika d < 0, M memiliki posisi diatas garis, oleh karena itu kita pilih E.

Karena kita mengambil E, maka langkah M ditambah satu kearah sumbu x. sehingga diperoleh d terbaru yaitu:

$$\boldsymbol{d}_{baru} = F(x_p + 2, y_p + ½) = a\,(x_p + 2) + b\,(y_p + ½) + c$$

$$\boldsymbol{d}_{lama} = F(x_p + 1, y_p + ½) = a\,(x_p + 1) + b\,(y_p + ½) + c$$

jadi $\boldsymbol{d}_{baru} - \boldsymbol{d}_{lama} = a = dy.$

Setelah kita memilih E, penjumlahan dalam *d* baru yaitu *a* = *dy*. Apabila kita memilih NE, maka lakukan penambahan M menuju ke *x* dan *y,* maka akan didapatkan :

$$d_{baru} = F(x_p + 2, y_p + 3/2) = a\,(x_p + 2) + b\,(y_p + 3/2) + c$$

$$d_{lama} = F(x_p + 1, y_p + ½) = a\,(x_p + 1) + b\,(y_p + ½) + c$$

jadi $d_{baru} - d_{lama} = a + b = dy - dx.$

Ketika algoritma dimulai maka persamaan awalnya adalah sebagai berikut (x_p , y_p) = ($x0$, $y0$). Dalam menjumlahkan nilai d di awal algoritma, maka langkah midpoint nya adalah ($x0$ + 1, $y0$ + ½) dan

$$F(M) = F(x_0 + 1, y_0 + ½) = a\,(x_0 + 1) + b\,(y_0 + ½) + c$$
$$= ax_0 + by_0 + c + a + b/2$$
$$= 0 + a + b/2 \quad \text{karena } (x_0, y_0) \text{ terletak pada garis.}$$

Jadi kita mulai dari $d = a + b/2 = dy - dx/2$

Supaya angka tersebut hilang, lakukan perhitungan F, caranya adalah melakukan perkalian 2, sehingga didapatkan F(*x,y*) = 2 (*ax* + *by* + *c*) = 0. Perhitungan ini membuat nilainya menjadi *d* = 2*dy* – *dx* Sehingga langkah-langkah memakai algortima bresenham midpoint (dengan nilai kemiringan 0 < m < 1) :

a. Pilihlah titik-titik yang ingin dihubungkan dalam membentuk sebuah garis.
b. Pilihlah dimana letak posisi awal (*x*0, *y*0) juga letak posisi akhir (*x*1, *y*1).
c. Hitunglah *dx*, *dy*, 2|*dy*| dan 2|*dy*| − 2|*dx*|
d. Jumlahkan parameter : *p*o = 2|*dy*| − |*dx*|
e. Dalam setiap *xk* sejajar dengan garis, kita mulai menggunakan k = 0

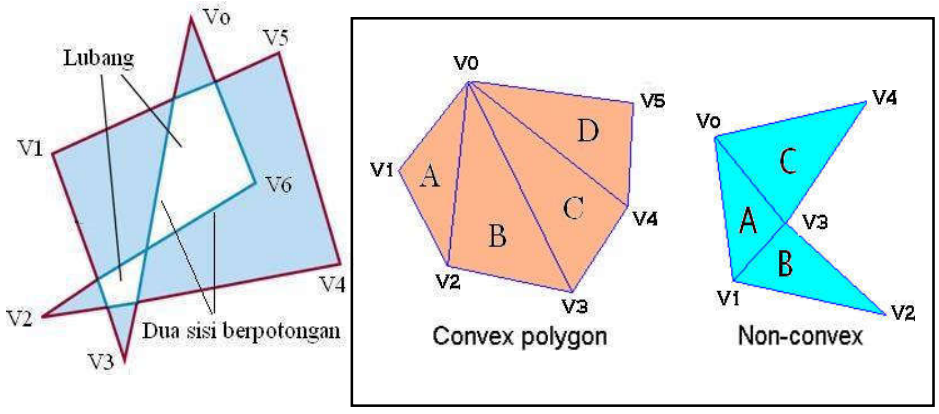
- jika $p_k < 0$ maka letak berikutnya yaitu : (x_k+1, y_k) dan $p_{k+1} = p_k + 2|dy|$
- jika bukan, maka posisi titik berikutnya yaitu :

$$(x_k+1, y_k+1) \text{ dan } p_{k+1} = p_k + 2|dy| - 2|dx|$$

f. Ulang langkah ke- 5 dalam mendapatkan dimana letak pixel selanjutnya hingga didapatkan $x = x1$ dan $y = y1$.

