

MODUL GRAFIKA KOMPUTER

Disusun oleh:

**Dr. Setiawan Hadi
NIP. 196207011993021001**



**Program Studi Teknik Informatika
Fakultas Matematika dan Ilmu Pengetahuan Alam
Universitas Padjadjaran
November 2014**

KATA PENGANTAR

Syukur kami panjatkan kepada Tuhan yang Maha Kuasa hingga pada akhirnya Modul Ajar Grafika Komputer ini bisa diselesaikan. Penulisan modul ajar ini sudah dimulai pada bulan Juli 2010, namun penulisan modul ajar ini banyak terhenti karena berbagai halangan. Di awal tahun 2014, LP3M Unpad melalui program penulisan modul ajar memberikan kesempatan kepada dosen pengampu mata kuliah untuk menulis modul ajar dan sekaligus video ajar sebagai bagian dari kegiatan E-Learning di Universitas Padjadjaran. Melalui program inilah akhirnya modul ajar ini dapat diselesaikan.

Grafika komputer (*Computer Graphics*) adalah mata kuliah keilmuan Informatika/Teknik Informatika/Ilmu Komputer yang tercantum dalam buku *Computing Curricula Computer Science* yang diterbitkan oleh himpunan profesi ilmuwan komputer/informatika IEEE Computer Society dan ACM edisi terbaru tahun 2013. Dengan demikian Grafika Komputer memiliki pijakan tidak hanya dari sisi keilmuan tetapi juga dari sisi pengembang kurikulum yang bertaraf internasional.

Penulisan modul ajar ini tidak lepas dari pengalaman penulis mengampu mata kuliah Grafika Komputer yang diadakan pada program S-1 Matematika bidang minat Ilmu Komputer, Program D-3 Ilmu Komputer dan yang terbaru pada Program Studi S-1 Teknik Informatika Universitas Padjadjaran. Selain itu bidang penelitian penulis juga yang memiliki relevansi dan korelasi yang cukup erat dengan Grafika Komputer. Perkembangan teknologi serta melimpahnya publikasi yang dapat dijumpai di dunia maya membuat perkembangan keilmuan Grafika Komputer juga berjalan sangatlah cepat. Untuk itu penulisan modul ajar ini, walaupun sebagian besar memaparkan keilmuan klasik grafika komputer, namun dalam beberapa bagian ditambahkan topik-topik terbaru yang relevan dan implementatif.

Sebagai mata kuliah yang diajarkan pada mahasiswa minimal di tahun ke-3 perguruan tinggi, maka Grafika Komputer membutuhkan pengetahuan pendahuluan (*prerequisite*) yang cukup sehingga materi-materi yang disajikan dapat difahami oleh mahasiswa. Selain itu, diperlukan pengetahuan dan kemampuan pemrograman yang kuat, agar konsep, teori, metode maupun algoritma yang dipelajari dapat diterapkan menggunakan bahasa komputer modern sehingga daya output dari perkuliahan ini dapat lebih terlihat nyata.

Modul ini masih belum sempurna dan belum tertata dengan rapi. Bahan-bahan yang melengkapi modul ajar ini masih harus disempurnakan dan ditambahkan. Namun paling tidak, bagi mahasiswa, adanya modul ini dapat menjadi bahan acuan dalam belajar. Bagi dosen, modul ini juga menjadi alat untuk selalu menyegarkan pengetahuannya sekaligus menunjukkan *expertise* nya dalam dunia atau komunitas ilmu yang relevan. Bagi institusi, tentunya adanya modul ini menjadi aset dan akan memperkuat posisi sebagai institusi pendidikan yang senantiasa menyediakan sarana yang menunjang kesuksesan proses pembelajaran di perguruan tinggi.

Terima kasih kami ucapkan kepada LP3M Unpad, FMIPA Unpad, Departemen Matematika, Program Studi Teknik Informatika, serta semua pihak yang sudah mendukung terciptanya modul ajar ini. Khususnya kepada keluarga, Lydia, Grace dan Karina, yang dengan sabar mendukung penuh penulisan modul ajar ini, kami ucapkan terima kasih. Semoga modul ajar ini bermanfaat bagi banyak pihak dan bisa terus disempurnakan.

SH. September 2014

DAFTAR ISI

KATA PENGANTAR	ii
DAFTAR ISI	iii
DESKRIPSI MATA KULIAH	vi

MODUL I TINJAUAN UMUM GRAFIKA KOMPUTER

I. 1	Apakah yang Dimaksud dengan Grafika Komputer	1
I. 2	Ruang Lingkup Grafika Komputer.....	2
I. 3	Sejarah Grafika Komputer.....	2
I. 4	Perkembangan Keilmuan dan Pustaka Grafika Komputer.....	4
I. 5	Aplikasi-aplikasi Grafika Komputer	6
I. 6	Perkakas Grafika Komputer	7
I. 7	Model Dasar Grafika Komputer.....	15
	Daftar Pustaka	19

MODUL II PENGAMBARAN OBJEK PRIMITIF

II.1	Pengertian objek primitif.....	20
II.2	Penggambaran Titik dan Garis	21
II.3	Algoritma Penggambaran Lingkaran	29
II.4	Implementasi Penggambaran Titik.....	33
	Daftar Pustaka	36

MODUL III ATRIBUT OUTPUT PRIMITIF

III. 1	Pengertian atribut output primitif.....	37
III. 2	Atribut Garis.....	37
III. 3	Fill Area Primitif	39
III. 4	Karakter dan Pembentukan Karakter	41
III. 5	Antialiasing	45
	Daftar Pustaka	49

MODUL IV WINDOWING DAN CLIPPING

IV. 1	Model Konseptual Grafika Komputer.....	50
IV. 2	Transformasi Windows-Viewport.....	51

IV. 3 Clipping.....	53
Daftar Pustaka	63

MODUL V TRANSFORMASI 2 DIMENSI

V. 1 Pengertian transformasi.....	64
V. 2 Translasi	65
V. 3 Penskalaan.....	66
V. 4 Rotasi.....	67
V. 5 Refleksi.....	70
V. 6 Shear.....	72
V. 7 Transformasi Homogen.....	74
Daftar Pustaka	79

MODUL VI TRANSFORMASI 3 DIMENSI

VI.1 Pengertian Transformasi 3D.....	80
VI.2 Operasi Dasar Transformasi 3D.....	81
VI.3 Sistem Koordinat Berganda	92
Daftar Pustaka	94

MODUL VII PROYEKSI GEOMETRI BIDANG

VII.1 Pengertian Proyeksi Geometri Bidang.....	95
VII.2 Taksonomi Proyeksi Geometri Bidang	96
VII.3 Proyeksi Paralel.....	97
VII.4 Proyeksi Perspektif.....	103
VII.5 Titik Hilang (Vanishing Points)	107
Daftar Pustaka	108

MODUL VIII KOMPONEN PENDUKUNG PEMROGRAMAN GRAFIS

VIII.1 Pemahaman GDI+	109
VIII.2 Eksplorasi Fungsionalitas GDI+	110
VIII.3 GDI+ Namespaces dan Classes dalam .NET	111
VIII.4 Mempersiapkan Penggunaan Grafis pada Visual Studio	118
VIII.5 Operasi Matriks Menggunakan C#	121
Daftar Pustaka	123

MODUL IX DASAR-DASAR PEMROGRAMAN GRAFIS

IX.1	Area Gambar	124
IX.2	Sistem Koordinat.....	125
IX.3	Membuat Program Pertama Aplikasi Grafika Komputer.....	127
IX.4	Menggunakan Struktur Point dan PointF.....	129
IX.5	Menggambar Objek-objek Grafis Lainnya.....	131
	Daftar Pustaka	151

MODUL X TOPIK-TOPIK GRAFIKA KOMPUTER LANJUT

X.1	Topik Grafika Komputer Lanjut.....	152
X.2	Pencahayaannya dan Warna.....	153
X.3	Tinjauan Fraktal Secara Umum.....	163
X.4	Ray Tracing	165
X.5	Pemrograman Grafika Komputer dengan OpenGL.....	173
	Daftar Pustaka	186

LAMPIRAN

	Soal Tugas.....	187
	Soal Quiz	188
	Soal Ujian Tengah Semester.....	189
	Soal Ujian Akhir Semester.....	190

DESKRIPSI MATA KULIAH

Nama Mata Kuliah : Grafika Komputer

SKS : 3

Silabus : Mata kuliah ini menjelaskan konsep dasar sistem representasi matematis objek-objek grafis pada peralatan komputasi yang meliputi: konsep dasar piksel, penggambaran objek-objek primitif, koordinat dua dimensi, transformasi objek dua dimensi dan transformasi homogen, koordinat tiga dimensi dan transformasinya, *windowing* dan *clipping*, dan proyeksi geometri bidang. Untuk meningkatkan pemahaman dilakukan implementasi konsep menggunakan bahasa pemrograman dan *library* tertentu.

Tujuan Pembelajaran Umum

: Setelah mengikuti perkuliahan mahasiswa akan mengerti dan mengetahui peralatan-peralatan dasar komputasi yang berkaitan dengan grafis, konsep manipulasi grafis pada komputer yang mencakup penggambaran objek-objek primitif berupa garis menggunakan algoritma dasar, DDA, dan Bresenham, algoritma penggambaran lingkaran menggunakan algoritma dasar, polar dan Bresenham, teknik pemotongan (*clipping*) dan tampilan pada jendela (*windowing*) menggunakan algoritma tertentu, sistem koordinat dua dimensi dan transformasi objek dua dimensi, teknik untuk transformasi sembarang (homogen), sistem koordinat dan transformasi objek tiga dimensi, teknik proyeksi dan perspektif dan titik hilang serta perhitungan-perhitungannya yang diarahkan kepada implementasinya pada peralatan komputasi. Mahasiswa juga akan menguasai paket program penggambaran geometris tertentu dan mengetahui jurnal(-jurnal) ilmiah dan program aplikasi yang berkaitan dengan komputasi grafis berbantuan komputer.

Buku Acuan Utama : 1. **Computer Graphics: Principles and Practice (3rd edition)**, John F. Hughes, Andries Van Dam, Morgan Mcguire, David F. Sklar, James D. Foley, Steven K. Feiner, Kurt Akeley, Addison-Wesley, 2014
2. **Computer Graphics with Open GL (4th Edition)**, Donald D. Hearn, M. Pauline, Warren Carithers Prentice-Hall, 2011
3. **Mathematical Elements for Computer Graphics (2nd edition)**, David F. Rogers, Alan J. Adams, McGraw-Hill, 1989

MODUL I TINJAUAN UMUM GRAFIKA KOMPUTER

Setelah membaca modul ini, mahasiswa akan memiliki pengetahuan dan mampu menjelaskan (i) apa yang dimaksud dengan grafika komputer (ii) ruang lingkup grafika komputer disertai dengan ilmu-ilmu yang berkait erat dengan grafika komputer (iii) sejarah grafika komputer secara umum (iv) aplikasi-aplikasi yang memanfaatkan fasilitas grafis pada komputer (v) perkakas atau tools berupa *hardware* maupun *software* yang menunjang (vi) model dasar grafika komputer serta elemen-elemen yang terkait di dalamnya

I. 1 Apakah yang Dimaksud dengan Grafika Komputer

Grafika komputer adalah teknik-teknik dalam ilmu komputer dan matematika untuk merepresentasikan dan memanipulasi data gambar menggunakan komputer. Dengan bahasa lain, istilah grafika komputer juga dapat diartikan segala sesuatu selain teks atau suara. Seiring dengan perkembangan teknologi dewasa ini, gambar-gambar yang dihasilkan dan ditampilkan pada komputer menjadi bagian kehidupan sehari-hari yang dapat ditemui misalnya pada televisi, koran dan majalah yang fungsinya untuk menampilkan hasil yang lebih komunikatif dan realistis. Selain itu juga grafika komputer ditemukan pada bidang-bidang kedokteran, geologi dan tak terkecuali dalam bidang pendidikan untuk pengajaran dan penulisan karya-karya ilmiah.

Salah satu aplikasi yang nyata dari grafika komputer adalah untuk visualisasi data dalam bentuk grafis 2D atau 3D dilengkapi dengan animasi. Walaupun bentuk grafis 3D lebih realistis, namun bentuk 2D masih banyak dipergunakan. Grafika komputer muncul sebagai bagian ilmu komputer yang mempelajari metode-metode sintesa dan manipulasi konten visual secara digital. Visualisasi informasi dan sains telah menjadi fokus penelitian terutama yang berkaitan dengan fenomena-fenomena 3D dalam bidang arsitektur, meteorologi, kedokteran, biologi dan sebagainya. Penekanan diberikan dalam rangka menjawab pertanyaan bagaimana menghasilkan gambar benda yang realistis sesuai dengan kondisi dan situasi yang terjadi.

I. 2 Ruang Lingkup Grafika Komputer

Grafika komputer bukan ilmu yang berdiri sendiri. Pada dasarnya banyak ilmu yang menyokong sekaligus menjadi dasar grafika komputer, misalnya ilmu matematika, geometri, analisis/metode numerik dan ilmu komputasi. Jika dikaitkan dengan konsep sistem, ada dua ilmu yang erat kaitannya dengan grafika komputer yaitu pengolahan citra dan visi komputer. Relasinya dapat digambarkan sebagai berikut:

Tabel 1.1 Kaitan Ilmu Grafika Komputer dengan Ilmu Lain

Input/output	Citra	Deskripsi
Gambar	Pengolahan Citra	Visi Komputer
Deskripsi	Grafika Komputer	AI

Dari Tabel 1.1 dapat dijelaskan bahwa objek yang menjadi masukan atau input untuk grafika komputer adalah deskripsi misalkan buatlah kursi, buatlah meja dan sebagainya, dan objek yang menjadi keluaran atau output dari grafika komputer adalah gambar atau citra digital sesuai dengan deskripsi masukan. Hal yang sama dapat disimpulkan untuk ketiga bidang ilmu terkait lainnya.

I. 3 Sejarah Grafika Komputer

Seperti bidang ilmu lain, sejarah grafika komputer sangat bervariasi tergantung darimana kita melihat sudut pandangnya. Namun beberapa nama menjadi pionir dalam pengembangan grafika komputer yaitu:



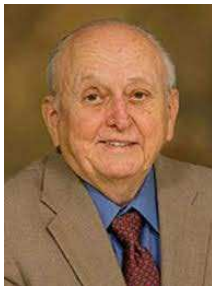
William Fetter. Mempopulerkan istilah computer graphics pada tahun 1960. Istilah ini digunakan untuk menjelaskan metode perancangan pesawat baru yang dikembangkan di tempat ia bekerja yaitu Boeing. Citra, yang direproduksi menggunakan plotter, menggambarkan rancangan kokpit menggunakan model 3D tubuh manusia.



Ivan Sutherland. Seorang mahasiswa MIT pada tahun 1961 menciptakan program komputer yang disebut Sketchpad. Dengan bantuan light pen seseorang dapat menggambar bangun sederhana pada layar komputer.



Steve Russell. Di tahun yang sama menciptakan video game pertama yang disebut Spacewar. Program ini dijalankan pada mesin DEC-PDP-1, dan sekaligus menjadi program uji bagi setiap komputer DEC yang dipasarkan.



E. E. Zajac. Seorang ilmuwan dari Bell Telephone Laboratory, tahun 1963, menciptakan sebuah film yang mensimulasikan gerakan-gerakan satelit pada saat mengorbit bumi. Animasinya dilakukan menggunakan komputer mainframe IBM 7090. Pada waktu yang sama beberapa ilmuwan lain menciptakan film untuk mensimulasikan hukum Newton, Gerakan fluida/cairan dan getaran.

Masih banyak nama-nama lain yang berkiprah dalam pengembangan grafika komputer, baik sebagai pionir, sebagai inventor, sebagai adapter maupun sebagai follower. Informasi ini dapat secara mudah diakses melalui jaringan internet.

Gambar 1.1 menunjukkan peralatan-peralatan berupa perangkat keras komputer dimasa lalu yang memanfaatkan grafika komputer yang pada konteks aplikasi, sangat sederhana dibandingkan dengan aplikasi pada masa sekarang.



Whirlwind Computer (MIT, 1950): input via keypunch, output via printer/plotter, both in batch

Cool facts: Whirlwind, built in the early 50's at MIT, cost \$4.5 million and could perform 40,000 additions/second. Mac 512K, list price \$3,195 in 1984, could do 500,000. Today, commodity PCs perform approximately one billion operations/second



Sketchpad (Sutherland, MIT, 1963): first truly interactive graphics system, using a CRT monitor, light pen, and function-key pad



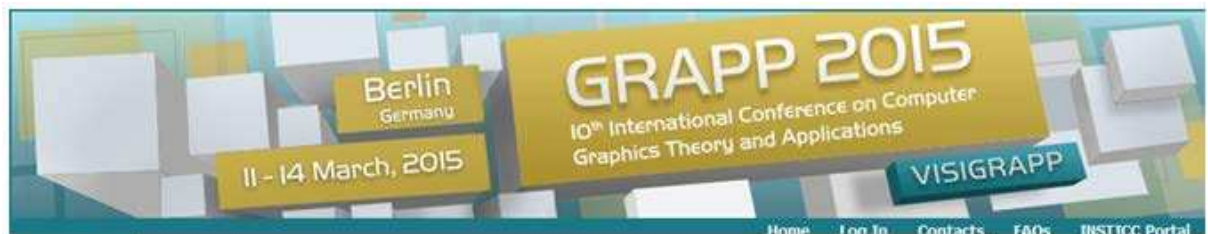
first **CAD** system (IBM, 1959)

Gambar 1. 1 Peralatan-peralatan Grafika Komputer Masa Lalu

I. 4 Perkembangan Keilmuan dan Pustaka Grafika Komputer

Sampai saat ini topik grafika komputer tetap menjadi topik penelitian yang menarik para peneliti di berbagai bidang. Banyak konferensi dan seminar yang berkaitan dengan grafika komputer dilakukan terutama di dunia internasional. Begitu juga dengan jurnal-jurnal yang relevan berkaitan dengan gaphics dan visualization, termasuk juga animasi. Buku-buku grafika komputer masih terus diperbaharui dan dibuat oleh para penulis di dunia keilmuan internasional.

Gambar 1.2 dan Gambar 1.3 menunjukkan konferensi/seminar internasional yang terbaru diadakan pada tahun 2014 dan 2015 dan juga beberapa penerbitan buku yang berkaitan dengan grafika komputer.



Gambar 1.2 Konferensi dan Seminar Internasional tentang Grafika Komputer



Gambar 1.3 Buku-buku Grafika Komputer Terbaru (2014)

I. 5 Aplikasi-aplikasi Grafika Komputer

Setidaknya ada tujuh area dimana aplikasi grafika komputer dapat dimanfaatkan, yaitu

User Interface. Penggunaan grafika komputer sebagai antar muka komputer pada sistem operasi dan aplikasi modern dewasa ini, misalnya Windows dan Visual Studio.

Membuat Presentasi. Digunakan untuk membuat diagram-diagram. Office Automation
Penggunaan grafis pada aplikasi otomatisasi perkantoran seperti Office sudah menjadi trend aplikasi modern dengan konsep point and click.

Percetakan Dijital. Penggunaan untuk percetakan, pembuatan brosur, billboard dijital, buku dan sebagainya

CAD/CAM (Computer-Aided Design/Computer-Aided Manufacturing). Aplikasi pada bidang teknik, misalnya untuk pembuatan rancang bangun rumah, kendaraan dan alat-alat suku cadang.

Seni dan Komersil. Aplikasi grafika komputer pada bidang seni dan komersil, misalnya lukisan dijital, promosi barang-barang yang dijual secara online yang dilengkapi dengan gambar

Pengontrolan Proses. Aplikasi visualisasi data dengan menghubungkan alat dengan komputer melalui saluran USB atau RS-232. Digunakan untuk memonitor lokasi, webcam dan pengawasan dan sebagainya.

Gambar 1.4 memperlihatkan film-film yang memanfaatkan grafika komputer dalam pembuatannya. Gambar 1.5 memberikan ilustrasi perangkat lunak yang memiliki fasilitas GUI atau *Graphical User Interface* dalam pengoperasiannya.



Gambar 1.4 Film-film yang Memanfaatkan Grafika Komputer



Gambar 1.5 Perangkat Lunak yang Memanfaatkan Fasilitas Grafik

I. 6 Perkakas Grafika Komputer

Sebagai bidang ilmu yang tidak hanya teoritis namun juga memiliki tingkat implementasi yang tinggi, grafika komputer membutuhkan perangkat keras, perangkat lunak serta sumber daya manusia (perangkat pikir) yang khusus.

Kebutuhan minimal untuk perangkat keras antara lain adalah Komputer, Mouse, DisplayMonitor, Printer, Plotter, Digitizer, Webcam, dan Scanner.

Perangkat lunak yang diperlukan (tidak semua) dalam mengembangkan aplikasi grafika komputer antara lain adalah:

- Perangkat lunak sistem operasi yang menunjang, misalnya Windows XP, Vista, Windows 7 atau 8
- Perangkat lunak aplikasi penggambaran dan editing gambar, misalnya Coreldraw, Photoshop dan sebagainya
- Perangkat lunak pembelajaran grafika dan animasi, misalnya Cabri, 3D Studio Max
- Perangkat lunak visualisasi, misalnya Matlab dan Maple
- Perangkat lunak untuk virtual reality, augmented reality dan sebagainya
- Perangkat Lunak pemrograman seperti Visual Studio.

Beberapa istilah yang penting diketahui dalam grafika komputer dan menjadi dasar pengetahuannya adalah antara lain adalah *vector graphics* dan *raster graphics*, Pixel, Bitmap, Resolution, CGA, EGA, VGA, SVGA, XGA, CRT, LCD, Plasma, LED, Dot Pitch, Interlace/Non-Interlace, Modeling, Rendering, Animation, Wireframe, JPG, GIF, PCX, BMP.

