Pengantar Computer Graphics Algoritma Dasar

Vincent Suhartono

Bacaan Teknologi Informasi

Pengantar
Computer Graphics
Algoritma Dasar

Penulis:

Vincent Suhartono

Bacaan Teknologi Informasi

Pengantar Computer Graphics: Bacaan Teknologi Informasi

Penulis :

Vincent Suhartono

ISBN : 978-602-72713-3-3

Editor :

Claudia Clarentia Ciptohartono

Penyunting :

Totok Sutoyo

Desain Sampul dan Tata Letak :

Claudia Clarentia Ciptohartono

Penerbit :

CV Mulia Jaya

Redaksi :

Jalan Anggajaya II No. 291-A,

Condong Catur

Kabupaten Sleman, Yogyakarta

Telp : 0812-4994-0973

Email : cv.muliajaya291@yahoo.com

Cetakan Pertama April 2016

KATA PENGANTAR

Puji sukur, penulis panjatkan kehadirat Allah YME yang telah memberikan segala karunia yang tak terhingga sehingga penulis dapat menyelesaikan buku ini. Buku ini ditulis dengan maksud untuk membantu pelajar dan mahasiswa sehingga dapat mengerti dan memahami seluk beluk *Computer Graphics* meskipun hanya dalam bentuk teori yang masih sangat mendasar.

Terima kasih sebesar-besarnya kepada semua pihak yang telah membantu dalam pembuatan buku ini, yaitu:

1. Universitas Dian Nuswantoro
2. Koordinator Matakuliah *Computer Graphics*
3. Teman-teman yang tidak dapat disebutkan satu per satu namanya

Kami menyadari bahwa dalam buku ini masih memiliki sejumlah kekurangan. Oleh karena itu, saran dan komentar sangat dinantikan untuk perbaikan selanjutnya. Meskipun demikian, kami tetap berharap semoga buku ini dapat membantu dan bermanfaat.

Semarang, 12 Januari 2016

Penulis

DAFTAR ISI

KATA PENGANTAR iii
DAFTAR ISI iv
DAFTAR GAMBAR vii
DAFTAR TABEL xi
Bab 1 12
Pendahuluan 12
1.1 Aplikasi *Computer Graphics* 14
1.2 *Graphics Languages* 20
1.3 Sistem Pemrograman Grafis 22
1.4 Komponen *Graphics Library* 22
Ringkasan Bab 1 24
Soal-Soal Latihan 25
Bab 2 26
Spesifikasi Graphics Primitif 26
2.1 *Pixel (Picture Element)* 27
2.2 Warna 31
2.3 Garis 33
Bab 3 37
Algoritma, Analisa, dan Tampilan Graphics Dua Dimensi (2D) 37
3.1 Algoritma Brute Force 38
3.2 Algoritma DDA (*Digital Diferential Analyzer*) 48
3.3 Algoritma Bressenham 53
3.4 Lingkaran 72
3.4.1 Simetris Delapan Titik 74
3.4.2 Algoritma Midpoint 75
3.5 Polygon 82

3.6 Filling Polygon 84
3.6.1 *Scan Line Polygon Fill Algorithms* 84
3.6.2 *Boundary-Fill Algorithm* 102
3.6.3 *Flood-Fill Algorithm* 108
Ringkasan Bab 3 110
Soal-Soal Latihan 111
Bab 4 113
Atribut Output Primitif 113
4.1 Atribut Titik 114
4.2 Atribut Garis 114
4.3 Tipe Garis 114
4.3.1 Ukuran Garis 115
4.3.2 Pen dan Brush 116
4.4 Warna Garis 117
Ringkasan Bab 4 118
Soal-Soal Latihan 119
Bab 5 121
Transformasi Geometri 121
5.1 Translasi (Pergeseran) 122
5.2 *Scalling* (Penskalaan) 123
5.3 Rotasi (Perputaran) 125
5.4 Sistem Koordinat Homogen 2D 126
5.5 Komposisi Matrik Transformasi 2D 129
5.8 Transformasi Geometri 3D 134
Ringkasan Bab 5 138
Soal-Soal Latihan 140
Daftar Istilah 142

Daftar Pustaka 144
Biografi Penulis 145

DAFTAR GAMBAR

Gambar 1. 1 Aplikasi grafika komputer pada bidang hiburan berupa film-film animasi 3D 14

Gambar 1. 2 (a) Visualisasi *Scientific* (b) Visualisasi aliran turbulen dari angin topan..... 15

Gambar 1. 3 Pembelajaran transformasi Fourier untuk peningkatan kualitas citra 17

Gambar 1. 4 Gambar kartun, digambar menggunakan Paint under window 18

Gambar 1. 5 Hasil perbaikan citra yang terkena noise domain frekuensi menggunakan transformasi Fourier..... 19

Gambar 1. 6 GUI dari aplikasi-aplikasi pada sistem operasi Windows XP 20

Gambar 1. 7 Sistem Pemrograman Grafis 22

Gambar 2. 1 Layar berisi M baris *piksel* dan N kolom *piksel.* Sebuah piksel (titik hitam) terletak pd (3,2)..... 27

Gambar 2. 2 Garis lurus yang melalui titik $P_1(x_1,y_1)$ dan $P_2(x_2,y_2)$. 34

Gambar 3. 1 Pembulatan nilai *y* dilakukan karena posisi pixel adalah bulat (integer)..... 39

Gambar 3. 2 Titik-titik pembentuk garis hasil perhitungan menggunakan algoritma *Brute Force* digambar pada *raster graphics.* 43

Gambar 3. 3 (a) Garis dengan kemiringan $m > 1$, tampak bahwa garis tidak kontinyu (b) setelah dilakukan interpolasi garis menjadi kontinyu..... 43

Gambar 3. 4 Titik-titik pembentuk garis hasil perhitungan menggunakan algoritma DDA digambar pada *raster graphics.* .. 52

Gambar 3. 5 Garis mempunyai kemiringan 0 < m < 1, pada titik $x = x_p+1$, garis berada diantara dua titik E dan NE yang mempunyai titik tengah di M. 54

Gambar 3. 6 Titik-titik pembentuk garis dengan kemiringan 0 < m < 1, hasil perhitungan menggunakan algoritma Bressenham yang digambar pada *raster graphics.* 60

Gambar 3. 7 Titik-titik pembentuk garis dengan kemiringan −1 < m < 0, hasil perhitungan menggunakan algoritma Bressenham yang digambar pada *raster graphics.* 64

Gambar 3. 8 Terjadinya gaps yang disebabkan adanya perubahan gradient menghasilkan posisi koordinat piksel yang tidak merata 73

Gambar 3. 9 Delapan titik simetris pada lingkaran 74

Gambar 3. 10 Garis lingkaran berada diantara titik (x_k+1, y_k) dan (x_k+1, y_k − 1) 75

Gambar 3. 11 Posisi piksel pada pembentukan lingkaran dengan titik pusat (0,0) dan jari-jari 8 82

Gambar 3. 12 Polygon sederhana dan polygon tidak sederhana 83

Gambar 3. 13 : Permukaan bola, torus dan teko yang terbentuk dari banyak polygon. 84

Gambar 3. 14 Metode scan-line 85

Gambar 3. 15 Aturan paritas ganjil-genap untuk mengisi warna 86

Gambar 3. 16 Perhitungan paritas pada verteks **a, b, c,** dan **d.** .. 87

Gambar 3. 17 Polygon dengan sisi-sisi horisontal 88

Gambar 3. 18 92

Gambar 3. 19 93

Gambar 3. 20 94

Gambar 3. 21 95

Gambar 3. 22 95

Gambar 3. 23 96

Gambar 3. 24 96
Gambar 3. 25 97
Gambar 3. 26 97
Gambar 3. 27 98
Gambar 3. 28 98
Gambar 3. 29 99
Gambar 3. 30 99
Gambar 3. 31 100
Gambar 3. 32 100
Gambar 3. 33 101
Gambar 3. 34 101
Gambar 3. 35 102
Gambar 3. 36 Metode *Boundary-Fill* 103
Gambar 3. 37 104
Gambar 3. 38 104
Gambar 3. 39 105
Gambar 3. 40 105
Gambar 3. 41 106
Gambar 3. 42 107
Gambar 3. 43 108
Gambar 3. 44 penggantian warna obyek menggunakan *Flood-Fill Algorithm* . (a) lingkaran berwarna merah, segitiga berwarna hijau dan persegi berwarna biru. Obyek tersebut warnanya diubah menjadi (b) lingkaran berwarna abu-abu, segitiga berwarna kuning dan persegi berwarna coklat 108

Gambar 4. 1 Type garis : solid line, dashed line, dotted line dan dashed-dotted line 115

Gambar 4. 2 Ukuran garis dari ¼ pt sampai dengan 6 pt pada software aplikasi Microsoft Word. 116

Gambar 4. 3 Beberapa bentuk pen atau brush pada paket program aplikasi Paint 116

Gambar 5. 1 130

Gambar 5. 2 131

Gambar 5. 3 131

Gambar 5. 4 132

Gambar 5. 5 133

DAFTAR TABEL

Tabel 2. 1 28
Tabel 2. 2 29
Tabel 2. 3 31
Tabel 2. 4 32
Tabel 2. 5 33

Bab 1

Pendahuluan

TUJUAN PEMBELAJARAN

- Agar pembaca mempunyai wawasan mengenai aplikasi grafika komputer di berbagai bidang.

OUTCOME PEMBELAJARAN

- Pembaca mampu menjelaskan pengertian grafika komputer dan bisa memberikan beberapa contoh aplikasi grafika komputer di berbagai bidang.

Pendahuluan

Berbagai informasi yang divisualisasikan dalam bentuk gambar sudah ada jauh sebelum komputer ditemukan. Sebagai contoh, gambar-gambar zaman purba yang masih terdapat pada gua-gua, juga tulisan Hieroglyph pada zaman mesir kuno juga dalam bentuk gambar. Pepatah mengatakan bahwa: "Satu gambar dapat menerangkan 1000 kata – dan tentu lebih bisa menerangkan 1000 bilangan".

Suatu laporan (informasi) yang disajikan dalam bentuk angka-angka sangat menyulitkan pembaca untuk bisa memahami secara cepat, bahkan konsentrasi akan menurun dalam waktu yang tidak begitu lama. Untuk menghindari hal ini maka informasi tersebut

perlu diubah hingga menjadi suatu gambaran grafis, sehingga jauh lebih mudah untuk dicerna. Masalah ini menjadi sangat penting jika laporan tadi akan dipergunakan sebagai acuan pengambilan keputusan, seperti misalnya untuk pembelian. Presentasi yang baik biasanya disertai dengan visualisasi terutama untuk tabel dengan bilangan yang sangat banyak. Juga arah pengembangan komputer sendiri tidak lepas dari visualisasi grafis. Sebagai contoh adalah GUI (*Graphical User Interface*), yang kita pakai setiap hari dalam penggunaan komputer. Lebih jauh lagi komputer sebagai bagian dari multimedia dan broadcasting dituntut untuk dapat mengolah gambar yang bergerak (video). Pengetahuan tentang grafik programming sangat diperlukan bahkan sebagai syarat untuk pengolahan gambar. Sehingga perlu adanya pengantar atau penggalian kembali topik akan grafika komputer.

Grafika komputer (*Computer graphics*) adalah bagian dari ilmu komputer yang mempelajari cara-cara pembuatan dan manipulasi gambar secara digital, sehingga dapat memudahkan komunikasi antara manusia dan komputer, atau manusia dengan manusia melalui gambar-gambar, bagan-bagan, tabel dan lain-lain. Teknik-teknik yang dipelajari dalam grafika komputer adalah teknik-teknik bagaimana membuat atau menciptakan gambar dengan menggunakan komputer. Bentuk sederhana dari grafika komputer adalah grafika komputer 2D, dengan teknik-teknik tertentu kemudian berkembang menjadi grafika komputer 3D.

1.1 Aplikasi *Computer Graphics*

Peran grafika komputer dalam menghasilkan gambar sangat penting sekali untuk perkembangan berbagai software aplikasi. Banyak sekali aplikasi-aplikasi tertentu yang memanfaatkan grafika komputer, diantaranya adalah:

1. Hiburan (*Entertainment*)

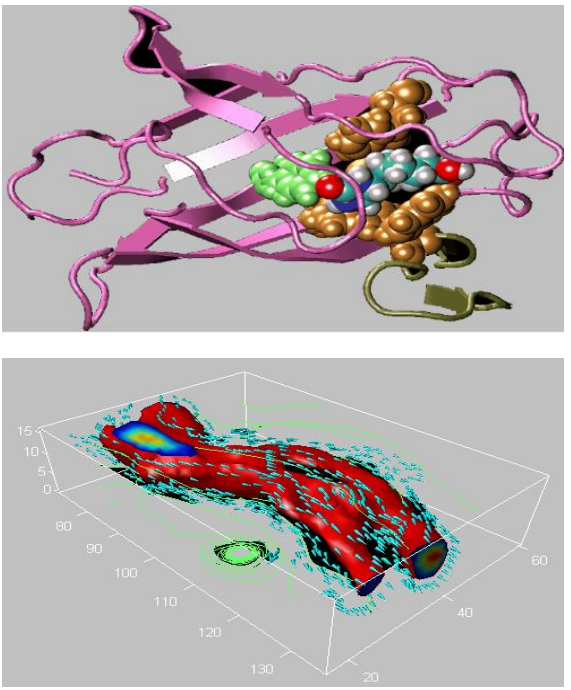
Saat ini grafika komputer banyak digunakan untuk menunjang pembuatan film, video musik, tayangan televisi, *motion picture*, animasi, dan *game* (permainan).

Gambar 1. 1 Aplikasi grafika komputer pada bidang hiburan berupa film-film animasi 3D

Sumber:*http://www.google.com/imgres?imgurl=http://hamdi.files.wordpress.com/2008/12/ upinipinbajubaru.jpg*

2. Visualisasi

Visualisasi (*visualization*) adalah teknik-teknik dalam pembuatan gambar, diagram atau animasi untuk mengkomunikasikan suatu informasi. Pada saat ini **visualisasi** telah menjadi cara yang efektif dalam mengkomunikasikan data atau ide abstrak maupun nyata sehingga cepat berkembang dan banyak dipakai untuk keperluan ilmu pengetahuan, rekayasa, visualisasi disain produk, pendidikan, multimedia interaktif, kedokteran, dan lain-lain.



Gambar 1. 2 (a) Visualisasi *Scientific* (b) Visualisasi aliran turbulen dari angin topan

Sumber : http://id.wikipedia.org/wiki/Visualisasi

3. *Computer-Aided Design* (CAD)

Grafika komputer digunakan dalam proses analisis dan desain, khususnya untuk sistem arsitektural dan engineering dalam bentuk aplikasi CAD (Computer-Aided Design). CAD banyak digunakan untuk mendesain bangunan, mobil, kapal, pesawat terbang, gedung, komputer, alat-alat elektronik, peralatan rumah tangga, dan berbagai produk lainnya. Contoh perangkat lunak : AutoCAD, 3D Studio Max dan lain-lain.

4. *Computer-Aided Sofware Engineering* (CASE)

CASE digunakan dalam bidang sofware engineering. CASE biasanya digunakan untuk memodelkan user requirement, pemodelan basisdata, workflow dalam proses bisnis, struktur program, dan sebagainya. Contoh perangkat lunak: Rational Rose, SyBase Power Designer dan lain-lain.

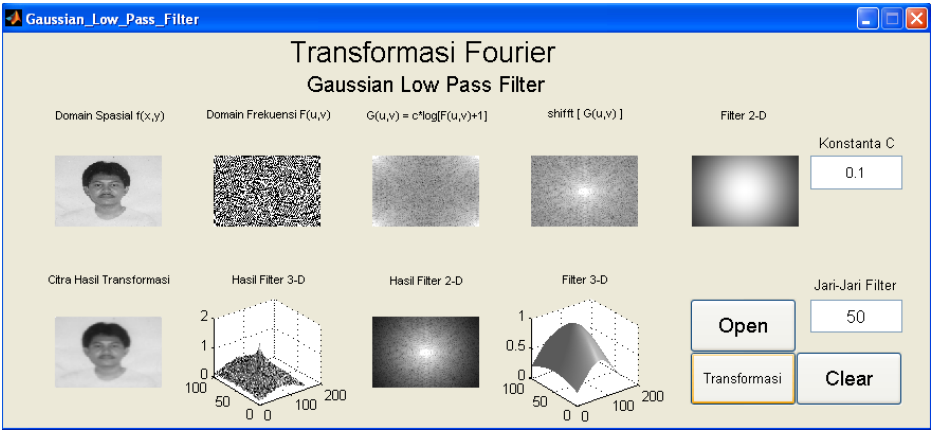
5. *Virtual Reality*

Virtual Reality adalah lingkungan virtual yang seakan-akan sama seperti lingkungan nyata. Pada lingkungan ini user dapat berinteraksi dengan objek-objek dalam lingkungan 3-D. Dibutuhkan perangkat keras khusus untuk memberikan efek pemandangan 3-D dan membuat user mampu beriteraksi dengan objek-objek yang berada di lingkungan tersebut. Contoh: aplikasi Virtual Reality pilot trainer yang digunakan untuk latihan mengendarai pesawat terbang. Aplikasi ini dapat memberikan keutungan berupa mengurangi resiko cedera selama latihan, mengurangi biaya penerbangan, melatih pilot-

pilot pemula sebelum melakukan penerbangan yang sesungguhnya.

6. Pendidikan dan Pelatihan

Komputer digunakan sebagai alat bantu pendidikan dan pelatihan, misalnya untuk membuat model-model proses fisika dan kimia, fungsi-fungsi psikologi, simulasi, dan sebagainya sehingga memudahkan sesorang untuk memahami bagaimana operasi atau proses yang terjadi dalam suatu sistem. Contoh: pembelajaran transformasi Fourier untuk peningkatan kualitas citra.



Gambar 1. 3 Pembelajaran transformasi Fourier untuk peningkatan kualitas citra

7. *Computer Art*

Computer art adalah penggunaan komputer grafis untuk menghasilkan karya-karya seni. Sebagai contoh grafika komputer digunakan untuk desktop publishing (cover buku), advertising (logo perusahaan), desain tekstil dan lain

sebagainya menggunakan perangkat lunak CorelDraw, Macromedia Freehand atau Adobe Ilustrator. Grafika komputer digunakan untuk pembuatan bermacam-macam gambar kartun sesuai dengan gagasan atau imajinasi seorang seniman menggunakan perangkat lunak yang berbasis paint contoh Paint, Corel Paint Shop Pro dan Adobe Photoshop.

Gambar 1. 4 Gambar kartun, digambar menggunakan Paint under window

8. Pengolahan Citra Digital

Pengolahan citra digital adalah teknik-teknik untuk mengolah citra digital. Pengolahan yang dilakukan meliputi, peningkatan kualitas citra, perbaikan citra, segmentasi citra, pengenalan pola menggunakan fitur-fitur yang ada dalam suatu citra. Contoh: perbaikan citra sehingga menjadi lebih jelas.

Gambar 1. 5 Hasil perbaikan citra yang terkena noise domain frekuensi menggunakan transformasi Fourier

9. *Graphical User Interface (GUI)*

Graphical User Inteface adalah antarmuka grafis yang berguna untuk mempermudah interaksi antara manusia dengan komputer dan alat-alat yang dikendalikan oleh komputer. Gambar 1.6 menunjukkan GUI berupa window-window yang digunakan pada sistem operasi Windows XP.

Gambar 1. 6 GUI dari aplikasi-aplikasi pada sistem operasi Windows XP

1.2 *Graphics Languages*

1) *IGL (Interactive Graphics Library)*

   Salah satu paket grafis yang mengandung pustaka fungsi grafis yang lengkap.

2) *GKS (Graphical Kernel System)*

   Rutin-rutin perangkat independen yang umum digunakan. Menggunakan pustaka UIS (*User Interface System*) untuk pelaksanaan ditingkat *low level*.

3) *HPGL (Hewlett Packard Graphics Library)*

Kumpulan Instruksi untuk berbagai fungsi output grafis untuk Plotter.

4) *ReGIS (Remote Graphics Instruction Set)*

   Khusus dikembangkan untuk pemrograman di terminal seri TV.

5) *PHIGS (Programmer's Hierarchical Interactive Graphics Standard)*

   Sebuah perangkat untuk keperluan umum dan merupakan paket grafis yang handal untuk mendukung sebagian besar bahasa pemrograman yang umum digunakan.

6) SG-GL (Silicon Graphics - Graphics Library)

   Berisi rutin-rutin yang cepat dan efisien yang dikembangkan oleh Silicon Graphics (misalnya OpenGL)

7) Computer-Aided Design (CAD)

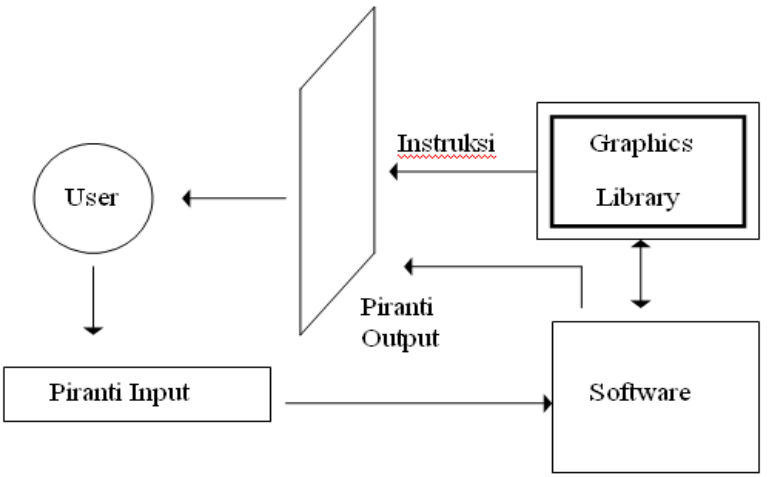
   Alat grafis untuk menggambar, desain berbantukan komputer dan Manufaktur

8) PC-Graphics

   Untuk pemrograman grafis pada lingkungan DOS dan Windows. Contohnya adalah: MVisual Basic, Visual C + + dengan GUI .

1.3 Sistem Pemrograman Grafis

Pada system pemrograman grafis, user menginput data (berupa program) melalui piranti input dan menggunakan software tertentu, kemudian software memanggil *graphics library* untuk mendapatkan instruksi-instruksi yang cepat, selanjutnya hasil dari program ditampilkan dilayar berupa gambar sesuai dengan apa yang diprogram.



Gambar 1. 7 Sistem Pemrograman Grafis

1.4 Komponen *Graphics Library*

1) Fungsi Output - Fungsi untuk menggambar primitif grafik, misalnya titik, garis, kurva lengkung, poligon, teks dan lain-lain.
2) Atribut Output - Fungsi untuk men-set atribut, misalnya tebal garis, isi warna, tinggi tkes dan lain-lain.