3. Switch

. Bentuk poligon merupakan gabungan dari beberapa garis lurus yang masing-masing terhubung sampai menjadi sebuah bentuk luasan (bangun luas). Garis di dalam bentuk poligon dikenal dengan nama edge (garis sisi poligon). Selain edge, dalam bentuk poligon terdapa pula yang disebut verteks yang merupakan titik dari pertemuan dua sisi setiap garis. Bentuk poligin umumnya disebut dengan kordinat verteks. Secara umum dikenal dua jenis poligon yaitu, poligon sederhana dan juga poligon tidak sederhana. Masing-masing poligon memiliki cirinya tersendiri. Untuk poligon sederhana memiliki ciri yaitu tidak semua letak posisi verteksnya ada di dalam bidang yang serupa, kemudan ciri lainnya adalah tidak memiliki sisi yang berpotongan, serta tidak memiliki lubang dalam bentuknya.



Gambar 3. 10 Poligon berbentuk sederhana (a) dan poligon bentuk tidak sederhana (b)

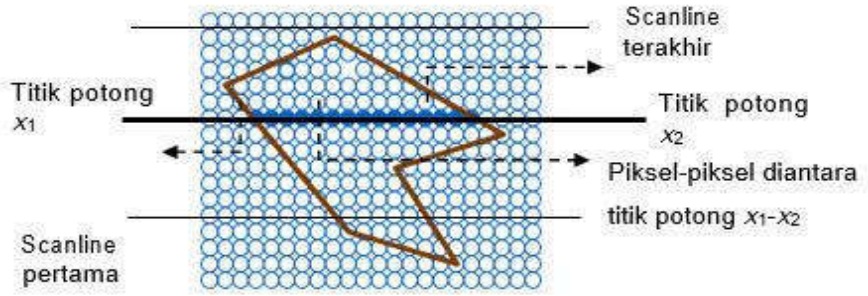
Secara umum poligon dapat dibedakan ke dalam dua bagian, pertama convex poligon dan kedua yaitu non convex poligon. Kedua poligon ini memiliki cirinya masing-masing. Untuk convex poligon bisa dilihat bahwa sudut interior poligonnya kurang dari 180 derajat, bisa dikatakan bahwa segmen yang dibuat

dari dua verteks berbentuk sembarang terletak di dalam poligon. Bangun poligon memiliki bentu sederhana yaitu segitiga, hal ini dikarenakan setiap poligon dapat dibuat menjadi bagian kecil yang akan berbentuk segitiga seperti yang terlihat dalam gambar 3.12 (b). Dalam prakteknya kita dapat membuat bentuk permukaan semua objek 3D apabila kita memiliki poligon dalam jumlah yang memadai.

Dalam dunia grafik, penambahan warna dilakukan sebagai upaya memperindah bentuk dan tampilan dari poligon. Maka, untuk melakukan hal tersebut dibutuhkan teknik algortima tersendiri dengan tujuan memberi warna did dalam poligon. Penambahan warna di dalam poligon disebut dengan teknik *area filling* atau *filling poligon*. Terdapat dua cara yang umum digunakan dalam melakukan *filling poligon* dalam sistem raster. Kedua teknik tersebut adalah *Boundary Fill Algorithm* dan *Scan Line Poligon Fill Algorithm.* Kita akan membahas penggunaan Scan Line Poligon Fill Algorithm terlebih dahulu.