I.6.1 Teknologi Output

Penggunaan alat utama untuk menampilkan output pada sistem grafika adalah video monitor. Operasi pada sebagian besar video monitor berdasarkan perancangan Cathode Ray Tube (CRT). Cara kerja dari operasi CRT adalah sebagai berikut :

- Sebuah electron gun memancarkan elektron, melalui focusing system (sistem untuk menentukan fokus), dan deflection system (sistem untuk mengatur pembelokan) sehingga pancaran elektron mencapai posisi tertentu dari lapisan fosfor pada layar.
- Kemudian, fosfor memancarkan sinar kecil pada setiap posisi yang berhubungan dengan pancaran elektron. Sinar yang dipancarkan dari fosfor cepat hilang, maka diperlukan pengaturan supaya fosfor tetap menyala. Hal ini dilakukan dengan cara refreshing, yaitu menembakkan elektron berulang kali pada posisi yang sama.
- Focusing system pada CRT diperlukan untuk mengarahkan pancaran elektron pada suatu titik tertentu dari lapisan fosfor. Pengaturan fokus dapat dilakukan pada electric dan magnetic field. Dengan electronic focusing, pancaran elektron melewati metal electrostatic yang berfungsi sebagai lensa untuk mengatur fokus dari pancaran elektron ke tengah monitor.
- Resolusi adalah jumlah titik per centimeter yang dapat ditempatkan menurut arah horizontal dan vertikal. Resolusi tergantung pada tipe fosfor, intensitas yang ditampilkan, serta focusing dan deflection system.

Raster-scan Display

Pada jenis ini pancaran elektron bergerak ke seluruh layar baris per baris dari atas ke bawah. Pada saat pancaran elektron bergerak pada tiap baris, intensitas pancaran timbul dan hilang untuk mendapatkan sinar spot. Definisi gambar disimpan dalam memori yang disebut refresh buffer atau frame buffer.

Refreshing pada raster-scan display mempunyai nilai 60 sampai 80 frame per detik. Kembalinya scan pada bagian kiri layar setelah refreshing tiap scan line disebut horizontal retrace. Sedangkan pada akhir dari tiap frame ($1/80$ sampai $1/60$ tiap detik) pancaran elektron yang kembali ke atas disebut vertical retrace.

Random-scan Display

Pada saat mengoperasikan unit random-scan display, pancaran elektron diarahkan hanya ke bagian layar di mana gambar dibuat. Random-scan monitor yang hanya membuat gambar dengan satu garis pada suatu saat disebut vector display, stroke writing, atau calligraphic display.

Refresh rate pada random-scan display tergantung dari jumlah garis yang ditampilkan. Definisi gambar disimpan sebagai satu blok perintah line drawing disebut refresh display file. Untuk menampilkan gambar tertentu, setelah semua perintah gambar diproses, siklus sistem kembali pada perintah baris pertama. Sistem random-scan dirancang untuk membuat gambar seluruh komponen garis dengan rate antara 30 sampai 60 tiap detik. Sistem dengan kualitas tinggi dapat menangani sampai 100.000 garis pendek setiap refreshing.

Monitor Color CRT

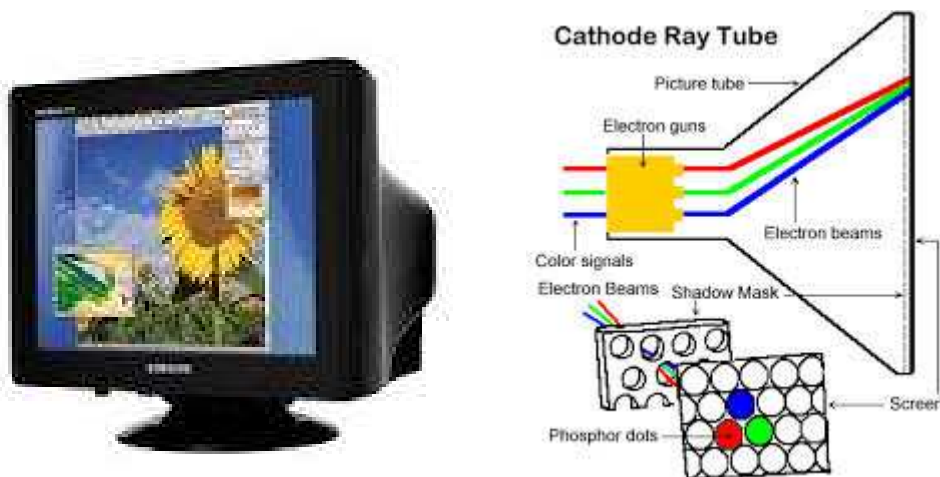
Color CRT menampilkan gambar dengan kombinasi fosfor yang memancarkan sinar warna yang berbeda. Dengan menggabungkan sinar dari fosfor yang berbeda, tingkat dari warna dapat ditampilkan. Terdapat dua teknik dasar untuk mendapatkan warna, yaitu beam penetration dan shadow mask.

Beam penetration digunakan untuk menampilkan gambar berwarna dengan random-scan monitor. Dua lapisan fosfor, biasanya merah dan hijau, dilapiskan pada bagian dalam dan warna yang dihasilkan tergantung dari seberapa besar pancaran electron menembus lapisan fosfor. Pancaran yang lemah hanya mencapai bagian luar lapisan merah. Pancaran yang lebih kuat dapat menembus lapisan merah dan mencapai bagian dalam dari lapisan hijau. Pada kecepatan menengah, kombinasi antara sinar merah dan hijau menghasilkan warna tambahan misal orange atau kuning.

Metode *shadow mask* biasanya digunakan pada *raster-scan system* termasuk TV. Metode ini menghasilkan tingkat warna yang lebih banyak dibandingkan dengan metode *beam penetration*. Shadow-mask CRT mempunyai 3 macam fosfor warna pada titik pixel yaitu merah, hijau, dan biru. CRT mempunyai tiga electron gun untuk setiap titik warna, sedangkan *shadow mask* terletak di belakang lapisan fosfor pada layar.

Pada saat ketiga pancaran elektron melewati suatu lubang pada *shadow mask*, *dot triangle* menjadi aktif. *Dot triangle* berupa titik warna yang kecil pada layar. Titik fosfor pada *triangle* diatur sehingga tiap elektron dapat mengamengaktifkan titik warna yang terhubung ketika melewati *shadow mask*.

Color CRT dalam sistem grafika dirancang sebagai RGB monitor. Monitor ini menggunakan metode *shadow mask* dan mengambil tingkat intensitas untuk setiap *electron gun* (*red, green, blue*) langsung dari sistem komputer tanpa pemrosesan antara.



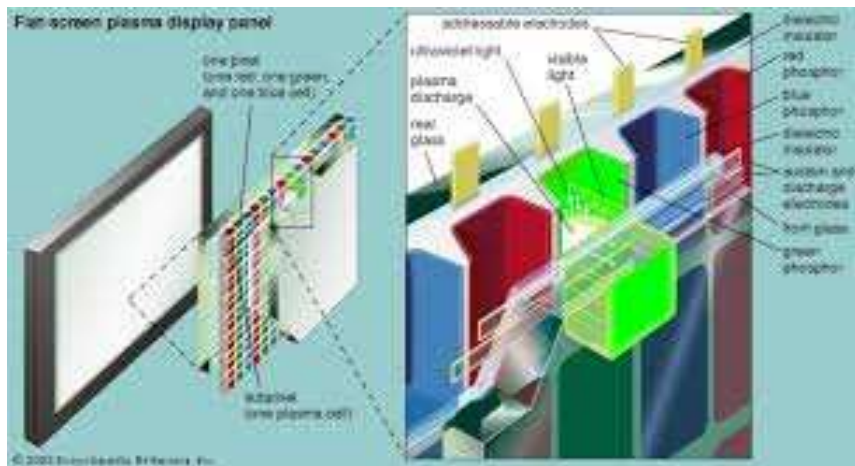
Gambar 1.66 Perangkat Displat CRT dan Anatominya

Flat Panel Display

Flat panel display mempunyai ukuran lebih tipis dari pada CRT. Penggunaan *flat panel display* diantaranya pada TV dengan ukuran kecil, kalkulator, komputer laptop, dan lain-lain. *Flat panel display* dapat dibagi menjadi dua kategori, yaitu *emissive display (emitters)* dan *nonemissive display*. *Emissive display* mengkonversi energi listrik menjadi sinar, contohnya yaitu plasma panel, *light emitting diode*. Nonemissive display menggunakan efek optik untuk mengkonversi sinar matahari atau sinar dari sumber lain ke dalam pola grafik, contohnya adalah *Liquid Crystal Display (LCD)*.

a. Plasma Panel

Plasma panel dibuat dengan mengisi ruangan antara pelat kaca dengan gas, biasanya gas neon. Satu set konduktor ditempatkan vertikal pada pelat pertama dan yang lainnya ditempatkan horizontal pada pelat kedua. Tegangan antara kedua pelat tersebut disebabkan oleh gas neon diantaranya. Definisi gambar disimpan dalam refresh buffer, dan tegangan menyebabkan refreshing pixel pada posisinya sebanyak 60 kali tiap detik.



Gambar 1.7 Perangkat Display Plasma dan Anatominya

b. *Liquid Crystal Display (LCD)*

LCD biasanya digunakan untuk suatu sistem yang kecil, seperti komputer laptop dan kalkulator. Nonemitters ini menghasilkan gambar dengan meneruskan sinar dari sekitarnya atau dari sinar di dalam yang menembus material *liquid-crystal*. *Liquid-crystal* terdiri dari susunan molekul yang dapat bergerak seperti cairan. Definisi gambar disimpan dalam *refresh buffer*, dan *refreshing* dilakukan dengan rate 60 *frame* per detik.



Gambar 1.8 Perangkat Display LCD

1.6.2 Peralatan Input Interaktif

Pada saat ini terdapat berbagai macam peralatan yang bisa dipergunakan untuk menginputkan data pada sistem grafis. Sebagian besar sistem menggunakan keyboard dan beberapa peralatan tambahan untuk input interaktif, misalnya: *mouse*, *trackball*, *spaceball*, *joystick*, *digitizer*, *dial*, dan *dial box*. Terdapat juga beberapa peralatan input khusus lain, seperti *data gloves*, *touch panel*, *image scanner* dan sistem suara.



Gambar 1.9 Perangkat Input Interaktif

1.6.3 Peralatan *Hardcopy*

Peralatan *hardcopy* yang umum dipergunakan adalah *printer* dan *plotter*. *Printer* menghasilkan output dengan dua metode, yaitu metode *impact* dan *non impact*. Metode *impact* menghasilkan output dengan menekan cetakan karakter pada pita karbon atau *ink ribbon* sehingga akan mengenai kertas dan output akan tercetak pada kertas. Pada metode *non*

impact, dipergunakan teknologi laser, *ink-jet spray*, proses *xerographic*, metode *electrostatic* dan metode *electrothermal* untuk menghasilkan gambar pada kertas.



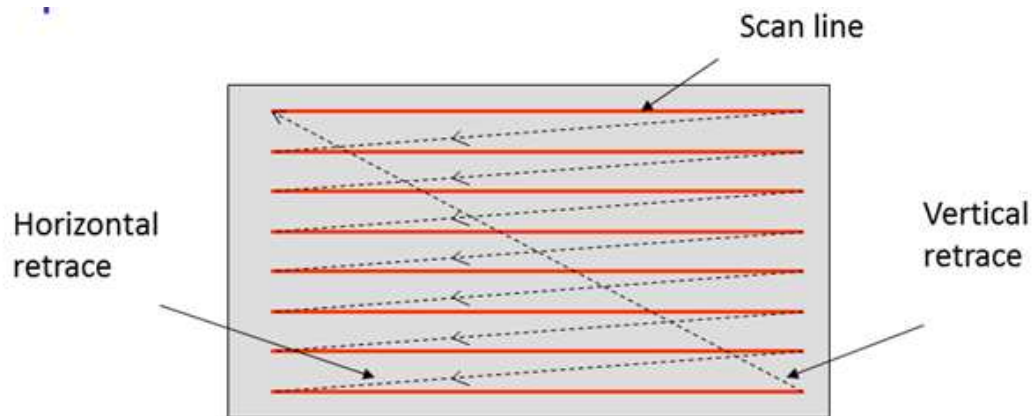
Gambar 1.10 Perangkat Output

1.6.4 Retrace

Dewasa ini teknologi output yang berkaitan dengan grafika komputer sudah berkembang dengan sangat pesat. Salah satu teknologi yang berkembang adalah teknologi output LCD dan LED. Namun teknologi perangkat keras output yang berkembang tidak merubah konsep penampilan objek di dalam perangkat tersebut.

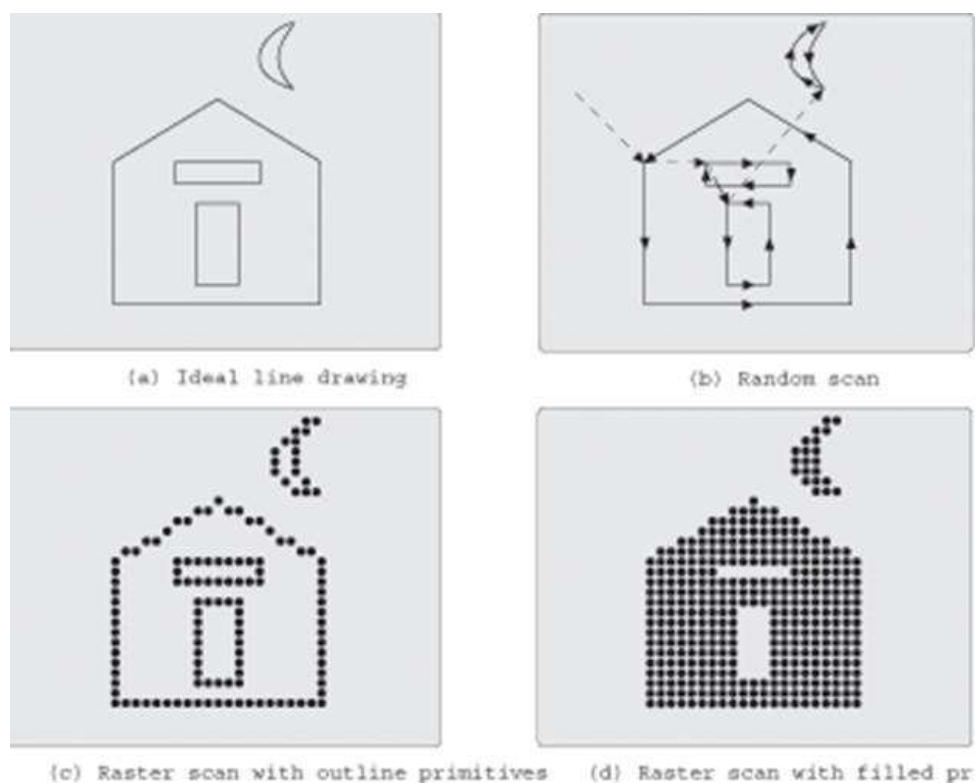
Terdapat dua jenis cara penampilan objek di dalam perangkat output yaitu teknologi raster dan teknologi vektor. Teknologi raster adalah teknologi yang berkembang dewasa ini dimana layar komputer dibayangkan sebagai kumpulan titik-titik yang disebut piksel. Titik-titik tersebut diaktifkan berdasarkan alamat atau address dari piksel tersebut pada layar mulai dari kiri atas ke kanan bawah. Aktivitas tersebut disebut retrace, dimana gerakan dari kiri ke

kanan sampai ke kanan bawah disebut *horizontal retrace*, dan gerakan kembali dari kanan bawah ke kiri atas disebut *vertical retrace*. Posisi horizontal yang dilalui disebut dengan scan line. Gambar di bawah ini memberikan ilustrasi teknologi *raster*.



Gambar 1.11 Monitor *Retrace*

Teknologi output vektor, tidak bekerja seperti teknologi raster atau bitmap. Kegiatan yang dilakukan seperti halnya manusia yaitu mirip dengan menggambar. Gambar berikut mengilustrasikan perbedaan teknologi raster dan teknologi vektor.



Gambar 1.11 Ilustrasi Penggambaran Raster dan Vektor

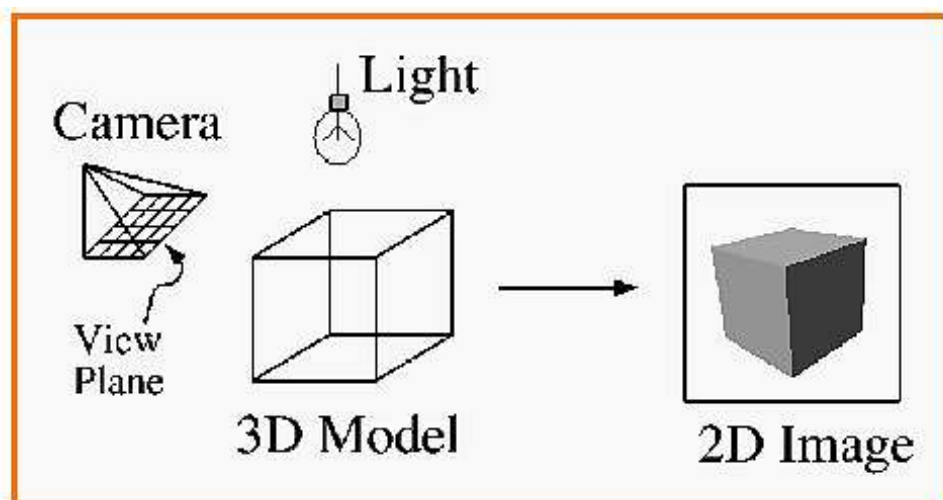
Gambar pada elemen kiri atas adalah gambar ideal yang ingin dibuat. Gambar di sebelah kanan atas adalah cara menggambar dengan memanfaatkan teknologi vektor, sedangkan gambar di kiri bawah menunjukkan cara menggambar menggunakan teknologi raster. Terlihat bahwa cara penggambaran secara vektor lebih halus dan lebih presisi dibandingkan dengan cara penggambaran *raster*.

Walupun lebih akurat namun penggambaran secara vektor kurang efisien terutama untuk gambar-gambar yang membutuhkan pengisian (*filling*) pada rongganya, sebagaimana yang ditunjukkan pada gambar kanan bawah. Selain itu, penggambaran secara vektor membutuhkan peralatan dan teknologi yang lebih tinggi sehingga harganya terlalu mahal. Sebaliknya teknologi *raster* semakin berkembang karena pembuatannya lebih murah dan dari hari ke hari semakin akurat karena ukuran piksel semakin kecil. Akibatnya teknologi *raster* menjadi lebih disukai dibandingkan dengan teknologi vektor.

I. 7 Model Dasar Grafika Komputer

Grafika komputer adalah ilmu yang berhubungan dengan pembuatan (produksi) gambar (citra) menggunakan komputer melalui tahapan (*tasks*):

1. Pemodelan
2. Rendering
3. Animasi



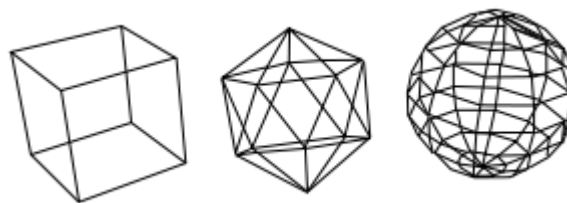
Gambar 1.12 Model Dasar Grafika Komputer

Gambar 1.12 menunjukkan model dasar dari sebuah sistem grafika komputer. Pemandangan di sekitar kita memiliki dimensi tiga dimana salah satunya adalah dimensi ruang. Namun dalam komputer, pada kenyatannya tidak dijumpai dimensi ruang tersebut. Efek tiga dimensi yang sering kita lihat pada layar komputer adalah efek visualisasi dimana efek ruang disimulasikan berdasarkan kaidah-kaidah geometri dalam bidang matematika. Secara sederhana grafika komputer juga dapat didefinisikan sebagai berikut:

Computer graphics is generating 2D images of a 3D world represented in a computer.

Modelling

Modelling atau pemodelan adalah upaya untuk menggambarkan objek nyata ke dalam objek yang memiliki karakteristik geometris. Pemodelan objek 3D dalam bentuk geometris ini dimaksudkan agar gambar dapat dimanipulasi tanpa kehilangan akurasi karena perhitungan dilakukan secara numeris berdasarkan kaidah matematis. Gambar-gambar geometris tersebut disebut *wireframe*. Gambar Gambar 1.13 menunjukkan contoh model *wireframe*.



Gambar 1.13 Wireframe Model

Secara umum pemodelan geometris dapat diartikan sebagai:

1. Memotret objek nyata dan lalu mengubahnya menjadi menjadi objek maya (virtual)
2. Menjelaskan dunia nyata atau objek nyata menggunakan matematika
3. Jika objek tidak ada, penggambaran dilakukan berdasarkan imajinasi artis

Rendering

Rendering adalah pemberian nuansa realistis kepada model-model geometris sehingga memiliki sifat/keadaan yang menyerupai sebenarnya. Gambar 1.14 menunjukkan contoh *graphics rendering*.

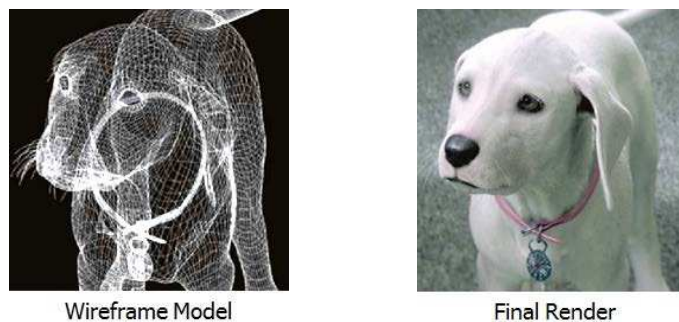


Gambar 1.14 Contoh *Graphics Rendering*

Langkah-langkah yang dilakukan pada proses rendering antara lain adalah:

- Penggambaran objek 3D dalam 2D
- Pemberian warna
- Pengaturan cahaya
- Pemberian gradasi warna
- Penambahan tekstur permukaan
- Pembuatan bayangan gambar
- Pantulan cahaya (*reflection*) maupun serapan cahaya (*transparancy*)
- Perhatian terhadap perpotongan antar objek
- Penghilangan objek-objek yang tersembunyi

Gambar di bawah ini mengilustrasikan proses rendering dari objek *wireframe*.