3) Fungsi Kontrol – Fungsi untuk menangani *event handling, error processing, I/O control* dan lain-lain.
4) Operasi File – Fungsi untuk menyimpan dan mengambil output grafik.
5) Fungsi Matematika and Graphics - Transformasi, *hidden line removal, rendering, clipping, window to viewport mapping, segments* dan lain-lain.
6) *Inquiry functions* - Fungsi untuk memberitahukan tentang status dan setting dari alat-alat output yang bekerja saat ini.

Ringkasan Bab 1

Grafika komputer (*Computer graphics*) adalah bagian dari ilmu komputer yang mempelajari cara-cara pembuatan dan manipulasi gambar secara digital, sehingga dapat memudahkan komunikasi antara manusia dan komputer, atau manusia dengan manusia melalui gambar-gambar, bagan-bagan, tabel dan lain-lain.

Grafika computer telah banyak di aplikasikan di berbagai bidang diantaranya adalah bidang hiburan, visualisasi, *CAD (Computer-Aided Design)*, *Virtual Reality*, Pendidikan dan Pelatihan, *Computer Art*, Pengolahan Citra Digital, *Graphical User Interface (GUI)*, dan sebagainya.

Soal-Soal Latihan

1. Jelaskan dan berikan contohnya peran grafika komputer dibidang :
    a. Hiburan
    b. Visualisasi
    c. CAD (Computer-Aided Design)
    d. Virtual Reality
    e. Pendidikan dan Pelatihan
    f. Computer Art
    g. Pengolahan Citra Digital
    h. Graphical User Interface (GUI).
2. Jelaskan peran grafika komputer untuk visualisasi pada teknologi informasi khususnya komputer!
3. Sebutkan beberapa bahasa pemrograman yang bisa digunakan untuk mengimplementasikan teori-teori grafika komputer!

Bab 2

Spesifikasi Graphics Primitif

TUJUAN PEMBELAJARAN

- Agar pembaca mengetahui definisi graphics primitif.
- Agar pembaca memahami primitif geometri titik dan garis.
- Agar pembaca memahami algoritma *Area Filling scan line* dan *Boundary Fill*

OUTCOME PEMBELAJARAN

- Pembaca dapat menjelaskan dan memberikan contoh tentang definisi graphics primitif.
- Pembaca dapat menjelaskan primitif geometri titik dan garis.
- Pembaca dapat menjelaskan perbedaan algoritma *Area Filling scan line* dan *Boundary Fill.*

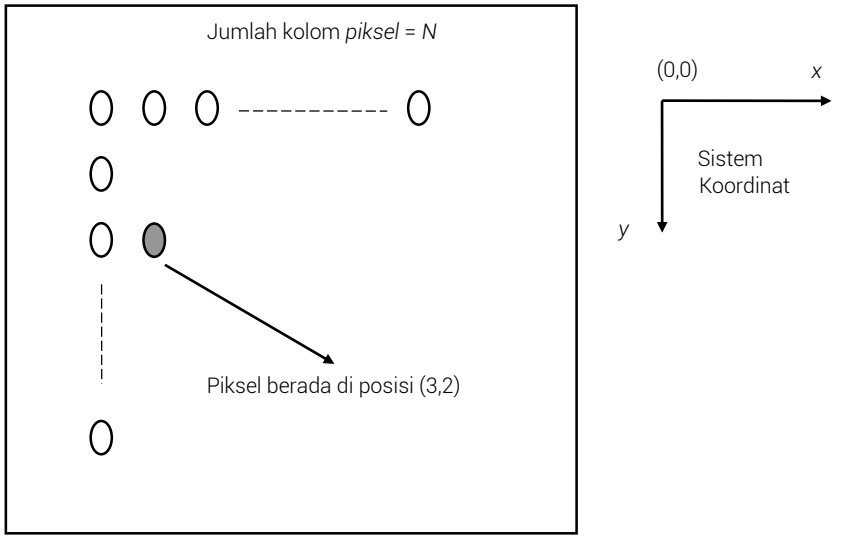
Pendahuluan

Graphics primitif adalah struktur dasar geometri yang paling sederhana dari gambar grafika komputer. Titik dan garis adalah contoh dari output primitif yang dapat digunakan untuk membentuk gambar, misalnya lingkaran, kerucut, permukaan berbentuk persegi, kurva dan permukaan berbentuk lengkung, warna area dan karakter, dan lain-lain. Gambar-gambar seperti

rumah, tembok, gedung dan lain-lain merupakan obyek yang lengkap yang ditempatkan pada koordinat tertentu pada layar, dimana obyek-obyek tadi sebenarnya tersusun dari kumpulan titik dan garis sebagai penyusunnya.

2.1 *Pixel (Picture Element)*

Pixel adalah elemen gambar terkecil berupa sebuah titik yang ditempatkan dilayar.



Gambar 2. 1 Layar berisi M baris *piksel* dan N kolom *piksel.* Sebuah piksel (titik hitam) terletak pd (3,2).

Gambar 2.1 menunjukkan sebuah piksel mempunyai koordinat (3,2) pada layar. Jumlah Piksel untuk setiap gambar tergantung dari kemampuan *Graphics card.* Terdapat beberapa tampilan format gambar:

Standart	x-maksimal	y-maksimal	Jumlah keseluruhan Pixel
VGA	640	480	307 200
SVGA	800	600	480 000
XGA	1024	768	786 432
SXGA	1280	1024	1 228 800

Tabel 2. 1

Sebagai pembanding adalah tayangan televisi standart dengan jumlah titik gambar sebesar 400.000 Piksel. Bandingkan dengan WebCam standar VGA dan juga dengan HP yang memiliki fasilitas perekam 2 Mega Piksel.

Resolusi Piksel adalah jumlah N piksel persatuan lebar (x) kali panjang (y) (x : width, y height)

Jika layar mempunyai dimensi p inchi sepanjang x dan q inchi sepanjang y, maka :

$$\text{resolusi-x} = \frac{N}{p} \text{ pixel/inchi (ppi)}$$

Persamaan 2 - 1

$$\text{resolusi-y} = \frac{M}{q} \text{ pixel/inchi (ppi)}$$

Persamaan 2 - 2

Aspek rasio = $\frac{resolusi - y}{resolusi - x}$ (Height : Width)

Persamaan 2 - 3

Frame buffer adalah area memory tempat informasi gambar disimpan. Jika system mempunyai *n* bit memori untuk tiap-tiap piksel, maka ukuran *frame buffer* adalah = H W *n* / 8 byte. Jumlah warna yang bisa ditampilkan secara bersama-sama dilayar adalah 2^n. Sistem dikatakan mempunyai *n* ***bit planes***.

Screen Mode	Pixels	Screen Ratio	Bit planes	Colors	Memory
CGA(med res)	320x200	4:3	2	4	16K
CGA(high res)	640x200	4:3	1	1	28K
EGA	640x350	4:3	4	16	112K
VGA	640x480	4:3	8	256	307K
Super VGA-1	800x600	4:3	24	16.7M	1.44M
Super VGA-2	1024x768	4:3	16	65K	1.57M

Tabel 2. 2

Atribut dari piksel adalah warnanya (indeks warna) *i*, dimana $0 \leq i \leq 2^n$. Biasanya indeks warna yang bernilai 0 melambangkan warna hitam.

Monitor monokrom yang mempunyai satu *bit plane* hanya bisa menampilkan dua warna, yaitu warna hitam (0) sebagai background dan warna putih (1) sebagai foreground. Monitor monokrom yang mempunyai 8 *bit plane* bisa menampilkan 256 warna yang berbeda, yang disebut dengan grayscale dengan indeks warna dimulai dari 0 (hitam) sampai dengan 255 (putih).

Contoh 2.1

Diketahui sebuah sistem dengan 1280 X 1024 frame buffer, mempunyai 4 bits per piksel. jika sistem tersebut dapat mentransfer data 10^4 bits per detik, berapa lama waktu yang dibutuhkan untuk loading ?

Jawab:

Jumlah piksel pada frame buffer = 1280 x 1024 = 1310720 piksel

Jumlah bit pada frame buffer = 1310720 piksel x 4 bit /piksel = 5242880 bit

Waktu loadingnya = $\frac{5242880\ bit}{10^4\ bit/sekon} = 524{,}28\ sekon$

2.2 Warna

Sudah dikatakan diatas bahwa warna merupakan atribut dari piksel. Selain piksel, yang lebih penting diperhitungkan lagi adalah warna dari piksel itu sendiri. Setiap piksel mengandung informasi mengenai warna dengan jumlah yang beragam. Pada awalnya saat teknik penggambaran belum begitu maju, hanya disediakan 16 kemungkinan warna setiap Pikselnya. Tetapi *Graphics card* sekarang ini sudah mampu menampilkan gambar dengan berbagai macam kedalaman seperti pada tabel dibawah ini:

Jumlah Byte setiap Pixel	Jumlah warna yang mampu ditampikan	Color Quality
1 (8 bit)	256	Low
2 (16 bit)	65 536	Medium
3 (24 bit)	16 777 216	High
4 (32 bit)	4 294 967 296	Highest

Tabel 2. 3

Untuk menampilkan suatu gambar pada layar monitor sering kali tidak cukup hanya dengan 256 tampilan warna. Tetapi jika jumlah tampilan warna diperbesar maka memerlukan ruang (memori) lebih banyak lagi. Lihat tabel kebutuhan memori di bawah ini:

x/y-Format	1 B / Piksel	2 B / Piksel	3 B / Piksel	4 B / Piksel
640 x 480	300 kB	600 kB	900 kB	1200 kB
800 x 600	469 kB	938 kB	1406 kB	1876 kB
1024 x 768	768 kB	1536 kB	2304 kB	3072 kB
1280 x 1024	1311 kB	2622 kB	3933 kB	5244 kB

Tabel 2. 4

Untuk menampilkan gambar pada layar monitor tabung sehingga tanpa efek kerdipan (*flicker-free*) diperlukan pengulangan tayang sebanyak 75 kali dalam satu detik. Bisa dicoba pada layar monitor, turunkan frekuensi monitor anda pada penyetelan: display – settings – advanced – Monitor – Screen refresh rate = 60. Terasa kerdipan yang mengganggu dan tidak nyaman di mata. Pengulangan tayangan per detik yang begitu tinggi (lebih besar dari 75) pada resolusi (ketajaman gambar) yang tinggi pula, dapat menyebabkan kesulitan dalam memindahkan gambar dari *graphic card* ke layar monitor (diperlukan *transfer-rate* yang besar). Berbeda dengan layer LCD (Liquid Crystal Display). Pada layer LCD tidak dijumpai efek kerdipan (*flicker*) sehingga tidak memerlukan frekuensi pengulangan gambar yang tinggi. Hanya saja, yang masih

menjadi masalah pada layar LCD adalah sudut pandang yang terbatas, *delay time*, dan tingginya harga. Secara teknis warna pada pesawat televisi tidak dipindahkan dalam bentuk Byte untuk setiap elemen gambarnya, akan tetapi sudah berupa sinyal warna analog. Sebagai pembanding *Photo-CD* yang dikeluarkan oleh Kodak memiliki resolusi sebesar 2500 x 3000 Piksel dengan kedalaman 3 Byte warna. Perbandingan ke tiga format dapat dilihat pada tabel dibawah ini:

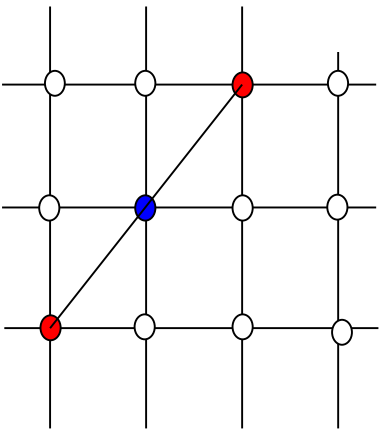
Televisi standar	Gambar komputer (600 x 800 x 3 Byte)	Photo-CD
800 kilo Byte	1.5 Mega Byte	20 Mega Byte

Tabel 2. 5

2.3 Garis

Software untuk menggambar garis lurus pada setiap paket grafika komputer secara tradisi sebagai fungsi elementarnya sudah tersedia, tetapi ada baiknya kita simak lebih seksama bagaimana algoritma sebagai dasar penggambaran titik-titik pada *raster graphics*. *Raster graphics* diartikan sebagai grafik yang dibangun atas dasar titik-titik pada kolom dan baris yang

juga mengandung informasi warna. Gambar 2.2 adalah sebuah garis lurus yang melalui titik (x_1,y_1) dan (x_2,y_2).



Gambar 2. 2 Garis lurus yang melalui titik $P_1(x_1,y_1)$ dan $P_2(x_2,y_2)$

Secara umum garis lurus dinyatakan dalam persamaan :

$$y = mx + c$$

Persamaan 2 - 4

dimana : m adalah gradient dan c adalah konstanta.

Bila garis melalui titik (x_1, y_1) dan titik (x_2, y_2) maka gradien m dihitung menggunakan persamaan (2-5)

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

Persamaan 2 - 5

Persamaan garis bergradien m yang melewati titik (x_1, y_1) adalah:

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

Persamaan 2 - 6

Berdasarkan persamaan (2-4), garis yang melalui titik (x_1, y_1) dan titik (x_2, y_2) dapat ditulis sebagai berikut,

$$c = y_1 - m\,.\,x_1 = y_2 - m.x_2$$

atau

$$y_2 = m\,(x_2 - x_1) + y_1$$

atau

$$y_2 = m\,\Delta \mathrm{x} + \mathrm{y}_1$$

Persamaan 2 - 7

bila Δx = 1, maka persamaan (2-7) menjadi

$$y_2 = m + y_1$$

Persamaan 2 - 8

Algoritma untuk menggambar garis pada komputer didasarkan pada persamaan (2-6) dan (2-8).

Tugas dari algoritma pembentuk garis adalah menentukan titik-titik diantara dua titik ujung (x_1,y_1) dan (x_2,y_2) yang akan digambar sebagai garis.

Kriteria algoritma pembentuk garis yang baik:

1) Garis antara dua titik ujung tersebut sebisa mungkin harus lurus.
2) Kerapatan titik-titik konstan (tidak ada *gap* antara dua titik yang bersebelahan).
3) Kerapatan titik-titik tidak tergantung pada kemiringan garis.
4) Waktu proses algoritma harus cepat.

Bab 3

Algoritma, Analisa, dan Tampilan Graphics Dua Dimensi (2D)

TUJUAN PEMBELAJARAN

- Agar pembaca memahami algoritma pembentukan garis *brute force*, DDA (*Digital Diferential Analyzer*) dan Bressenham.
- Agar pembaca memahami algoritma pembentukan lingkaran.
- Agar pembaca memahami algoritma *Area Filling scan line* dan *Boundary Fill.*

OUTCOME PEMBELAJARAN

- Pembaca dapat menjelaskan perbedaan algoritma pembentukan garis brute force, DDA dan Bressenham.
- Pembaca dapat menjelaskan algoritma pembentukan lingkaran.
- Pembaca dapat menjelaskan perbedaan algoritma *Area Filling scan line* dan *Boundary Fill*.

Pendahuluan

Output primitif adalah struktur dasar geometri yang paling sederhana dari gambar grafika komputer. Titik dan garis adalah contoh dari output primitif yang dapat digunakan untuk membentuk gambar, misalnya lingkaran, kerucut, permukaan berbentuk persegi, kurva dan permukaan berbentuk lengkung, warna area dan karakter, dan lain-lain. Gambar-gambar seperti rumah, tembok, gedung dan lain-lain merupakan obyek yang lengkap yang ditempatkan pada koordinat tertentu pada layar, dimana obyek-obyek tadi sebenarnya tersusun dari kumpulan titik dan garis sebagai penyusunnya.

3.1 Algoritma Brute Force

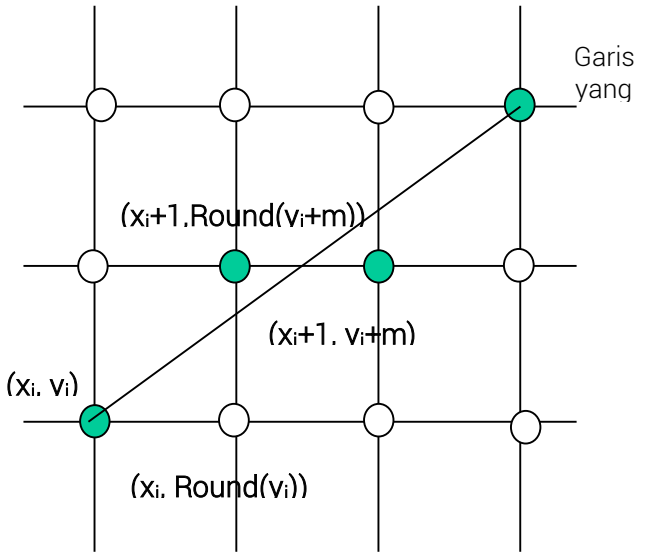
Algoritma brute force untuk membentuk garis didasarkan pada persamaan (2-6), yaitu :

1. Tentukan dua titik ujung (x_1,y_1) dan (x_2,y_2)
2. Jika $x_1 = x_2$ (garis vertikal), maka
   (a) $y = y + 1$ dan x tetap
   (b) gambar titik (x,y) di layar
   (c) Selesai
3. Jika $y_1 = y_2$ (garis horisontal), maka
   (a) $x = x + 1$ dan y tetap
   (b) gambar titik (x,y) di layar
   (c) Selesai

{anggap $x_2 > x_1$, (jika sebaliknya, gantilah x_2 dengan x_1)}

4. Hitung kemiringan garis $m = (y_2 - y_1)/(x_2 - x_1)$
5. $N = x_2 - x_1 + 1$
6. $x = x_1$
7. Ulang sebanyak N kali:
   (a) $y = m(x - x_1) + y_1$
   (b) lakukan pembulatan y_a = Round(y),
   (c) gambar titik (x, y_a) di layar
   (d) $x = x + 1$
8. selesai

Perhatikan langkah 7(b), karena posisi piksel adalah integer (bilangan bulat), dan nilai gradien *m* biasanya pecahan, maka nilai *y* adalah bilangan pecahan, sehingga kita perlu melakukan pembulatan terhadap nilai *y*. Perhatikan Gambar 2.3 berikut.



Gambar 3. 1 Pembulatan nilai *y* dilakukan karena posisi pixel adalah bulat (integer)

Contoh 2.2

Diketahui 2 buah titik A(2,1) dan titik B(8,5) bila titik A sebagai titik awal dan titik B sebagai titik akhir, maka buatlah garis yang menghubungkan titik tersebut dengan menggunakan algoritma *Brute Force*.

Jawab:

1. titik ujung $(x_1,y_1) = (2,1)$ dan $(x_2,y_2) = (8,5)$
2. tidak dipenuhi
3. tidak dipenuhi
4. $m = (5 - 1)/(8 - 2) = 0{,}67$
5. $N = 8 - 2 + 1 = 7$
6. untuk $x = 2$ $y = 0{,}67\ (x - 2) + 1$
7. Ulang sebanyak 7 kali

iterasi ke-1:

$x = 2$; $y = 0{,}67.\ (2 - 2) + 1 = 1$

pembulatan: $y = 1$

gambar titik (2,1) di layar

$x = 2 + 1 = 3$

==========

iterasi ke-2:

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

pembulatan: $y = 2$

gambar titik (3,2) di layar .

$x = 3 + 1 = 4$

==========

iterasi ke-3:

$x = 4$; $y = 0{,}67. (4 - 2) + 1 = 2{,}34$

pembulatan: $y = 2$

gambar titik (4,2) di layar .

$x = 4 + 1 = 5$

==========

iterasi ke-4:

$x = 5$; $y = 0{,}67. (5 - 2) + 1 = 3{,}01$

pembulatan: $y = 3$

gambar titik (5,3) di layar .

$x = 5 + 1 = 6$

==========

iterasi ke-5:

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

pembulatan: $y = 4$

gambar titik (6,4) di layar .

$x = 6 + 1 = 7$

==========

iterasi ke-6:

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

pembulatan: $y = 4$

gambar titik (7,4) di layar .

$x = 7 + 1 = 8$

==========

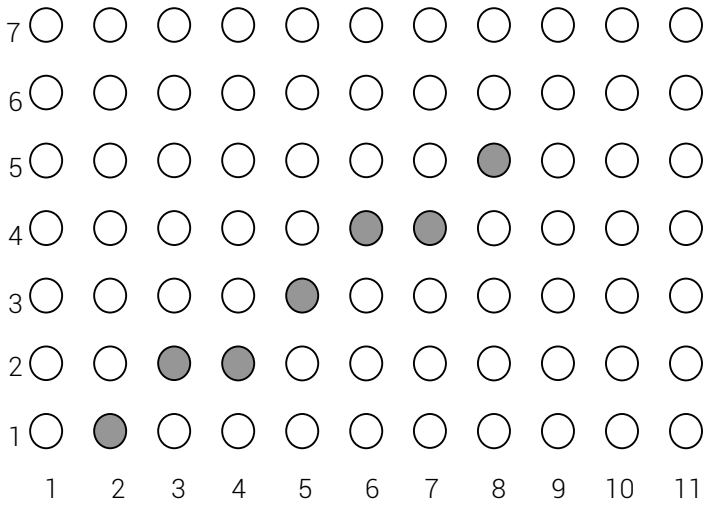
iterasi ke-7:

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

pembulatan: $y = 5$

gambar titik (8,5) di layar .

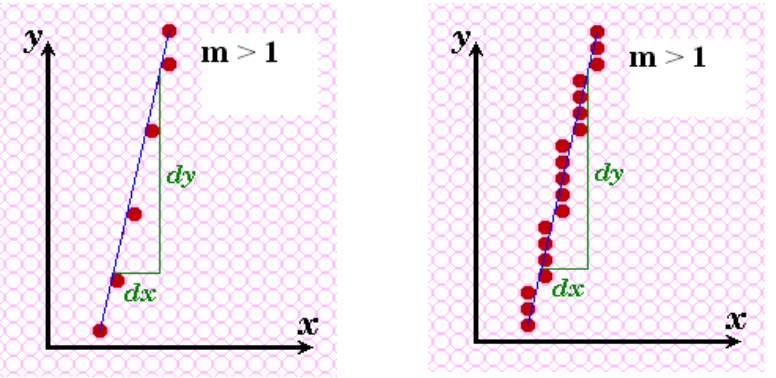
Maka diperoleh titik-titik pembentuk garis : (2,1), (3,2), (4,2), (5,3), (6,4), (7,4), (8,5).



Gambar 3. 2 Titik-titik pembentuk garis hasil perhitungan menggunakan algoritma *Brute Force* digambar pada *raster graphics*.

Masalah :

Untuk kemiringan $m > 1$, garis menjadi tidak kontinyu (Gambar 2.5(a)) yang mengakibatkan terjadinya *gap* antar piksel, sehingga diperlukan interpolasi (Gambar 2.5(b)).



(a) (b)

Gambar 3. 3 (a) Garis dengan kemiringan $m > 1$, tampak bahwa garis tidak kontinyu (b) setelah dilakukan interpolasi garis menjadi kontinyu.