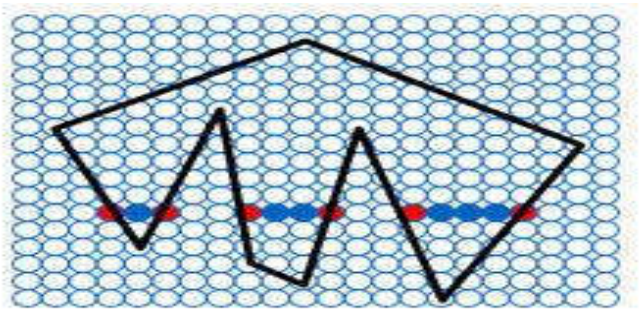
a. *Scan Line Poligon Fill Algorithms*

Cara ini dilakukan dengan penambahan warna dalam poligon melalui scan dengan arah horizontal dari sisi kiri ke arah sisi kanan. Teknik ini dipakai supaya didapatkan titik potong yang tepat dengan bagian pinggir poligon. Selanjutnya adalah mengurutkan titik perpotongan x sisi kiri menuju sisi kanan untuk selanjutnya diberikan pewarnaan di pixel yang ada antara kedua poligon yang bersisian atau ($x1$- $x2$). Teknik *scan line* dimulai dengan garis terletak di bagian bawah yang memiliki nilai *y* yang nilainya kecil sampai ke bagian garis scan yang terletak di bagian atas. Teknik ini dapat dipakai untuk melakukan penambahan warna dalam objek-objek sederhana misalnya elips, lingkaran dan bentuk sederhana lainnya. Seperti terlihat pada gambar berikut :



Gambar 3. 11 ilustrasi penggunaan metode scan line

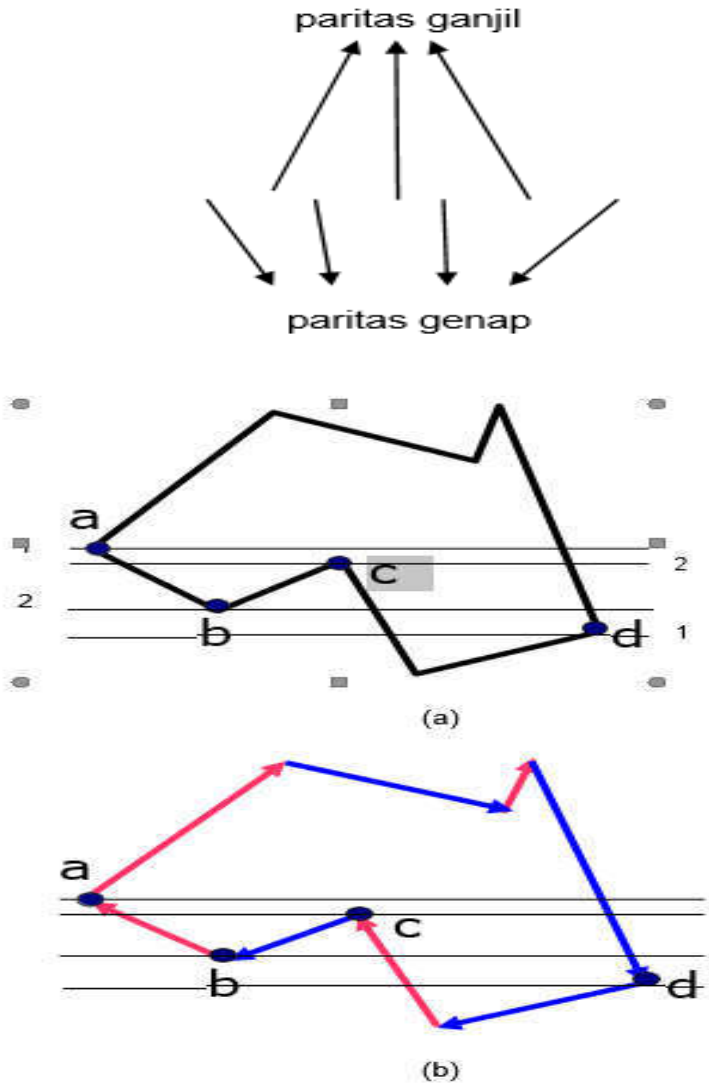
Permasalahan umum yang sering terjadi adalah bagaimana kita bisa menentukan apakah piksel tersebut ada di dalam atau di luar poligon. Salah satu caranya adalah dengan menggunakan *Inside-Outside test.* Dalam membedakan antara bagian sisi dalam dengan sisi luar, kita dapat menggunakanj aturan paritas ganjil genap (*odd even rule*). Langkah awal yang dilakukan adalah menngatur paritas dengan nilai paritas genap. Apabila kita menemukan titik potong dimana nilai paritas yang dibalik, maka yang semula nilainya genap kita balik sehingga menjadi nilai ganjil begitupula dengan sebaliknya. Kemudian adalah kita memberi warna piksel apabila nilai paritasnya adalah ganjil.