Gambar 1.15 Contoh *Graphics Rendering* (2)

Animation

Animation atau animasi adalah teknik-teknik untuk memberikan efek gerakan atau *motion* pada objek grafis. Pemberian efek gerak ini harus mengikuti kaidah-kaidah normal dari gerakan baik gerakan manusia, gerakan alam maupun gerakan objek-objek lainnya.

Efek animasi merupakan efek yang paling penting khususnya dalam pembuatan film-film yang bersifat banyak gerak. Dengan adanya animasi komputer maka terjadi efisiensi dalam hal pembuatan film sekaligus juga menciptakan kreativitas-kreativitas baru yang terkadang cukup sesasional. Saat ini efek animasi sudah sedemikian realistisnya sehingga kadang-kadang sukar dibedakan apakah yang ada dalam film itu aktor sesungguhnya atau hanya aknot palsu (*synthetic actor*).

Beberapa film kolosal yang memanfaatkan efek animasi dalam grafika komputer antara lain adalah Titanic, Jurassic Park, Dragonheart.



Gambar 1.16 Contoh Film Menggunakan Animasi Komputer

Latihan

1. Tentukan kepanjangan dan atau definisi istilah-istilah grafis sebagai berikut
 - a. CGA
 - b. CRT
 - c. LCD
 - d. JPG
 - e. GIF
 - f. BMP
2. Carilah aplikasi-aplikasi grafis terbaru dalam literatur, khususnya melalui internet. Aplikasi-aplikasi bisa berupa teknologi grafika komputer maupun implementasi grafika komputer dalam berbagai bidang.

Daftar Pustaka

1. Edward Angel, Interactive Computer Graphics: A Top-Down Approach with OpenGL 2nd, Addison Wesley, 2005
2. John F. Hughes, Andries Van Dam, Morgan Mcguire, David F. Sklar, James D. Foley, Steven K. Feiner, Kurt Akeley, Computer Graphics: Principles and Practice (3rd edition), Addison-Wesley, 2014

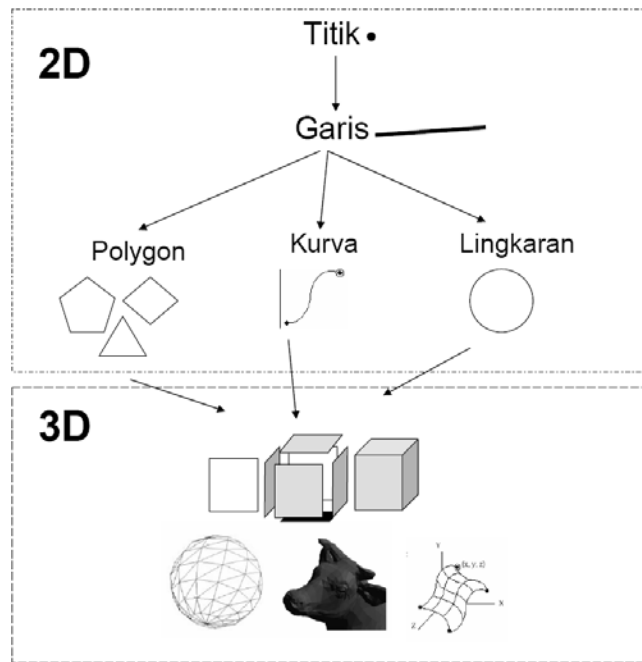
MODUL II PENGAMBARAN OBJEK PRIMITIF

Setelah membaca modul ini, mahasiswa akan memiliki pengetahuan dan mampu menjelaskan (i) apakah yang dimaksud dengan objek primitif (ii) algoritma-algoritma pembentukan garis (iii) algoritma-algoritma pembentukan lingkaran (iv) mengaplikasikan algoritma dalam program komputer

II.1 Pengertian objek primitif

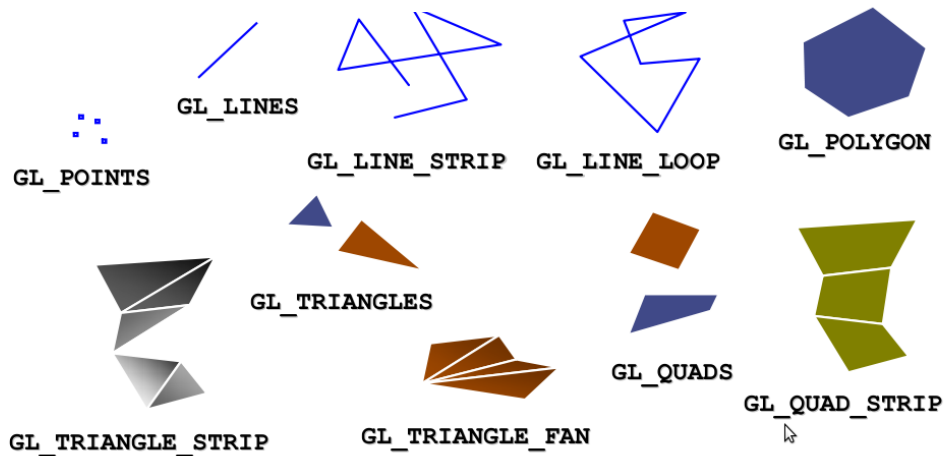
Gambar dapat dijelaskan dengan beberapa cara, bila menggunakan raster display, gambar ditentukan oleh satu set intensitas untuk posisi display pada display. Sedangkan dengan *scene* tampilan gambar dengan *loading array* dari *pixel* ke dalam *buffer* atau dengan mengkonversikan scan dari grafik geometri tertentu ke dalam pola *pixel*. Paket grafika dilengkapi dengan fungsi untuk menyatakan *scene* dalam bentuk struktur. Paket pemrograman grafika dilengkapi dengan fungsi untuk menyatakan *scene* dalam bentuk struktur dasar geometri yang disebut output primitif, dengan memasukkan output primitif tersebut sebagai struktur yang lebih kompleks.

Gambar 2.1 menunjukkan bahwa dari sebuah titik dapat dibentuk objek garis, dimana garis dibentuk dari 2 titik. Dari garis dapat dibentuk poligon, kurva maupun lingkaran. Dengan dasar bangun ini maka dapat dibentuk objek-objek lain yang lebih kompleks di antara objek-objek 3 dimensi misalnya kubus, bola, bahkan objek-objek gabungan semua elemen.



Gambar 2.1 Elemen-elemen Pembentuk Objek Grafis

OpenGL, salahsatu graphics engine, mendefinsiikan objek primitif pada gambar berikut ini.



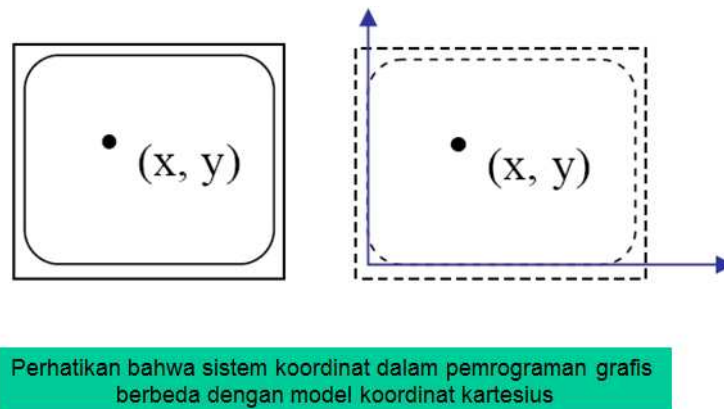
Gambar 2.2 Definisi Objek Grafis dalam OpenGL

II.2 Penggambaran Titik dan Garis

Pembentukan titik dilakukan dengan mengkonversi suatu posisi titik koordinat dengan program aplikasi ke dalam suatu operasi tertentu menggunakan output. *Random-scan* (vektor) *system* menyimpan instruksi pembentukan titik pada *display list* dan nilai koordinat

menentukan posisi pancaran *electron* ke arah lapisan fosfor pada layer. Garis dibuat dengan menentukan posisi titik diantara titik awal dan akhir dari suatu garis.

Perlu diperhatikan bahwa sistem koordinat biasa dan sistem koordinat grafika sedikit berbeda seperti ditunjukkan pada gambar berikut.

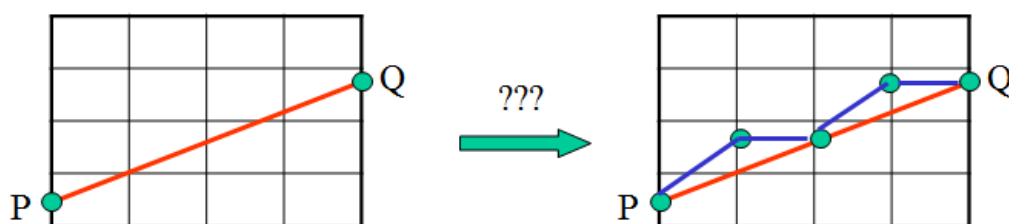


Gambar 2.3 Sistem Koordinat pada Layar dan Kartesius

Pada sistem grafika, posisi (0,0) ada pada kiri atas dari layar, sedangkan menurut sistem koordinat kartesius posisi (0,0) ada di tengah-tengah bidang gambar.

Algoritma Pembentukan Garis

Pada dasarnya, algoritma penggambaran atau pembentukan garis berusaha mencari suatu cara membentuk garis sedemikian rupa sehingga masalah jaggies dapat dihindarkan se optimal mungkin. Gambar di bawah ini mengilustrasikan keadaan tersebut.



Gambar 2.4 Pembentukan Garis Secara Diskrit

Algoritma Penggambaran Garis Dasar

Persamaan garis menurut koordinat Cartesian adalah:

$$y = m.x + b$$

dimana m adalah slope (kemiringan) dari garis yang dibentuk oleh dua titik yaitu (x_1, y_1) dan (x_2, y_2) . Untuk penambahan x sepanjang garis yaitu dx akan mendapatkan penambahan y sebesar $\Delta y = m \cdot \Delta x$. Dari formulasi di atas dikembangkan algoritma dasar untuk penggambaran garis.

Contoh: Diketahui dua buah titik A(2, 1) dan B(6, 4). Tentukan titik-titik digital yang dilalui oleh garis yang melalui kedua titik tersebut!

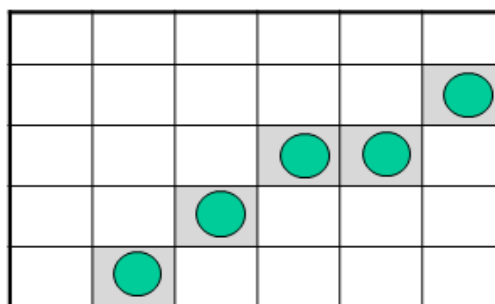
Jawab:

$$\text{Hitung nilai } m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{4 - 1}{6 - 2} = \frac{3}{4} = 0.75$$

Kemudian buat tabel berikut:

x	Δx	x^*	y	Δy	y^*	$[x]$	$[y]$
2	-	2	1	-	1	2	1
2	1	3	1	0.75	1.75	3	2
3	1	4	1.75	0.75	2.50	4	3
4	1	5	2.50	0.75	3.25	5	3
5	1	6	3.25	0.75	4.00	6	4

Jadi titik-titik digitalnya adalah (2,1), (3,2), (4,3), (5,3) dan (6,4).



Latihan: Dengan menggunakan algoritma dasar, tentukan koordinat titik-titik digital untuk garis yang dibentuk oleh dua titik sebagai berikut:

- (-5,5) dan (1,2)
- (4,3) dan (8,-2)
- (2,3) dan (5,3)

Catatan: Pada saat anda mengerjakan latihan di atas, akan ditemui masalah yang tidak dapat diselesaikan oleh algoritma dasar. Untuk itu dikembangkan algoritma lain yang dijelaskan berikut ini.

- d. (2,3) dan (2,5)
- e. (6,4) dan (2,1)

Algoritma DDA (Digital Differential Analyzer)

DDA adalah algoritma pembentukan garis berdasarkan perhitungan Δx dan Δy , menggunakan rumus $y = m \cdot \Delta x$. Garis dibuat dengan menentukan dua endpoint yaitu titik awal dan titik akhir. Setiap koordinat titik yang membentuk garis diperoleh dari perhitungan, kemudian dikonversikan menjadi nilai integer. Langkah-langkah pembentukan menurut algoritma DDA, yaitu :

1. Tentukan dua titik yang akan dihubungkan.
2. Tentukan salah satu titik sebagai titik awal (x_0, y_0) dan titik akhir (x_1, y_1).
3. Hitung $\Delta x = x_1 - x_0$ dan $\Delta y = y_1 - y_0$.
4. Tentukan step, yaitu jarak maksimum jumlah penambahan nilai x maupun nilai y dengan cara :
5. bila nilai $|\Delta y| > |\Delta x|$ maka step = nilai $|\Delta y|$.
6. bila tidak maka step = $|\Delta x|$.
7. Hitung penambahan koordinat pixel yaitu $x_increment = \Delta x / step$ dan $y_increment = \Delta y / step$.
8. Koordinat selanjutnya ($x+x_increment, y+y_increment$).
9. Posisi pixel pada layer ditentukan dengan pembulatan nilai koordinasi tersebut.
10. Ulangi step 6 dan 7 untuk menentukan posisi pixel selanjutnya, sampai $x = x_1$ dan $y = y_1$

Contoh :

Untuk menggambarkan algoritma DDA dalam pembentukan suatu garis yang menghubungkan titik (10,10) dan (17,16), pertama-tama ditentukan dx dan dy, kemudian dicari step untuk mendapatkan $x_increment$ dan $y_increment$.

$$\Delta x = x_1 - x_0 = 17 - 10 = 7$$

$$\Delta y = y_1 - y_0 = 16 - 10 = 6$$

selanjutnya hitung dan bandingkan nilai absolutnya.

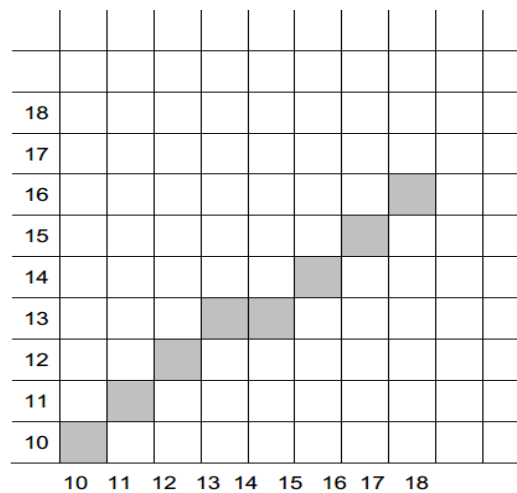
$$|\Delta x| = 7, |\Delta y| = 6$$

karena $|\Delta x| > |\Delta y|$, maka step = $|\Delta x| = 7$, maka diperoleh :

$$x_inc = 7/7 = 1$$

$$y_inc = 6/7 = 0,86$$

k	x	y	round(x), round(y)
			(10,10)
0	11	10,86	(11,11)
1	12	11,72	(12,12)
2	13	12,58	(13,13)
3	14	13,44	(14,13)
4	15	14,3	(15,14)
5	16	15,16	(16,15)
6	17	16,02	(17,16)



Latihan: Dengan menggunakan algoritma DDA, tentukan koordinat titik-titik digital untuk garis yang dibentuk oleh dua titik sebagai berikut:

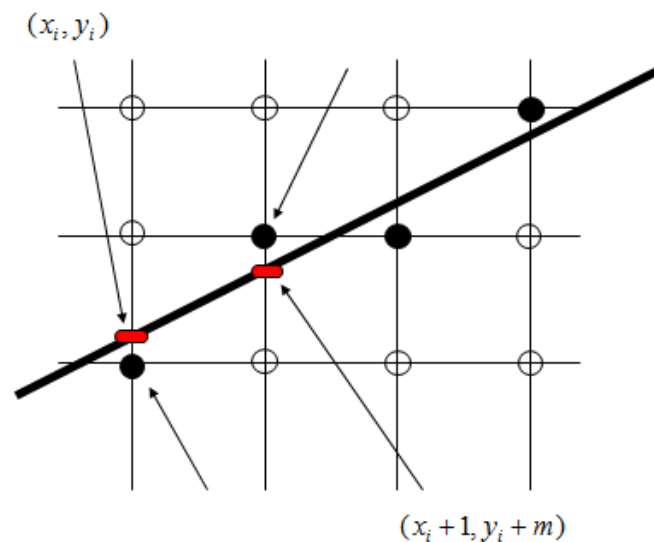
- (-5,5) dan (1,2)
- (4,3) dan (8,-2)
- (2,3) dan (5,3)
- (2,3) dan (2,5)
- (6,4) dan (2,1)

Catatan: Apakah semua soal dapat diselesaikan ?

Algoritma-algoritma sebelumnya secara umum sudah dapat menjawab masalah penggambaran garis, namun dalam komputasinya masih mengandung perhitungan aritmatik yang secara analisis membuat kebutuhan sumber daya komputasi menjadi besar karena operasi *floating points*. Untuk itu dikembangkan algoritma lain yaitu algoritma Bresenham yang bekerja pada domain bilangan integer sehingga walaupun algoritma menjadi panjang namun lebih efisien.

Algoritma Bresenham

Prosedur untuk menggambar kembali garis dengan membulatkan nilai x atau y kebilangan integer membutuhkan waktu, serta variable x, y dan m merupakan bilangan real karena kemiringan merupakan nilai pecahan. Bresenham mengembangkan algoritma klasik yang lebih menarik, karena hanya menggunakan perhitungan matematika dengan bilangan integer. Dengan demikian tidak perlu membulatkan nilai posisi setiap pixel setiap waktu. Algoritma garis Bresenham disebut juga midpoint line algorithm adalah algoritma konversi penambahan nilai integer yang juga dapat diadaptasi untuk menggambar sebuah lingkaran.



Gambar 2.5 Penggambaran Garis dengan Teknik Bresenham

Langkah-langkah untuk membentuk garis menurut algoritma ini adalah :

1. Tentukan dua titik yang akan dihubungkan dalam pembentukan garis.

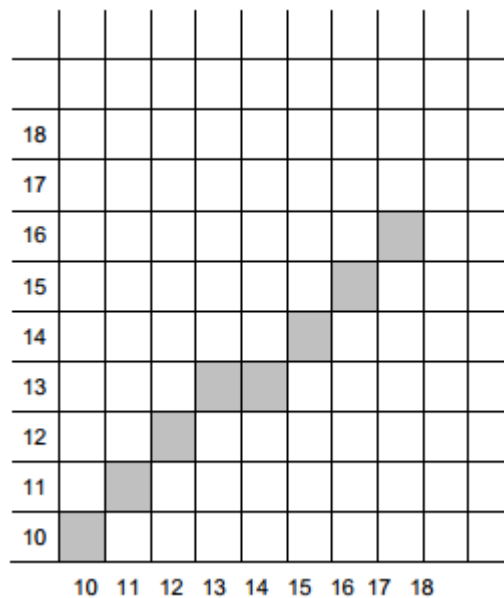
2. Tentukan salah satu titik disebelah kiri sebagai titik awal (x_0, y_0) dan titik lainnya sebagai titik akhir (x_1, y_1).
3. Hitung Δx , Δy , $2\Delta x$, dan $2\Delta y - 2\Delta x$.
4. Hitung parameter $p_0 = 2\Delta y - \Delta x$.
5. Untuk setiap x_k sepanjang jalur garis, dimulai dengan $k = 0$
6. bila $p_k < 0$ maka titik selanjutnya (x_{k+1}, y_k) dan $p_{k+1} = p_k + 2\Delta y$
7. bila tidak maka titik selanjutnya adalah (x_{k+1}, y_{k+1})
8. dan $p_{k+1} = p_k + 2\Delta y - 2\Delta x$.
9. Ulangi langkah nomor 5 untuk menentukan posisi pixel selanjutnya, sampai $x = x_n$.