Selain menggunakan interpolasi, untuk kemiringan garis $m > 1$, tukarlah x dengan y maka sudah tidak terjadi *gap* antara titik yang satu dengan yang lain. Sehingga algoritma pembentukan garis untuk $m > 1$ adalah sebagai berikut :

1. Tentukan dua titik ujung (x_1,y_1) dan (x_2,y_2)
2. Jika $x_1 = x_2$ (garis vertikal), maka
   (a) $y = y + 1$ dan x tetap
   (b) gambar titik (x,y) di layar
   (c) Selesai
3. Jika $y_1 = y_2$ (garis horisontal), maka
   (a) $x = x + 1$ dan y tetap
   (b) gambar titik (x,y) di layar
   (c) Selesai

{anggap $y_2 > y_1$, (jika sebaliknya, gantilah y_2 dengan y_1)}

4. Hitung kemiringan garis $m = (x_2 - x_1) / (y_2 - y_1)$
5. $N = y_2 - y_1 + 1$
6. $y = y_1$
7. Ulang sebanyak N kali:
   (a) $x = m(y - y_1) + x_1$
   (b) lakukan pembulatan x_a = Round(x),
   (c) gambar titik (x_a,y) di layar

(d) $y = y + 1$

8. selesai

Contoh 2.3

Diketahui 2 buah titik A(4,3) dan titik B(7,8) bila titik A sebagai titik awal dan titik B sebagai titik akhir, maka buatlah garis yang menghubungkan titik tersebut dengan menggunakan algoritma *Brute Force*.

Jawab:

1. titik ujung A(4,3) dan B(7,8)
2. tidak dipenuhi
3. tidak dipenuhi
4. $m = (7 - 4)/(8 - 3) = 0{,}6$
5. $N = 8 - 3 + 1 = 6$
6. untuk $y = 3$ dan $x = 0{,}6. (y - 3) + 4$
7. Ulang sebanyak 6 kali

   iterasi ke-1:

   $y = 3$; $x = 0{,}6. (3 - 3) + 4 = 4$

   pembulatan: $x = 4$

   gambar titik (4,3) di layar .

   $y = 3 + 1 = 4$

==========

iterasi ke-2:

y= 4; x = 0,6. (4 – 3) + 4 =4,6

pembulatan: x = 5

gambar titik (5,4) di layar .

y = 4 + 1 = 5

==========

iterasi ke-3:

y= 5; x = 0,6. (5 – 3) + 4 =5,2

pembulatan: x = 5

gambar titik (5,5) di layar .

y = 5 + 1 = 6

==========

iterasi ke-4:

y= 6; x = 0,6. (6 – 3) + 4 =5,8

pembulatan: x = 6

gambar titik (6,6) di layar .

y = 6 + 1 = 7

==========

iterasi ke-5:

y= 7; *x* = 0,6. (7 – 3) + 4 =6,4

pembulatan: *x* = 6

gambar titik (6,7) di layar .

y = 7 + 1 = 8

==========

iterasi ke-6:

y= 8; *x* = 0,6. (8 – 3) + 4 =7

pembulatan: *x* = 7

gambar titik (7,8) di layar .

y = 8 + 1 = 9

titik-titik pembentuk garis : (4,3), (5,4), (5,5), (6,6), (6,7), (7,8).

Kelemahan algoritma *Brute Force* :

Algoritma ini terlalu lambat karena: pada setiap iterasi terdapat perkalian bilangan pecahan (floating point), penjumlahan bilangan pecahan, dan proses pembulatan angka pecahan menjadi bulat (integer).

3.2 Algoritma DDA (*Digital Diferential Analyzer*)

DDA adalah algoritma pembentuk garis yang didasarkan pada perasamaan (2-8). Garis dibuat menggunakan titik awal (x_1, y_1) dan titik akhir (x_2, y_2). Setiap koordinat titik (x_k, y_k) yang membentuk garis diperoleh dari perhitungan, kemudian hasil perhitungan dikonversikan menjadi nilai integer. Algoritma ini bisa digunakan untuk menghitung garis dengan semua kemiringan. { $0 < m < 1$; $m>1$; $-1 < m < 0$; $m < -1$ }. Berikut adalah langkah-langkah pembentukan garis berdasarkan algoritma DDA:

1. Tentukan dua titik yang akan dihubungkan dalam pembentukan garis.
2. Tentukan salah satunya sebagai titik awal (x_1, y_1) dan yang lain sebagai titik akhir (x_2, y_2).
3. Hitung : $dx = x_2 - x_1$ dan $dy = y_2 - y_1$
4. Tentukan *step,* dengan ketentuan berikut:
   - bila $|dx| > |dy|$ maka $step = |dx|$
   - bila tidak, maka $step = |dy|$
5. Hitung penambahan koordinat piksel dengan persamaan:

   $x\_inc = dx / step$

   $y\_inc = dy / step$
6. Koordinat selanjutnya :

   $x = x + x\_inc$ $\qquad$ $y = y + y\_inc$

7. Lakukan pembulatan u = Round(x), v = Round(x), kemudian plot piksel (u, v) pada layar
8. Ulangi point 6 dan 7 untuk menentukan posisi piksel berikutnya sampai $x = x_2$ dan $y = y_2$.

Contoh 2.4

Diketahui 2 buah titik A(2,1) dan titik B(8,5) bila titik A sebagai titik awal dan titik B sebagai titik akhir, maka buatlah garis yang menghubungkan titik tersebut dengan menggunakan algoritma DDA.

Jawab:

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 $step = |dx| = 6$

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

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

==========

Iterasi ke-1: (x,y) = (2,1)

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

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

Koordinat selanjutnya : (x,y) = (3; 1,67)

Pembulatan (3; 1,67) ≈ (3,2). Gambar titik (3,2) dilayar

==========

Iterasi ke-2: (*x*,*y*) = (3; 1,67)

x+*x_inc* = 3 + 1 = 4

y+*y_inc* = 1,67 + 0,67 = 2,34

Koordinat selanjutnya : (*x*,*y*) = (4; 2,34)

Pembulatan (4; 2,34) ≈ (4,2). Gambar titik (4,2) dilayar

==========

Iterasi ke-3: (*x*,*y*) = (4; 2,34)

x_A+*x_inc* = 4 + 1 = 5

y_A+*y_inc* = 2,34 + 0,67= 3,01

Koordinat selanjutnya : (*x*,*y*) = (5; 3,01)

Pembulatan (5; 3,01) ≈ (5,3). Gambar titik (5,3) dilayar

==========

Iterasi ke-4: (*x*,*y*) = (5; 3,01)

x_A+*x_inc* = 5 + 1 = 6

y_A+*y_inc* = 3,01 + 0,67 = 3,68

Koordinat selanjutnya : (*x*,*y*) = (6; 3,68)

Pembulatan (6; 3,68) ≈ (6,4). Gambar titik (6,4) dilayar

==========

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

x_A+x_inc = 6 + 1 = 7

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

Koordinat selanjutnya : (x,y) = (7; 4,35)

Pembulatan (7; 4,35) ≈ (7,4). Gambar titik (7,4) dilayar

==========

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

x_A+x_inc = 7 + 1 = 8

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

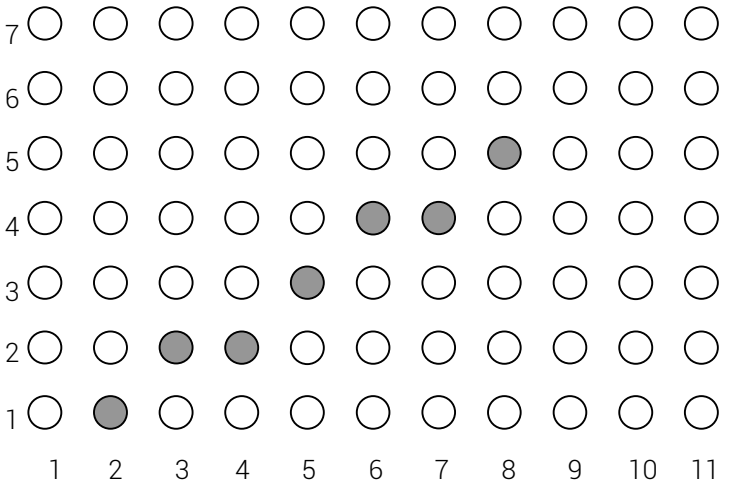
Koordinat selanjutnya : (x,y) = (8; 5,02)

Pembulatan (8; 5,02) ≈ (8,5). Gambar titik (8,5) dilayar

Karena $x = x_2 = 8$, maka iterasi dihentikan, sehingga diperoleh titik-titik pembentuk garis sebagai berikut: (2,1), (3,2), (4,2), (5,3), (6,4), (7,4) dan (8,5).

==========

Bila digambar pada *raster graphics* diperoleh gambar 2.6:



Gambar 3. 4 Titik-titik pembentuk garis hasil perhitungan menggunakan algoritma DDA digambar pada *raster graphics.*

Kelebihan Algoritma DDA dibanding Algoritma *Brute Force*

Algoritma DDA lebih cepat dibanding dengan algoritma Brute Force dan baik digunakan untuk kemiringan garis $m > 1$.

Kelemahan Algoritma DDA

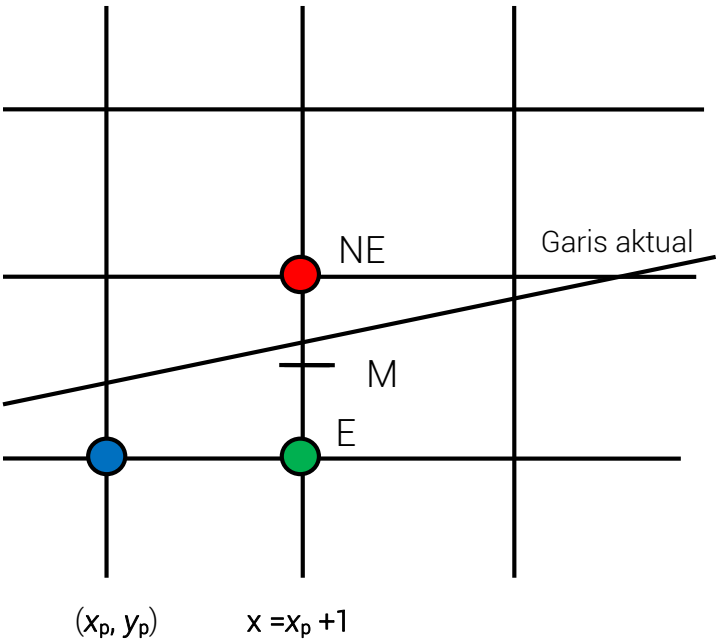
- *Prosedur untuk menggambar garis masih menggunakan fungsi pembulatan x maupun y, sehingga memerlukan waktu.*
- *variabel x, y maupun m memerlukan bilangan real karena kemiringan merupakan nilai pecahan.*

3.3 Algoritma Bressenham

J.E. Bressenham pada th.1965 mengumumkan penemuan algoritma untuk mengoptimalkan gambar garis pada *raster*. Algoritma garis Bressenham disebut juga Midpoint Line Algorithm.

Algoritma temuannya tidak lagi menggunakan *floating point arihtmetic,* sehingga tidak perlu membulatkan nilai posisi piksel setiap waktu dan dalam pengulangan (*looping)* hanya menggunakan pengoperasian incremental. Algoritma ini hampir dapat dipergunakan dan diimplementasikan pada semua hardware dan software untuk keperluan penggambaran garis. Algoritma Bressenham dapat memperbaiki pengendalian plotter, sehingga dia mendapatkan hak paten. Algoritma ini hanya berlaku untuk nilai kemiringan garis : $0 < m < 1$.

Perhatikan Gambar 2.7. Misalkan kemiringan garis m, yang besarnya adalah $0 < m < 1$, titik awal garis di (x_0, y_0) dan titik akhir garis di (x_1, y_1). Anggap posisi titik sekarang (x_p, y_p) dan sekarang kita harus memilih antara titik E (*East*) atau NE (*Northeast*). Misalkan Q adalah titik potong garis dengan garis $x = x_p+1$, dan M adalah titik tengah antara E dan NE. Jika titik tengah M terletak diatas garis, E dipilih, tetapi jika titik tengah M terletak dibawah garis, maka NE dipilih.



Gambar 3. 5 Garis mempunyai kemiringan $0 < m < 1$, pada titik $x = x_p+1$, garis berada diantara dua titik E dan NE yang mempunyai titik tengah di M.

Dalam hal ini kita harus menentukan apakah garis berada diatas titik tengah M atau dibawah titik tengah M. Untuk melakukan ini, kita amati bahwa setiap garis bisa dinyatakan sebagai berikut;

$$F(x, y) = ax + by + c = 0$$

Persamaan 3 - 1

Hitung: $dy = y_1 - y_0$ dan $dx = x_1 - x_0$, maka garis juga bisa ditulis sebagai:

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

atau

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

Kalikan dengan dx menjadi

$$(dy)x - (dx)y + (dx)B = 0$$

Diperoleh $\boldsymbol{a} = dy$, $\boldsymbol{b} = -dx$ dan $\boldsymbol{c} = (dx)B$.

- Jika $F(x, y) = 0$, maka (x, y) terletak pada garis
- Jika $F(x, y) > 0$, maka (x, y) terletak dibawah garis
- jika $F(x, y) < 0$, maka (x, y) terletak diatas garis

Untuk menerapkan kriteria midpoint, kita hitung:

$F(M) = F(x_p + 1, y_p + ½) = a\,(x_p + 1) + b\,(y_p + ½) + c$

Dan menguji tanda dari $\boldsymbol{d} = F(M)$.

- Jika $\boldsymbol{d} > 0$, M terletak dibawah garis, maka NE dipilih.
- Jika $\boldsymbol{d} = 0$, pilih E (sebenarnya memilih NE juga bisa)
- Jika $\boldsymbol{d} < 0$, M terletak diatas garis, maka E dipilih.

Misalkan E dipilih, M bertambah satu langkah dalam arah x, dan kita mempunyai $\boldsymbol{d}$ baru misalnya d_{baru}.

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

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

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

Jadi setelah E dipilih, pertambahan untuk mendapatkan d yang baru adalah $a = dy$.

Misalkan NE dipilih, M adalah pertambahan satu langkah kearah x dan y sehingga

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

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

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

Untuk memulai algoritma, titik awal adalah $(x_p , y_p) = (x_0, y_0)$. Untuk menghitung nilai *d* yang pertama untuk memulai algoritma, *midpoint* pertama adalah $(x_0 + 1, y_0 + ½)$ dan

$F(M) = F(x_0 + 1, y_0 + ½) = a\,(x_0 + 1) + b\,(y_0 + ½) + c$

$= ax_0 + by_0 + c + a + b/2$

$= 0 + a + b/2$ karena (x_0, y_0) terletak pada garis.

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

Untuk menghilangkan pecahan, definisikan F dengan mengalikan 2, diperoleh

$F(x,y) = 2\,(ax + by + c) = 0$.

Definisi ini akan menyebabkan nilai *d* menjadi $d = 2dy - dx$

Dengan begini algoritma *midpoint* Bresenham (untuk kemiringan $0 < m < 1$) adalah :

1) Tentukan dua titik yang akan dihubungkan dalam pembentukan garis.
2) Tentukan salah satu sebagai titik awal (x_0, y_0) dan titik akhir (x_1, y_1).
3) Hitung *dx*, *dy*, $2|dy|$ dan $2|dy| - 2|dx|$
4) Hitung parameter : $po = 2|dy| - |dx|$
5) Untuk setiap x_k sepanjang jalur garis, dimulai dengan k = 0

- bila $p_k < 0$ maka titik selanjutnya adalah:

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

- bila tidak, titik selanjutnya adalah:

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

6) Ulangi nomor 5 untuk menentukan posisi piksel berikutnya, sampai

   $x = x_1$ dan $y = y_1$.

Contoh 2.5

Diketahui 2 buah titik A(2,1) dan titik B(8,5) bila titik A sebagai titik awal dan titik B sebagai titik akhir, maka buatlah garis yang menghubungkan titik tersebut dengan menggunakan algoritma Bressenham.

Jawab:

Titik awal (x_0, y_0) = A(2,1) dan Titik akhir (x_1, y_1) = B(8,5)

$dx = x_1 - x_0 = 8 - 2 = 6$ dan $dy = y_1 - y_0 = 5 - 1 = 4$

$m = dy/dx = 4/6$; kemiringan garis berada diantara 0 dan 1 : $0 < m < 1$

$2|dx| = 2.6 = 12$; $2|dy| = 2.4 = 8$ dan $2|dy| - 2|dx| = 8 - 12 = -4$

$p_o = 2|dy| - |dx| = 8 - 6 = 2$

==========

Iterasi ke-1 (k = 0):

Titik awal = (2,1)

*P*o = 2 > 0, maka titik selanjutnya adalah

$x = 2 + 1 = 3$ dan $y = 1 + 1 = 2$, koordinat selanjutnya : (3,2)

$p_1 = p_0 + 2|dy| - 2|dx| = 2 - 4 = -2$

==========

Iterasi ke-2 (k = 1):

Titik awal = (3,2)

$P_1 = -2 < 0$, maka titik selanjutnya adalah

$x = 3 + 1 = 4$ dan $y = 2$, koordinat selanjutnya : (4,2)

$p_2 = p_1 + 2|dy| = -2 + 8 = 6$

==========

Iterasi ke-3 (k = 2):

Titik awal = (4,2)

$P_2 = 6 > 0$, maka titik selanjutnya adalah

$x = 4 + 1 = 5$ dan $y = 2 + 1 = 3$, koordinat selanjutnya : (5,3)

$p_3 = p_2 + 2|dy| - 2|dx| = 6 - 4 = 2$

==========

Iterasi ke-4 (k = 3):

Titik awal = (5,3)

$P_3 = 2 > 0$, maka titik selanjutnya adalah

$x = 5 + 1 = 6$ dan $y = 3 + 1 = 4$, koordinat selanjutnya : (6,4)

$p_4 = p_3 + 2|dy| - 2|dx| = 2 - 4 = -2$

==========

Iterasi ke-5 (k = 4):

Titik awal = (6,4)

$P_3 = -2 < 0$, maka titik selanjutnya adalah

$x = 6 + 1 = 7$ dan $y = 4$, koordinat selanjutnya : (7,4)

$p_5 = p_4 + 2|dy| = -2 + 8 = 6$

==========

Iterasi ke-6 (k = 5):

Titik awal = (7,4)

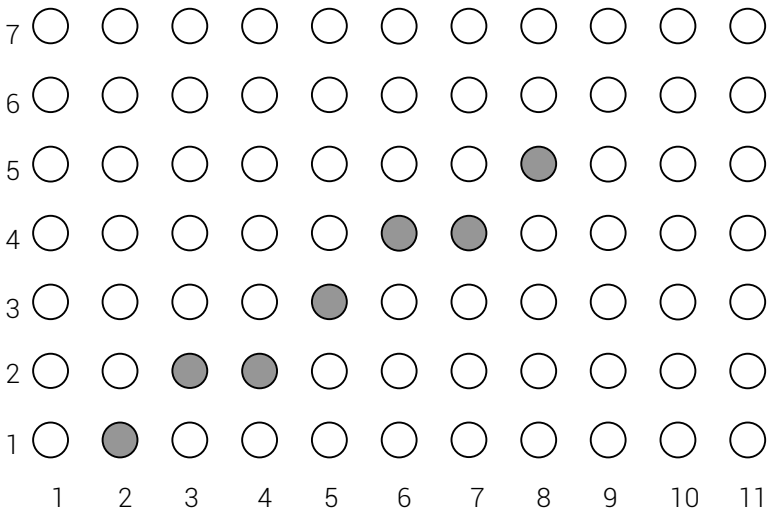
$P_5 = 6 > 0$, maka titik selanjutnya adalah

$x = 7 + 1 = 8$ dan $y = 4 + 1 = 5$, koordinat selanjutnya : (8,5)

==========

Karena $x = x_2 = 8$, maka iterasi dihentikan, sehingga diperoleh titik-titik penyusun garis sebagai berikut: (2,1), (3,2), (4,2), (5,3), (6,4), (7,4) dan (8,5).

Bila digambar pada *raster graphics* diperoleh Gambar 2.8:



Gambar 3. 6 Titik-titik pembentuk garis dengan kemiringan 0 < m < 1, hasil perhitungan menggunakan algoritma Bressenham yang digambar pada *raster graphics*.

Untuk kemiringan −1 < m < 0 , kita tinggal mengganti komponen berikut:

Untuk setiap x_k sepanjang jalur garis, dimulai dengan k = 0

- bila $p_k < 0$ maka titik selanjutnya adalah:
  (x_k+1, y_k) dan $p_{k+1} = p_k + 2|dy|$
- bila tidak, titik selanjutnya adalah:
  (x_k+1, y_k-1) dan $p_{k+1} = p_k + 2|dy| - 2|dx|$

Contoh 2.6

Diketahui 2 buah titik A(2,9) dan titik B(8,5) bila titik A sebagai titik awal dan titik B sebagai titik akhir, maka buatlah garis yang menghubungkan titik tersebut dengan menggunakan algoritma Bressenham.