Gambar 3. 12 Ilustrasi penggunaan aturan paritas dan pengisian warna

Permasalahan lainnya antara lain bisa kita lihat pada gambar 3.15. Dapat dilihat bahwa verteks dari a, kemudian b, c dan juga d yang menjadi titik bertemunya kedua segmen garis tersebut. Pertanyaannya adalah

kenapa verteks a dan juga d hanya dilakukan penghitungan sebanyak satu kali saja, berbeda dengan verteks b dan c yang dilakukan dengan 2 kali penghitungan ?

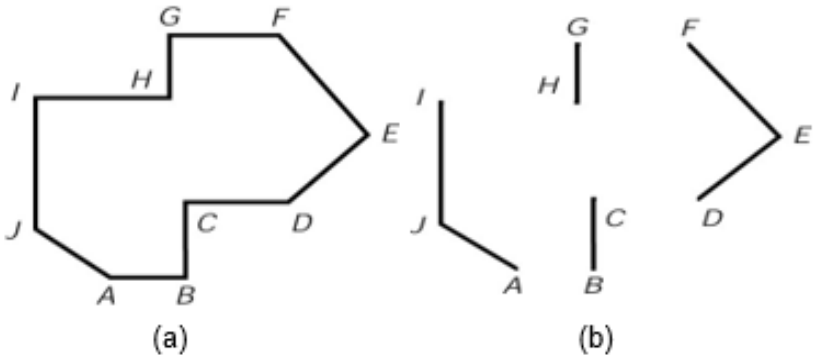


Gambar 3. 13 ilustrasi penghitungan aturan paritas dalam gambar verteks.

Jalan keluar untuk mengatasi hal ini dapati dilakukan cara mengubah arah putaran bagian tepi poligon menjadi searah dengan jarum jam, yang dapat dilihat di gambar 3.15.(b). Periksalah apabila terdapat verteks yang arah panahnya mengalami perubahan (dari mengarah keatas menjadi

kebawah, begitupula sebaliknya) mka penghitungan dalam verteks itu kita hitung sebanyak dua kali hitungan. Namun, apabila tidak terdapat perubahan arah panah pada verteks, artinya verteks tersebut hanya dihitung sekali saja.

Permasalahan lain yang umum terjadi adalah bagaimana cara melakukan perhitungan di garis tepi horizontal. Seperti yang terlihat dalam gambar 3.16.(a) Bagaimana cara menghitung bagian pinggir horizontal AB, CD, GF, dan juga HI ?



Gambar 3. 14 Bentuk poligon dan bagian sis horizontal

Cara menyelesaikan perhitungan tersebut adalah dengan mengabaikan verteks yang memiliki letak di pinggir dari horizontal dan jangan dimasukkan sebagai bagian dari perhitungan aturan paritas, seperti dapat dilihat pada gambar 3.16.(b).

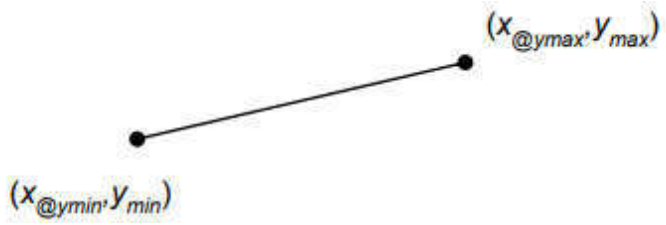
Algoritma pengisian warna scan line memakai aturan paritas ganjil genap sebagai berikut :

1) Pilihlah titik potong garis dengan semua bagian dari poligon
2) Atur dan urutkan semua titik dengan mendasarkan pada kordinat sumbu x
3) Beri warna pada semua piksel yang ada pada pasangan titik (x1-x2) yang posisinya berada di dalam poligon dengan menggunakan aturan paritas ganjil – genap.

Akan tetapi algoritma *scan fill* juga memiliki kelemahan. Kelemahan dari algoritma *scan fill* adalah diperlukan biaya yang besar, hal ini dikarenakan semua sisi poligon perlu dilakukan pengjujian khususnya terhadap masing-masing piksel.

Salah satu jalan keluar untuk mengatasi kelemahan tersebut adalah dengan penggunaan *edge table concept*. *Edge Table* (ET) sendiri merupakan sebuah tabel berisi sisi- sisi dari poligon. Selain itu, terdapat juga *Active Edge Table* (AET) dan juga *Global Edge Table* (GET). Kedua edge table ini memiliki fungsi yang berbeda, AET memiliki fungsi menyimpan seluruh bagian sisi dari poligon yang memiliki perpotongan dengan garis scan. Kemudian, untuk GET fungsinya adalah menyimpan seluruh bagian sisi dari poligon dan melakukan update dari AET.