Contoh :

Untuk menggambarkan algoritma Bresenham dalam pembentukan suatu garis yang menghubungkan titik (10,10) dan (17,16), pertama-tama ditentukan bahwa titik (10,10) berada disebelah kiri merupakan titik awal, sedangkan (17,16) merupakan titik akhir. Posisi yang membentuk garis dapat ditentukan dengan perhitungan sebagai berikut :

- $x = x_1 - x_0$ dan $y = y_1 - y_0$
- $x = 7$ dan $y = 6$
- parameter $p_0 = 2\Delta y - x$
- $p_0 = 5$
- increment $2\Delta y = 12$ $2\Delta y - 2\Delta x = -2$

k	p_k	(x_{k+1}, y_{k+1})
		(10,10)
0	5	(11,11)
1	3	(12,12)
2	1	(13,13)
3	-1	(14,13)
4	11	(15,14)
5	9	(16,15)
6	7	(17,16)



Pseudocode Algoritma Bresenham untuk Penggambaran Garis

```

procedure bres1(x1,y1,x2,y2:integer);
Var dx, dy, x, y, x_end, p, da, db, m1, m2 : integer;
begin
  dx := x2-x1;
  dy := y2-y1;
  if abs(dx)>=abs(dy) then da:=abs(dx) else da:=abs(dy);
  if abs(dx)>=abs(dy) then db:=abs(dy) else db:=abs(dx);
  if (abs(dx)>=abs(dy)) and (dx>=0) then m1:=3;
  if (abs(dx)>=abs(dy)) and (dx<0) then m1:=7;
  if (abs(dx)<abs(dy)) and (dy>=0) then m1:=1;
  if (abs(dx)<abs(dy)) and (dy<0) then m1:=5;
  if (dx>=0) and (dy>=0) then m2:=2;
  if (dx>=0) and (dy<0) then m2:=4;

```

```

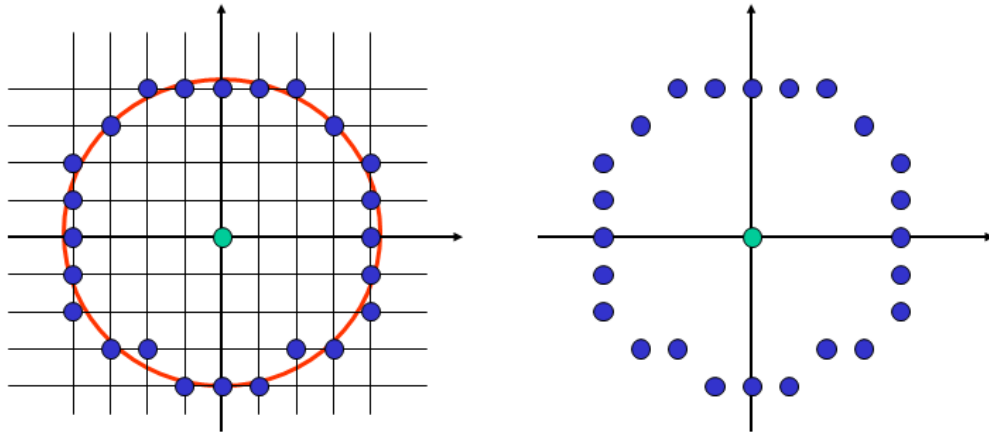
if (dx<0)    and (dy<0)  then m2:=6;
if (dx<0)    and (dy>=0) then m2:=8;
p := 2 * db - da;
x:=x1;
y:=y1;
set_pixel(x,y);
while (x <> x2) or (y<>y2)
begin
  if p>=0 then
  begin
    p := p + 2*db - 2*da;
    case m2 of
      2:begin x:=x+1;y:=y+1;end;
      4:begin x:=x+1;y:=y-1;end;
      6:begin x:=x-1;y:=y-1;end;
      8:begin x:=x-1;y:=y+1;end;
    end;
  end
  else
  begin
    case m1 of
      1:y:=y+1;
      3:x:=x+1;
      5:y:=y-1;
      7:x:=x-1;
    end;
    p := p + 2* db;
  end;
  set_pixel(x,y);
end;
end;

```

II.3 Algoritma Penggambaran Lingkaran

Lingkaran adalah kumpulan dari titik-titik yang memiliki jarak dari titik pusat yang sama untuk semua titik. Lingkaran dibuat dengan menggambarkan seperempat lingkaran, karena bagian lain dapat dibuat sebagai bagian yang simetris. Penambahan x dapat dilakukan dari 0 ke r sebesar unit step, yaitu menambahkan $\pm y$ untuk setiap step.

Pada dasarnya, mirip dengan penggambaran garis, algoritma penggambaran lingkaran pun berusaha mencari solusi optimal sedemikian rupa sehingga lingkaran yang dibentuk adalah sesempurna mungkin. Gambar di bawah ini memberikan ilustrasi penggambaran lingkaran.



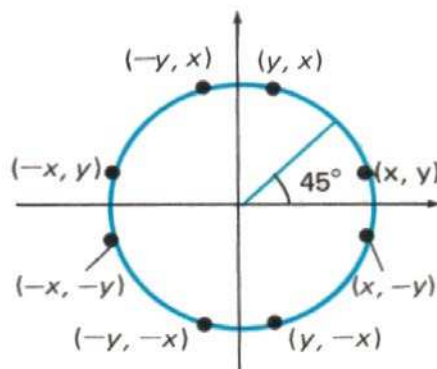
Simetris delapan titik

Pada umumnya, lingkaran digunakan sebagai komponen dari suatu gambar. Prosedur untuk menampilkan lingkaran dan elips dibuat dengan persamaan dasar dari lingkaran $x^2 + y^2 = r^2$.

Proses pembuatan lingkaran dapat dilakukan dengan menentukan satu titik awal. Bila titik awal pada lingkaran (x, y) , maka terdapat tiga posisi lain, sehingga dapat diperoleh delapan titik. Dengan demikian, hanya diperlukan untuk menghitung segmen 45o dalam menentukan lingkaran selengkapnya. Delapan titik simetris, yaitu :

- Kuadran I $(x, y), (y, x)$
- Kuadran II $(-x, y), (-y, x)$
- Kuadran III $(-x, -y), (-y, -x)$
- Kuadran IV $(x, -y), (y, -x)$

Gambar di bawah ini mengilustrasikan kuadran tersebut.



Algoritma lingkaran midpoint disebut juga algoritma lingkaran Bresenham. Algoritma yang digunakan membentuk semua titik berdasarkan titik pusat dengan penambahan semau

jalur disekeliling lingkaran. Dalam hal ini hanya diperhatikan bagian 45o dari suatu lingkaran, yaitu oktan kedua dari $x = 0$ ke $x = R/\sqrt{2}$, dan menggunakan prosedur circle point untuk menampilkan titik dari seluruh lingkaran.

$$f_{\text{circle}}(x,y) \begin{cases} < 0, \text{ bila } (x,y) \text{ di dalam garis lingkaran} \\ = 0, \text{ bila } (x,y) \text{ di garis lingkaran} \\ > 0, \text{ bila } (x,y) \text{ di luar garis lingkaran} \end{cases}$$

fungsi lingkaran menggambarkan posisi midpoint antara pixel yang terdekat dengan jalur lingkaran setiap step. Fungsi lingkaran menentukan parameter pada algoritma lingkaran.

Langkah-langkah pembentukan lingkaran :

1. Tentukan radius r dengan titik pusat lingkaran (x_c, y_c) kemudian diperoleh $(x_c, y_c) = (0, r)$.
2. Hitung nilai dari parameter $P_0 = 1 - r$
3. Tentukan nilai awal $k = 0$, untuk setiap posisi x_k berlaku sbb :
 - a. Bila $p_k < 0$, maka titik selanjutnya adalah (x_{k+1}, y_k)

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

- b. Bila $p_k > 0$, maka titik selanjutnya adalah (x_{k+1}, y_{k-1})

$$P_{k+1} = p_k + 2x_{k+1} + 1 - 2y_{k+1}$$

$$\text{Dimana } 2x_{k+1} = 2x_k + 2 \text{ dan } 2y_{k+1} = 2y_k - 2$$

4. Tentukan titik simetris pada ketujuh oktan yang lain.
5. Gerakkan setiap posisi pixel (x, y) pada garis melingkar dari lingkaran dengan titik pusat (x_c, y_c) dan tentukan nilai koordinat : $x = x + x_c$ dan $y = y + y_c$
6. Ulangi langkah ke 3 -5, sampai dengan $x \geq y$

Contoh :

Untuk menggambarkan algoritma Bresenham dalam pembentukan suatu lingkaran dengan titik pusat (0,0) dan radius 10, perhitungan berdasarkan pada oktan dari kuadran pertama dimana $x=0$ sampai $x=y$.

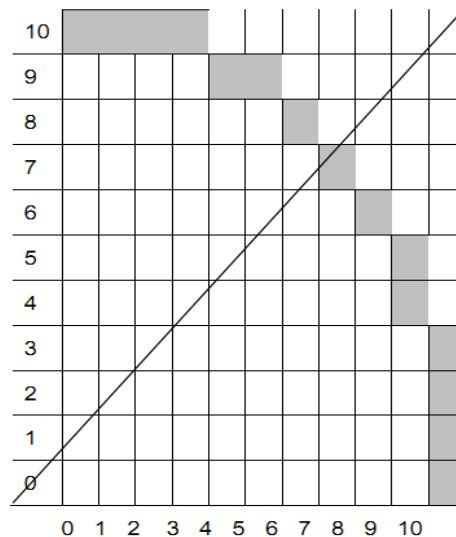
Penyelesaian :

$$(x_0, y_0) = (0, 0) \quad r = 10$$

$$(x_0, y_0) = (0, 10) \quad 2x_0 = 0, 2y_0 = 20$$

$$\text{parameter } p_0 = 1 - r = -9$$

k	p_k	(x_{k+1}, y_{k+1})	$2x_{k+1}$	$2y_{k+1}$
0	-9	(1,10)	2	20
1	-6	(2,10)	4	20
2	-1	(3,10)	6	20
3	6	(4,9)	8	18
4	-3	(5,9)	10	18
5	8	(6,8)	12	16
6	5	(7,7)	14	14



Pseudocode Algoritma Bresenman Untuk Lingkaran

```

BresenhamCircle(Xc, Yc, R)
Set X = 0 and Y = R
Set D = 3 - 2R
Repeat While (X < Y)
    Call DrawCircle(Xc, Yc, X, Y)
    Set X = X + 1
    If (D < 0) Then
        D = D + 4X + 6
    Else
        Set Y = Y - 1
        D = D + 4(X - Y) + 10
    Call Draw Circle(Xc, Yc, X, Y)

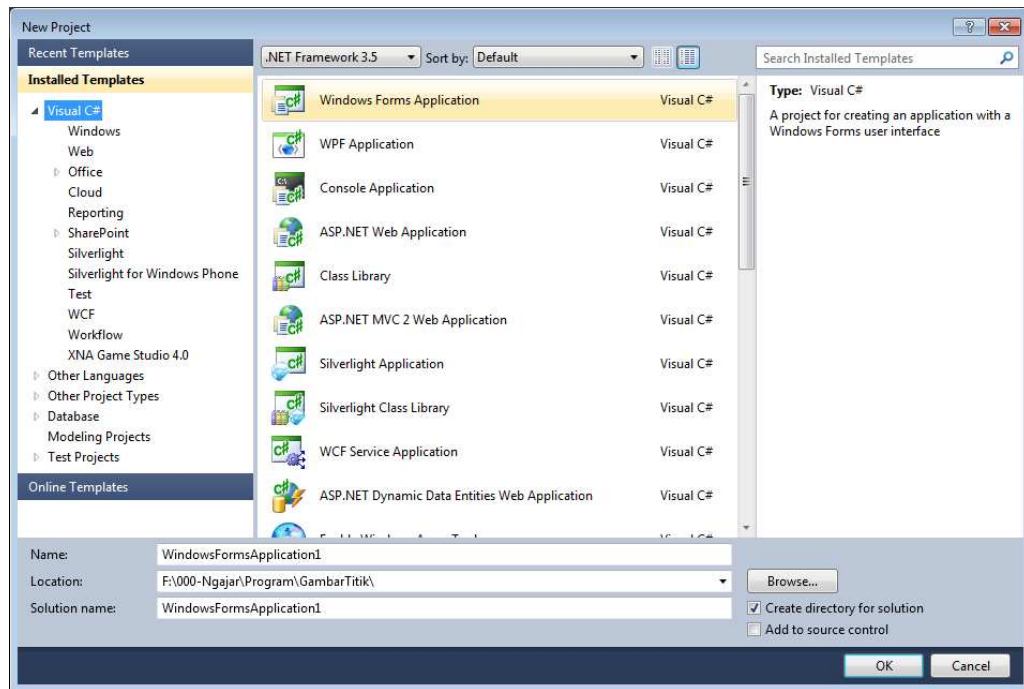
DrawCircle(Xc, Yc, X, Y)
Call Put Pixel(Xc+ X, Yc, + Y)
Call PutPixel(Xc-X, Yc, + Y)
Call PutPixel(Xc+ X, Yc,-Y)
Call PutPixel(Xc-X, Yc,-Y)
Call PutPixel(Xc+ Y, Yc, + X)
Call PutPixel(Xc-Y, Yc, + X)
Call PutPixel(Xc+ Y, Yc,-X)
Call PutPixel(Xc-Y, Yc,-X)

```

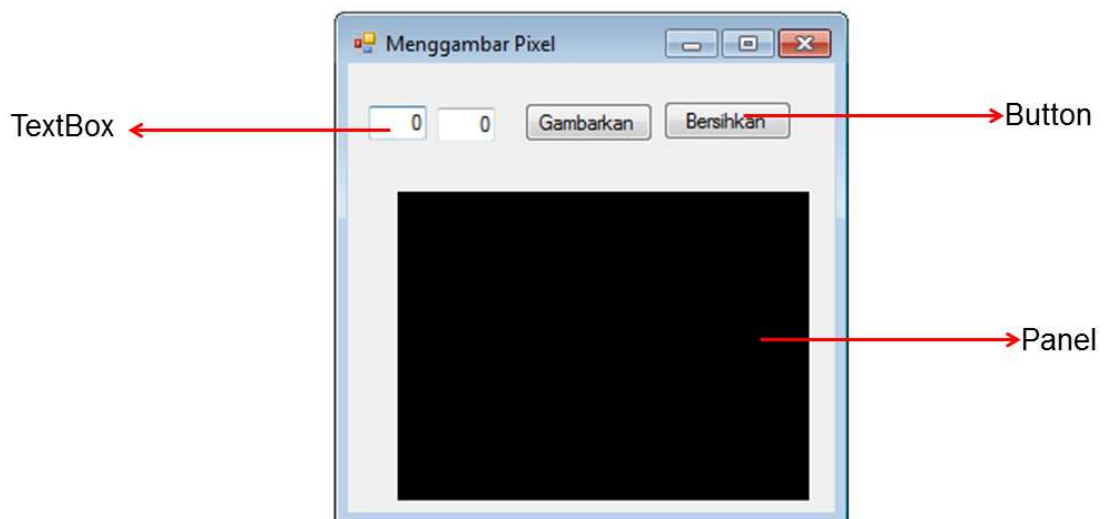
II.4 Implementasi Penggambaran Titik

Sebelum kita bisa melakukan pemrograman, harus dipersiapkan dulu perangkat lunak Visual Studio. Sata ini versi yang terakhir dari visual studio adalah Visual Studio 2013. Setelah Visual Studio di-install, jalankan Visual Studio dan lakukan langkah-langkah sebagai berikut. Dalam contoh ini akan digunakan Visual Studio 2010 Ultimate. Diasumsikan proses instalasi telah sukses dilakukan.

1. Jalankan Visual Studio
2. Buat Proyek Baru (File-New-Project)
3. Pilih Windows Form Application, yakinkan bahwa template yang digunakan adalah Visual C#



4. Tentukan Name, Location dan Solution Name sesuai dengan keinginan pemrogram
5. Jika sudah lengkap maka pada layar akan ditampilkan form kosong.
6. Dengan memilih komponen dalam ToolBox, buatlah form sebagai berikut:



7. Ketikkan program berikut pada tab Form1.cs

```
Graphics g;
int x, y;
Brush aBrush = (Brush)Brushes.White;

private void Form1_Load(object sender, EventArgs e)
```



```

{
    g = canvas.CreateGraphics();
}

private void DrawPixel_Click(object sender,
    EventArgs e)
{
    x=Convert.ToInt16(PointX.Text);
    y =Convert.ToInt16(PointY.Text);
    g.FillRectangle(aBrush, x, y, 1, 1);
}

private void ClearScreen_Click(object sender,
    EventArgs e)
{
    canvas.Refresh();
}

```

8. Jalankan program tersebut dan lihat hasilnya.

Latihan

1. Tentukan koordinat titik-titik digital untuk garis yang dibentuk oleh dua titik sebagai berikut:
 - a. (-5,5) dan (1,2)
 - b. (4,3) dan (8,-2)
 - c. (2,3) dan (5,3)
 - d. (2,3) dan (2,5)
 - e. (6,4) dan (2,1)

2. Gunakan algoritma DDA dan Bresenham untuk menentukan titik-titik digital antara
 - a. (-3,3) dan (-1,3)
 - b. (7,-1) dan (-4,-6)
 - c. (-3,3) dan (-1,-3)

Daftar Pustaka

1. Donald D. Hearn, M. Pauline, Warren Carithers, Computer Graphics with Open GL (4th Edition), Prentice-Hall, 2011
2. Mahesh Chand, Graphics Programming with GDI+, Addison-Wesley, 2003

MODUL III ATRIBUT OUTPUT PRIMITIF

Setelah membaca modul ini, mahasiswa akan memiliki pengetahuan dan mampu menjelaskan (i) apakah yang dimaksud dengan atribut output yang primitif (ii) algoritma untuk pengisian (*filling*) dari sebuah objek grafis (iii) karakter dan pembentukan karakter grafis (iv) *antialiasing* untuk optimasi grafis

III. 1 Pengertian atribut output primitif

Pada umumnya, setiap parameter yang memberi pengaruh pada output primitive ditampilkan sesuai dengan parameter atribut. Beberapa parameter atribut, seperti ukuran dan warna ditentukan sebagai karakteristik dasar dari parameter. Sedangkan yang lain ditentukan untuk penampilan pada kondisi tertentu.

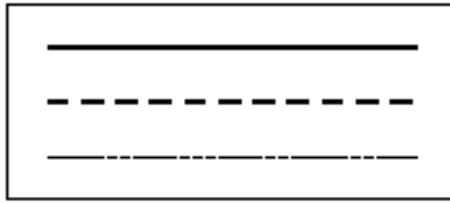
Teks dapat dibaca dari kiri ke kanan, miring searah diagonal (*slanted diagonal*), atau vertikal sesuai kolom. Salah satu cara untuk mengatur atribut output primitif, yaitu dengan daftar parameter fungsi yang berkaitan, contohnya fungsi menggambar garis dapat berisi parameter untuk warna, tebal, dan lainnya.

III. 2 Atribut Garis

Atribut dasar untuk garis lurus adalah *type* (tipe), *width* (tebal), dan *color* (warna). Dalam berapa paket aplikasi grafik, garis dapat ditampilkan dengan menggunakan pilihan *pen* atau *brush*.

Tipe Garis

Garis mempunyai beberapa *linetype* (tipe garis) diantaranya *solid line*, *dashed line* (garis putus), dan *dotted line* (garis titik-titik). Algoritma pembentukan garis dilengkapi dengan pengaturan panjang dan jarak yang menampilkan bagian solid sepanjang garis. Garis putus dibuat dengan memberikan nilai jarak dengan bagian *solid* yang sama. Garis titik-titik dapat ditampilkan dengan memberikan jarak yang lebih besar dari bagian *solid*.



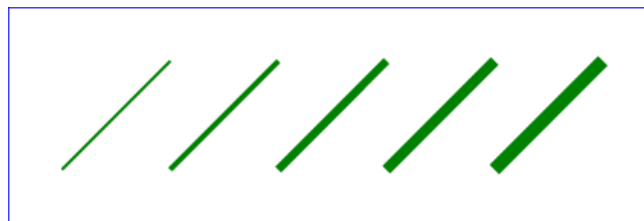
Gambar 3.1 Tipe Garis

Tebal Garis

Implementasi dari tebal garis tergantung dari kemampuan alat output yang digunakan. Garis tebal pada video monitor dapat ditampilkan sebagai garis *adjacent parallel* (kumpulan garis sejajar yang berdekatan), sedangkan pada *plotter* mungkin menggunakan ukuran pen yang berbeda.

Pada implementasi raster, tebal garis standar diperoleh dengan menempatkan satu piksel pada tiap posisi, seperti algoritma Bresenham. Garis dengan ketebalan didapatkan dengan perkalian integer positif dari garis standar, dan menempatkan tambahan piksel pada posisi sejajar. Untuk garis dengan *slope* kurang dari 1, rutin pembentukan garis dapat dimodifikasi untuk menampilkan ketebalan garis dengan menempatkan pada posisi vertikal setiap posisi x sepanjang garis.

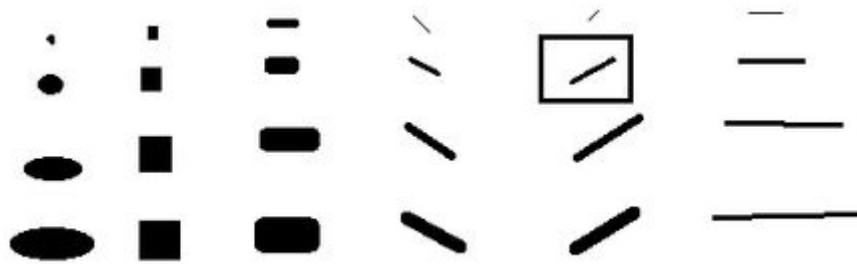
Untuk garis dengan *slope* lebih besar dari 1, ketebalan garis dapat dibuat dengan horizontal span.



Gambar 3.2 Ketebalan Garis

Pilihan Pen dan Brush

Pada beberapa paket aplikasi grafik, dapat ditampilkan dengan pilihan *pen* maupun *brush*. Kategori ini meliputi bentuk, ukuran, dan pola (*pattern*). Ketebalan yang bermacam-macam dari garis yang mempunyai bentuk pen dan brush dapat ditampilkan dengan cara mengubah ukuran dari *mask*.



Gambar 3.3 Berbagai Jenis *Pen* dan *Brush*

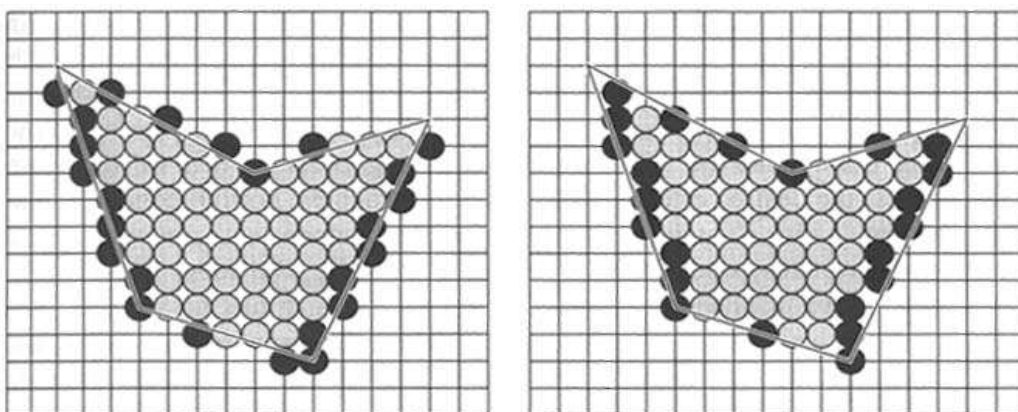
Warna Garis

Bila suatu sistem dilengkapi dengan pilihan warna (atau intensitas), parameter yang akan diberikan pada indeks warna termasuk dalam daftar nilai atribut dari sistem. Rutin *polyline* membuat garis pada warna tertentu dengan mengatur nilai warna pada *frame buffer* untuk setiap posisi piksel, menggunakan prosedur set piksel. Jumlah warna tergantung pada jumlah bit yang akan digunakan untuk menyimpan informasi warna.