Jawab:

Titik awal (x_0, y_0) = A(2,9) dan Titik akhir (x_1, y_1) = B(8,5)

$dx = x_1 - x_0 = 8 - 2 = 6$ dan $dy = y_1 - y_0 = 5 - 9 = -4$

$m = dy/dx = -4/6$; kemiringan garis berada diantara −1 dan 0 : $-1 < m < 0$

$2|dx| = 2.6 = 12$; $2|dy| = 2.|-4| = 8$ dan $2|dy| - 2|dx| = 8 - 12 = -4$

$p_o = 2|dy| - |dx| = 8 - 6 = 2$

==========

Iterasi ke-1 (k = 0):

Titik awal = (2,9)

$Po = 2 > 0$, maka titik selanjutnya adalah

$x = 2 + 1 = 3$ dan $y = 9 - 1 = 8$, koordinat selanjutnya : (3,8)

$p_1 = p_0 + 2|dy| - 2|dx| = 2 - 4 = -2$

==========

Iterasi ke-2 (k = 1):

Titik awal = (3,8)

$P_1 = -2 < 0$, maka titik selanjutnya adalah

$x = 3 + 1 = 4$ dan $y = 8$, koordinat selanjutnya : (4,8)

$p_2 = p_1 + 2|dy| = -2 + 8 = 6$

==========

Iterasi ke-3 (k = 2):

Titik awal = (4,8)

$P_2 = 6 > 0$, maka titik selanjutnya adalah

$x = 4 + 1 = 5$ dan $y = 8 - 1 = 7$, koordinat selanjutnya : (5,7)

$p_3 = p_2 + 2|dy| - 2|dx| = = 6 - 4 = 2$

==========

Iterasi ke-4 (k = 3):

Titik awal = (5,7)

$P_2 = 2 > 0$, maka titik selanjutnya adalah

$x = 5 + 1 = 6$ dan $y = 7 - 1 = 6$, koordinat selanjutnya : (6,6)

$p_4 = p_3 + 2|dy| - 2|dx| = = 2 - 4 = -2$

==========

Iterasi ke-5 (k = 4):

Titik awal = (6,6)

$P_4 = -2 < 0$, maka titik selanjutnya adalah

$x = 6 + 1 = 7$ dan $y = 6$, koordinat selanjutnya : (7,6)

$p_5 = p_4 + 2|dy| = -2 + 8 = 6$

==========

Iterasi ke-6 (k = 5):

Titik awal = (7,6)

$P_5 = 6 > 0$, maka titik selanjutnya adalah

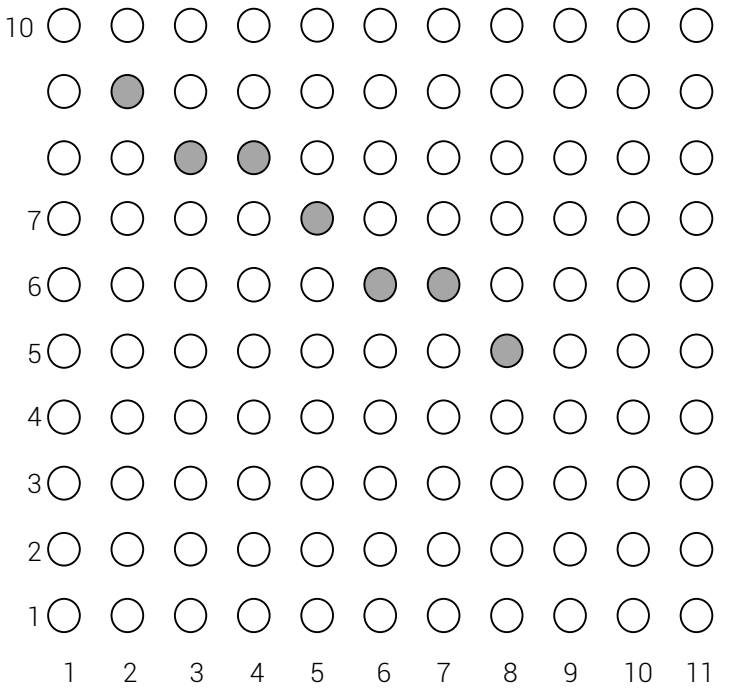
$x = 7 + 1 = 8$ dan $y = 6 - 1 = 5$, koordinat selanjutnya : (8,5)

$p_6 = p_5 + 2|dy| - 2|dx| = 6 - 4 = 2$

==========

Karena $x = x_2 = 8$, maka iterasi dihentikan, sehingga diperoleh titik-titik penyusun garis sebagai berikut: (2,9), (3,8), (4,8), (5,7), (6,6), (7,6) dan (8,5).

Bila digambar pada *raster graphics* diperoleh Gambar 2.9:



Gambar 3. 7 Titik-titik pembentuk garis dengan kemiringan −1 < m < 0, hasil perhitungan menggunakan algoritma Bressenham yang digambar pada *raster graphics*.

Untuk kemiringan garis $m > 1$, tukarlah x dengan y, sehingga algoritma pembentukan garis untuk $m > 1$ adalah sebagai berikut:

1. Tentukan dua titik yang akan dihubungkan dalam pembentukan garis.
2. Tentukan salah satu sebagai titik awal (x_0, y_0) dan titik akhir (x_1, y_1).
3. Hitung dx, dy, $2|dx|$ dan $2|dx| - 2|dy|$
4. Hitung parameter : $po = 2|dx| - |dy|$

5. Untuk setiap y_k sepanjang jalur garis, dimulai dengan k = 0

   - bila $p_k < 0$ maka titik selanjutnya adalah:

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

   - bila tidak, titik selanjutnya adalah:

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

6. Ulangi nomor 5 untuk menentukan posisi piksel berikutnya, sampai

   $x = x_1$ dan $y = y_1$.

Contoh 2.7

Diketahui 2 buah titik A(2,1) dan titik B(4,7) bila titik A sebagai titik awal dan titik B sebagai titik akhir, maka buatlah garis yang menghubungkan titik tersebut dengan menggunakan algoritma Bressenham.

Jawab:

Titik awal (x_0, y_0) = A(2,1) dan Titik akhir (x_1, y_1) = B(4,7)

$dx = x_1 - x_0 = 4 - 2 = 2$ dan $dy = y_1 - y_0 = 7 - 1 = 6$

$m = dy/dx = 6/2$; kemiringan garis lebih besar dari 1 : m > 1

$2|dx| = 4$; $2|dy| = 12$ dan $2|dx| - 2|dy| = 4 - 12 = -8$

$p_o = 2|dx| - |dy| = 4 - 6 = -2$

==========

Iterasi ke-1 (k = 0):

Titik awal = (2,1)

$p_o = -2 < 0$, maka titik selanjutnya adalah

$x = 2$ dan $y = 1 + 1 = 2$, koordinat selanjutnya : (2,2)

$p_1 = p_o + 2|dx| = -2 + 4 = 2$

==========

Iterasi ke-2 (k = 1):

Titik awal = (2,2)

$p_1 = 2 > 0$, maka titik selanjutnya adalah

$x = 2+1 = 3$ dan $y = 2 + 1 = 3$, koordinat selanjutnya : (3,3)

$p_2 = p_1 + 2|dx| - 2|dy| = 2 - 8 = -6$

==========

Iterasi ke-3 (k = 2):

Titik awal = (3,3)

$p_2 = -6 < 0$, maka titik selanjutnya adalah

$x = 3$ dan $y = 3 + 1 = 4$, koordinat selanjutnya : (3,4)

$p_3 = p_2 + 2|dx| = -6 + 4 = -2$

==========

Iterasi ke-4 (k = 3):

Titik awal = (3,4)

$p_3 = -2 < 0$, maka titik selanjutnya adalah

$x = 3$ dan $y = 4 + 1 = 5$, koordinat selanjutnya : (3,5)

$p_4 = p_3 + 2|dx| = -2 + 4 = 2$

==========

Iterasi ke-5 (k = 4):

Titik awal = (3,5)

$p_4 = 2 > 0$, maka titik selanjutnya adalah

$x = 3+1=4$ dan $y = 5 + 1 = 6$, koordinat selanjutnya : (4,6)

$p_5 = p_4 + 2|dx| - 2|dy| = 2 - 8 = -6$

==========

Iterasi ke-6 (k = 5):

Titik awal = (4,6)

$p_5 = -6 < 0$, maka titik selanjutnya adalah

$x = 4$ dan $y = 6 + 1 = 7$, koordinat selanjutnya : (4,7)

$p_6 = p_5 + 2|dx| = -6 + 4 = -2$

==========

Sampai disini iterasi dihentikan.

Untuk kemiringan garis $m < -1$, kita tinggal mengganti komponen berikut:

Untuk setiap y_k sepanjang jalur garis, dimulai dengan k = 0

- bila $p_k < 0$ maka titik selanjutnya adalah:

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

- bila tidak, titik selanjutnya adalah:

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

Contoh 2.8

Diketahui 2 buah titik A(−6,10) dan titik B(0,0) bila titik A sebagai titik awal dan titik B sebagai titik akhir, maka buatlah garis yang menghubungkan titik tersebut dengan menggunakan algoritma Bressenham.

Jawab:

Titik awal (x_0, y_0) = A(−6,10) dan Titik akhir (x_1, y_1) = B(0,0)

$dx = x_1 - x_0 = 0 - (-6) = 6$ dan $dy = y_1 - y_0 = 0 - 10 = -10$

$m = dy/dx = -10/6$; kemiringan garis lebih kecil dari −1 : $m < -1$

$2|dx| = 12$; $2|dy| = 20$ dan $2|dx| - 2|dy| = 12 - 20 = -8$

$p_o = 2|dx| - |dy| = 12 - 10 = 2$

==========

Iterasi ke-1 (k = 0):

Titik awal = (−6,10)

$p_o = 2 > 0$, maka titik selanjutnya adalah

$x = -6 + 1 = -5$ dan $y = 10 - 1 = 9$, koordinat selanjutnya : (−5,9)

$p_1 = p_o + 2|dx| - 2|dy| = 2 - 8 = -6$

==========

Iterasi ke-2 (k = 1):

Titik awal = (−5,9)

$P_1 = -6 < 0$, maka titik selanjutnya adalah

$x = -5$ dan $y = 9 - 1 = 8$, koordinat selanjutnya : (−5,8)

$p_2 = p_1 + 2|dx| = -6 + 12 = 6$

==========

Iterasi ke-3 (k = 2):

Titik awal = (−5,8)

$P_2 = 6 > 0$, maka titik selanjutnya adalah

$x = -5 + 1 = -4$ dan $y = 8 - 1 = 7$, koordinat selanjutnya : $(-4,7)$

$p_3 = p_2 + 2|dx| - 2|dy| = 6 - 8 = -2$

==========

Iterasi ke-4 (k = 3):

Titik awal = $(-4,7)$

$P_3 = -2 < 0$, maka titik selanjutnya adalah

$x = -4$ dan $y = 7 - 1 = 6$, koordinat selanjutnya : $(-4,6)$

$p_4 = p_3 + 2|dx| = -2 + 12 = 10$

==========

Iterasi ke-5 (k = 4):

Titik awal = $(-4,6)$

$P_4 = 10 > 0$, maka titik selanjutnya adalah

$x = -4 + 1 = -3$ dan $y = 6 - 1 = 5$, koordinat selanjutnya : $(-3,5)$

$p_5 = p_4 + 2|dx| - 2|dy| = 10 - 8 = 2$

==========

Iterasi ke-6 (k = 5):

Titik awal = $(-3,5)$

$P_5 = 2 > 0$, maka titik selanjutnya adalah

$x = -3 + 1 = -2$ dan $y = 5 - 1 = 4$, koordinat selanjutnya : $(-2,4)$

$p_5 = p_4 + 2|dx| - 2|dy| = 2 - 8 = -6$

==========

Iterasi ke-7 (k = 6):

Titik awal = $(-2,4)$

$P_3 = -6 < 0$, maka titik selanjutnya adalah

$x = -2$ dan $y = 4 - 1 = 3$, koordinat selanjutnya : $(-2,3)$

$p_4 = p_3 + 2|dx| = -6 + 12 = 6$

==========

Iterasi ke-8 (k = 7):

Titik awal = $(-2,3)$

$P_7 = 6 > 0$, maka titik selanjutnya adalah

$x = -2 + 1 = -1$ dan $y = 3 - 1 = 2$, koordinat selanjutnya : $(-1,2)$

$p_8 = p_7 + 2|dx| - 2|dy| = 6 - 8 = -2$

==========

Iterasi ke-9 (k = 8):

Titik awal = $(-1,2)$

$P_8 = -2 < 0$, maka titik selanjutnya adalah

$x = -1$ dan $y = 2 - 1 = 1$, koordinat selanjutnya : $(-1,1)$

$p_9 = p_8 + 2|dx| = -2 + 12 = 10$

==========

Iterasi ke-10 (k = 9):

Titik awal = (−1,1)

$P_9 = 10 > 0$, maka titik selanjutnya adalah

$x = -1 + 1 = 0$ dan $y = 1 - 1 = 0$, koordinat selanjutnya : (0,0)

$p_{10} = p_9 + 2|dx| - 2|dy| = 10 - 8 = 2$

==========

Sampai disini iterasi dihentikan.

3.4 Lingkaran

Persamaan umum lingkaran dengan pusat lingkaran (x_p, y_p) dan jari-jari ***r*** adalah:

$$(x - x_p)^2 + (y - y_p)^2 = r^2$$

Persamaan 3 - 2

Atau

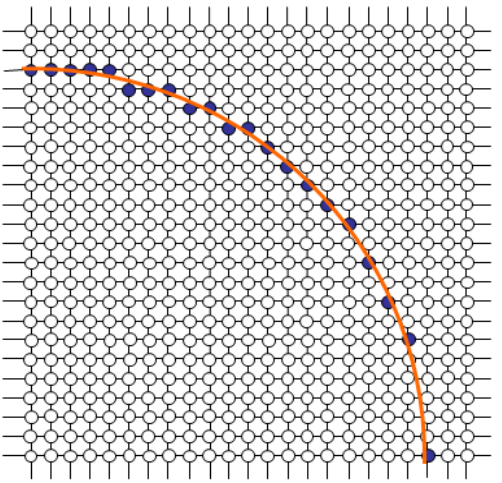
$$y = y_p \pm \sqrt{[r^2 - (x - x_p)^2]}$$

Persamaan 3 - 3

Untuk menggambar piksel-piksel dalam kurva lingkaran, dapat digunakan sumbu *x* dari $x = (x_p - r)$ sampai $x = (x_p + r)$ sebagai parameter dan sumbu *y* sebagai hasil dari persamaan (2-10).

Algoritma ini memerlukan waktu operasi yang besar, karena mengandung operasi perkalian dua bilangan integer, membutuhkan fungsi SQRT (untuk menghitung akar suatu bilangan) dan fungsi ROUND (untuk membulatkan bilangan pecahan menjadi bilangan integer), dan menghasilkan posisi koordinat piksel yang tidak merata, karena terjadinya gaps yang disebabkan adanya perubahan gradient seperti tampak pada Gambar 3.8.

Gambar 3. 8 Terjadinya gaps yang disebabkan adanya perubahan gradient



$$y_0 = \sqrt{20^2 - 0^2} \approx 20$$

$$y_1 = \sqrt{20^2 - 1^2} \approx 20$$

$$y_2 = \sqrt{20^2 - 2^2} \approx 20$$

$$\vdots$$

$$y_{19} = \sqrt{20^2 - 19^2} \approx 6$$

$$y_{20} = \sqrt{20^2 - 20^2} \approx 0$$

menghasilkan posisi koordinat piksel yang tidak merata

Untuk menghindari posisi koordinat piksel yang tidak merata, koordinat piksel (*x*,*y*) dinyatakan dengan menggunakan koordinat polar dalam persamaan (3 - 4)

$$x = x_p + r\cos\theta \qquad \text{dan} \qquad y = y_p + r\sin\theta$$

Persamaan 3 - 4

Akan tetapi penggambaran lingkaran menggunakan persamaan (3-4) memerlukan waktu operasi yang besar karena mengandung operasi perkalian bilangan riel, perhitungan trigonometri, dan membutuhkan banyak segmen garis.

3.4.1 Simetris Delapan Titik

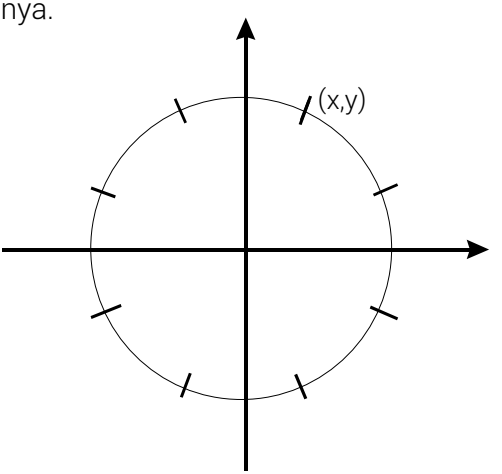
Pembuatan kurva lingkaran dapat dilakukan dengan menentukan titik awal (x,y) yang terletak pada lingkaran, maka tujuh titik yang lain (yang terletak pada lingkaran juga) dapat ditentukan sebagai berikut :

$(-x,y), (x, -y), (-x, -y), (y,x), (-y,x), (y, -x), (-y, -x)$

Sehingga terbentuk delapan titik:

$(x, y), (-x,y), (x, -y), (-x, -y), (y,x), (-y,x), (y, -x), (-y, -x)$

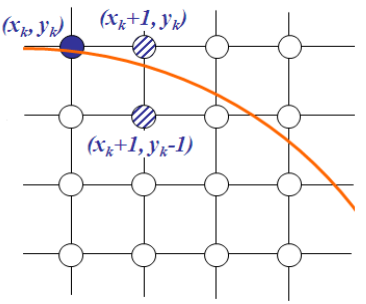
Dengan demikian sebenarnya hanya diperlukan untuk menghitung segmen 45° dalam menentukan lingkaran selengkapnya.



Gambar 3. 9 Delapan titik simetris pada lingkaran

3.4.2 Algoritma Midpoint

Algoritma *midpoint* juga disebut algoritma lingkaran Bressenham. Pembentukan semua titik berdasarkan titik pusat dengan penambahan semua jalur di sekeliling lingkaran. 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. Perhatikan Gambar 3.10.



Gambar 3. 10 Garis lingkaran berada diantara titik (x_k+1, y_k) dan (x_k+1, y_k – 1)

Anggap bahwa kita mempunyai titik (x_k, y_k). Selanjutnya adalah memilih titik (x_k+1, y_k) atau (x_k+1, y_k – 1) yang paling mendekati lingkaran. Persamaan untuk lingkaran

$$f(x,y) = x^2 + y^2 - r^2$$

Sehingga,

$$f\ (x,y) \begin{cases} < 0, & \text{jika } (x,y) \text{ didalam lingkaran} \\ = 0, & \text{jika } (x,y) \text{ terletak pada lingkaran} \\ > 0, & \text{jika } (x,y) \text{ terletak diluar lingkaran} \end{cases}$$

Untuk memilih titik (x_k+1, y_k) atau (x_k+1, y_k – 1) yang paling mendekati lingkaran, kriteria yang digunakan adalah titik tengah (x_k+1, y_k – ½) yaitu

$$p_k = f\ (x_k + 1, y_k - \tfrac{1}{2}) = (x_k + 1)^2 + (y_k - \tfrac{1}{2})^2 - r^2$$

Jika p_k< 0 titik tengah berada didalam lingkaran, maka titik (x_k+1, y_k) dipilih.

Jika p_k> 0 titik tengah berada diluar lingkaran, maka titik (x_k+1, y_k – 1) dipilih.

hitung

$$p_{k+1} = f\ \left(x_{k+1} + 1, y_{k+1} - \tfrac{1}{2}\right) = [(x_k + 1) + 1]^2 + \left(y_{k+1} - \tfrac{1}{2}\right)^2 - r^2$$

Atau

$$p_{k+1} = p_k + 2(x_k + 1) + (y_{k+1}^2 - y_k^2) - (y_{k+1} - y_k) + 1$$

Dimana y_{k+1} adalah y_k atau y_k-1 tergantung tanda dari p_k. Dalam hal ini kriteria pertama yang dihitung adalah

$$p_0 = f(1, r - \tfrac{1}{2}) = 1 + (r - \tfrac{1}{2})^2 - r^2 = \tfrac{5}{4} - r$$

Jika jari-jari lingkaran bernilai integer (bilangan bulat), kita bisa membulatkan nilai p0 menjadi berikut

$$p_0 = 1 - r$$

Kemudian jika $p_k < 0$ maka:

$$p_{k+1} = p_k + 2x_{k+1} + 1$$

Jika $p_k > 0$ maka:

$$p_{k+1} = p_k + 2x_{k+1} + 1 - 2y_k + 1$$

Dari sini diperoleh langkah-langkah algoritma pembentuk lingkaran.

- Langkah-langkah Algoritma Lingkaran Midpoint adalah:

1. Tentukan jari-jari r dan pusat lingkaran (x_p, y_p), kemudian setting sedemikian rupa sehingga titik awal berada pada: $(x_0, y_0) = (0 , r)$
2. Hitung nilai parameter :

$$p_0 = \frac{5}{4} - r$$

Jika jari-jari r pecahan

$$p_0 = 1 - r$$

Jika jari-jari r bulat

3. Untuk setiap posisi x_k, dimulai dengan k = 0 berlaku ketentuan:

- bila $p_k < 0$ maka titik selanjutnya adalah (x_k+1, y_k) dan $p_{k+1} = p_k + 2\, x_{k+1} + 1$

- bila tidak, titik selanjutnya adalah (x_k+1, y_k – 1) dan $p_{k+1} = p_k + 2\,x_{k+1} + 1 - 2\,y_{k+1}$

4. Tentukan titik simetris pada ketujuh oktan yang lain
5. Gerakkan setiap posisi piksel (x, y) pada garis lingkaran dengan titik pusat (x_p, y_p) dan plot nilai koordinat : $x = x + x_p$, $y = y + y_p$
6. Ulangi langkah 3 sampai dengan 5 hingga $x \geq y$

Contoh 2.9

Buatlah gambar kurva lingkaran dengan pusat lingkaran (4,6) dan jari-jari 8, perhitungan berdasarkan dari oktan kuadran pertama dimana $x = 0$ sampai $x = y$. Koordinat titik awal dimulai dari (x,r) = (0,8). Karena jari-jari r bulat, maka gunakan $P_0 = 1 - r$.