Cara perhitungan garis scan dan titik potongnya :



Persamaan Umum :

$$y = mx + b, \quad m = \frac{y_{\max} - y_{\min}}{x_{@y\max} - x_{@y\min}}$$

Semua *scan line* baru maka ketentuan yang digunakan yaitu :

$$y_{i\square 1} \square y_i \square 1$$

Perhitungan x kepada semua *scan line* :

$$x = \frac{y - b}{m}$$

.

Sehingga didapatkan persamaan seperti berikut :

$$x_i = \frac{y_i - b}{m}, \qquad x_{i+1} = \frac{y_{i+1} - b}{m}$$

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

Maka hasil akhirnya adalah :

$$x_{i+1} = \frac{y_i + 1 - b}{m} = \frac{y_i - b}{m} + \frac{1}{m} = x_i + \frac{1}{m}$$

Dimana hal ini merupakan salah satu teknik yang memiliki efisiensi yang baik dalam melakukan perhitungan terhadap nilai *x*.

1) Penjelasan AET
   a) Merupakan tabel yang mengandung setiap entry dari sisi yang berpotongan dengan *scan line* yang aktif.
   b) Untuk semua *scan line* baru maka :
      (1) Lakukan perhitungan perpotongan baru pada seluruh sisi dengan memakai rumus :
      (2) Menambahkan seluruh sisi yang baru terbentuk dan berpotongan.
      (3) Kemudian hilangkan/hapus seluruh bagian sisi yang tidak memiliki perpotongan.
      (4) Dalam memaksimalkan efisiensi dalam update Active Entry Table, maka kita harus menjaga Global Entry Table.
2) Penjelasan GET
   a) GET merupakan tabel yang memuat informasi mengenai seluruh bagian sisi dari poligon.
   b) GET memiliki sebuah tempat untuk tiap-tiap *scan line*.
      (1) Seluruh tabel mempunyai daftar setiap bagian sisi dan juga memiliki nilai ymin
      (2) Tiap-tiap bagian sisi bisa ditempatkan ke dalam satu tempat.
   c) Setiap entry untuk di GET memiliki :
      (1) Angka *y* max di setiap bagian sisi
      (2) Angka *x*@ymin (angka *x* dalam letak posisi y_{min})
      (3) Angka kenaikan dari *x* (1/m)

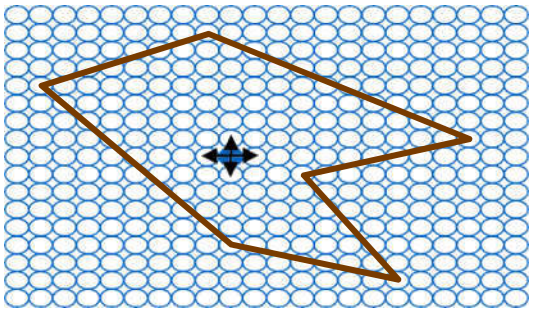
Penggunaan teknik ***Scan Line Poligon Fill Algorithms*** memakai bantuan GET, ET dan juga AET. Scan

1) Jumlahkan bagian sisi dari poligon ke dalam GET
2) Aturlah *y* ke kordinat *y* yang paling kecil ke arah GET
3) Aturlah set AET = habis
4) Ulangi langkah tersebut sampa AET dan GET benar-benar kosong
   a) Jumlahkan bagian sisi GET ke dalam AET dimana $y_{min} = y$.
   b) Hapus bagian sisi AET jika $y_{max} = y$.
   c) Susun secara berurutan AET dengan berdasar kepada *x*
   d) Beri warna pada piksel yang posisinya berada diantara titik potong yang terdapat di AET.
   e) Disetiap bagian sisi dalam dari AET, ubah nilai x menjadi x = 1/m
   f) Aturlah $y = y + 1$ supaya berpindah ke bagian line selanjutnya.

b. Boundary-Fill Algorithm

Penggunaan teknik Boundary Fill Algorithm memiliki prosedur dengan 3 parameter utama. Parameter tersebut antara lain kordinat dari letak (x,y), kemudian pewarnaan pengisian kemudian batas garis warna. Jadi, penambahan warna dalam poligon diawali di kordinat letak (x,y), selanjutnya adalah mengecek letak dari titik tetangganya, apakah terdapat warna batas atau tidak. Apabila tidak terdapat warna batas, maka beri warna titik itu dengan menggunakan warna tertentu. Kemudian, cek kembali letak dan warna titik tetangganya. Ulangi proses itu sampai semua titik dalam wilayah pengisian sudah selesai diuji semuanya.