III. 3 *Fill Area Primitif*

Fill area (pengisian area) output primitif standar pada paket aplikasi grafika pada umumnya adalah warna solid atau pola raster. Terdapat dua dasar pendekatan untuk mengisi area pada sistem raster::

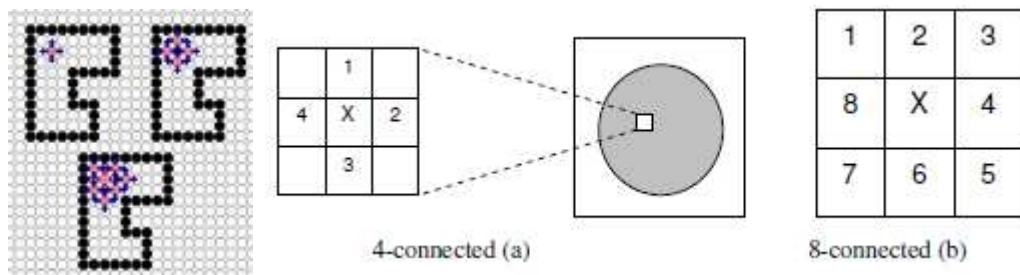
- Menentukan overlap interval untuk *scan line* yang melintasi area
- Dengan memulai dari titik tertentu pada posisi di dalam poligon dan menggambar dengan arah menyebar ke pinggir, sampai batas poligon.



Gambar 3.4 Teknik *Fill Area*

Algoritma *Boundary Fill*

Metode ini bermanfaat untuk paket aplikasi grafik interaktif, dimana titik dalam dapat dengan mudah ditentukan. Prosedurnya yaitu menerima input koordinat dari suatu titik (x,y), warna isi dan warna garis batas. Dimulai dari titik (x,y) prosedur memeriksa posisi titik tetangga, yaitu apakah merupakan warna batas, bila tidak maka titik tersebut digambarkan dengan warna isi. Proses ini dilanjutkan sampai semua titik pada batas diperiksa. Ada dua macam metode yaitu *4-connected* dan *8-connected*. Ilustrasi dapat dilihat pada gambar di bawah.



Gambar 3.5 Ilustrasi *Boundary Fill*

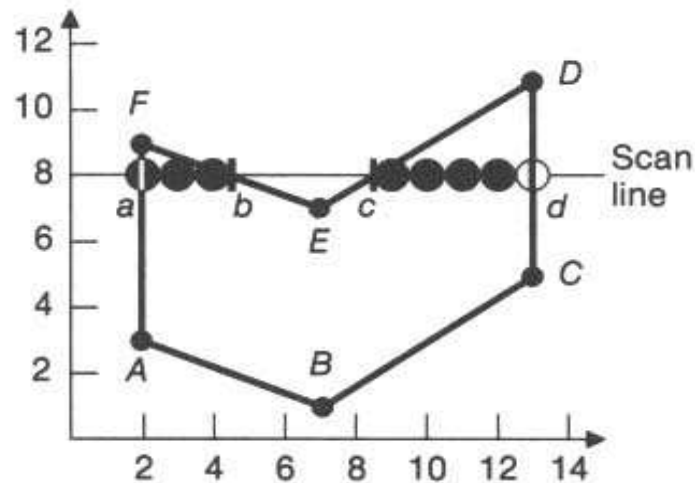
Algoritma Boundary-Fill adalah sebagai berikut:

```
Procedure BoundaryFill (x,y,fill,boundary : Integer);
Var
    Current : integer;
Begin
    Current = getpiksel(x,y);
    If (Current<>boundary) and (Current<>fill) then
        Begin
            setpiksel (x,y,fill);
            Boundaryfill4 (x+1,y,fill, boundary);
            Boundaryfill4 (x-1,y,fill, boundary);
            Boundaryfill4 (x,y+1,fill, boundary);
            Boundaryfill4 (x,y-1,fill, boundary);
        End;
    End;
```

Algoritma *Flood Fill*

Metode ini dimulai pada titik (x,y) dan mendefinisikan seluruh piksel pada bidang tersebut dengan warna yang sama. Bila bidang yang akan diisi warna mempunyai beberapa warna, pertama-tama yang dilakukan adalah membuat nilai piksel yang baru, sehingga semua piksel mempunyai warna yang sama.

Algoritma *Flood Fill* bisa lebih dioptimalkan dengan menambahkan fasilitas scan line sehingga yang tadinya nilai piksel yang akan dimasukan disimpan pada *stack* namun sekarang tidak dilakukan tetapi cukup dengan menginspeksi nilai piksel di sekitar, Dengan demikian terjadi optimasi terhadap memory. Gambar di bawah mengilustrasikan proses ini.



Gambar 3.6 Ilustrasi *Flood Fill*

Algoritmanya secara umum adalah sebagai berikut:

1. Find the intersections of the scan line with all edges, sort them in increasing order of x-coordinates: $\{a_1, a_2, \dots, a_n\} = \{a, b, c, d\}$
2. Fill in all piksels between a_{2k} and a_{2k+1} . E.g. those between a and b, and between c and d

III. 4 Karakter dan Pembentukan Karakter

Huruf, angka dan karakter lain dapat ditampilkan dalam berbagai ukuran (size) dan style. Jenis huruf atau typeface dikelompokkan menjadi beberapa kelompok antara lain menjadi 4 macam, yaitu serif, sanserif, egyptian dan dekoratif.

- **Serif**

Huruf dalam kategori serif mempunyai kait pada ujungnya. Misalnya : Times New Roman, Book Antiqua.

- **Sanserif**

Huruf dalam kategori sanserif tidak mempunyai kait pada ujungnya. Misalnya : Arial, Helvetica, Tahoma.

- **Egyptian**

Huruf dalam kategori *egyptian* mempunyai kait dengan bentuk segi empat yang mempunyai karakter kokoh. Dikenal juga sebagai Slab serif, Mechanistic, Square serif.

- **Dekoratif**

Huruf dalam kategori dekoratif mempunyai bentuk indah. Misalnya : monotype corsiva

- **Monospace**

Huruf dalam kategori monospace disebut juga *non proportional spacing* dimana ukuran huruf tidak bervariasi atau tetap. Misalnya antara huruf I dan huruf M, ukuran lebarnya sama.

Berikut ini adalah visualisasi tipe huruf yang populer.

POPULAR SERIF FONTS:

Book Antigua

Bookman Old Style

Cambria

Cambria Math

Century

Century Schoolbook

Chaparral Pro

CHARLEMAGNE STD BOLD

Cooper Black

Garamond

Garamond Premier Pro

Adobe Garamond Pro

Goudy Old Style

GOUDY STOUT

Goudy Old Style

Georgia

High Tower Text

Lucida Bright

Palatino Linotype

Perpetua

PERPETUA TITLING MT

Poor Richard

Rockwell

Rockwell Condensed

Rockwell Extra Bold

Times New Roman

TRAJAN PRO

POPULAR SANS SERIF FONTS:

Agency FB regular

Arial Narrow

Arial Regular

Arial Black

Arial Rounded MT Bold

Bauhaus 93

Berlin Sans FB

Calibri

Century Gothic

Franklin Gothic Book

Franklin Gothic Demi

Franklin Gothic Heavy

Gill Sans MT

Gill Sans MT Condensed

Gill Sans Ultra Bold Condensed
Impact

GT-HelveticaPPlot

Lucida Sans

Microsoft Sans Serif

Myriad Roman

Myriad Pro Condensed

Myriad Pro Regular

Segoe UI

Tahoma

Trebuchet MS

Tw Cen MT

Tw Cen MT Condensed

Verdana

BLOCK & SLAB SERIF

AACHEN BT

ACADEMIC M54

ATHLETIC REGULAR

BLOCK

BULLPEN

COLLEGE-BLACK

COLLEGIATE-INSIDE

COMMANDO

COMPACTA BLACK BT

FACE OFF M54

GEOSLAB703 XBD BT

MACHINE BT

SPORTS JERSEY

SQUARE SLABSERIF 711 BOLD

Decorative

Textile

Texton

Swing Bold

Skia

Kidprint

French Script

Comic Sans

*more
examples
here*

Courier New (Windows 3.1)

10pt Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean auctor. Vestibulum justo. Sed eu felis. Donec laoreet, justo ut auctor tristique, elit erat suscipit mauris, eget volutpat justo erat sed metus.

Lucida Console

10pt Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean auctor. Vestibulum justo. Sed eu felis. Donec laoreet, justo ut auctor tristique, elit erat suscipit mauris, eget volutpat justo erat sed metus.

Consolas (Office 2007, Vista)

10pt Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean auctor. Vestibulum justo. Sed eu felis. Donec laoreet, justo ut auctor tristique, elit erat suscipit mauris, eget volutpat justo erat sed metus.

Gambar 3.7 Contoh Berbagai-macam Font

Tiga macam metode dapat digunakan untuk menyimpan jenis huruf dalam komputer yaitu

- *Bitmap*
- *Outline*
- *Stroke-based*

Metode sederhana bitmap menggunakan pola grid dengan bentuk segi empat, dan karakternya disebut dengan *bitmap font*. Grid dari karakter dipetakan pada posisi *frame buffer*, bit yang mempunyai nilai 1 berhubungan dengan tampilan piksel pada monitor.

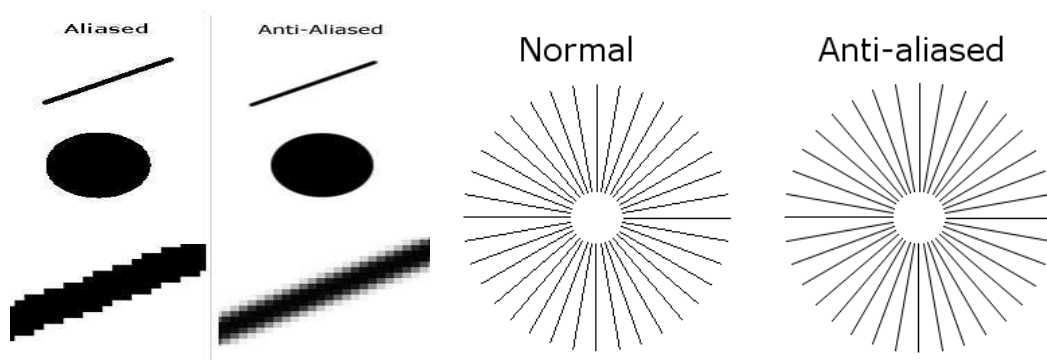
Metode *Outline* menggunakan jenis gambar vektor untuk mempresentasikan font. Jenis ini biasanya digunakan pada printer laser dan teknik yang berkualitas tinggi. Contohnya adalah postscript dan truetype

Metode *stroke-based* yaitu dengan stroke menggunakan garis lurus dan kurva, karakternya disebut dengan *outline font*. Huruf ditampilkan menurut koordinat relatif (x,y) dimana pusat dari koordinat adalah pada posisi kiri bawah dimana karakter pertama yang ditampilkan.

III. 5 Antialiasing

Seperti yang telah dikatakan sebelumnya bahwa konversi *raster-scan* adalah pengisian nilai-nilai elemen suatu "matriks" (yaitu *frame buffer*) sedemikian rupa sehingga secara visual "tergambarkan" primitif-primitif grafik yang bersangkutan. Jadi pada dasarnya adalah semacam diskretisasi obyek tersebut. Selanjutnya sebagai sesuatu yang diskret, masalah yang timbul adalah distorsi informasi yang disebut *aliasing*. Secara visual obyek garis atau batas suatu area akan terlihat sebagai tangga (efek tangga atau "*jaggies*"). Peningkatan resolusi *frame buffer* dapat mengurangi efek ini namun tidak dapat dihilangkan sama sekali karena keterbatasan teknologi (ingat faktor-faktor yang menentukan resolusi: *refresh rate*, dan ukuran *frame buffer*).

Pada sistem raster dengan tingkat intensitas > 2 bisa diaplikasikan metoda *antialiasing* dengan memodifikasi intensitas piksel-piksel "batas" obyek dengan latar atau obyek lainnya. Modifikasi tersebut akan memperhalus batas-batas tersebut sehingga mengurangi penampakan yang "*jaggies*" tersebut. Gambar berikut mengilustrasikan gambar sebelum dan sesudah pengaktifan *antialiasing*.



Gambar 3.8 Ilustrasi *Antialiasing*

Proses antialiasing dapat dilakukan melalui 3 pendekatan yaitu:

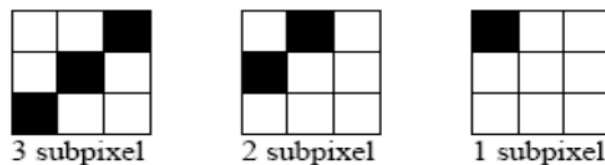
- *Supersampling (postfiltering)*
- *Area sampling*
- *Piksel phasing*

III.5.1 Supersampling dan Postfiltering

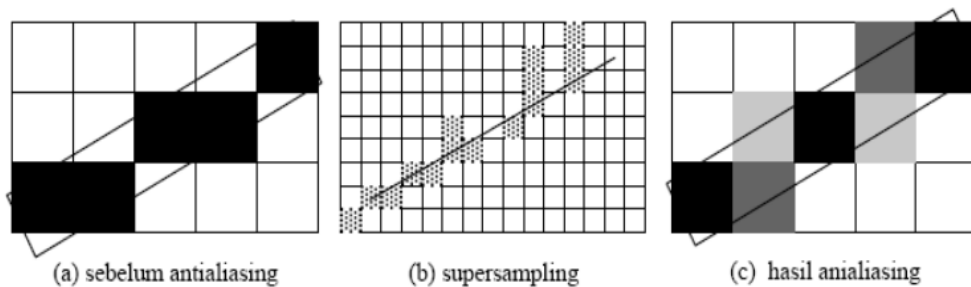
Berdasarkan logika metoda ini "memperhalus" ukuran piksel ke dalam subpiksel-subpiksel dan "menggambarkan" garis pada grid subpiksel tersebut. lalu nilai intensitas suatu piksel ditentukan sesuai dengan berapa banyak subpikselya dikenai "garis" tersebut. Relasi: intensitas piksel \sim jumlah subpiksel pada garis.

Ada dua cara penghitungan relasi tersebut :

- Menganggap garis adalah garis dengan ketebalan infinitesimal 0 (hanya garis lojik). Untuk subsampling 3x3 ada 4 kemungkinan tingkatan: 3 subpiksel, 2 subpiksel, 1 subpiksel, dan tidak ada. Pemberian intensitas sesuai dengan keempat tingkat tersebut.

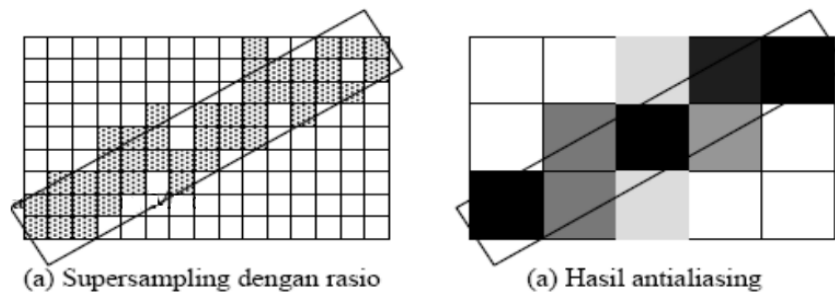


Contoh:



Gambar 3.9 Ilustrasi Supersampling dan Postfiltering

- Menganggap garis adalah garis dengan tebal tetap yaitu 1 piksel (yaitu suatu segiempat dengan lebar 1 piksel) dan intensitas dihitung sesuai dengan jumlah subpiksel yang "tertutupi" oleh segi empat ini (Perlu diambil acuan bahwa suatu subpiksel "tertutupi", misalnya jika sudut kiri bawah subpiksel ada di dalam segi empat). Yang paling sederhana adalah menggunakan nilai rasio jumlah subpiksel terhadap total subpiksel pada piksel sebagai fungsi intensitas. Untuk subsampling 3x3 total subpiksel adalah 9 sehingga ada 10 tingkat intensitas yang bisa diberikan. Khusus titik ujung yang bernilai bilangan bulat (karena bisa untuk koordinat bilangan real) Akan diberi nilai penuh.



Gambar 3.10 Ilustrasi *Supersampling* dengan Rasio

Alternatif penghitungan sederhana (rasio tsb.) ini adalah dengan pembobotan dengan mask diskret (*Pixelweighting Mask*), dan pembobotan dengan *mask* kontinu (*continuous filtering*).

Pixelweighting Mask

Alternatif menggunakan rasio secara langsung di atas, teknik *filtering* dalam pengolahan citra (bedanya: pengolahan citra pada piksel sedangkan di sini pada subpiksel) dengan suatu mask (atau kernel) sesuai dengan subdivision piksel misalnya 3x3 subpiksel digunakan untuk menghitung. Ada beberapa bentuk mask.

Contohnya:

- box *mask* (berefek *averaging*)
- gaussian *mask*

Kadang-kadang mask meliputi juga subpiksel di piksel tetangganya untuk mendapatkan hasil yang lebih *smooth*.

Continuous Filtering

Smoothing mirip *weighting mask* di atas pada subpiksel-subpiksel (dari piksel ybs. dan juga dari subpiksel tetangganya) namun menggunakan fungsi permukaan kontinu: box, konus, atau Gaussian. Jadi secara teoritis dilakukan konvolusi antara fungsi filter dengan fungsi citra pada tingkat subpiksel. Secara praktis untuk mengurangi komputasi digunakan suatu *table-lookup* dari kombinasi piksel dengan piksel-piksel tetangganya.

III.5.2 Area Sampling

Pada *Unweighted Area Sampling* suatu garis dianggap sebagai segi empat dengan lebar 1 piksel seperti halnya pada *supersampling* cara kedua di atas. Yang dihitung adalah luas bagian piksel yang tertutup "segiempat" garis tersebut secara geometris. Penghitungan lebih akurat tetapi karena memerlukan perhitungan yang lebih rumit maka metoda ini lebih banyak digunakan untuk *anti-aliasing* batas dari *fill-area*. Metoda ini menghitung luas bagian dari piksel yang tertutup area (garis atau fill-area) dan dari rasio luas tsb. terhadap luas piksel dapat ditentukan bobot *foreground* terhadap *background* untuk mendapatkan intensitas piksel. Cara penghitungannya?

Untuk *fill-area* dengan memodifikasi *midpoint algorithm* untuk garis sehingga fungsi diskriminan p menentukan juga persentasi tsb. Dalam algoritma ini pada persamaan garis

$$y = m x + b, m > 1$$

digunakan fungsi keputusan:

$$p = m (x_i + 1) + b - (y_i + \frac{1}{2})$$

Sementara bagian piksel yang tertutup area di bawah garis tersebut adalah suatu trapesium dengan ketinggian kiri $y = m (x_i - \frac{1}{2}) + b - (y_i - \frac{1}{2})$ dan ketinggian kanan $y = m (x_i + \frac{1}{2}) + b - (y_i - \frac{1}{2})$ serta lebar 1 (satuan piksel). Luas trapesium ini adalah $= m x_i + b - (y_i - 0.5) = p - (1 - m)$

III.5.3 Pixel Phasing

Pergeseran mikro (*microposition*) yang dilakukan oleh deflektor elektron sebesar $\frac{1}{4}$, $\frac{1}{2}$ atau $\frac{3}{4}$ diameter piksel. Metode ini biasanya dipasang built-in pada *chipset* grafis dan pada *graphics driver*.

III.5.4 Kompensasi Perbedaan Intensitas Garis

Secara normal garis diagonal (tanpa antialiasing) lebih tipis dari garis horisontal/vertikal karena pada garis tsb. piksel-piksel lebih *spanned* dari pada piksel-piksel pada garis horisontal/diagonal. Jadi secara visual efek ini dapat juga dikurangi dengan menaikkan intensitas garis yang mengarah diagonal sesuai dengan sudut.

Latihan

1. Sebutkan karakteristik atau properties dari sebuah garis
2. Jelaskan perbedaan anda *proportional spacing font* dan *monospace font* !