Iterasi ke-1:

K = 0 $X_0 = 0$ $Y_0 = r = 8$ $P_0 = 1 - r = 1 - 8 = -7$

Karena $P_0 < 0$, maka :

$X_1 = X_0 + 1 = 0 + 1 = 1$ dan $Y_1 = Y_0 = 8$, jadi Titik selanjutnya : (1,8)

$P_1 = p_0 + 2\,x_1 + 1 = -7 + 2.(1) + 1 = -4$

Dengan algoritma simetris delapan titik, maka diperoleh titik-titik berikut :

(1,8), (−1,8), (1, −8), (−1, −8), (8,1), (−8,1), (8, −1), (−8, −1)

Gerakkan setiap posisi piksel (*x, y*) pada garis lingkaran dengan titik pusat (4,6) diperoleh titik-titik berikut

(5, 14), (3, 14), (5, −2), (3, −2), (12, 7), (−4, 7), (12, 5), (−4, 5)

==========

Iterasi ke-2:

K = 1 X_1 = 1 Y_1 = 8 P_1 = − 4

Karena P1 < 0, maka

$X_2 = X_1 + 1 = 1 + 1 = 2$ dan $Y_2 = Y_1 = 8$, jadi Titik selanjutnya : (2,8)

$P_2 = p_1 + 2 x_2 + 1 = -4 + 2.(2) + 1 = 1$

Dengan algoritma simetris delapan titik, maka diperoleh titik-titik berikut :

(2,8), (−2,8), (2, −8), (−2, −8), (8,2), (−8,2), (8, −2), (−8, −2)

Gerakkan setiap posisi piksel (*x, y*) pada garis lingkaran dengan titik pusat (4,6) diperoleh titik-titik berikut

(6, 14), (2, 14), (6, −2), (2, −2), (12, 8), (−4, 8), (12, 4), (−4, 4)

==========

Iterasi ke-3:

K = 2 X_2 = 2 Y_2 = 8 P_2 = 1

Karena $P_2 > 0$, maka

$X_3 = X_2 + 1 = 2 + 1 = 3$ dan $Y_3 = Y_2 - 1 = 8 - 1 = 7$, jadi Titik selanjutnya : (3,7)

$P_3 = p_2 + 2\,x_3 + 1 - 2\,y_3 = 1 + 2.(3) + 1 - 2.(7) = -6$

Dengan algoritma simetris delapan titik, maka diperoleh titik-titik berikut :

(3,7), (−3,7), (3, −7), (−3, −7), (7,3), (−7,3), (7, −3), (−7, −3)

==========

Gerakkan setiap posisi piksel (*x, y*) pada garis lingkaran dengan titik pusat (4,6) diperoleh titik-titik berikut

(7, 13), (1, 13), (7, −1), (1, −1), (11, 9), (−3, 9), (11, 3), (−3, 3)

Iterasi ke-4:

$K = 3$ $X_3 = 3$ $Y_3 = 7$ $P_3 = -6$

Karena $P_3 < 0$, maka

$X_4 = X_3 + 1 = 3 + 1 = 4$ dan $Y_4 = Y_3 = 7$, jadi Titik selanjutnya : (4,7)

$P_4 = p_3 + 2\,x_4 + 1 = -6 + 2.(4) + 1 = 3$

Dengan algoritma simetris delapan titik, maka diperoleh titik-titik berikut :

(4,7), (−4,7), (4, −7), (−4, −7), (7,4), (−7,4), (7, −4), (−7, −4)

Gerakkan setiap posisi piksel (*x, y*) pada garis lingkaran dengan titik pusat (4,6) diperoleh titik-titik berikut

(8, 13), (0, 13), (8, −1), (0, −1), (11, 10), (−3, 10), (11, 2), (−3, 2)

==========

Iterasi ke-5:

K = 4 $X_4 = 4$ $Y_4 = 7$ $P_4 = 3$

Karena $P_4 > 0$, maka

$X_5 = X_4 + 1 = 4 + 1 = 5$ dan $Y_5 = Y_4 - 1 = 7 - 1 = 6$, jadi Titik selanjutnya : (5,6)

$P_5 = p_4 + 2\,x_4 + 1 - 2\,y_4 = 3 + 2.(5) + 1 - 2.(6) = 2$

Dengan algoritma simetris delapan titik, maka diperoleh titik-titik berikut :

(5,6), (−5,6), (5, −6), (−5, −6), (6,5), (−6,5), (6, −5), (−6, −5)

Gerakkan setiap posisi piksel (*x, y*) pada garis lingkaran dengan titik pusat (4,6) diperoleh titik-titik berikut

(9, 12), (−1, 12), (9, 0), (−1, 0), (10, 11), (−2, 11), (10, 1), (−2, 1)

==========

Iterasi ke-6:

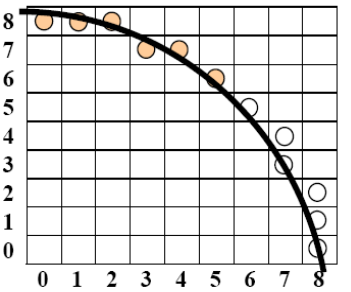
K = 5 $X_5 = 5$ $Y_5 = 6$ $P_5 = 2$

Karena $P_5 > 0$, maka

$X_6 = X_5 + 1 = 5 + 1 = 6$ dan $Y_6 = Y_5 - 1 = 6 - 1 = 5$, jadi Titik selanjutnya : (6,5)

Iterasi dihentikan karena X > Y.

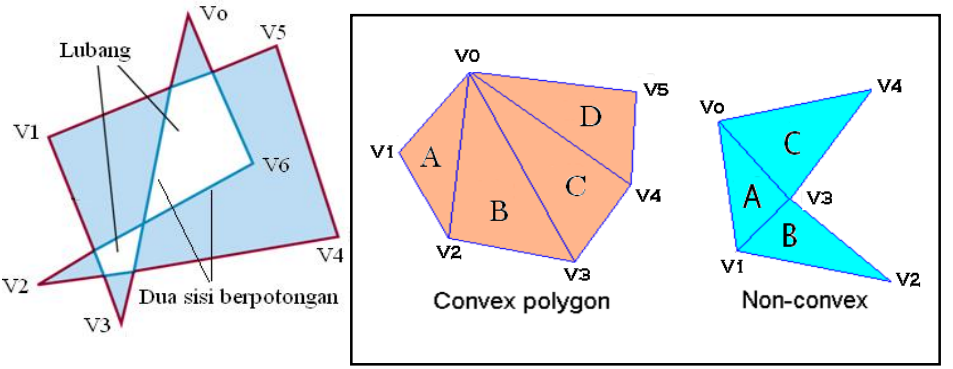
Bila digambar, hasil untuk oktan ke-1 ditunjukkan oleh gambar (3.11).



Gambar 3. 11 Posisi piksel pada pembentukan lingkaran dengan titik pusat (0,0) dan jari-jari 8

3.5 Polygon

Polygon adalah kumpulan garis lurus yang saling menyambung hingga membentuk suatu luasan. Garis-garis ini disebut *edge* (sisi polygon). Titik pertemuan tiap dua sisi disebut verteks. Biasanya polygon dinyatakan dengan koordinat verteks-verteks ini. Ada dua jenis polygon yaitu polygon sederhana dan polygon tidak sederhana. Ciri polygon sederhana adalah tidak semua verteks berada pada bidang yang sama, tidak mempunyai sisi-sisi yang berpotongan, dan tidak mempunyai lubang.



(a)

Gambar 3. 12 Polygon sederhana dan polygon tidak sederhana

Polygon sederhana dibagi menjadi dua yaitu convex polygon dan non convex polygon. Ciri convex polygon adalah semua sudut interiornya < 180°, atau setiap segmen garis yang dihasilkan dari dua buah verteks sembarang dalam polygon berada didalam polygon. Bentuk polygon yang paling sederhana adalah segitiga, karena semua polygon dapat dipecah-pecah menjadi bagian yang terkecil yaitu segitiga seperti pada Gambar 3.12(b). Mengapa polygon ? Dengan polygon secara praktek kita bisa melakukan pendekatan untuk membentuk permukaan setiap obyek 3-D, jika kita mempunyai jumlah polygon yang cukup. Sebagai contoh permukaan bola, torus, dan teko seperti pada Gambar 3.13 dapat dibuat dari beberapa polygon.

Gambar 3. 13 : Permukaan bola, torus dan teko yang terbentuk dari banyak polygon.

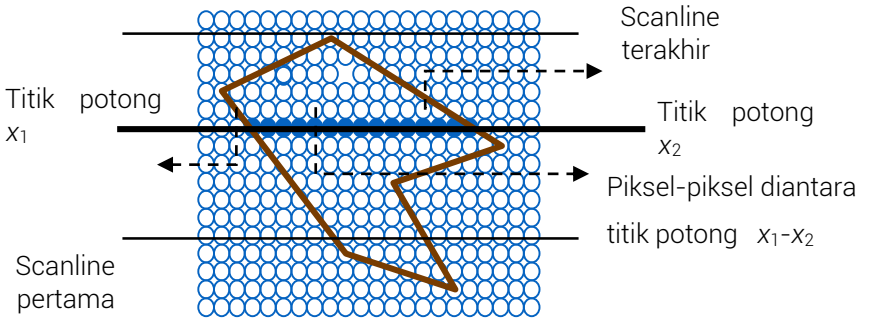
3.6 Filling Polygon

Dalam grafis, pemberian warna dibutuhkan untuk mempercantik tampilan polygon. Karena itu diperlukan algoritma khusus untuk mengisi warna pada polygon tersebut. Teknik atau algoritma untuk pengisian warna pada polygon disebut *filling polygon* atau *area filling*. Ada dua macam dasar pendekatan *filling polygon* pada sistem raster yaitu scan-line *Polygon Fill Algorithm* dan *Boundary-Fill Algorithm*:

3.6.1 *Scan Line Polygon Fill Algorithms*

Pemberian warna pada polygon dilakukan dengan cara men-*scan* secara horisontal dari kiri ke kanan. Hal ini dilakukan untuk mendapatkan titik potong dengan tepi polygon, kemudian mengurutkan nilai-nilai titik potong *x* dari kiri ke kanan dan memberi warna pada piksel-piksel diantara dua pasangan berurutan (x_1- x_2). Hal ini dilakukan dari garis scan yang paling bawah (nilai y terkecil) hingga garis scan yang paling atas seperti pada Gambar 3.14. Metode ini bisa juga digunakan

untuk pengisian warna pada obyek-obyek sederhana lainnya, misalnya lingkaran, ellip dan lain-lain.

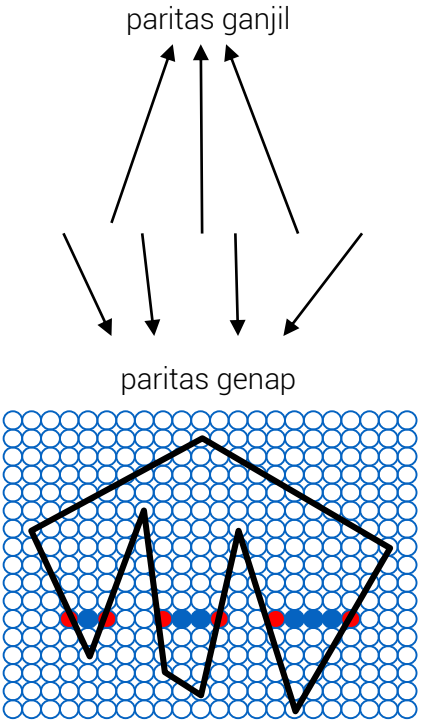


Gambar 3. 14 Metode scan-line

Bagaimana kita tahu bahwa piksel tersebut berada didalam polygon atau diluar polygon ?

Inside-Outside test

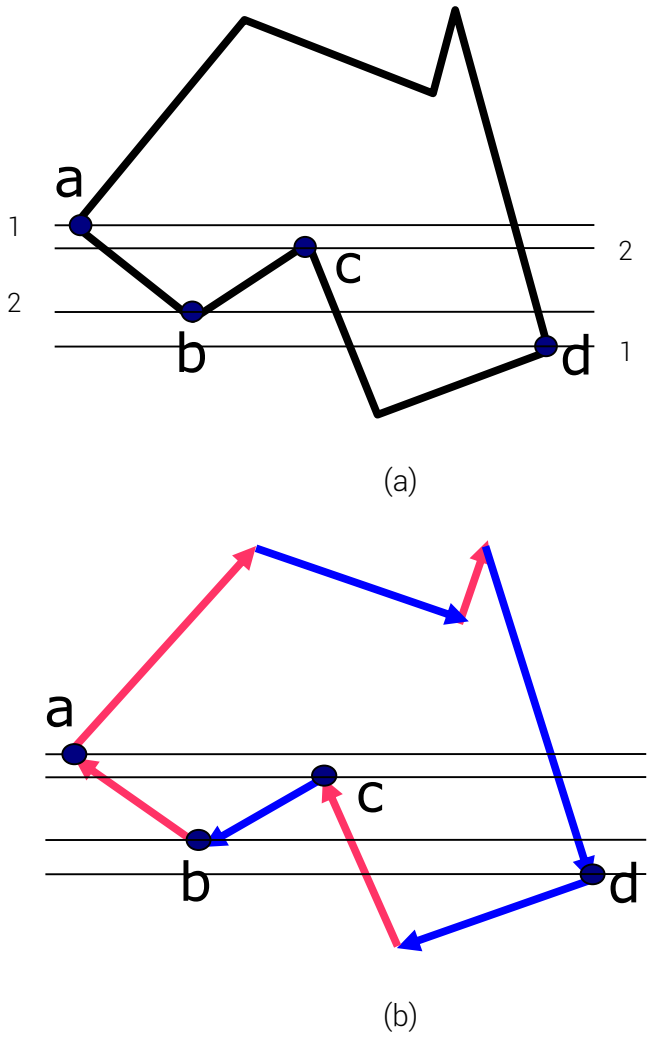
Perhatikan Gambar 3.15. Untuk mengidentifikasi bagian dalam dan bagian luar digunakan aturan paritas ganjil-genap yang biasa disebut sebagai *odd-even rule* atau *odd parity rule*. Semula kita men-set paritas dengan nilai genap. Setiap ditemukan titik potong nilai paritas dibalik, yang semula genap dibalik menjadi ganjil, sebaliknya yang semula ganjil dibalik menjadi genap. Beri warna pada piksel jika paritasnya Ganjil.



Gambar 3. 15 Aturan paritas ganjil-genap untuk mengisi warna

Masalah:

Pada Gambar 3.16(a), verteks a, b, c, dan d merupakan pertemuan dari dua buah segmen garis. Mengapa pada verteks a dan d dihitung sekali, sedangkan pada verteks b dan c dihitung dua kali ?



Gambar 3. 16 Perhitungan paritas pada verteks **a, b, c,** dan **d.**

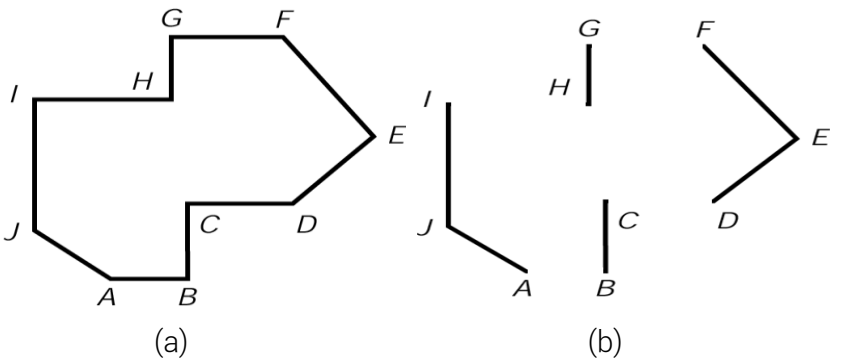
Solusi:

Buatlah perputaran tepi polygon searah jarum jam seperti pada Gambar 2.16(b). Check, apabila pada suatu verteks arah anak

panahnya berubah (pada verteks b dan c yaitu: naik-turun atau turun-naik) maka pada verteks tersebut dihitung dua kali. Sebaliknya jika arah anak panahnya tidak berubah (pada verteks a dan d yaitu: naik-naik atau turun-turun) maka pada verteks tersebut dihitung sekali.

Masalah:

Perhatikan Gambar 3.17(a), Bagaimana perhitungan tepi horisontal AB, CD, GF, dan HI ?



Gambar 3. 17 Polygon dengan sisi-sisi horisontal

Solusi:

Abaikan verteks-verteks yang terletak pada tepi horisontal, atau jangan dimasukkan dalam perhitungan paritas ganjil-genap, sehingga tampak seperti pada Gambar 3.17(b).

Algoritma scanline menggunakan kaidah paritas ganjil-genap:

- Tentukan titik potong garis scan dengan semua sisi polygon
- Urutkan titik potong tadi berdasarkan koordinat sumbu *x*
- Warnai semua piksel diantara pasangan titik potong (x_1-x_2) yang terletak didalam polygon menggunakan kaidah paritas ganjil-genap.

Kelemahan algoritma ini adalah:

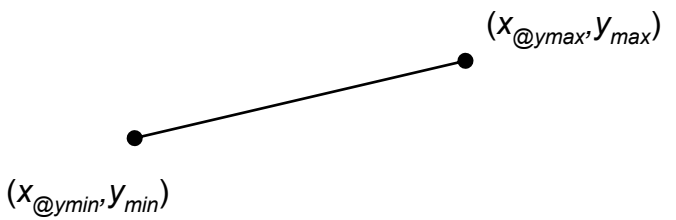
Memerlukan biaya tinggi (*big cost*) karena pada setiap sisi polygon selalu dilakukan pengujian terhadap piksel-piksel.

Solusi:

Menggunakan konsep *edge table* (ET)

Edge Table (ET), *edge table* (ET) adalah tabel yang berisi sisi-sisi polygon. Kita akan menggunakan dua *edge table* yang berbeda, yaitu: *Active Edge Table* (AET) dan *Global Edge Table* (GET). (AET) digunakan untuk menyimpan semua sisi polygon yang berpotongan dengan garis scan, sedangkan GET digunakan untuk menyimpan semua sisi polygon dan meng-*update* AET.

Perhitungan titik potong dengan garis scan



Kita tahu bahwa:

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

Setiap garis scan yang baru berlaku :

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

Kita perlu menghitung *x* untuk setiap garis scan:

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

Sehingga,

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

Dan

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

Maka

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

Ini adalah cara yang efisien untuk menghitung nilai *x*.

Active Edge Table (AET)

- Tabel berisi satu entry per sisi yang berpotongan dengan garis scan aktif.
- Pada setiap garis scan yang baru :
    - Hitung titik potong baru untuk semua sisi menggunakan rumus :
    - Tambahkan setiap sisi baru yang berpotongan
    - Hapus setiap sisi yang tidak berpotongan
- Untuk efisiensi Update AET, kita tetap menjaga GET.

Global Edge Table (GET)

- Tabel yang berisi informasi tentang semua sisi-sisi polygon
- GET mempunyai satu tempat unttuk setiap garis scan
    - Setiap tempat menyimpan daftar sisi yang mempunyai nilai y_{min}
    - Setiap sisi ditentukan hanya dalam satu tempat
- Tiap-tiap entry dalam GET berisi
    - Nilai y_{max} dari sisi
    - Nilai $x_{@ymin}$ (nilai *x* pada titik y_{min})
    - Nilai pertambahan *x* (1/m)

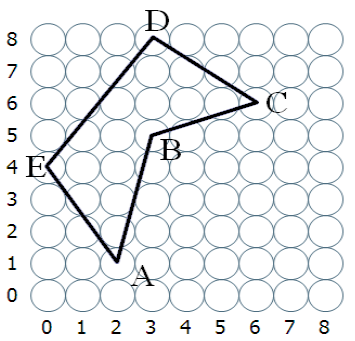
Scan Line Polygon Fill Algorithms menggunakan ET, GET dan AET

1. Tambahkan sisi-sisi polygon ke GET
2. Set *y* ke koordinat *y* terkecil dalam GET

3. Inisialisasi, set AET = kosong
4. Ulang sampai AET dan GET kosong
    a. Tambahkan sisi-sisi dari GET ke AET yang mana $y_{min} = y$.
    b. Hilangkan sisi-sisi dari AET bila $y_{max} = y$.
    c. Urutkan AET berdasarkan x
    d. Warnai piksel yang terletak diantara pasangan titik potong dalam AET
    e. Untuk setiap sisi dalam AET, ganti x dengan $x + 1/m$
    f. Set $y = y + 1$ untuk bergerak ke garis scan berikutnya

Contoh 2.10

Diketahui polygon berikut, gunakan konsep *edge table* untuk mewarnai polygon tersebut.



Masukan GET $(y_{max}, x_{@ymin}, 1/m)$

AB → (5, 2, ¼)

BC → (6, 3, 3)

CD → (8, 6, -3/2)

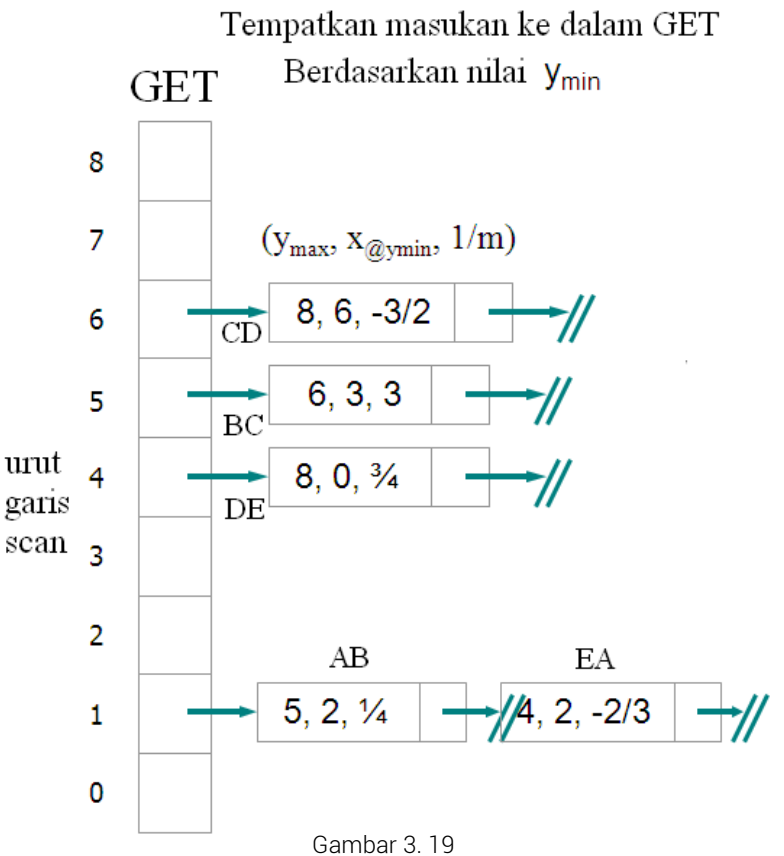
DE → (8, 0, 3/4)

EA → (4, 2, -2/3)

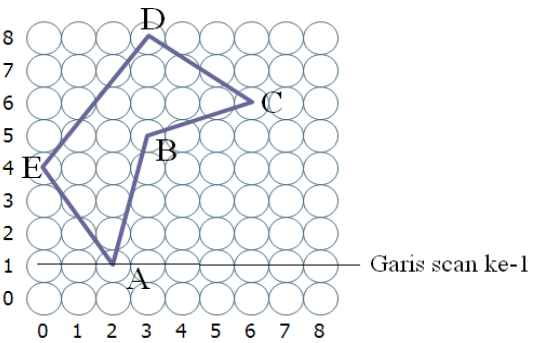
Gambar 3. 18

Sisi-sisi pembentuk polygon:

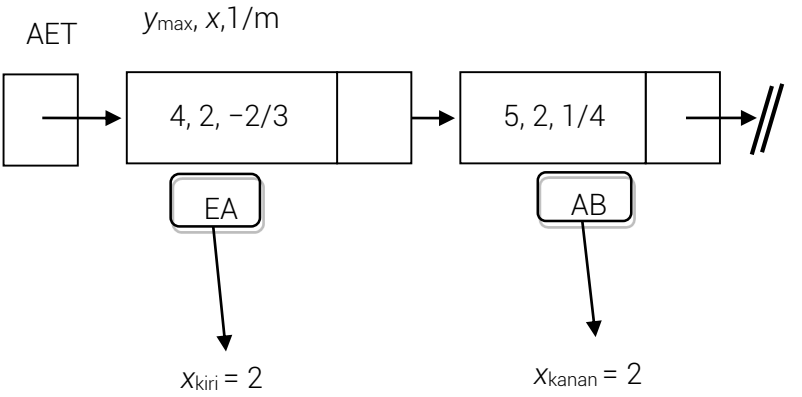
AB = (2, 1), (3, 5)
BC = (3, 5), (6, 6)
CD = (6, 6), (3, 8)
DE = (3, 8), (0, 4)
EA = (0, 4), (2, 1)