Dalam teknik Boundary Fill, proses pengisian warna diawali dengan sebuah titik yang terletak di dalam wilayah (x,y) poligon kemudian beri warna piksel hingga seluruh piksel di dalam bentuk poligon selesai diberikan warna, kita dapat melihatnya dalam gambar 3.15.

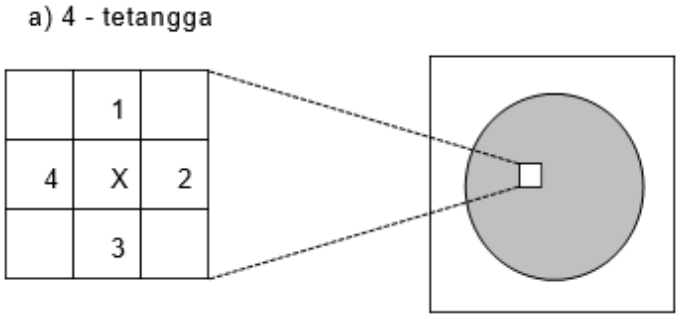


Gambar 3. 15 Ilustrasi penggunaan Boundary Fill pada raster

Untuk mengetahui dimana letak titik tetangga berada, terdapat 2 (dua) macam cara dalam melihat letak dari titik tetangga tersebut, yaitu :

1) Mencari titik yang memiliki posisi di atas, sebelah kanan, sebelah kiri dan bagian bawah dari titik pusat, 4 titik tetangga.
2) Mencari titik yang memiliki posisi di atas, kanan, kiri, pojok kiri atas, pokok kanan atas, pojok kanan bawah, pojok kiri bawah dan bagian bawah dari titik pusat, 8 titik tetangga.
3) Mencari letak posisi yang terletak diatas, kanan, bawah, kiri, pojok kanan atas, pojok kiri atas, pojok kanan bawah dan pojok kiri bawah.

Contohnya bisa terlihat seperti gambar 3.18 :



b)8-tetangga

1	2	3
8	X	4
7	6	5

Gambar 3. 16 Ilustrasi bentuk titik tetangga, 4 titik dan 8 titik tetangga

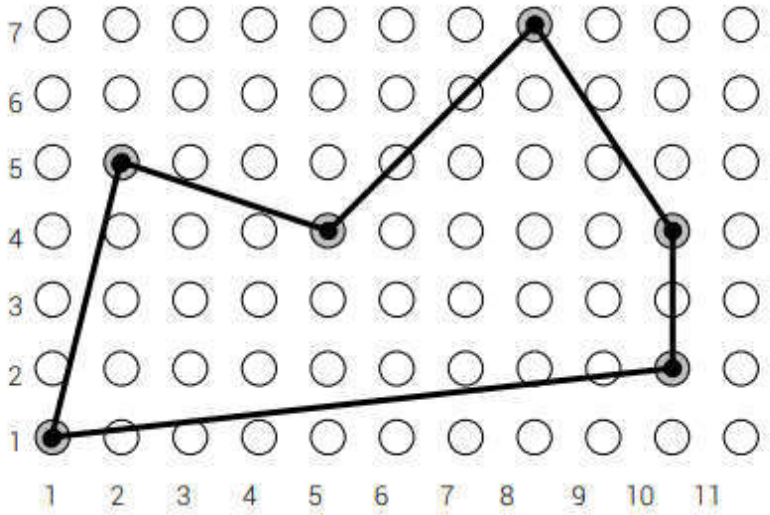
Contoh 1:

Contoh lainnya, Terdapat sebuah poligon dengan titik = (1,1), (2,5), (5,4), (8,7), (10,4), (10,2) dan (1,1), dengan memakai algoritma Boundary Fill 4-Connected, lakukan pengisian warna terhadap poligon tersebut,