3. Gambarkan apa yang dimaksud dengan antialiasing, metode apa saja yang termasuk di dalamnya

Daftar Pustaka

1. David F. Rogers, Alan J. Adams , Mathematical Elements for Computer Graphics (2nd edition), McGraw-Hill, 1989
2. Donald D. Hearn, M. Pauline, Warren Carithers, Computer Graphics with Open GL (4th Edition), Prentice-Hall, 2011
3. John F. Hughes, Andries Van Dam, Morgan Mcguire, David F. Sklar, James D. Foley, Steven K. Feiner, Kurt Akeley, Computer Graphics: Principles and Practice (3rd edition), Addison-Wesley, 2014

MODUL IV *WINDOWING* DAN *CLIPPING*

Setelah membaca modul ini, mahasiswa akan memiliki pengetahuan dan mampu menjelaskan (i) konsep transformasi umum dalam konteks konseptual grafika komputer (ii) proses transformasi *windows-viewport* serta komputasinya (iii) proses *clipping* dengan algoritma-algoritma standar

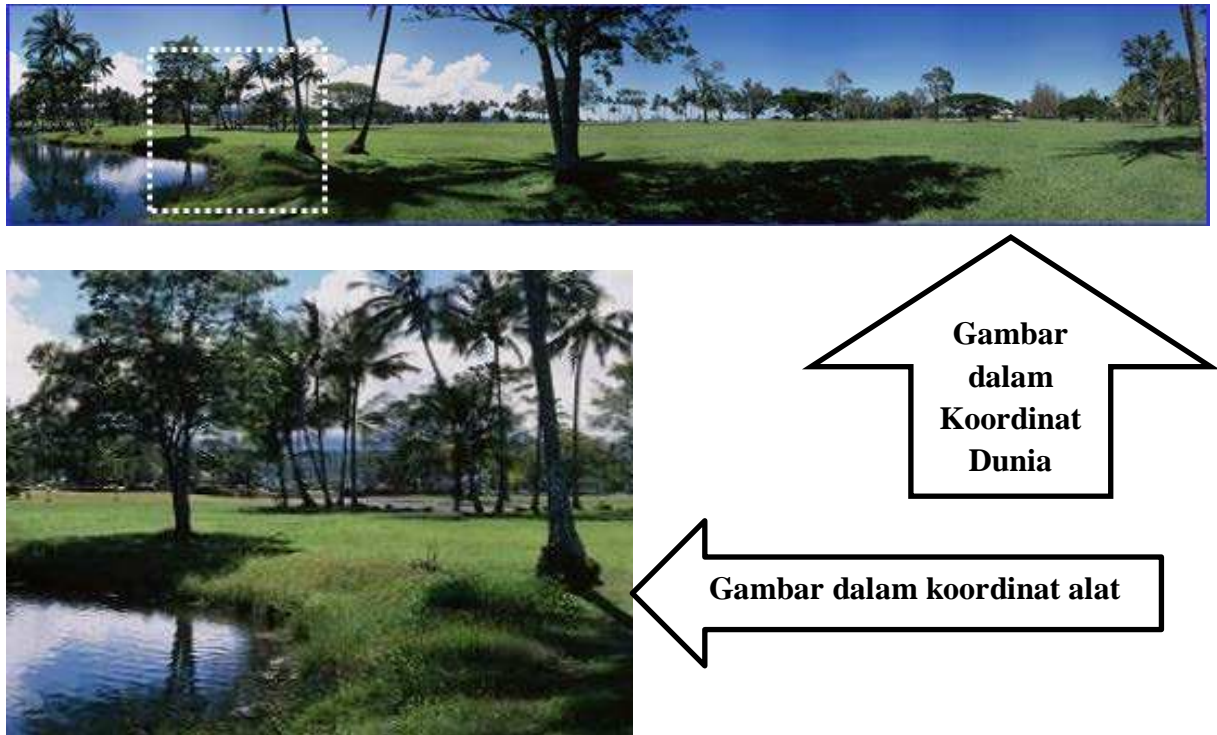
IV. 1 Model Konseptual Grafika Komputer

Sebagaimana sudah dijelaskan pada bagian awal buku ini, grafika komputer adalah ilmu yang dipelajari dan dikembangkan untuk mentransformasikan suasana atau pemandangan (*scene*) nyata yang ada dalam ruang 3 dimensi ke dalam peralatan komputer, dalam hal ini adalah layar monitor, yang pada dasarnya bekerja dalam 2 dimensi.

Proses transformasi pemandangan nyata yang begitu luas ke dalam monitor komputer yang relatif sempit memberikan pemahaman baru akan perlunya *windowing* dalam proses tersebut. Proses *windowing* akan membatasi luas pandang dari objek sesuai dengan ukuran *window*.

Berikut ini adalah definisi dari beberapa istilah yang bisa mengantarkan kita kepada pemahaman konsep *windowing*. *Window* adalah sebuah area pada koordinat dunia yang dipilih untuk ditampilkan pada alat *display*; sedangkan *Viewport* adalah sebuah area pada alat *display* yang merupakan hasil pemetaan dari *window*. Ilustrasi pada Gambar 4.1 akan memberikan kejelasan tentang proses *windowing*.

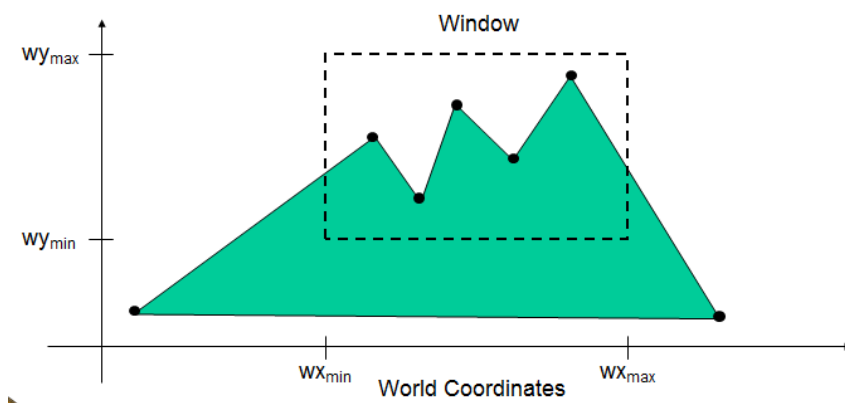
Gambar 4.1 adalah pemandangan nyata dalam koordinat dunia. Region persegi empat adalah windows yang tentunya memiliki ukuran tertentu. Pada saat kita akan memindahkan objek yang ada pada *window* ke dalam alat, maka kita melakukan proses *windowing*. Hasilnya seperti ditunjukkan pada gambar di bawahnya. Tentu saja sistem koordinat pada alat berbeda dengan sistem koordinat nyata. Koordinat nyata bekerja pada domain derajat, sedangkan koordinat alat bekerja pada domain piksel. Hal ini yang membuat diperlukannya pemetaan atau *mapping* dari koordinat dunia ke koordinat alat.



Gambar 4.1 Ilustrasi Windowing

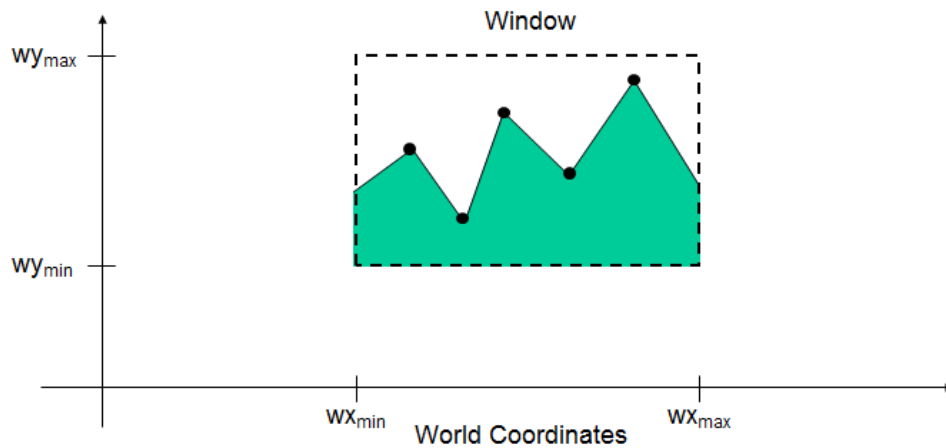
IV. 2 Transformasi *Windows-Viewport*

Ketika pemandangan ditampilkan pada layar, maka yang kelihatan hanya yang ada di dalam window sebagaimana ditunjukkan pada gambar di bawah ini.



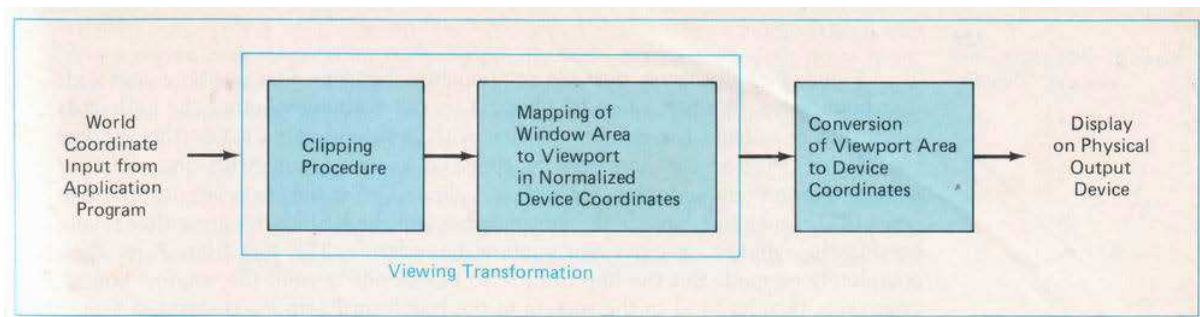
Gambar 4.2 Konsep Windowing

Proses pencuplikan pada *windows* disebut juga proses *clipping*, dimana yang terlihat oleh alat hanyalah yang ada di dalam *windows* saja, sebagaimana diilustrasikan pada gambar berikut.



Gambar 4.3 Konsep *Clipping*

Langkah-langkah yang kelihatannya sederhana tersebut dapat digambarkan dengan skema sebagai berikut:



Gambar 4.4 Diagram Blok Transformasi *Windowing*

Proses pemetaan yang dilakukan melibatkan proses Matematika yang pada dasarnya tidak terlalu rumit. Di bawah ini disajikan penurunan formulasi untuk mendapatkan korelasi antara kordinat dunia dan (*window*) dan koordinat alat (*viewport*).

Apabila diasumsikan ada sebuah titik pada kordinat dunia (X_w, Y_w), dan diketahui ukuran *windows* adalah ($X_{wmin}, Y_{wmin}, X_{wmax}, Y_{wmax}$), dan ukuran *viewport* adalah ($X_{vmin}, Y_{vmin}, X_{vmax}, Y_{vmax}$), maka koordinat titik (X_w, Y_w) pada *viewport* (X_v, Y_v) dapat dihitung dengan formulasi berikut ini.

$$\frac{x_v - x_{v \min}}{x_{v \max} - x_{v \min}} = \frac{x_w - x_{w \min}}{x_{w \max} - x_{w \min}}$$

$$x_v = x_{v \min} + (x_w - x_{w \min}) \cdot S_x$$

$$S_x = \frac{x_{v \max} - x_{v \min}}{x_{w \max} - x_{w \min}}$$

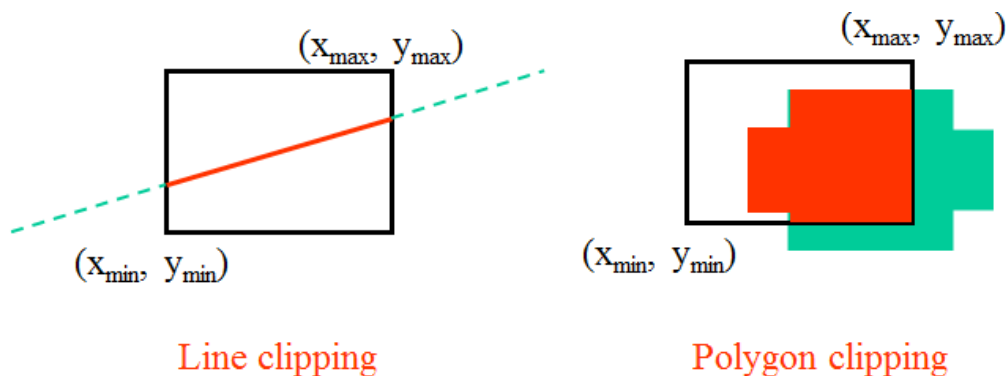
$$\frac{y_v - y_{v \min}}{y_{v \max} - y_{v \min}} = \frac{y_w - y_{w \min}}{y_{w \max} - y_{w \min}}$$

$$y_v = y_{v \min} + (y_w - y_{w \min}) \cdot S_y$$

$$S_y = \frac{y_{v \max} - y_{v \min}}{y_{w \max} - y_{w \min}}$$

IV. 3 Clipping

Clipping adalah proses pemotongan objek atau pengguntingan objek sehingga hanya objek yang berada pada area yang menjadi perhatian saja yang terlihat. Proses ini merupakan hal yang bisa dengan teknologi yang ada dewasa ini, namun proses internal pemrograman di dalamnya tidak sesederhana memakainya. Gambar di bawah ini mengilustrasikan proses clipping garis dan *clipping polygon*.

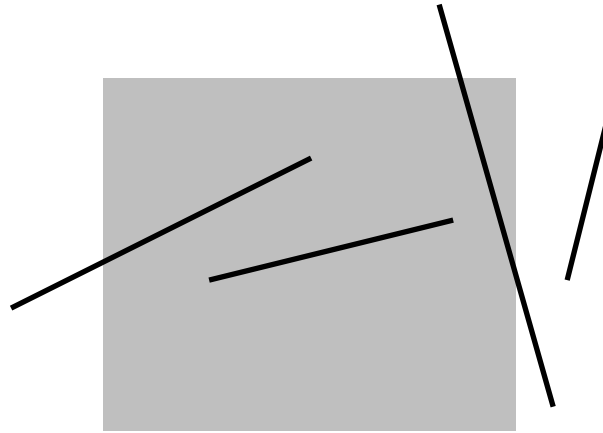


Gambar 4.5 Line Clipping dan Polygon Clipping

Penampakan Garis

Garis-garis yang tampak pada area gambar atau *viewport* dapat dikelompokkan menjadi tiga yaitu:

1. Garis yang terlihat seluruhnya (*Fully visible*).
2. Garis yang hanya terlihat sebagian (*Partiality Visible*).
3. Garis yang tidak terlihat sama sekali (*Fully Invisible*).

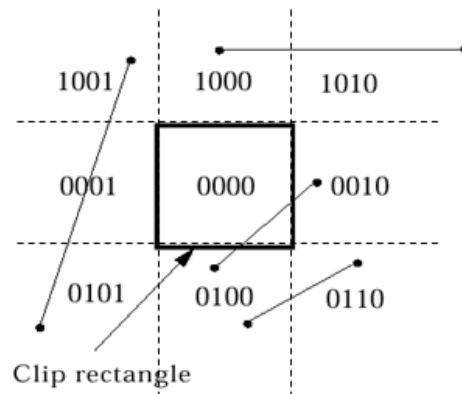


Gambar 4.6 Konsep Garis dalam Viewport

Proses *clipping* dilakukan terhadap garis-garis yang berada pada kondisi ke-2 yaitu garis-garis yang hanya terlihat sebagian saja.

Algoritma Clipping Garis Cohen-Shuterland

Pada algoritma Cohen-Sutherland, region *viewport* dibagi menjadi 9 dan masing-masing memiliki kode bit atau *bit code* yang terdiri dari 4 bit yang menyatakan kondisi dari garis yang melalui viewport atau region yang dimaksud.



Gambar 4.7 Bit Code dalam Cohen-Sutherland

Kode empat bit menunjukkan posisi ujung garis pada region *viewport*. Sebagai contoh garis dengan ujung yang memiliki kode bit 1001 artinya ujungnya ada di kiri atas *viewport*. Ujung dengan kode bit 0010 berarti ada di kanan *viewport*. Dengan mengacu kepada kode bit maka proses *clipping* akan dilakukan secara lebih mudah dan efisien.

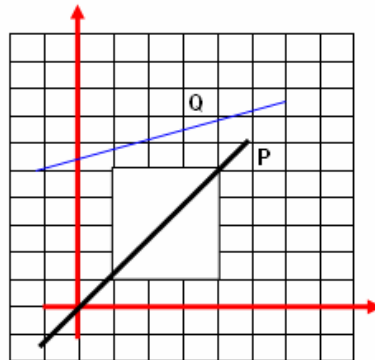
Contoh perhitungan

Diketahui koordinat *viewport* atau area gambar adalah (xMin, yMin, xMax, yMax) yaitu (1,1,4,5) dan dua buah garis P dengan koordinat (-1,2), (5,6) dan garis Q dengan koordinat (-1,5), (6,7).

1. Tentukan *region code* atau kode bit dari garis-garis tersebut
2. Apabila statusnya partially visible, tentukan titik potongnya dengan batas *viewport*

Jawab:

Permasalahan dapat divisualkan pada gambar berikut.



Garis P:

Ujung garis P(-1, -2)

L = 1; karena $x < x_{\text{Min}}$ atau $-1 < 1$

R = 0; karena $x < x_{\text{Max}}$ atau $-1 < 4$

B = 1; karena $y < y_{\text{Min}}$ atau $-2 < 1$

T = 0; karena $y < y_{\text{Max}}$ atau $-2 < 5$

Dengan demikian *region code* untuk ujung P(-1,-2) adalah 0101

Ujung garis P(5,6)

L = 0; karena $x > x_{\text{Min}}$ atau $5 > 1$

R = 1; karena $x > x_{\text{Max}}$ atau $5 > 4$

B = 0; karena $y > y_{\text{Min}}$ atau $6 > 1$

T = 1; karena $y > y_{\text{Max}}$ atau $6 > 5$

Dengan demikian *region code* untuk ujung P(5,6) adalah 1010

Karena *region code* dari kedua ujung garis tidak sama dengan 0000 maka garis P bersifat partially invisible dan perlu dipotong.

Garis Q:

Dengan cara yang sama pada garis P maka akan ditentukan region code:

Ujung garis Q(-1,5) mempunyai region code = 0001

Ujung garis Q(6,7) mempunyai region code = 1010

Karena region code tidak sama dengan 0000 maka garis Q bersifat kemungkinan partialy invisible dan perlu dipotong.

Penentuan Titik Potong dengan Region

Dilakukan berdasarkan tabel berikut.

Region Bit	Berpotongan Dengan	Dicari	Titik Potong
L=1	xMin	yP1	(xMin, yP1)
R=1	xMax	yP2	(xMax, yP2)
B=1	yMin	xP1	(xP1, yMin)
T=1	yMax	xP2	(xP2, yMax)

xP1, xP2, yP1 dan yP2 dihitung dengan formulasi berikut ini.

$$xP1 = x1 + \frac{yMin - y1}{M}$$

$$xP2 = x1 + \frac{yMax - y1}{M}$$

$$yP1 = y1 + M * (xMin - x1)$$

$$yP2 = y1 + M * (xMax - x1)$$

Dimana M dihitung dengan formula

$$M = \frac{Y2 - Y1}{X2 - X1}$$

Titik potong P (-1,-2), (5,6)

$$M = \frac{6 - (-2)}{5 - (-1)} = \frac{8}{6}$$

Region code di (-1,2) adalah 0101 (TBRL), berarti B=1 dan L=1

$$L=1 \text{ berarti } yP1 = y1 + M * (X \text{ min} - x1) = -2 + \frac{8}{6} * (1 - (-1)) = \frac{2}{3} = 0.86$$

Titik potongnya adalah (1, 0.86)

$$B=1 \text{ berarti } xP1 = x1 + \frac{yMin - y1}{M} = -1 + \frac{1 - (-2)}{\frac{8}{6}} = 1.25$$

Titik potongnya adalah (1.25,1)

Region code di (5,6) adalah 1010 (TBRL), berarti T=1 dan R=1

$$R=1 \text{ berarti } yP2 = y1 + M * (xMax - x1) = 6 + \frac{8}{6} * (4 - 5) = 4.7$$

Titik potongnya adalah (4, 4.7)

$$T=1 \text{ berarti } xP2 = x1 + \frac{yMax - y1}{M} = 5 + \frac{5 - 6}{\frac{8}{6}} = 4.25$$

Titik potongnya adalah (4.25,5)

Ada titik potong pada garis P yaitu (1, 0.67) , (1.25 ,1) , (4 , 4.7) , (4.25 , 5).

Pilih titik potong yang terdapat dalam viewport yaitu (1.25 , 1) dan (4 , 4.7)

Contoh Program dalam Visual Basic

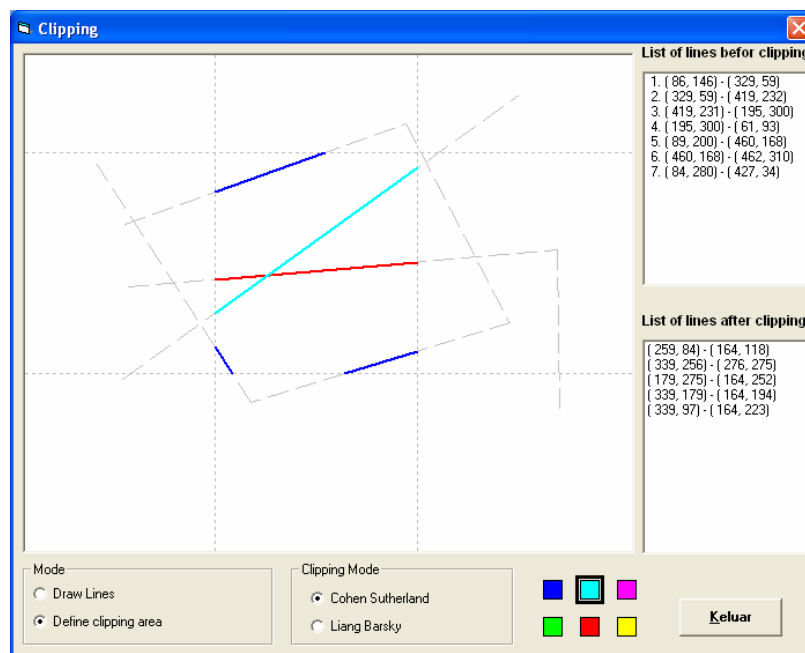
```
'Clipping routine for Cohen-Sutherland
'-----
Function clipLine(l As line, r As line)
Dim p1 As Point
Dim p2 As Point
Dim c1 As Code
Dim c2 As Code
Dim t As line
Dim done As Boolean
Dim draw As Boolean
Dim m As Variant
'MsgBox toString(l)
fixRegion r
p1 = l.p1
p2 = l.p2
done = False
draw = False
While done = False
c1 = getCode(p1, r)
c2 = getCode(p2, r)
If accept(c1, c2) Then
done = True
draw = True
ElseIf reject(c1, c2) Then
done = True
Else
110
If isInside(c1) Then
```

```

swapPts p1, p2
swapCodes c1, c2
End If
m = (p2.Y - p1.Y) / (p2.X - p1.X)
If c1.c(1) Then
'crosses left
p1.Y = p1.Y + (r.p1.X - p1.X) * m
p1.X = r.p1.X
ElseIf c1.c(2) Then
'crosses right
p1.Y = p1.Y + (r.p2.X - p1.X) * m
p1.X = r.p2.X
ElseIf c1.c(3) Then
'crosses bottom
p1.X = p1.X + (r.p1.Y - p1.Y) / m
p1.Y = r.p1.Y
ElseIf c1.c(4) Then
'crosses bottom
p1.X = p1.X + (r.p2.Y - p1.Y) / m
p1.Y = r.p2.Y
End If
End If
Wend
t.p1 = p1
t.p2 = p2
t.c = l.c
If draw Then
drawLine t, 0, Form1.Picture1
Form1.List2.AddItem (toString(t))
End If
End Function

```

Contoh: Penerapan clipping Cohen-Sutherland dengan menggunakan VB.