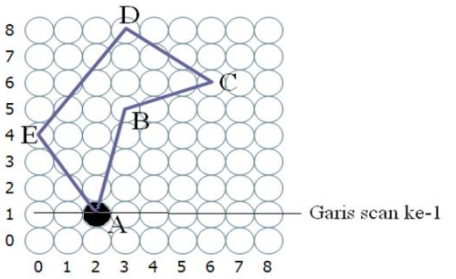
Gambar 3. 19



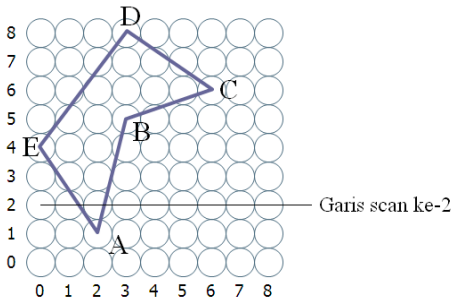
Gambar 3. 20



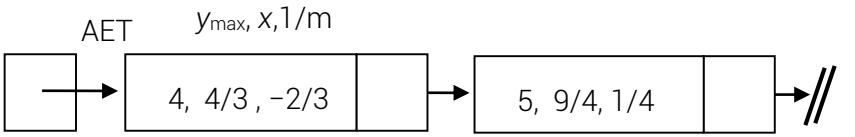
Pewarnaan dilakukan diantara titik potong (x_{kiri} – x_{kanan}) = (2 – 2), hasilnya adalah;



Gambar 3. 21



Gambar 3. 22



EA

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

$= 2 - 2/3$

$= 4/3$

$x_{kiri} = 4/3 \approx 1$

dan

AB

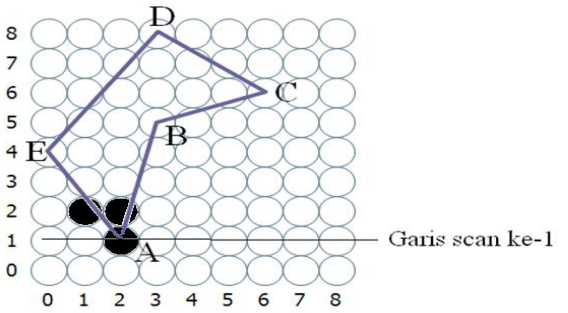
$x_{i+1} = x_i + 1/m$

$= 2 + 1/4$

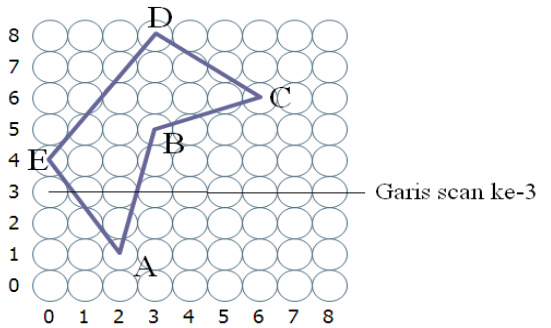
$= 9/4$

$x_{kanan} = 9/4 \approx 2$

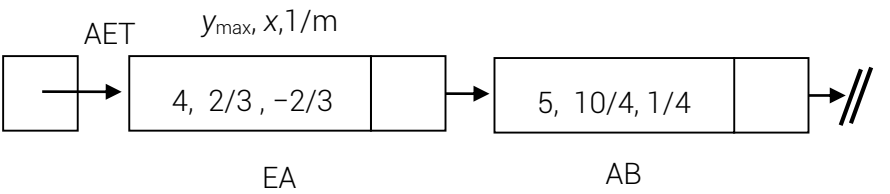
Pewarnaan dilakukan diantara titik potong ($x_{kiri} - x_{kanan}$) = (1 – 2), hasilnya adalah



Gambar 3. 23



Gambar 3. 24



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

$= 4/3 - 2/3$

$= 2/3$

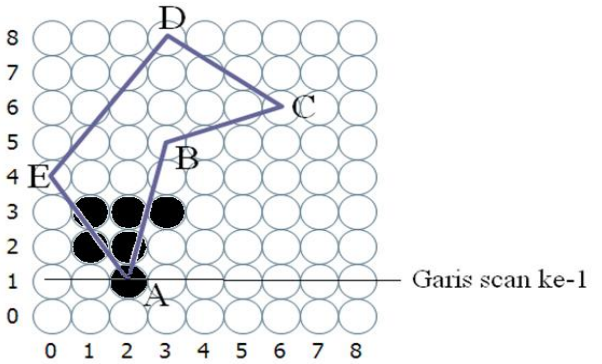
$x_{i+1} = x_i + 1/m$

$= 9/4 + 1/4$

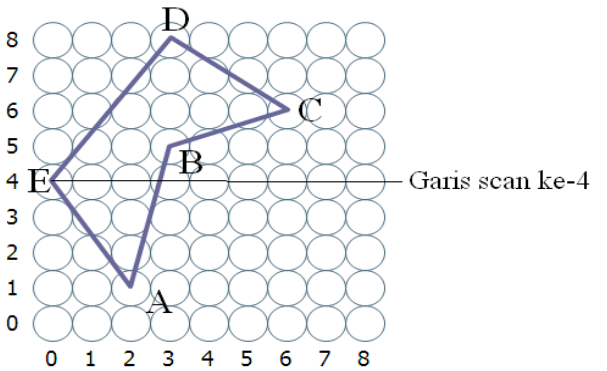
$= 10/4$

$x_{kiri} = 2/3 \approx 1$ dan $x_{kanan} = 10/4 \approx 3$

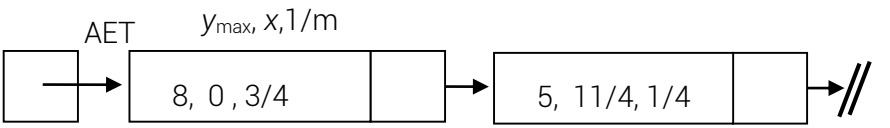
Pewarnaan dilakukan diantara titik potong $(x_{kiri} - x_{kanan}) = (1 - 3)$, hasilnya adalah;



Gambar 3. 25



Gambar 3. 26



~~EA~~ = DE

AB

pada EA, y_{max} = 4. EA harus dihapus dari AET. Dalam GET sisi DE tersimpan pada y = 4. Jadi sisi EA diganti dengan sisi DE

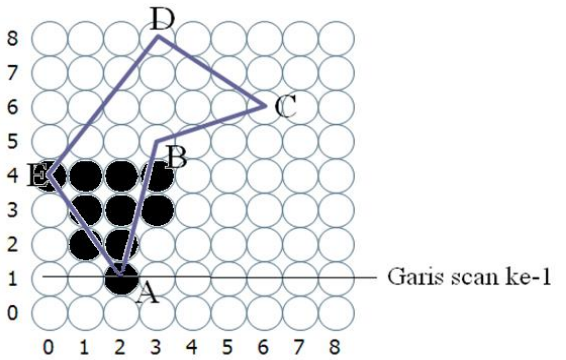
$x_{i+1} = x_i + 1/m$

$= 10/4 + 1/4$

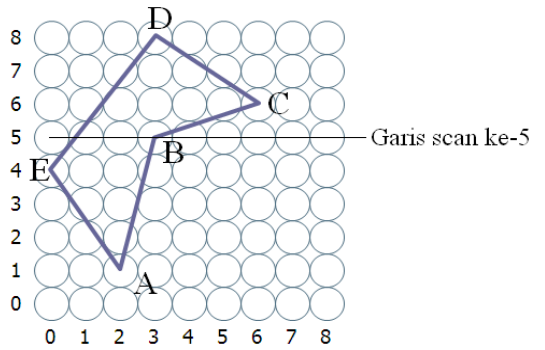
$= 11/4$

$x_{kiri} = 0$ dan $x_{kanan} = 11/4 \approx 3$

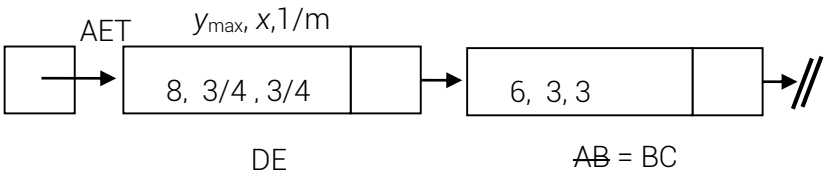
Pewarnaan dilakukan diantara titik potong (x_{kiri} – x_{kanan}) = (0 – 3), hasilnya adalah



Gambar 3. 27



Gambar 3. 28



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

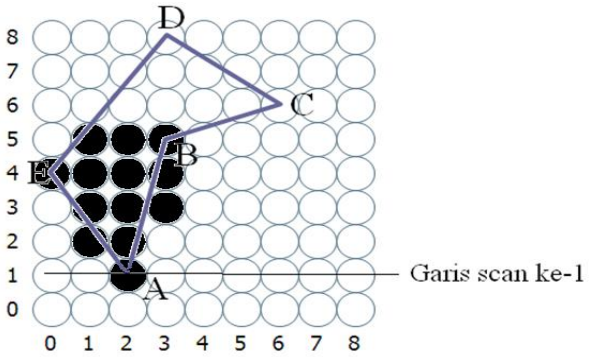
$= 0 + 3/4$

$= 3/4$

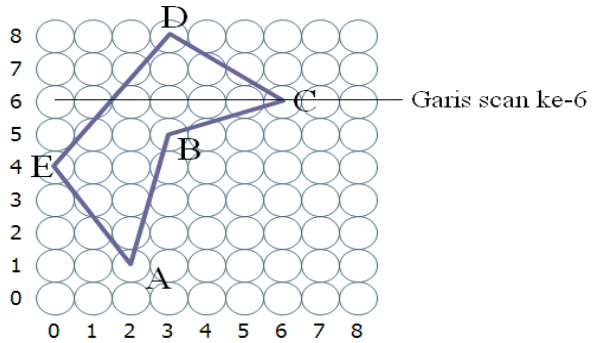
pada AB, y_{max} = 5. AB harus dihapus dari AET. Dalam GET sisi BC tersimpan pada y = 5. Jadi sisi AB diganti dengan sisi BC

x_{kiri} = 3/4 ≈ 1 dan x_{kanan} = 3

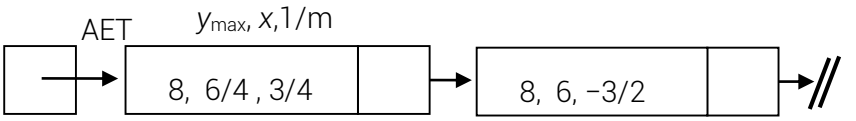
Pewarnaan dilakukan diantara titik potong ($x_{kiri} - x_{kanan}$) = (1 – 3), hasilnya adalah;



Gambar 3. 29



Gambar 3. 30



DE

~~BC~~ = CD

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

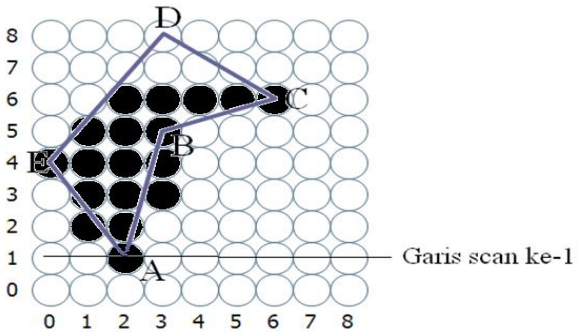
$= 3/4 + 3/4$

$= 6/4$

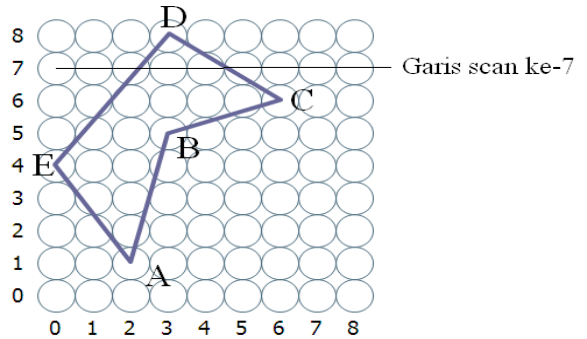
$x_{kiri} = 6/4 \approx 2$ dan $x_{kanan} = 6$

pada BC, y_{max} = 6. BC harus dihapus dari AET. Dalam GET sisi CD tersimpan pada y = 6. Jadi sisi BC diganti dengan sisi CD

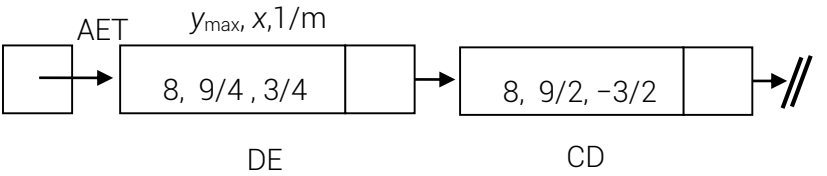
Pewarnaan dilakukan diantara titik potong (x_{kiri} – x_{kanan}) = (2 – 6), hasilnya adalah;



Gambar 3. 31



Gambar 3. 32



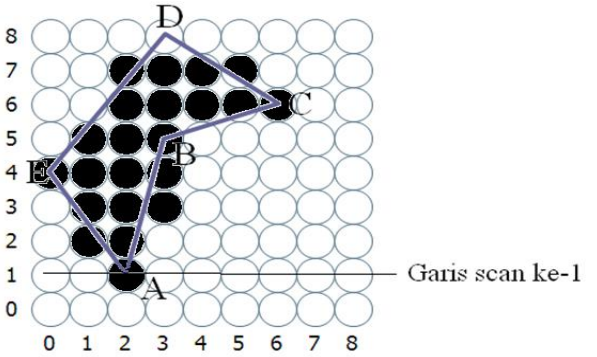
$x_{i+1} = x_i + 1/m$ $\quad$ $x_{i+1} = x_i + 1/m$

$= 6/4 + 3/4$ $\quad$ $= 6 - 3/2$

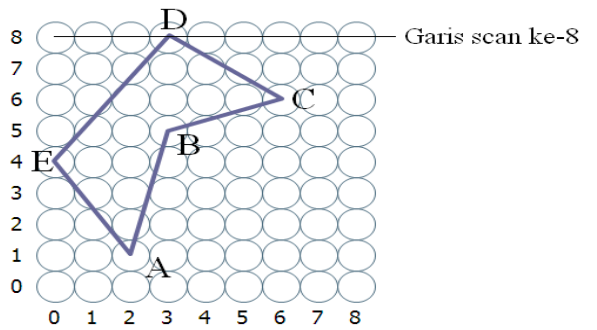
$= 9/4$ $\quad$ $= 9/2$

$x_{kiri} = 9/4 \approx 2$ dan $x_{kanan} = 9/2 \approx 5$

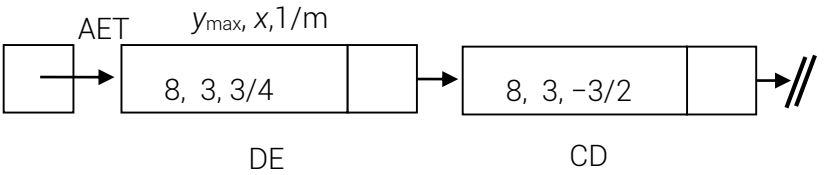
Pewarnaan dilakukan diantara titik potong (x_{kiri} – x_{kanan}) = (2 – 5), hasilnya adalah;



Gambar 3. 33



Gambar 3. 34

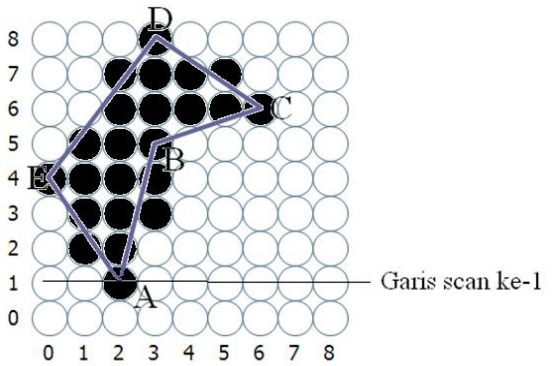


$x_{i+1} = x_i + 1/m$
$= 9/4 + 3/4$
$= 3$

$x_{i+1} = x_i + 1/m$
$= 9/2 - 3/2$
$= 3$

$x_{kiri} = 3$ dan $x_{kanan} = 3$

Pewarnaan dilakukan diantara titik potong ($x_{kiri} - x_{kanan}$) = (3 – 3), hasilnya adalah;



Gambar 3. 35

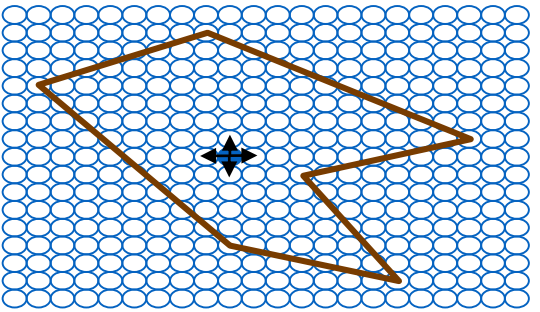
Karena sisi polygon dalam GET ataupun AET sudah habis, maka proses dihentikan.

3.6.2 *Boundary-Fill Algorithm*

Prosedur *boundary-fill* menerima tiga parameter yaitu: koordinat titik (*x,y*), warna isi dan warna garis batas. Proses pengisian warna tertentu dimulai dari titik (*x,y*), kemudian memeriksa posisi titik tetangganya, apakah titik tetangga tersebut memiliki warna batas:

- Jika tidak, warnai titik tersebut dengan warna tertentu.
- Selanjutnya periksa lagi posisi dan warna titik tetangganya.
- Proses diulangi terus hingga seluruh titik pada area pengisian telah diuji.

Dengan teknik ini pengisian warna dimulai dari sebuah titik yang berada didalam area (*x*,y) polygon dan mewarnai piksel dari titik tetangganya hingga semua piksel yang berada didalam polygon telah diwarnai seperti pada Gambar 2.20.

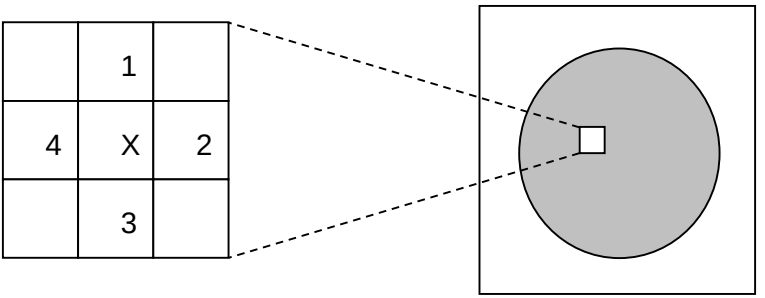


Gambar 3. 36 Metode *Boundary-Fill*

Ada 2 macam cara untuk melihat titik tetangga, yaitu :

- 4 tetangga, melihat titik yang berada diatas, bawah, kanan dan kiri
- 8 tetangga, melihat titik yang berada diatas, bawah, kanan, kiri, pojok kiri atas, pojok kiri bawah, pojok kanan atas dan pojok kanan bawah.

a) 4 - tetangga



Gambar 3. 37

b) 8 - tetangga

1	2	3
8	X	4
7	6	5

Gambar 3. 38

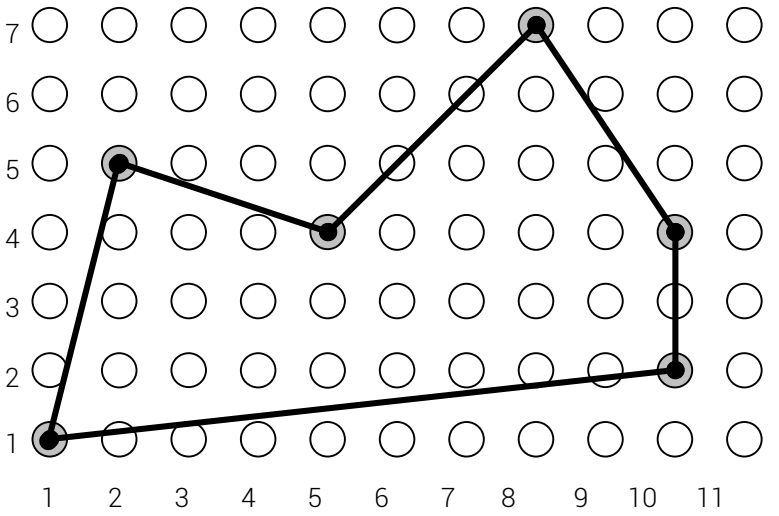
Contoh 2.11

diketahui : polygon = {(1,1), (2,,5), (5,4), (8,7), (10,4), (10,2), (1,1)}, lakukan *Area Filling* menggunakan algoritma *Boundary Fill Algorithm* 4-tetangga.

Jawab:

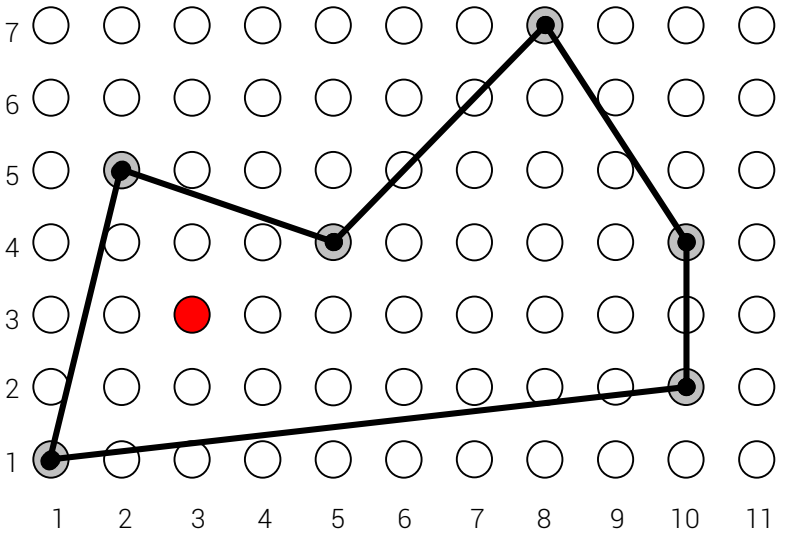
titik-titik sebagai pembentuk polygon = {(1,1), (2,,5), (5,4), (8,7), (10,4), (10,2), (1,1)}.