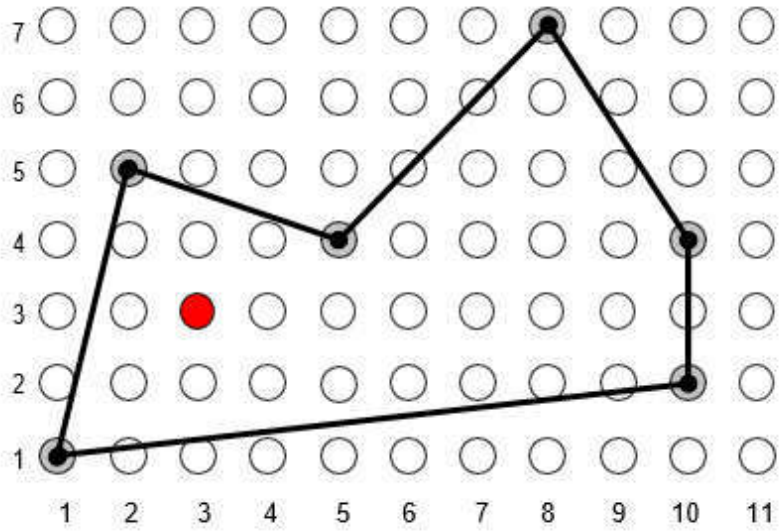
Cara Menjawab :

Diketahui titik pembentuk bangun poligon = {(1,1), (2,,5), (5,4), (8,7), (10,4), (10,2), (1,1)}.

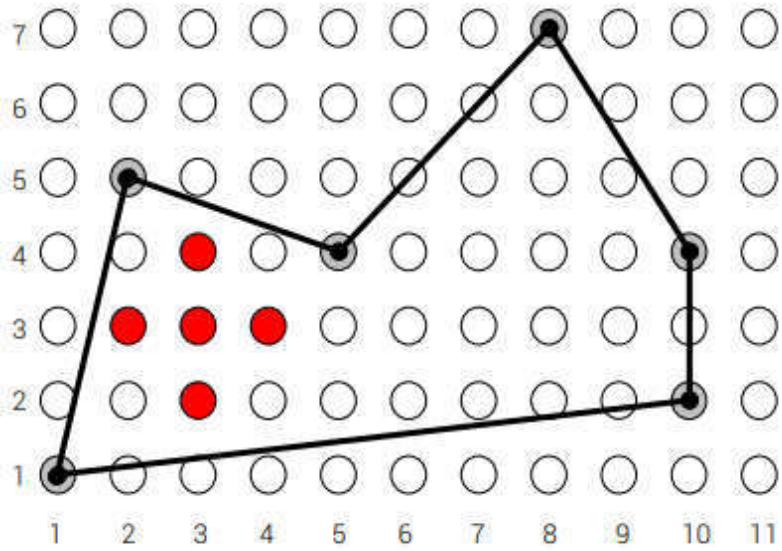
Jika dilakukan penggambaran, maka akan didapatkan gambar poligon seperti berikut ini :



Jika titik awalan yaitu (3,3), maka berikan tanda di posisi kordinat (3,3) menggunakan warna merah sebagai contohnya. Selanjutnya lihat ke 4 bagian titik tetangganya (3,4), (3,2), (4,3), (2,3) seperti berikut ini



Keempat titik itu bukanlah garis batas dari poligon, maka keempat titik itu diberi warna merah.



Letak titik yang sudah selesai mengalami pemrosesan : (3,3)

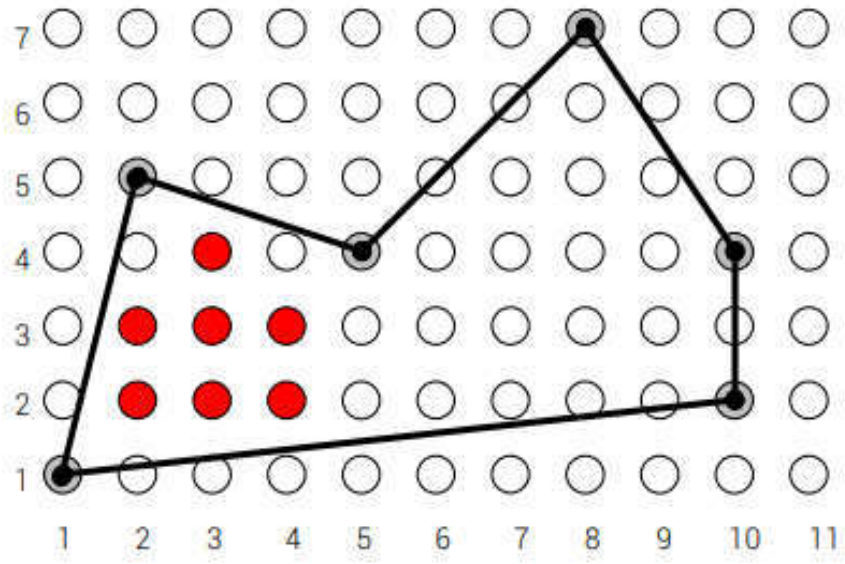
Letak titik belum mengalami pemrosesan : (3,2), (3,4), (2,3), (4,3)