Gambar 4.8 Output Program Clipping

Algoritma Clipping Garis Liang-Barsky

Liang-Barsky menemukan algoritma *clipping* garis yang lebih cepat. *Clipping* yang lebih cepat dikembangkan berdasarkan persamaan parametrik dari segmen garis dapat ditulis dalam bentuk:

$$x = x_1 + u \cdot dx$$

$$y = y_1 + u \cdot dy$$

Dimana $dx = x_2 - x_1$ dan $dy = y_2 - y_1$. Dimana nilai $u \in [0,1]$. Menurut Liang dan Barsky

bentuk pertidaksamaan sebagai berikut:

$$x_{wmin} \leq x_1 + u \cdot dx \leq x_{wmax}$$

$$y_{wmin} \leq y_1 + u \cdot dy \leq y_{wmax}$$

Dengan

$$u \cdot p_k \leq q_k, k=1,2,3,4$$

Dimana parameter p dan q ditentukan sebagai berikut:

$$k = 1 \text{ (Kiri): } p_1 = -dx, q_1 = x_1 - x_{wmin}$$

$$k = 2 \text{ (Kanan): } p_2 = dx, q_2 = x_{wmax} - x_1$$

$$k = 3 \text{ (Bawah): } p_3 = -dy, q_3 = y_1 - y_{wmin}$$

$$k = 4 \text{ (Atas): } p_4 = dy, q_4 = y_{wmax} - y_1$$

Garis yang sejajar dengan salah satu batas *clipping* mempunyai $p_k = 0$ untuk nilai $k=1,2,3,4$ yaitu *left*, *right*, *bottom*, dan *top*. Untuk setiap nilai k , juga diperoleh $q_k < 0$, maka garis sepenuhnya diluar batas *clipping*. Bila $q_k \geq 0$, maka garis didalam dan sejajar batas *clipping*.

Bila $p_k < 0$, garis memotong batas *clipping* dari luar ke dalam, dan bila $p_k > 0$, garis memotong batas *clipping* dari dalam ke luar. Untuk nilai p_k yang tidak sama dengan 0, nilai u dapat diperoleh dengan

$$u = q_k / p_k$$

Untuk setiap garis, dapat dihitung nilai dan parameter u_1 dan u_2 yang menentukan posisi garis dalam bidang *clipping*. Nilai u_1 diperlihatkan dengan batas *clipping* dimana garis memotong batas *clipping* dari luar ke dalam ($p < 0$).

$$rk = q_k / p_k$$

Dengan nilai u_1 adalah nilai maksimum dari nilai 0 dan bermacam-macam nilai r . Sebaliknya nilai u_2 ditentukan dengan memeriksa batas dimana *clipping* dipotong oleh garis dari dalam keluar ($p > 0$). nilai rk dihitung untuk setiap batas *clipping*, dan nilai u_2 merupakan nilai minimum dari sekumpulan nilai yang terdiri dari 1 dan nilai r yang dihasilkan. Bila $u_1 > u_2$, maka garis sepenuhnya berada di luar *clip window* dan dapat dihilangkan; sebaliknya bila tidak ujung dari garis yang di clip dihitung dari dua nilai parameter u .

Untuk ($p_i < 0$) $t_1 = \text{Max}(rk)$

Untuk ($p_i > 0$) $t_2 = \text{Min}(rk)$

Jika $t_1 < t_2$ cari nilai ujung yang baru.

$t_1 \rightarrow (x_1 + dx * t_1, y_1 + dy * t_1)$ titik awal garis yang baru

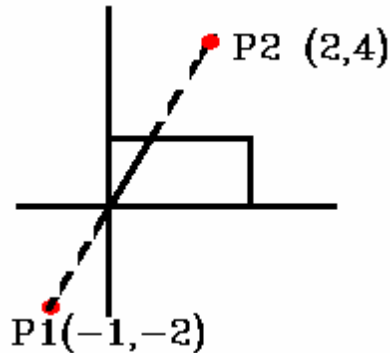
$t_2 \rightarrow (x_1 + dx * t_1, y_1 + dy * t_1)$ titik ujung garis yang baru

Algoritma Liang-Barsky lebih efisien dibandingkan dengan Cohen-Sutherland karena perhitungan titik potong dihilangkan. Algoritma adalah sebagai berikut.

1. Initialize line intersection parameter: $m_1 \leftarrow 0, m_2 \leftarrow 1$
2. Compute dx, dy
3. For each window boundary
 - Repeat
 - Compute q, p
 - If $p < 0$ (outside \rightarrow inside) Then
 - Compute $Int = q / p$
 - If $Int > m_2$ Then reject
 - Else If $Int > m_1$ Then $m_1 \leftarrow Int$
 - Else If $p > 0$ (inside \rightarrow outside) Then
 - Compute $Int \leftarrow q/p$
 - If $Int < m_1$ Then Reject
 - Else If $Int < m_2$ Then $m_2 \leftarrow Int$ Else $p = 0$
 - If $q < 0$ reject
4. If m_1 is greater Than 0 (has been modified) compute new x_1, y_1
5. If m_2 is less Than 1 (has been modified) compute new x_2, y_2

Contoh Perhitungan

Diketahui titik garis P1(-1,-2) dan P2(2,4) dan viewport $x_l = 0$, $x_r = 1$, $y_b = 0$ dan $y_t = 1$.
Tentukan endpoint baru.



Jawab:

P1(-1,-2) , P2(2,4)

$x_L = 0$, $x_R = 1$, $y_B = 0$, $y_T = 1$

$$\begin{aligned} dx &= x_2 - x_1 \\ &= 2 - (-1) = 3 \end{aligned}$$

$$\begin{aligned} p_1 &= -dx, \\ &= -3 \end{aligned}$$

$$\begin{aligned} P_2 &= dx, \\ &= 3 \end{aligned}$$

$$\begin{aligned} P_3 &= -dy, \\ &= -6 \end{aligned}$$

$$\begin{aligned} P_4 &= dy, \\ &= 6 \end{aligned}$$

$$\begin{aligned} dy &= y_2 - y_1 \\ &= 4 - (-2) = 6 \end{aligned}$$

$$\begin{aligned} q_1 &= x_1 - x_L \\ &= -1 - 0 = -1 \end{aligned}$$

$$\begin{aligned} q_2 &= x_R - x_1 \\ &= 1 - (-1) \end{aligned}$$

$$\begin{aligned} q_3 &= y_1 - y_B \\ &= -2 - 0 = -2 \end{aligned}$$

$$\begin{aligned} q_4 &= y_T - y_1 \\ &= 1 - (-2) = 3 \end{aligned}$$

$$\begin{aligned} \rightarrow q_1/p_1 &= -1/-3 \\ &= 1/3 \end{aligned}$$

$$\begin{aligned} \rightarrow q_2/p_2 &= 2/3 \\ &= 2 \end{aligned}$$

$$\begin{aligned} \rightarrow q_3/p_3 &= -2/-6 \\ &= 1/3 \end{aligned}$$

$$\begin{aligned} \rightarrow q_4/p_4 &= 3/6 \\ &= 1/2 \end{aligned}$$

Untuk ($p_i < 0$) $T_1 = \text{"Max"} (1/3, 1/3, 0) = 1/3$

Untuk ($p_i > 0$) $T_2 = \text{Min} (2/3, 1/2, 1) = 1/2$

$T_1 < T_2$

Perhitungan ujung baru

$T_1 = 1/3$

$$\begin{aligned} X_1' &= x_1 + dx * t_1 \\ &= -1 + (3 * 1/3) = -1 + 1 = 0 \end{aligned}$$

$$\begin{aligned} Y_1' &= y_1 + dy * t_1 \\ &= -2 + 6 * 1/3 = -2 + 2 = 0 \end{aligned}$$

$$(X_1', Y_1') = (0, 0)$$

$$T2 = 1/2$$

$$\begin{aligned} X2' &= x1 + dx * t2 \\ &= -1 + (3 * 1/2) = 1/2 \end{aligned}$$

$$\begin{aligned} Y2' &= y1 + dy * t2 \\ &= -2 + 6 * 1/2 = 1 \end{aligned}$$

$$(X2', Y2') = (1/2, 1)$$

Program Clipping Garis Liang-Barsky

```
Function clipLiangBarsky(region As line, p As PictureBox)
Form1.PictureBox1.Cls
Form1.List2.Clear
p.DrawWidth = 1
p.DrawStyle = 2
p.Line (0, region.p1.Y)-(p.Width, region.p1.Y), QBColor(7)
p.Line (0, region.p2.Y)-(p.Width, region.p2.Y), QBColor(7)
p.Line (region.p1.X, 0)-(region.p1.X, p.Height), QBColor(7)
p.Line (region.p2.X, 0)-(region.p2.X, p.Height), QBColor(7)
p.DrawStyle = 1
For i = 0 To n
    drawLine Garis(i), 7, p
Next i
p.DrawStyle = 0
p.DrawWidth = 2
fixRegion region
For i = 0 To (n - 1)
    clipLine2 Garis(i), region, p
Next i
End Function

'=====
' Liang-Barsky Clipping
'=====

Function clipTest(p As Double, q As Double, u1 As Double, u2 As Double)
As Boolean
Dim r As Double
clipTest = True
If p < 0 Then
    r = q / p
    If r > u2 Then
        clipTest = False
    ElseIf r > u1 Then
        u1 = r
    End If
Else
    If p > 0 Then
        r = q / p
        If r < u1 Then
            clipTest = False
        ElseIf r < u2 Then
            u2 = r
        End If
    End If
End Function
```

```

ElseIf q < 0 Then
    clipTest = False
End If
End If
End Function

```

Latihan

1. Diketahui titik awal P (1,1) dan titik akhir di Q (10,10), dengan area *clipping* xmin = 1, ymin=1, xmax= 7 dan ymax=7. Selesaikan masalah ini dengan clipping *Cohen-Sutherland*.
2. Berdasarkan soal no 1 lakukan *clipping* menggunakan algoritma Liang-Barsky dimana xl=1, xr= 7, yb = 1 dan yt = 7.

Daftar Pustaka

1. David F. Rogers, Alan J. Adams , Mathematical Elements for Computer Graphics (2nd edition), McGraw-Hill, 1989
2. Nick Symmonds, GDI+ Programming in C# and VB .NET, Apress, 2002

MODUL V TRANSFORMASI 2 DIMENSI

Setelah membaca modul ini, mahasiswa akan memiliki pengetahuan dan mampu menjelaskan (i) apa yang dimaksud dengan transformasi 2 dimensi pada objek grafik (ii) proses transformasi dasar dan melakukan proses komputasi transformasi dasar (iii) proses transformasi homogen dan mengimplementasikannya.

V. 1 Pengertian transformasi

Grafika komputer merupakan bidang yang menarik minat banyak orang. Salah sub bagian dari grafika komputer adalah pemodelan objek (*object modelling*). Dalam pemodelan objek dua dimensi (2D), didapati berbagai objek dapat dimodelkan menurut kondisi tertentu, objek yang dimodelkan itu perlu dimodifikasi. Pemodifikasian objek ini dapat dilakukan dengan melakukan berbagai operasi fungsi atau operasi transformasi geometri. Transformasi ini dapat berupa transformasi dasar ataupun gabungan dari berbagai transformasi geometri.

Contoh transformasi geometri adalah translasi, penskalaan, putaran (rotasi), balikan, *shearing* dan gabungan. Transformasi ini dikenal dengan transformasi *affine*. Pada dasarnya, transformasi ini adalah memindahkan objek tanpa merusak bentuk.

Tujuan transformasi adalah :

- Merubah atau menyesuaikan komposisi pemandangan
- Memudahkan membuat objek yang simetris
- Melihat objek dari sudut pandang yang berbeda
- Memindahkan satu atau beberapa objek dari satu tempat ke tempat lain, ini biasa dipakai untuk animasi komputer.

Proses transformasi dilakukan dengan mengalikan matriks objekl dengan matriks transformasi, sehingga menghasilkan matriks baru yang berisi koordinat objek hasil transformasi.

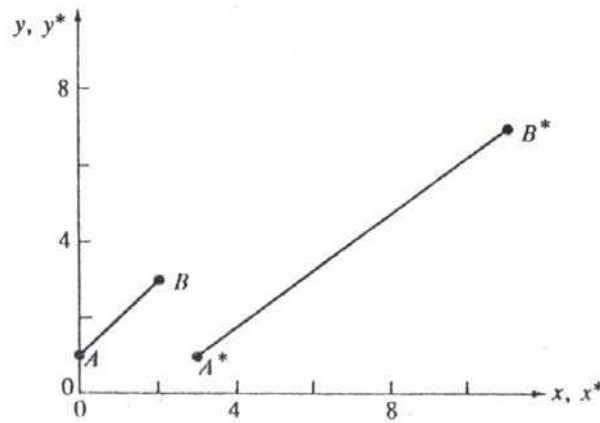
Sebagai contoh, apabila sebuah garis yang melalui titik A(0,1) dan titik B(2,3) ditransformasikan dengan matriks

$$[T] = \begin{bmatrix} 1 & 2 \\ 3 & 1 \end{bmatrix}$$

maka hasilnya adalah

$$[L][T] = \begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 3 & 1 \end{bmatrix} = \begin{bmatrix} 3 & 1 \\ 11 & 7 \end{bmatrix} [L^*]$$

Secara visual proses transformasi ini dapat dilihat pada gambar berikut:



Gambar 5.1 Ilustrasi Transformasi Sebuah Garis

V. 2 Translasi

Transformasi translasi merupakan suatu operasi yang menyebabkan perpindahan objek 2D dari satu tempat ke tempat yang lain. Perubahan ini berlaku dalam arah yang sejajar dengan sumbu X dan sumbu Y. Translasi dilakukan dengan penambahan translasi pada suatu titik koordinat dengan translation vector, yaitu (t_x, t_y) , dimana t_x adalah translasi menurut sumbu x dan t_y adalah translasi menurut sumbu y. Koordinat baru titik yang ditranslasi dapat diperoleh dengan menggunakan rumus :

$$x' = x + t_x \quad (x, y) \quad = \text{titik asal sebelum translasi}$$

$$y' = y + t_y \quad (x', y') \quad = \text{titik baru hasil translasi}$$

Translasi adalah transformasi dengan bentuk yang tetap, memindahkan objek apa adanya. Setiap titik dari objek akan ditranslasikan dengan besaran yang sama.

Dalam operasi translasi, setiap titik pada suatu entitas yang ditranslasi bergerak dalam jarak yang sama. Pergerakan tersebut dapat berlaku dalam arah sumbu X saja, atau dalam arah sumbu Y saja atau keduanya.

Translasi juga berlaku pada garis, objek atau gabungan objek 2D yang lain. Untuk hal ini, setiap titik pada garis atau objek yang ditranslasi dalam arah x dan y masing-masing sebesar t_x, t_y .

Contoh

Untuk menggambarkan translasi suatu objek berupa segitiga dengan koordinat A(10,10) B(30,10) dan C(10,30) dengan $t_x, t_y(10,20)$, tentukan koordinat yang barunya !

Jawab

$$A : x' = 10 + 10 = 20$$

$$y' = 10 + 20 = 30$$

$$A' = (20, 30)$$

$$B : x' = 30 + 10 = 40$$

$$y' = 10 + 20 = 30$$

$$B' = (40, 30)$$

$$C : x' = 10 + 10 = 20$$

$$y' = 30 + 20 = 50$$

$$C' = (20, 50)$$

V. 3 Penskalaan

Penskalaan adalah suatu operasi yang membuat suatu objek berubah ukurannya baik menjadi mengecil ataupun membesar secara seragam atau tidak seragam tergantung pada faktor penskalaan (*scaling factor*) yaitu (s_x, s_y) yang diberikan. s_x adalah faktor penskalaan menurut sumbu x dan s_y faktor penskalaan menurut sumbu y. Koordinat baru diperoleh dengan

$$x' = x + s_x \quad (x, y) \quad = \text{titik asal sebelum diskala}$$

$$y' = y + s_y \quad (x', y') \quad = \text{titik setelah diskala}$$

Nilai lebih dari 1 menyebabkan objek diperbesar, sebaliknya bila nilai lebih kecil dari 1, maka objek akan diperkecil. Bila (sx, sy) mempunyai nilai yang sama, maka skala disebut dengan *uniform scaling*.

Contoh

Untuk menggambarkan skala suatu objek berupa segitiga dengan koordinat A(10,10) B(30,10) dan C(10,30) dengan (sx, sy) (3,2), tentukan koordinat yang barunya!

Jawab:

$$A : \quad x' = 10 \cdot 3 = 30$$

$$y' = 10 \cdot 2 = 20$$

$$A' = (30, 20)$$

$$B : \quad x' = 30 \cdot 3 = 90$$

$$y' = 10 \cdot 2 = 20$$

$$B' = (90, 20)$$

$$C : \quad x' = 10 \cdot 3 = 30$$

$$y' = 30 \cdot 2 = 60$$

$$C' = (30, 60)$$

V. 4 Rotasi

Putaran adalah suatu operasi yang menyebabkan objek bergerak berputar pada titik pusat atau pada sumbu putar yang dipilih berdasarkan sudut putaran tertentu. Untuk melakukan rotasi diperlukan sudut rotasi dan *pivot point* (x_p, y_p) dimana objek akan dirotasi.

Putaran biasa dilakukan pada satu titik terhadap sesuatu sumbu tertentu misalnya sumbu x, sumbu y atau garis tertentu yang sejajar dengan sembarang sumbu tersebut. Titik acuan putaran dapat sembarang baik di titik pusat atau pada titik yang lain. Aturan dalam geometri, jika putaran dilakukan searah jarum jam, maka nilai sudutnya adalah negatif. Sebaliknya, jika dilakukan berlawanan arah dengan arah jarum jam nilai sudutnya adalah positif.

Rotasi dapat dinyatakan dengan :

$$x' = r \cos(\phi + \theta) = r \cos\phi \cos\theta - r \sin\phi \sin\theta$$

$$y' = r \sin(\phi + \theta) = r \cos\phi \sin\theta + r \sin\phi \cos\theta$$

sedangkan diketahui

$$x = r \cos \phi, y = r \sin \phi$$

lakukan substitusi, maka :

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

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

Matriks rotasi dinyatakan dengan :

$$P' = R.P$$

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

Rotasi suatu titik terhadap pivot point (xp,yp) :

$$x' = xp + (x - xp) \cos \theta - (y - yp) \sin \theta$$

$$y' = yp + (x - xp) \sin \theta + (y - yp) \cos \theta$$

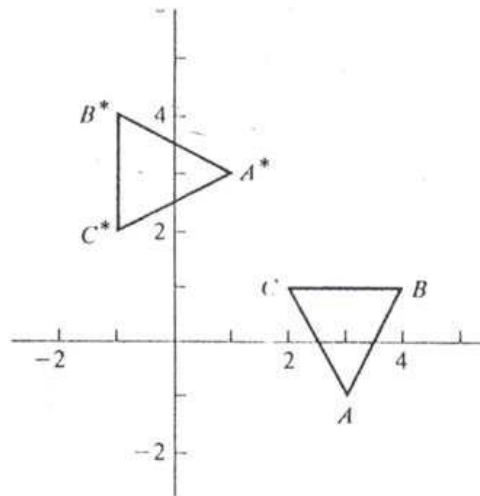
Contoh 1. Diketahui koordinat titik yang membentuk segitiga {(3, -1), (4, 1), (2, 1)}. Gambarkan objek tersebut kemudian gambarkan pula objek baru yang merupakan transformasi rotasi objek lama sebesar 90° CCW dengan pusat rotasi (0,0).

Jawab:

Matriks transformasi umum adalah

$$\begin{aligned} [T_{90}] &= \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} & [T_{180}] &= \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \\ [T_{270}] &= \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} & [T_{360}] &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \end{aligned}$$

Maka dengan mengalikan titik-titik segitiga tersebut dengan matriks transformasi yang sesuai, maka akan diperoleh hasil yang digambarkan sebagai berikut:



Gambar 4.2 Ilustrasi Rotasi

Contoh 2. Untuk menggambarkan rotasi suatu objek berupa segitiga dengan koordinat A(10,10), B(30,10) dan C(10,30) dengan sudut rotasi 300° terhadap titik pusat cartesian (10,10), dilakukan dengan menghitung koordinat hasil rotasi tiap titik satu demi satu.