Bila poligon tersebut digambar, diperoleh gambar berikut :



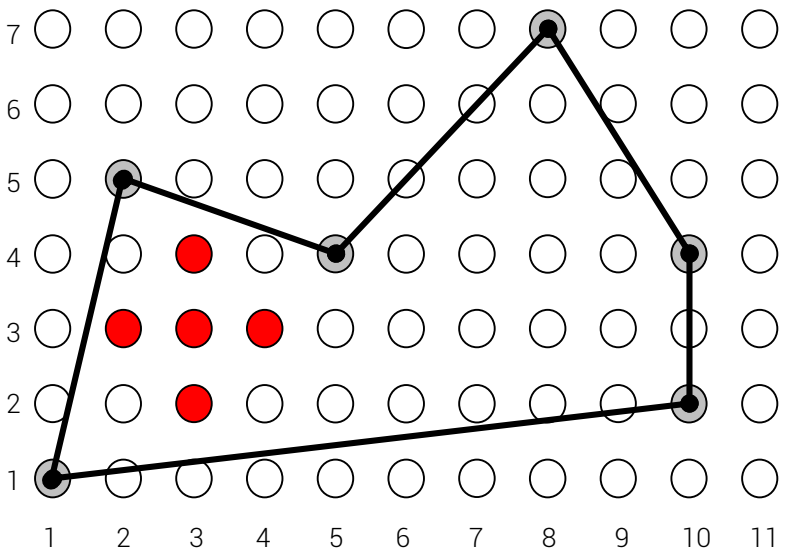
Gambar 3. 39

Misalkan titik awal pencarian adalah (3,3). Tandai titik (3,3) dengan warna tertentu, misalnya warna merah. Lihat 4-tetangganya, yaitu titik (3,2), (3,4), (2,3), (4,3).



Gambar 3. 40

Ke-4 tetangga tersebut bukan garis batas poligon, sehingga 4-titik tersebut diwarnai merah.



Gambar 3. 41

Titik yang telah diproses: (3,3)

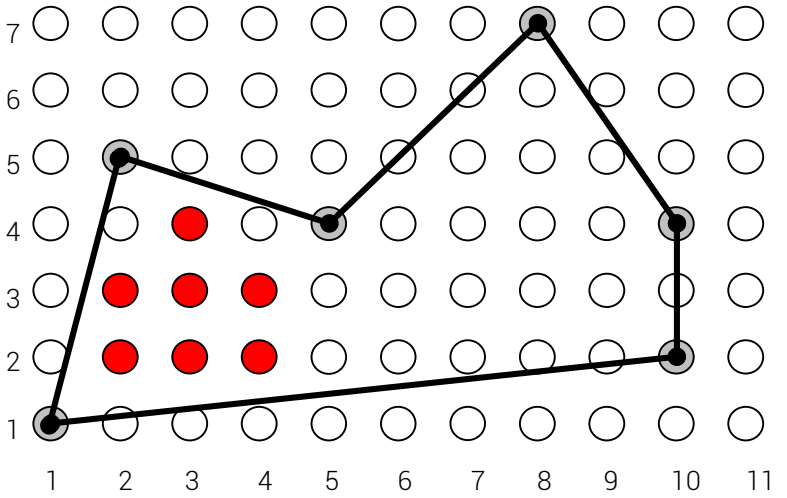
Titik yang belum diproses : (3,2), (3,4), (2,3), (4,3)

Ambil titik (3,2).

Titik yang telah diproses: (3,2), (3,3)

Titik yang belum diproses : (3,4), (2,3), (4,3)

4-tetangga titik tersebut adalah (3,3), (3,1), (2,2), (4,2). Terlihat bahwa titik (4,2) dan (2,2) bukan garis batas poligon, sehingga diwarnai dengan warna merah. Titik (3,3) sudah diwarnai. Titik (3,1) adalah garis batas jadi tidak diwarnai.



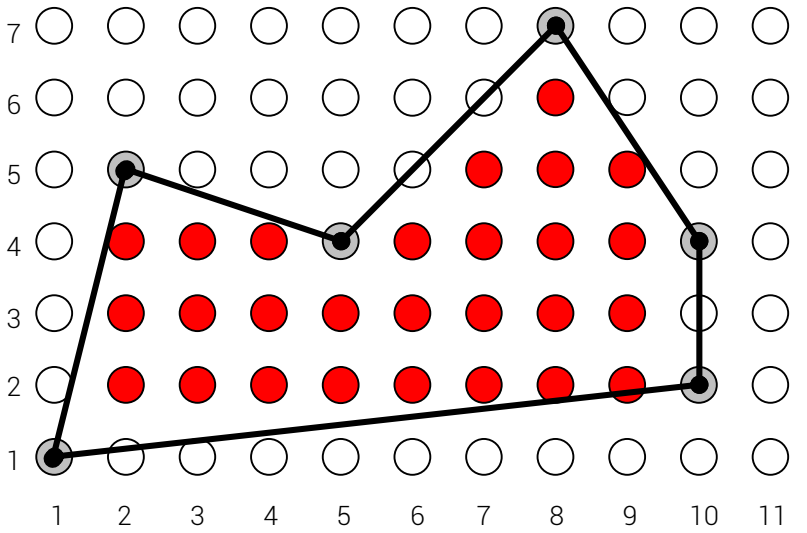
Gambar 3. 42

Titik yang telah diproses: (3,3)(3,2)

Titik yang belum diproses : (3,4), (2,3), (4,3) (2,2), (4,2)

Ambil titik (3,4). 4-tetangga titik tersebut adalah (3,3), (3,5), (2,4), (4,4). Titik (3,3) sudah diwarnai. Titik (3,5), (2,4) dan (4,4) adalah garis batas jadi tidak diwarnai.

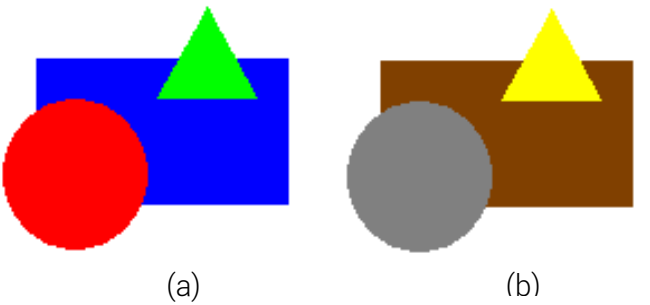
Proses diulang sehingga seluruh bagian dalam poligon diwarnai dengan warna merah.



Gambar 3. 43

3.6.3 *Flood-Fill Algorithm*

Terkadang kita ingin mewarnai (atau memberi warna yang baru) pada sebuah area yang mempunyai warna lebih dari satu. Perhatikan gambar 2.21 berikut:



Gambar 3. 44 penggantian warna obyek menggunakan *Flood-Fill Algorithm* . (a) lingkaran berwarna merah, segitiga berwarna hijau dan persegi berwarna biru. Obyek tersebut warnanya diubah menjadi (b) lingkaran berwarna abu-abu, segitiga berwarna kuning dan persegi berwarna coklat

Algoritma ini dimulai dari titik yang berada didalam area (*x,y*) dan mengganti semua pikselnya dengan warna baru sehingga bagian dalam area mempunyai warna yang sama. Pengujian titik tetangga bisa menggunakan 4-tetangga atau 8-tetangga.

Ringkasan Bab 3

- Output primitif adalah struktur dasar geometri yang paling sederhana dari gambar grafika komputer. Titik dan garis adalah contoh dari output primitif yang dapat digunakan untuk membentuk gambar, misalnya lingkaran, kerucut, permukaan berbentuk persegi, kurva dan permukaan berbentuk lengkung, warna area dan karakter, dan lain-lain.
- *Piksel* adalah elemen gambar terkecil berupa sebuah titik yang ditempatkan dilayar. warna merupakan atribut dari piksel.
- *Frame buffer* adalah area memory tempat informasi gambar disimpan.
- Ada 3 algoritma untuk membentuk garis, yaitu Algoritma *brute force*, DDA, dan Bressenham. Sedangkan untuk membentuk lingkaran digunakan algoritma lingkaran midpoint.
- Polygon adalah kumpulan garis lurus yang saling menyambung hingga membentuk suatu luasan. Garis-garis ini disebut *edge* (sisi polygon). Titik pertemuan tiap dua sisi disebut verteks.
- Untuk mengisi warna pada polygon diperlukan algoritma khusus. Teknik atau algoritma untuk pengisian warna pada polygon disebut *filling polygon* atau *area filling*. Ada dua macam dasar pendekatan *filling polygon* pada sistem raster yaitu scan-line *Polygon Fill Algorithm* dan *Boundary-Fill Algorithm*.

Soal-Soal Latihan

1. Apa yang dimaksud dengan output primitif ? Sebutkan !

2. Diketahui 2 buah titik A dan titik B. Bila titik A sebagai titik awal dan titik B sebagai titik akhir, tentukan titik-titik antara yang menghubungkan titik A dan titik B sehingga membentuk garis AB dengan menggunakan (a) Algoritma *brute force* (b) algoritma DDA (c) algoritma Bressenham , jika :

   i) A(3,2) dan B(11, 6)

   ii) A(3,2) dan B(7, 7)

   iii) A(–5, 10) dan B(0, 0)

   iv) A(–5, 4) dan B(0,0)

3 Berdasarkan soal no.2, bagaimana menurut saudara, mana algoritma yang lebih baik, Algoritma *brute force* , algoritma DDA atau algoritma Bressenham ? Mengapa ?

4. (a) Buatlah gambar kurva lingkaran dengan pusat lingkaran (0,0) dan jari-jari 6, perhitungan berdasarkan dari oktan kuadran pertama dimana $x = 0$ sampai $y = r$. Koordinat titik awal dimulai dari $(x,r) = (0,6)$. Untuk mempermudah perhitungan gunakan $P_0 = 1 - r$ (sekali lagi, ini hanya untuk mempermudah perhitungan dalam contoh). (b) sama seperti soal (a), tetapi pusat lingkarang di P(2,5).

5. Diketahui : polygon = {(2,1), (3,6), (5,4), (8,8), (10,4), (12,2), (2,1)}, lakukan *Area Filling* menggunakan (a) algoritma *Scan Line Polygon* (b) algoritma *Boundary Fill*.

Bab 4

Atribut Output Primitif

TUJUAN PEMBELAJARAN

- Agar pembaca memahami pengertian atribut output primitive
- Agar pembaca memahami atribut titik
- Agar pembaca memahami atribut garis

OUTCOME PEMBELAJARAN

- Pembaca bisa mendefinisikan pengertian output primitive
- Pembaca bisa menjelaskan atribut titik
- Pembaca bisa menjelaskan atribut garis

Pendahuluan

Setelah berhasil membangun output primitif, langkah selanjutnya adalah melakukan pengaturan terhadap atribut output primitif. Atribut adalah semua parameter yang mempengaruhi bagaimana primitive grafis ditampilkan. Atribut dari output primitif dapat berupa:

- Ukuran garis batas
- Tipe garis batas

- Warna garis batas
- Warna objek (Fill color / Area Filling)

4.1 Atribut Titik

Atribut dasar untuk titik adalah ukuran dan warna. Ukuran titik direpresentasikan sebagai beberapa piksel. Sedangkan warna titik bisa berupa monokrom (hitam/putih), grayscale (abu-abu), ataupun berwarna (RGB).

titik ukuran 20 point, dengan warna biru

titik ukuran 26 point, dengan warna abu-abu

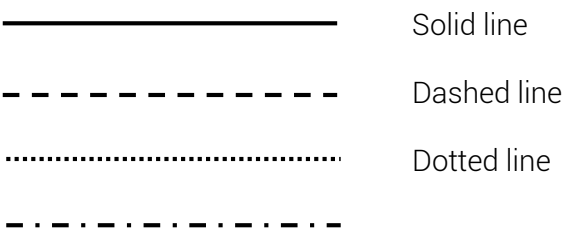
4.2 Atribut Garis

Atribut dasar untuk garis adalah tipe (type), tebal (width) dan warna (color). Dalam beberapa paket program aplikasi grafika seperti paint, photo shop, corel draw dan lain-lain, garis dapat ditampilkan dengan menggunakan pilihan pen atau brush.

4.3 Tipe Garis

Garis memiliki beberapa tipe, seperti garis tanpa terputus (solid line), garis putus-putus (dashed line), garis titik-titik (dotted line) dan kombinasi garis dan titik (dash-dotted line). Garis putus-putus dibuat dengan memberikan jarak dengan bagian solid yang sama. Garis titik-titik dapat dibuat dengan cara memberikan jarak yang lebih besar dari bagian solid line. Garis putus-putus dapat juga dihasilkan dari system raster

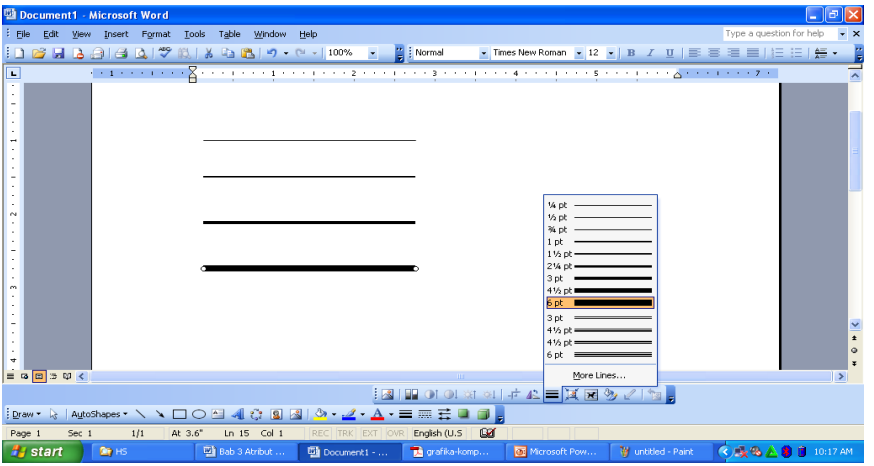
menggunakan pixel mask, contohnya 11100000 akan menampilkan garis putus-putus dengan panjang dash 3 dan jarak antar dash 5. Gambar 4.1 memperlihatkan tipe-tipe garis:



Gambar 4. 1 Type garis : solid line, dashed line, dotted line dan dashed-dotted line

4.3.1 Ukuran Garis

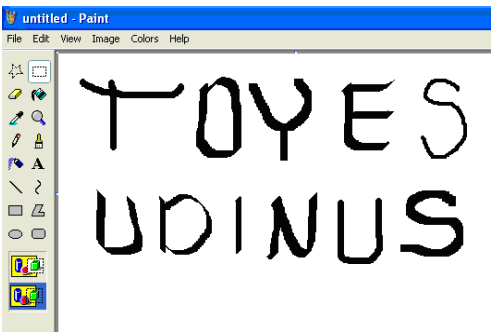
Implementasi ukuran garis tergantung pada kemampuan device output yang digunakan. Pada video monitor garis tebal ditampilkan sebagai kumpulan garis sejajar yang berdekatan, sedangkan pada plotter akan menggunakan ukuran pen yang berbeda-beda. Pada raster, tebal garis standar diperoleh dengan cara menempatkan satu pixel pada tiap posisi, seperti pada algortima Bressenham. Garis dengan ketebalan yang lain diperoleh dengan perkalian integer positif dan garis standar. Gambar 3-2 menunjukkan berbagai ukuran garis mulai ¼ pt sampai dengan 6 pt yang terdapat pada software aplikasi Microsoft Word.



Gambar 4. 2 Ukuran garis dari ¼ pt sampai dengan 6 pt pada software aplikasi Microsoft Word.

4.3.2 Pen dan Brush

Penggunaan pen dan brush pada aplikasi tertentu terkadang sangat dibutuhkan. Pen dan Brush merupakan atribut lain dari garis yang mempunyai beberapa kategori yaitu, bentuk, ukuran dan pola. Beberapa bentuk pen atau brush dapat dilihat pada Gambar 4.3 (diambil dari aplikasi Paint) :



Gambar 4. 3 Beberapa bentuk pen atau brush pada paket program aplikasi Paint

4.4 Warna Garis

Pada system raster, pixel merupakan komponen dasar penyusun garis. Sehingga atribut warna garis ditentukan oleh atribut warna dari pixel penyusunnya. Jumlah warna setiap pixel bergantung pada jumlah bit yang tersedia per pixel pada frame buffer. Bila satu pixel mempunyai jumlah 1 bit, maka pixel tersebut berwarna hitam atau putih. Artinya pixel tersebut mempunyai 2 kemungkinan warna yaitu hitam atau putih. Bila satu pixel mempunyai jumlah 8 bit, maka pixel tersebut mempunyai 256 kemungkinan warna atau biasa disebut sebagai grayscale. Bila satu pixel mempunyai jumlah 24 bit, maka pixel tersebut mempunyai sekitar 16 juta kemungkinan warna atau biasa disebut sebagai true color atau RGB.

Ringkasan Bab 4

Atribut adalah semua parameter yang mempengaruhi bagaimana primitive grafis ditampilkan. Atribut dari output primitif dapat berupa:

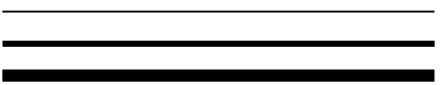
- Ukuran garis batas
- Tipe garis batas
- Warna garis batas
- Warna objek (Fill color / Area Filling)

Atribut dasar untuk titik adalah ukuran dan warna.

Atribut dasar untuk garis adalah tipe (type), tebal (width) dan warna (color).

Soal-Soal Latihan

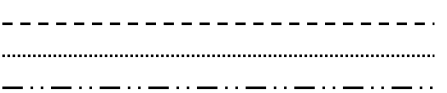
1. Buatlah algoritma untuk membuat garis dengan ketebalan tertentu.



2. Buatlah algoritma untuk membuat garis dengan ketebalan tertentu dan warna tertentu.

3. Buatlah algoritma untuk membuat garis dengan ketebalan tertentu, warna biru dan warna tepi garis adalah merah.

4. Ubahlah algoritma pembentukan garis DDA dan Bressenham untuk membuat garis putus-putus (dashed), garis titik-titik (dotted) dan garis putus-titik-titik.



Bab 5

Transformasi Geometri

TUJUAN PEMBELAJARAN

- Pembaca bisa memahami konsep transformasi geometri 2-D dan 3-D: translasi, rotasi, Refleksi, Shear dan scalling.

OUTCOME PEMBELAJARAN

- Pembaca bisa menghitung transformasi geometri 2-D secara manual

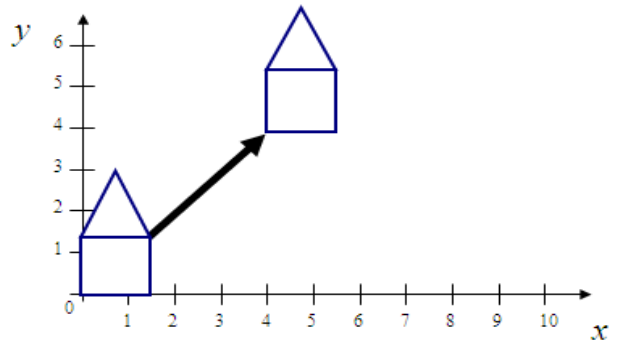
Pendahuluan

Transformasi geometri digunakan untuk memberikan metode-metode perubahan bentuk dan posisi dari sebuah obyek. Sehingga transformasi merupakan alat yang paling mendasar yang digunakan untuk grafika komputer. Transformasi membantu menyederhanakan tugas-tugas dari pemodelan geometri (geometric modeling), animasi, dan rendering.

Transformasi yang paling banyak digunakan di dalam grafika komputer adalah transformasi *affin* (*affine tranformation*), yang mempunyai bentuk sangat sederhana. Sejumlah transformasi dasar dari transformasi *affin* antara lain adalah : penggeseran (*translation*), penskalaan (*scaling*), dan pemutaran (*rotation*).

5.1 Translasi (Pergeseran)

Sebuah titik A(x,y) digeser searah sumbu X sejauh t_x dan searah sumbu Y sejauh t_y (perhatikan Gambar 5.1),



Pergeseran posisi obyek relatif terhadap titik asal

maka titik hasil pergeseran tersebut dapat ditulis sebagai berikut :

$$x' = x + t_x$$

$$y' = y + t_y$$

atau dapat disusun sebagai berikut :

$$x' = x + 0.y + t_x$$

$$y' = 0.x + y + t_y$$

atau dalam bentuk matriks :

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

Contoh 5.1

Tentukan posisi dari segitiga ABC yang dibentuk oleh titik-titik A(20,20), B(100,20) dan C(60,120) jika dilakukan penggeseran pada $\begin{bmatrix} 80 \\ 70 \end{bmatrix}$.

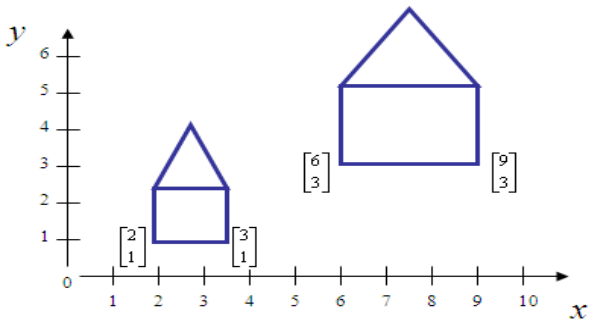
Jawab:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 20 & 100 & 60 \\ 20 & 20 & 120 \end{bmatrix} + \begin{bmatrix} 80 & 80 & 80 \\ 70 & 70 & 70 \end{bmatrix} = \begin{bmatrix} 100 & 180 & 140 \\ 90 & 90 & 190 \end{bmatrix}$$

Yaitu A'(100,90), B'(180,90) dan C'(140,190)

5.2 *Scalling* (Penskalaan)

Penskalaan adalah proses untuk memperbesar atau memperkecil suatu obyek atau gambar. Misalkan titik A(*x*,*y*) diskalakan terhadap titik P(*a*,*b*) dengan faktor skala sebesar S_x searah sumbu X dan sebesar S_y searah sumbu Y (perhatikan Gambar 5.1,



Obyek mengalami perbesaran relatif terhadap titik asal

maka koordinat hasil penskalaan dapat ditentukan sebagai berikut :

$x' = S_x(x-a) + a$

$y' = S_y(y-b) + b$

atau

$x' = S_x x + a - S_x a$

$y' = S_y y + b - S_y b$

atau dalam bentuk matriks :

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} a - S_x a \\ b - S_y b \end{bmatrix}$$

Jika pusat penskalaannya adalah sumbu koordinat P(0,0), maka $a = 0$ dan $b = 0$, sehingga persamaannya menjadi :

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

Matrik penyajian untuk penskalaan terhadap titik pusat P(0,0) adalah;

$$S = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix}.$$

Contoh 5.2

Tentukan posisi dari segitiga ABC yang dibentuk oleh titik-titik A(20,20), B(100,20) dan C(60,120), jika dilakukan penskalaan dengan faktor skala $\begin{bmatrix} 4 \\ 2 \end{bmatrix}$ terhadap titik pusat P(0,0)

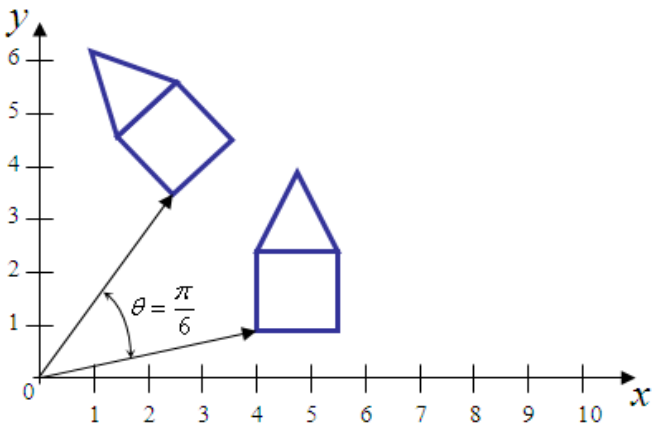
Jawab:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 4 & 0 \\ 0 & 2 \end{bmatrix} \cdot \begin{bmatrix} 20 & 100 & 60 \\ 20 & 20 & 120 \end{bmatrix} = \begin{bmatrix} 80 & 400 & 240 \\ 40 & 40 & 240 \end{bmatrix}$$

Yaitu A'(80,40), B'(400,40) dan C'(240,240)

5.3 Rotasi (Perputaran)

Seperti halnya pergeseran dan penskalaan, untuk pemutaran sembarang obyek dilakukan dengan pemutaran setiap titik ujung garis. Pemutaran searah jarum jam akan dinyatakan dengan sudut negatif, sedangkan pemutaran berlawanan dengan jarum jam dinyatakan dengan sudut postif. Dengan menganggap besarnya sudut putar adalah θ, maka hasil pemutaran titik A(x,y) dengan pusat putar P(0,0) akan dihasilkan titik A'(x',y') seperti yang diperlihatkan pada Gambar 5.3 berikut:



$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

Matrik penyajian untuk rotasi terhadap titik pusat P(0,0) adalah

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}.$$

Contoh 5.3

Tentukan posisi dari segitiga ABC yang dibentuk oleh titik-titik A(20,20), B(100,20) dan C(60,120), jika dilakukan pemutaran dengan pusat sumbu koordinat dengan rotasi putarnya 180 derajat berlawanan arah dengan arah jarum jam.