Pilih titik (3,2).

Letak titik sudah selesai mengalami pemrosesan: (3,2), (3,3)

Letak titik belum mengalami pemrosesan : (3,4), (2,3), (4,3)

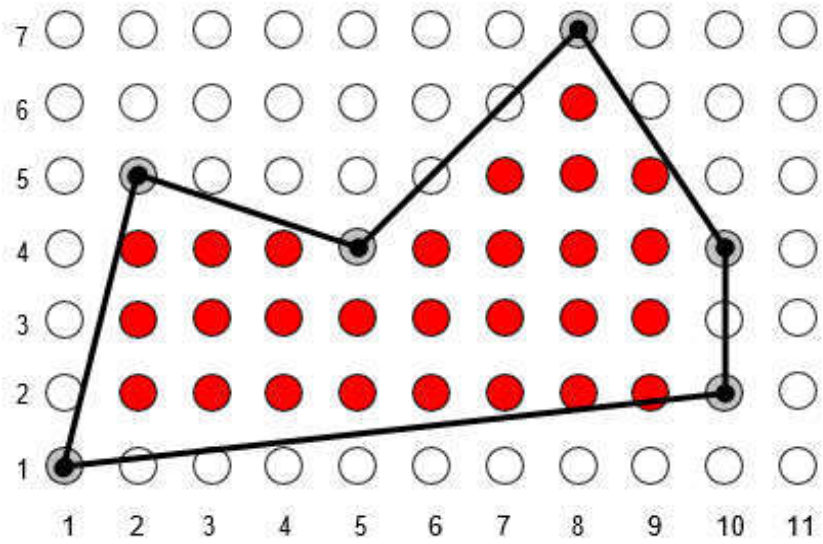
Keempat letak kordinat itu antara lain (3,3), (3,1), (2,2), (4,2). Dapat dilihat di kordinat (4,2) kemudian titik (2,2) bukanlah merupakan titik yang menjadi batas dari poligon. Maka titik tersebut juga diberikan pewarnaan merah. Selanjutnya kordinatk (3,3) sudah diberikan warna, kordinat (3,1) yang menjadi pembatas dari poligon tersebut tidaklah perlu diberikan warna.



Letak titik yang sudah selesai mengalami pemrosesan : (3,2)(3,3)

Letak titik belum mengalami pemrosesan : (2,3), (3,4), (4,2) (2,2), (4,3)

Pilih titik (3,4). Keempat titik tetangga tersebut terletak pada titik (3,5), (3,3), (4,4), (2,4). Titik (3,3) diberikan warna. Sedangkan titik yang terletak di (2,4), (4,4) dan (3,5) menjadi garis batas sehingga tidak diberikan warna. Ulangi proses tersebut sampai semua bagian di dalam bangun poligon diberi warna merah,


7
6
5
4
3
2
1
1 2 3 4 5 6 7 8 9 10 11

C. SOAL LATIHAN/TUGAS

Latihan	Petunjuk Pengerjaan Tugas
Latihan Pertemuan 3	1. Terdapat dua simbol yaitu simbol A dan simbol B. Jika simbol A dijadikan sebagai posisi awalan dan simbol B menjadi posisi akhir, cobalah buat titik penghubung diantara simbol A dan simbol B sampai menjadi sebuah garis AB memakai algoritma : a. Brute Force Algorithm b. Digital Differential Analyzer Algorithm c. Bresenham Algorithm Dengan menggunakan ketentuan sebagai berikut : Titik A terletak pada kordinat (3,2) Titik B terletak pada kordinat (11,6)

D. REFERENSI