Jawab :

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} 20 & 100 & 60 \\ 20 & 20 & 120 \end{bmatrix} = \begin{bmatrix} -20 & -100 & -60 \\ -20 & -20 & -120 \end{bmatrix}$$

Yaitu A'(-20, -20), B'(-100, -20) dan C'(-60,-120)

5.4 Sistem Koordinat Homogen 2D

Dari berbagai bentuk matrik penyajian, terlihat bahwa hanya transformasi translasi saja yang belum bisa dinyatakan sebagai matrik penyajian, karena diperlukan operasi perkalian dan penjumlahan, sedangkan pada jenis transformasi yang lain cukup diperlukan operasi perkalian matriks saja, sehingga perlu dicari suatu cara agar translasipun juga bisa dinyatakan dalam

operasi perkalian matriks. Hal ini dapat dilakukan dengan dengan menggunakan sistem koordinat homogen.

Sistem koordinat homogen adalah sistem koordinat yang mempunyai satu dimensi lebih tinggi dari sistem koordinat yang ditinjau. Sebagai contoh, sistem koordinat homogen dari sistem koordinat dua dimensi adalah sistem koordinat 3 dimensi dengan cara menentukan salah satu sumbunya sebagai suatu konstanta. Dengan menggunakan sistem koordinat homogen, persamaan umum transformasi titik A(x,y) menjadi A'(x',y') dapat ditulis sebagai :

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = . \begin{bmatrix} a & c & t_x \\ b & d & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Dari persamaan tersebut, maka masing-masing transformasi diatas bisa dituliskan sebagai berikut :

- Matrik penyajian Translasi : T = $\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$, sehingga

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- Matrik penyajian *Scalling* : S = $\begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$, sehingga

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- Matrik penyajian Rotasi: R = $\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$, sehingga

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Contoh 5.4

Tentukan posisi dari segitiga ABC yang dibentuk oleh titik-titik A(10,2), B(10,8) dan C(3,2) jika dilakukan transformasi berikut :

a) Translasi kearah sumbu x = 4, kearah sumbu y = −2

b) Scalling dengan skala kearah sumbu x = 2, kearah sumbu y = −2

c) Diputar 90^o berlawanan jarum jam

Jawab:

a) $\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 4 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 10 & 10 & 3 \\ 2 & 8 & 2 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 14 & 14 & 7 \\ 0 & 6 & 0 \\ 1 & 1 & 1 \end{bmatrix}$

diperoleh A′(14,0), B′(14,6) dan C′(7,0)

b) $\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 10 & 10 & 3 \\ 2 & 8 & 2 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 20 & 20 & 6 \\ -4 & -16 & -4 \\ 1 & 1 & 1 \end{bmatrix}$

diperoleh A′(20, −4), B′(20, −16) dan C′(6, −4)

c) $\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos 90^0 & -\sin 90^0 & 0 \\ \sin 90^0 & \cos 90^0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 10 & 10 & 3 \\ 2 & 8 & 2 \\ 1 & 1 & 1 \end{bmatrix} =$

$\begin{bmatrix} -2 & -8 & -2 \\ 10 & 10 & 3 \\ 1 & 1 & 1 \end{bmatrix}$

diperoleh A′(−2, 10), B′(−8, 10) dan C′(−2, 3)

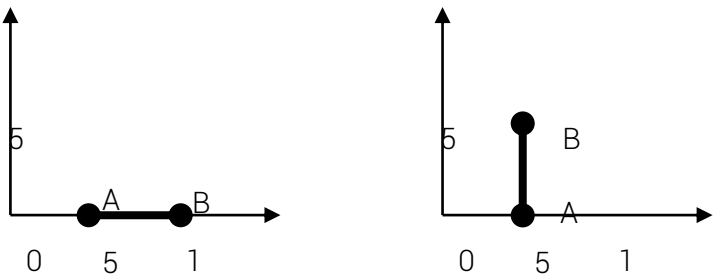
5.5 Komposisi Matrik Transformasi 2D

Dengan menggunakan matrik penyajian, kita bisa menyusun transformasi secara berurutan yang biasa disebut sebagai komposisi matrik tranformasi, yaitu dengan cara menghitung

perkalian matrik penyajian secara berurutan. Karena perkalian matrik tidak bersifat komutatif, maka urut-urutan perkalian matrik harus diperhatian (tidak boleh kebalik) karena menunjukkan urut-urutan transformasi yang dilakukan.

Contoh 5.5

Sebuah segmen garis AB dimana A(0,5) dan B(10,0) akan dirotasikan 90^o terhadap titik A. Bagaimana proses ini dilakukan? Apakah langsung dilakukan proses rotasi 90^o ?



Gambar 5. 1

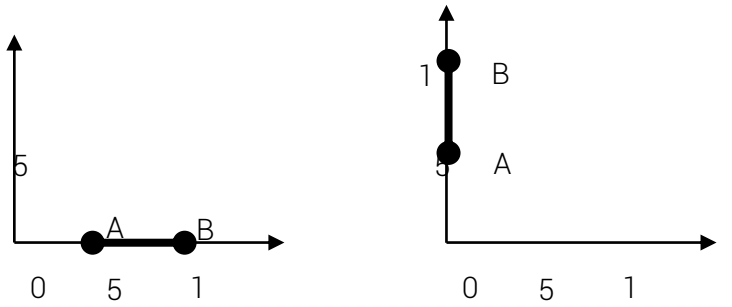
Jawab:

Jika langsung dirotasikan 90^o hasilnya sebagai berikut:

$$\begin{bmatrix} x'_x \\ y'_y \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(90) & -\sin(90) & 0 \\ \sin(90) & \cos(90) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 5 & 10 \\ 0 & 0 \\ 1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} x'_x \\ y'_y \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 5 & 10 \\ 0 & 0 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 5 & 10 \\ 1 & 1 \end{bmatrix}$$

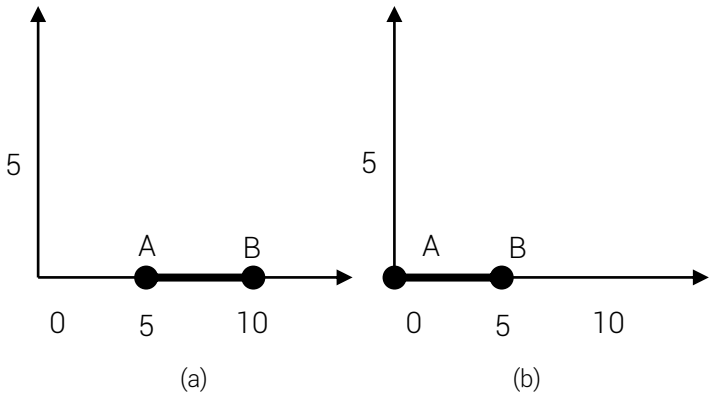
Diperoleh titik hasil transformasi : A'(0,5) dan B'(0,10)



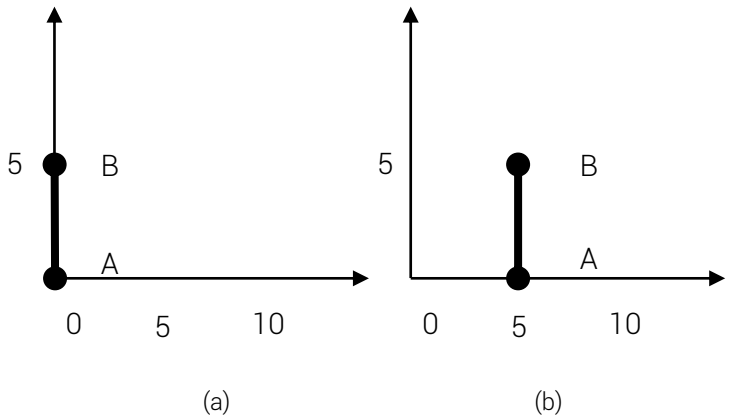
Gambar 5. 2

Tentu saja hasil ini tidak seperti yang kita harapkan (hasilnya salah).

Cara yang benar adalah menggunakan komposisi transformasi, yaitu (a) translasikan segmen garis tersebut hingga titik A (sebagai pusat rotasi) berada di titik asal (0,0). (b) Kemudian rotasikan 90° terhadap titik asal, dan terakhir (c) translasikan titik A sehingga posisinya berada di tempat semula.



Gambar 5. 3



Gambar 5. 4

Secara matematis dapat ditulis sebagai berikut : AB′ = T(5) R(90) T(−5) AB

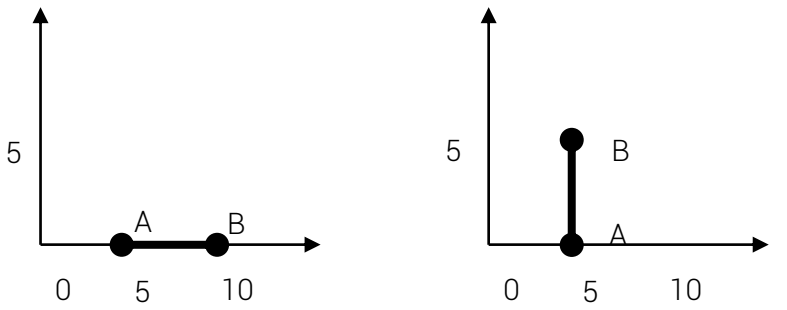
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} \cos(90) & -\sin(90) & 0 \\ \sin(90) & \cos(90) & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -5 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} \cos(90) & -\sin(90) & 0 \\ \sin(90) & \cos(90) & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -5 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 5 & 10 \\ 0 & 0 \\ 1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -5 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 5 & 10 \\ 0 & 0 \\ 1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 5 & 5 \\ 0 & 5 \\ 1 & 1 \end{bmatrix}$$

Diperoleh titik hasil transformasi : A'(5,0) dan B'(5,5)



Gambar 5. 5

Contoh 5.6

Tentukan posisi dari segitiga ABC yang dibentuk oleh titik-titik A(5,5), B(10,5) dan C(8,10) jika dilakukan komposisi transformasi berikut: penggeseran pada $\begin{bmatrix} 6 \\ 6 \end{bmatrix}$, dilanjutkan dengan rotasi 90º berlawanan jarum jam, kemudian diakhiri dengan penskalaan dengan faktor skala $\begin{bmatrix} 3 \\ 3 \end{bmatrix}$ terhadap titik pusat P(0,0).

Jawab:

Dalam hal ini kita harus menghitung matrik komposisi berikut

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = S * R * T * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Proses dilakukan terhadap operasi translasi terlebih dahulu

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} \cos(90) & -\sin(90) & 0 \\ \sin(90) & \cos(90) & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 6 \\ 0 & 1 & 6 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 5 & 10 & 8 \\ 5 & 5 & 10 \\ 0 & 0 & 1 \end{bmatrix}$$

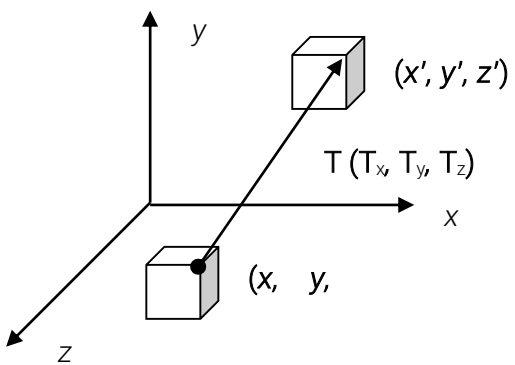
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} -33 & -33 & -48 \\ 33 & 48 & 42 \\ 1 & 1 & 1 \end{bmatrix}$$

Diperoleh titik hasil transformasi: A′(−33, 33), B′(−33, 48) dan C′(−48, 42)

5.8 Transformasi Geometri 3D

Transformasi geometri 3D merupakan pengembangan dari transformasi geometri 2D. Secara umum representasi transformasi pada 3D juga dibuat dalam bentuk matrik untuk memudahkan perhitungan.

a) Translasi



$$x' = x + T_x$$

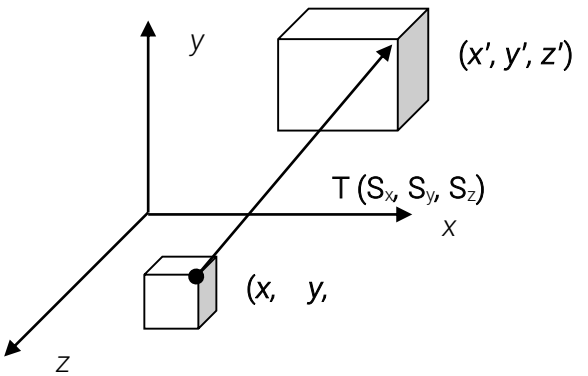
Dalam hal ini, $$y' = y + T_y$$

$$z' = z + T_z$$

Dalam bentuk matrik

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

b) Penskalaan (Scalling)



$$x' = S_x . x$$

Dalam hal ini, $$y' = S_y . y$$.

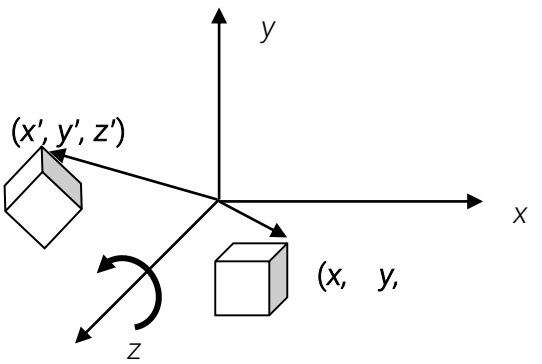
$$z' = S_z . z$$

Dalam bentuk matrik

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

c) Rotasi

Rotasi terhadap sumbu-*z* sebesar sudut Θ.



$$x' = x\cos\theta - y\sin\theta$$

Dalam hal ini, $y' = x\sin\theta + y\cos\theta$

$$z' = z$$

Dalam bentuk matrik rotasi terhadap sumbu-*z* sebesar sudut Θ.

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Dengan cara yang sama diperoleh rotasi terhadap sumbu-*x* sebesar sudut Θ.

$$x' = x$$
$$y' = y\cos\theta - z\sin\theta$$
$$z' = y\sin\theta + z\cos\theta$$

Dalam bentuk matrik rotasi terhadap sumbu-*x* sebesar sudut Θ.

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

rotasi terhadap sumbu-*y* sebesar sudut Θ.

$$x' = x\cos\theta + z\sin\theta$$
$$y' = y$$
$$z' = -x\sin\theta + z\cos\theta$$

Dalam bentuk matrik rotasi terhadap sumbu-*y* sebesar sudut Θ.

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Ringkasan Bab 5

Transformasi geometri pada dasarnya adalah proses yang mengubah kedudukan setiap titik yang membentuk obyek. Transformasi yang banyak digunakan di dalam grafika komputer adalah transformasi affin (*affine tranformation*), yaitu: penggeseran (*translation*), penskalaan (*scaling*), pemutaran (*rotation*) dan *shearing*.

Sistem koordinat homogen adalah sistem koordinat yang mempunyai satu dimensi lebih tinggi dari sistem koordinat yang ditinjau. Berikut adalah bentuk matrik Tranformasi 2D

Translasi : $\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

Penskalaan : $\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

Rotasi : $\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

Berikut adalah bentuk matrik Tranformasi 3D

Translasi: $\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$

Penskalaan: $\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$

Rotasi terhadap sumbu-*z* : $\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$

Rotasi terhadap sumbu-*x*: $\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$

Rotasi terhadap sumbu-*y*: $\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$

Soal-Soal Latihan

1. Tentukan posisi dari segitiga ABC yang dibentuk oleh titik-titik A(30,25), B(80,50) dan C(60,20) jika dilakukan penggeseran pada $\begin{bmatrix} 10 \\ 35 \end{bmatrix}$.

2. Tentukan posisi dari segitiga ABC yang dibentuk oleh titik-titik A(23,42), B(30,22) dan C(60,12), jika dilakukan penskalaan dengan faktor skala $\begin{bmatrix} 4 \\ 2 \end{bmatrix}$ terhadap titik pusat P(0,0)

3. Tentukan posisi dari segitiga ABC yang dibentuk oleh titik-titik A(20,24), B(15,20) dan C(60,12), jika dilakukan pemutaran dengan pusat sumbu koordinat dengan rotasi putarnya 30 derajat berlawanan arah dengan arah jarum jam.

4. Tentukan posisi dari segitiga ABC yang dibentuk oleh titik-titik A(10,5,6), B(15,18,2) dan C(30,25,10) jika dilakukan transformasi berikut :

a) translasi kearah sumbu $x = 2$, sumbu $y = -3$, sumbu $z = 4$

b) Scalling dengan skala kearah sumbu $x = 5$, sumbu $y = -3$, sumbu $z = 7$

c) Diputar 45^o terhadap sumbu y

Daftar Istilah

AVI: "AVI" definisi Microsoft untuk Audio Video Interleave. Standar format file video untuk Platform Ms Windows.

Binary: Salah satu jenis pada sistem Digital untuk menggambarkan pengkodean pada komputer, sehingga angka numerik hanya dapat terdiri atas nilai nol dan satu (On atau Off) saja.

CGI: Kependekan dari *Computer Graphic Imagery*.

Realtime: Istilah dalam dunia computer yang merupakan bagian dari jenis pengoperasian dimana penerimaan dan proses data begitu pula pengembalian hasil berlangsung begitu cepat sehingga berkesan bahwa input-proses-output serentak. Pada suatu Sistem-NLE pada saat penambahan efek dan overblending proses render segera dilaksanakan dan tanpa ada kesan *interrupt* (penundaan karena perhitungan render).

Rendering: Proses perhitungan matematik sebagai hasil dari efek-efek transformasi untuk frame video (Contoh: perubahan ukuran, efek-efek baik image maupun video, pergerakan).

Resolution: Jumlah Informasi pada satu Frame Video, biasanya diukur dari jumlah pixel horisontal dikalikan jumlah pixel vertikal (Contoh misalnya standar VGA dengan resolusi 640x480 pixel). Jika faktor-faktor lainnya tidak ada perobahan maka resolusi yang lebih tinggi berarti kualitas gambar lebih baik.

Streaming: Pengiriman data video melalui internet atau jaringan lain, dimana video langsung dapat dilihat meskipun proses pengiriman masih berlangsung.

24-Bit-Color: Standar warna yang dewasa ini dipakai pada Komputer. Masing-masing komponen warna merah(Red), hijau(Green) dan biru(Blue) disimpan pada informasi 8 Bit (3X8 Bit=24 Bit). Dengan komposisi warna 24 Bit dapat menghasilkan lebih dari satu juta (1048576) variasi warna g.

YCC: Suatu sinyal Video yang terdiri atas Luminance „Y"-Komponen dan dua Crominance „C"-Komponen.

Daftar Pustaka

Andreas Butz, *Computergrafik* 1, SS 2015, Ludwig-Maximilians Universitaet Muenchen

Asthana & Sinha, *Computer Graphics for Scientists and Engineers*, Second Revised Edition, New Age International Publishers, New Delhi, 1996 – 1998 (Revised) – 2003 (Reprint), ISBN : 81-224-0874-5

Edhi Nugroho, *Teori dan Praktek Grafika Komputer menggunakan DELPHI dan OpenGL*, Penerbit Graha Ilmu, Yogyakarta, Cetakan Pertama 2005, ISBN : 979-756-070-X

Francis S.Hill Jr, *Computer Graphics, Department of Electrical and Computer Engineering University of Massachusetts*, Macmillan Publishing Company, New York, 1990, ISBN 0-02-354860-6

James Foley, Van Daam, dkk., *Grundlagen der Computergraphik Einfuehrung, Konzepte, Methoden*, Addison-Wesley Publishing Company, Bonn, 1.Auflage 1994, ISBN 3-89319-647-1

Biografi Penulis

Vincent Suhartono adalah pengajar di dua Fakultas yaitu Fakultas Teknik Jurusan Elektro dan Fakultas Ilmu Komputer Jurusan Teknik Informatika. Pada saat ini menjabat sebagai Ketua Program Studi, Program Sarjana Teknik Biomedis, Universitas Dian Nuswantoro (UDINUS). Penulis pernah mengajar di Fakultas Teknik Jurusan Teknik Elektro Universitas Kristen Satya Wacana (UKSW) Salatiga. Dan sebelum terjun di dunia pendidikan, pernah sebagai praktisi dengan menjabat sebagai Electronic Data Processing (EDP) Manager pada perusahaan swasta asing (PMA) pada PT Floetotto Indonesia di Semarang.

Penulis memiliki 3 gelar pendidikan di Jerman: Ing.-(grad) dari Fachhochschule Bielefeld, Dipl.-Ing. dan Dr.-Ing. dari Universitas Bremen. Bidang Kepakaran penulis adalah: *Nachhrichtentechnik* (Teknik Komunikasi), *Künstliche Intelligenz* (Intelligence Control - Robotik) dan *Realzeit Programmierung* (Real Time Operating System).

Sebagai Coauthor, penulis sudah terlibat dalam berbagai buku seperti; buku *Teori Pengolahan Citra Digital* (Ed.I , ANDI Offset – UDINUS , Yogyakarta-Semarang, 2009 dengan ISBN:978-979-29-0974-6) dan *Kecerdasan Buatan* (Ed.I , ANDI Offset – UDINUS, Yogyakarta-Semarang, 2011 dengan ISBN:978-979-29-2761-0).