Jawab:

Titik A

$$\begin{aligned} x' &= x_p + (x - x_p) \cos \theta - (y - y_p) \sin \theta \\ &= 10 + (10 - 10) * 0.9 - (10 - 10) * 0.5 = 10 \\ y' &= y_p + (x - x_p) \sin \theta + (y - y_p) \cos \theta \\ &= 10 + (10 - 10) * 0.5 - (10 - 10) * 0.9 = 10 \end{aligned}$$

Titik A'(10,10)

Titik B

$$\begin{aligned} x' &= x_p + (x - x_p) \cos \theta - (y - y_p) \sin \theta \\ &= 10 + (30 - 10) * 0.9 - (10 - 10) * 0.5 = 28 \\ y' &= y_p + (x - x_p) \sin \theta + (y - y_p) \cos \theta \\ &= 10 + (30 - 10) * 0.5 - (10 - 10) * 0.9 = 20 \end{aligned}$$

Titik B'(28,20)

Titik C

$$\begin{aligned}x' &= x_p + (x - x_p) \cos \theta - (y - y_p) \sin \theta \\&= 10 + (10 - 10) * 0.9 - (30 - 10) * 0.5 = 0\end{aligned}$$

$$\begin{aligned}y' &= y_p + (x - x_p) \sin \theta + (y - y_p) \cos \theta \\&= 10 + (10 - 10) * 0.5 - (30 - 10) * 0.9 = 28\end{aligned}$$

Titik C'(0,28)

V.5 Refleksi

Refleksi adalah transformasi yang membuat mirror (pencerminan) dari image suatu objek. Image mirror untuk refleksi 2D dibuat relatif terhadap sumbu dari refleksi dengan memutar 180° terhadap refleksi. Sumbu refleksi dapat dipilih pada bidang x,y. Refleksi terhadap garis $y=0$, yaitu sumbu x dinyatakan dengan matriks

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

Transformasi membuat nilai x sama tetapi membalikan nilai y berlawanan dengan posisi koordinat. Langkah :

- Objek diangkat
- Putar 180° terhadap sumbu x dalam 3D
- Letakkan pada bidang x,y dengan posisi berlawanan
- Refleksi terhadap sumbu y membalikan koordinat dengan nilai y tetap.

$$\begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

Refleksi terhadap sumbu x dan y sekaligus dilakukan dengan refleksi pada sumbu x terlebih dahulu, hasilnya kemudian direfleksikan terhadap sumbu y. Transformasi ini dinyatakan dengan :

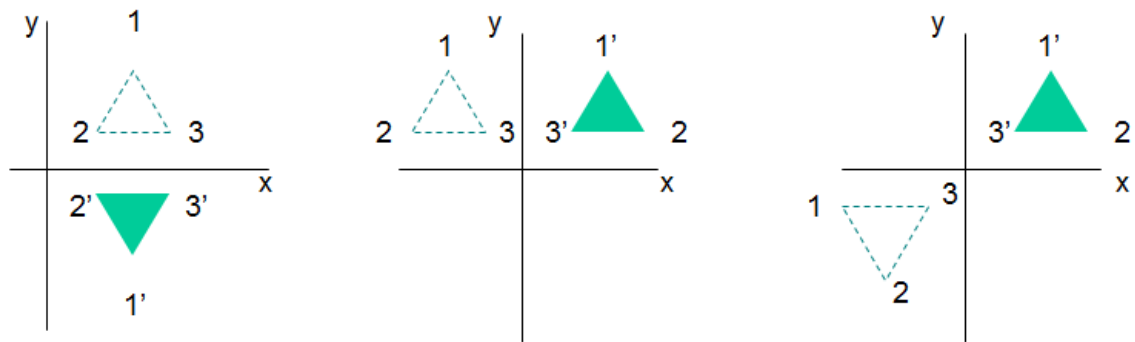
$$\begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

Refleksi ini sama dengan rotasi 180° pada bidang xy dengan koordinat menggunakan

titik pusat koordinat sebagai pivot point. Refleksi suatu objek terhadap garis $y=x$ dinyatakan dengan bentuk matriks.

$$\begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

Ilustrasi proses refleksi pada sumbu-sumbu utama digambarkan pada gambar berikut:



Gambar 4.3 Ilustrasi Refleksi

Matriks dapat diturunkan dengan menggabungkan suatu sekuen rotasi dari sumbu koordinat mereflesi matriks. Pertama-tama dilakukan rotasi searah jarum jam dengan sudut 45° yang memutar garis $y=x$ terhadap sumbu x . Kemudian objek direflesi terhadap sumbu y , setelah itu objek dan garis $y=x$ dirotasi kembali ke arah posisi semula berlawanan arah dengan jarum jam dengan sudut rotasi 90° .

Untuk mendapatkan refleksi terhadap garis $y=-x$ dapat dilakukan dengan tahap :

- Rotasi 45° searah jarum jam
- Refleksi terhadap axis y
- Rotasi 90° berlawanan arah dengan jarum jam

Dinyatakan dengan bentuk matriks

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \end{pmatrix}$$

Refleksi terhadap garis $y=mx+b$ pada bidang xy merupakan kombinasi transformasi translasi – rotasi – refleksi .

- Lakukan translasi mencapai titik perpotongan koordinat

- Rotasi ke salah satu sumbu
- Refleksi objek menurut sumbu tersebut

Latihan

Diketahui sebuah objek dengan pasangan koordinat $\{(4,1), (5,2), (4,3)\}$.

- Refleksikan pada cermin yang terletak pada sumbu x
- Refleksikan pada garis $y=-x$.

V. 6 *Shear*

Shear adalah bentuk transformasi yang membuat distorsi dari bentuk suatu objek, seperti menggeser sisi tertentu. Terdapat dua macam *shear* yaitu *shear* terhadap sumbu x dan *shear* terhadap sumbu y.

Shear terhadap sumbu x

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \end{pmatrix}$$

Dengan koordinat transformasi

$$x' = x + shx.y \qquad y' = y$$

Parameter shx dinyatakan dengan sembarang bilangan. Posisi kemudian digeser menurut arah *horizontal*.

Shear terhadap sumbu y

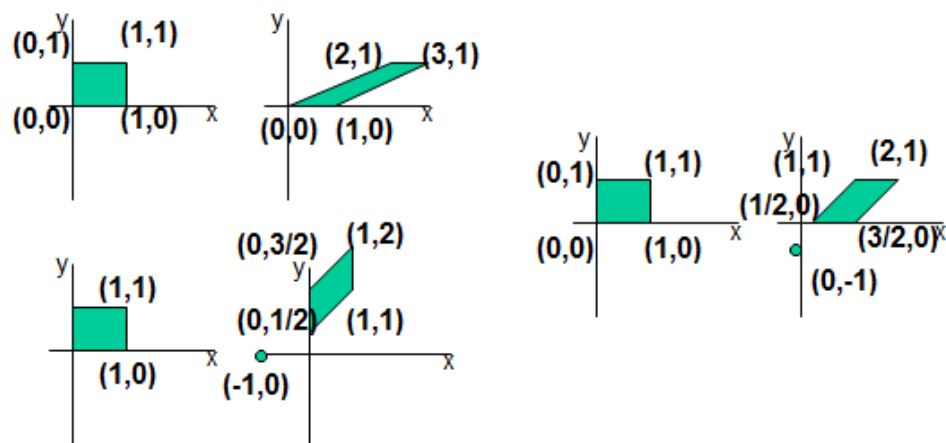
$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \end{pmatrix}$$

Dengan koordinat transformasi

$$x' = x \qquad y' = shy.x + y$$

Parameter sh_y dinyatakan dengan sembarang bilangan. Posisi koordinat kemudian menurut arah vertikal.

Gambar di bawah mengilustrasikan proses shearing.



Gambar 4.4 Ilustrasi Proses *Shearing*

Latihan

Diketahui sebuah bidang segiempat dengan koordinat A(3,1), B(10,1), C(3,5) dan D(10,5). Tentukan koordinat baru dari bidang tersebut dengan melakukan translasi dengan faktor translasi (4,3)

1. Lakukan penskalaan dengan faktor skala (3,2)
2. Dari hasil (1) lakukan rotasi terhadap titik pusat (A) dengan sudut rotasi 30° .
3. Transformasi shear dengan nilai $sh_x = 2$ dengan koordinat A(0,0), B(1,0), C(1,1), dan D(0,1)
4. Transformasi shear dengan nilai $sh_y = 2$ dengan koordinat A(0,0), B(1,0), C(1,1), dan D(0,1)

V. 7 Transformasi Homogen

Transformasi homogen merupakan transformasi yang memberikan cakupan proses transformasi secara umum. Dalam dunia nyata dimana objek nyata merupakan elemen yang kompleks dan memiliki koordinat masing-masing, maka peran transformasi homogen sangat diperlukan untuk menyelesaikan berbagai permasalahan yang ada.

Beberapa hal yang perlu diperhatikan dalam hal implemengtasi transformasi homogen adalah sebagai berikut:

- Origin bersifat INVARIANT. Koordinatnya tidak akan pernah berubah. Jika ditransformasikan, akan tetap di (0,0).
- Dalam kondisi nyata, origin tidak harus selalu absolut di (0,0). Untuk itu digunakan koordinat homogen
- Koordinat homogen memetakan titik (0,0) ke posisi lain. Untuk itu ada elemen tambahan pada matriks transformasi

Untuk itu maka didefinisikan Matriks Transformasi Umum (MTU) sebagai berikut:

$$[T] = \begin{bmatrix} a & b & p \\ c & d & q \\ m & n & s \end{bmatrix}$$

Dimana a, b, c, d merupakan elemen untuk skala, rotasi, refleksi dan *shearing*; m, n merupakan elemen untuk translasi; s adalah elemen untuk *overal scaling*; dan p, q adalah elemen untuk proyeksi.

Rotasi pada Sumbu Sembarang

Jika sebuah objek dirotasikan sebesar θ° dengan pusat rotasi (m, n), maka langkah-langkah yang harus dilakukan adalah

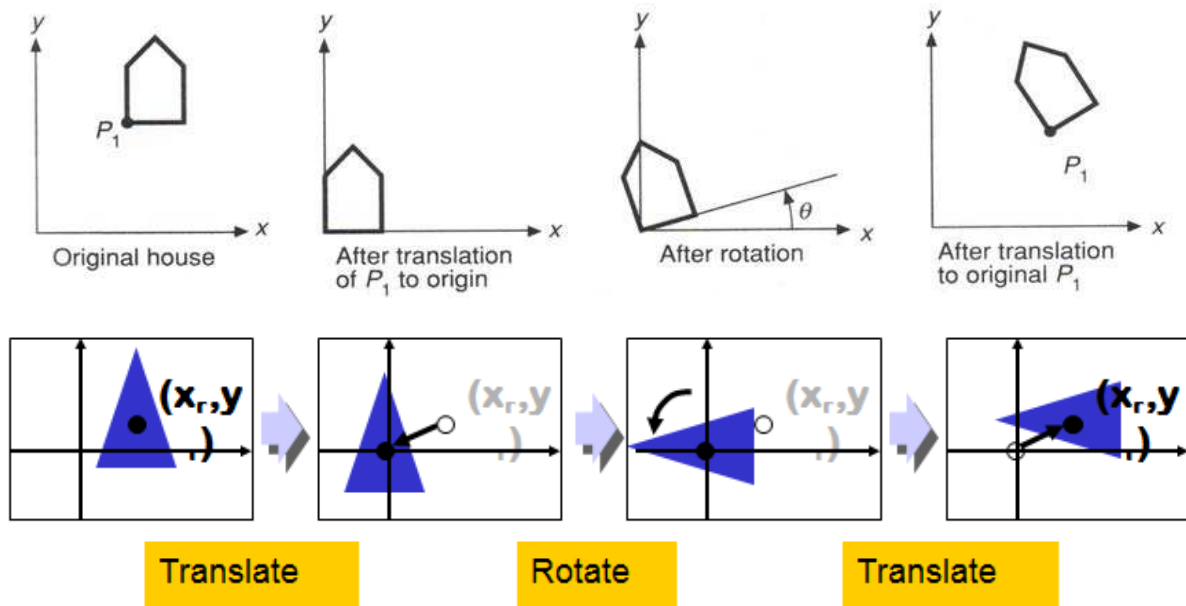
- a. Translasikan pusat rotasi ke (0, 0); karena yang kita ketahui hanyalah rumus rotasi pada origin
- b. Lakukan rotasi sebesar yang diinginkan
- c. Re-translasi pusat rotasi ke posisi semula

Dengan demikian matriks transformasinya menjadi

$$[T] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -m & -n & 1 \end{bmatrix} \begin{bmatrix} \cos \theta & \sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ m & n & 1 \end{bmatrix}$$

$$\begin{bmatrix} x^* & y^* & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} [T]$$

Proses ini diilustrasikan sebagai berikut:



Gambar 4.5 Ilustrasi Rotasi Pada Sumbu Sembarang

Refleksi pada Garis Sembarang

Jika sebuah objek direfleksikan pada sebuah garis maka langkah-langkah yang harus dilakukan adalah

- Translasikan cermin sedemikian rupa sehingga menyentuh titik origin
- Rotasikan cermin sehingga berimpit dengan salah satu sumbu utama
- Refleksikan objek
- Re-rotasi
- Re-translasi

Jadi MTU terdiri dari 5 buah matriks transformasi sebagai berikut:

$$[T] = [T_r][Rot][R][Rot^{-1}][T_r^{-1}]$$

Latihan:

- Diketahui sebuah objek dengan koordinat

$\{(0,0), (2,2), (2,1), (6,1), (6,-1), (2, -1), (-2,-2)\}$

- Rotasikan objek sebesar 45° CCW dengan pusat rotasi pada $(9, 4)$
- Rotasikan objek sebesar 30° CW dengan pusat rotasi pada $(-3,5)$

Dari operasi transformasi

- Gambarkan objek asli
- Tentukan MTU
- Tentukan Koordinat Objek Baru
- Gambarkan objek hasil transformasi

- Diketahui sebuah objek dengan koordinat

$\{(0, 0), (1, -2), (3, 3), (2, 3), (1, 1), (0, 2), (-1, 1), (-2, 3), (-3, 3), (-1, -2), (0, 0)\}$.

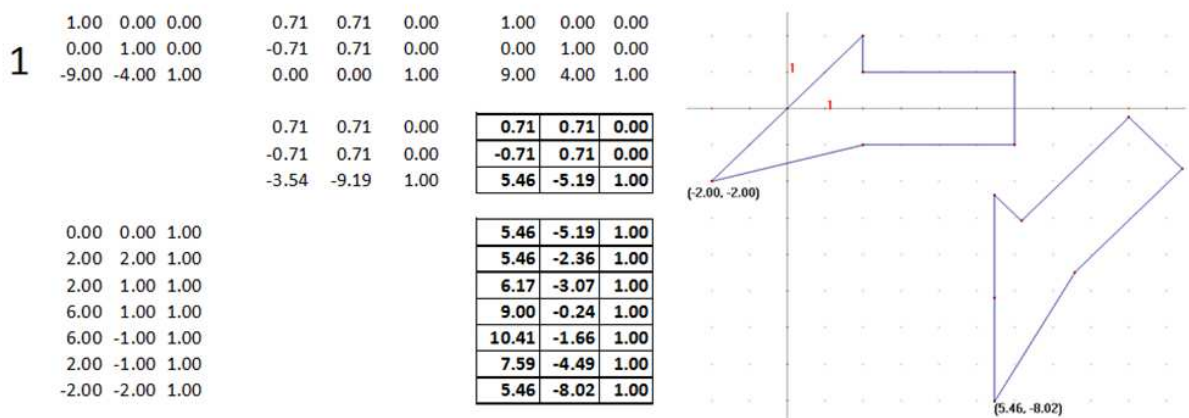
- Refleksikan objek di atas pada cermin yang berimpit dengan garis $y = -x+9$.
- Refleksikan objek di atas pada cermin yang berimpit dengan garis $y = x+9$.

Dari operasi transformasi

- Gambarkan objek asli
- Tentukan MTU
- Tentukan Koordinat Objek Baru
- Gambarkan objek baru hasil transformasi

Jawaban

1.a



1.b

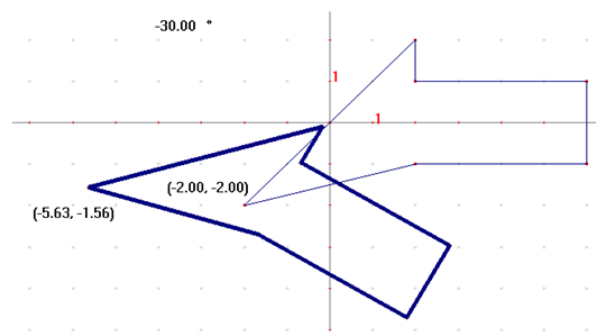
1

1.00	0.00	0.00	0.87	-0.50	0.00	1.00	0.00	0.00
0.00	1.00	0.00	0.50	0.87	0.00	0.00	1.00	0.00
3.00	-5.00	1.00	0.00	0.00	1.00	-3.00	5.00	1.00

0.87	-0.50	0.00	0.87	-0.50	0.00
0.50	0.87	0.00	0.50	0.87	0.00
0.10	-5.83	1.00	-2.90	-0.83	1.00

0.00	0.00	1.00
2.00	2.00	1.00
2.00	1.00	1.00
6.00	1.00	1.00
6.00	-1.00	1.00
2.00	-1.00	1.00
-2.00	-2.00	1.00

0.87	-0.50	0.00
0.50	0.87	0.00
-2.90	-0.83	1.00
-2.90	-0.83	1.00
-0.17	-0.10	1.00
-0.67	-0.96	1.00
2.79	-2.96	1.00
1.79	-4.70	1.00
-1.67	-2.70	1.00
-5.63	-1.56	1.00



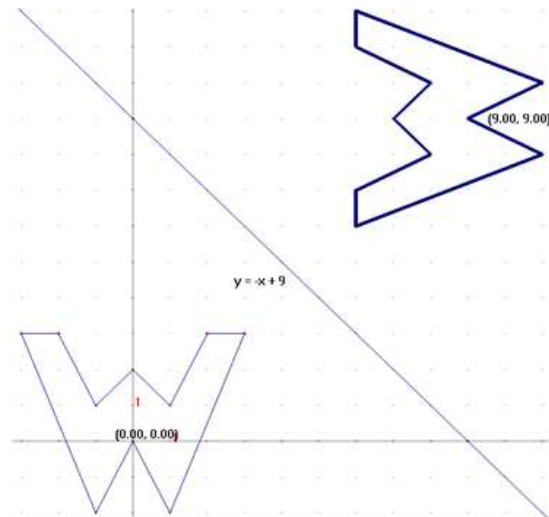
2.a

2

y=-x+9				-0.79	-45.00	135.00									
1.00	0.00	0.00		-0.71	-0.71	0.00	1.00	0.00	0.00	-0.71	0.71	0.00	1.00	0.00	0.00
0.00	1.00	0.00		0.71	-0.71	0.00	0.00	-1.00	0.00	-0.71	-0.71	0.00	0.00	1.00	0.00
0.00	-9.00	1.00		0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	9.00	1.00

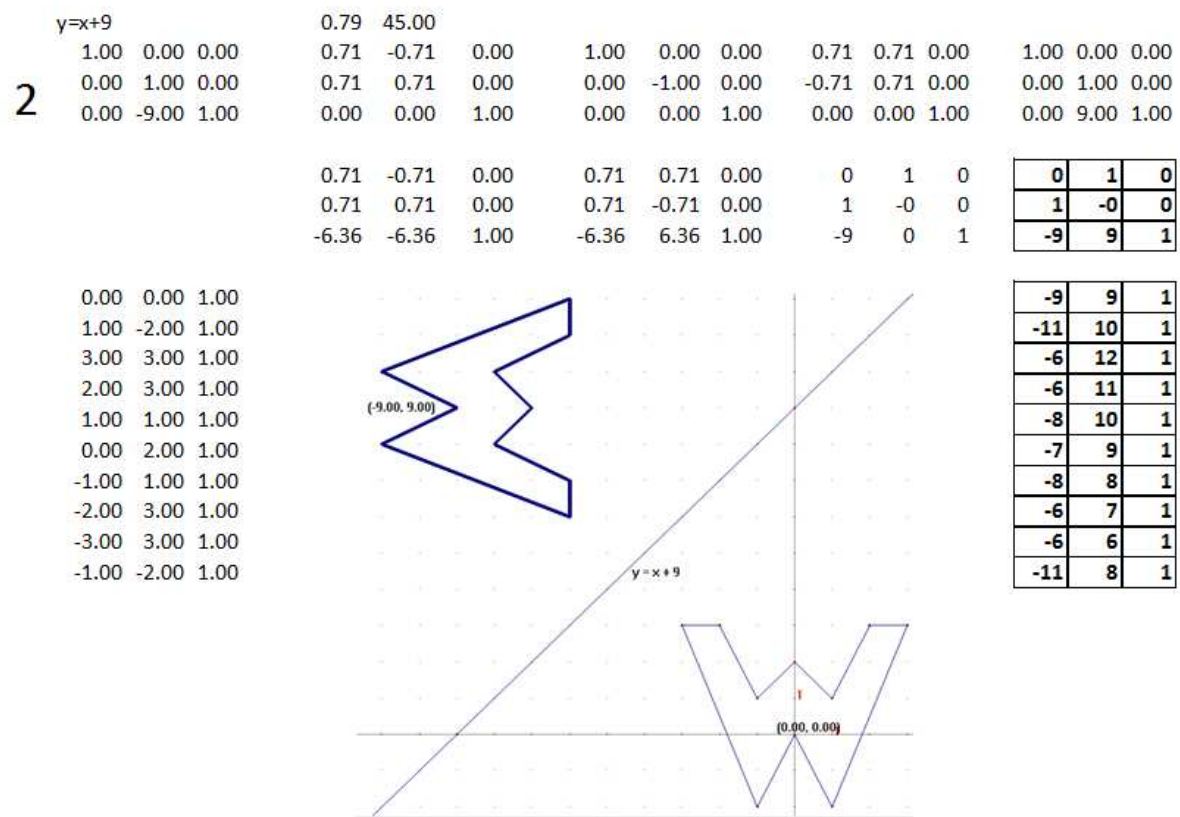
-0.71	-0.71	0.00	-0.71	0.71	0.00	-0	-1	0	-0	-1	0
0.71	-0.71	0.00	0.71	0.71	0.00	-1	0	0	-1	0	0
-6.36	6.36	1.00	-6.36	-6.36	1.00	9	-0	1	9	9	1

0.00	0.00	1.00
1.00	-2.00	1.00
3.00	3.00	1.00
2.00	3.00	1.00
1.00	1.00	1.00
0.00	2.00	1.00
-1.00	1.00	1.00
-2.00	3.00	1.00
-3.00	3.00	1.00
-1.00	-2.00	1.00



9	9	1
11	8	1
6	6	1
6	7	1
8	8	1
7	9	1
8	10	1
6	11	1
6	12	1
11	10	1

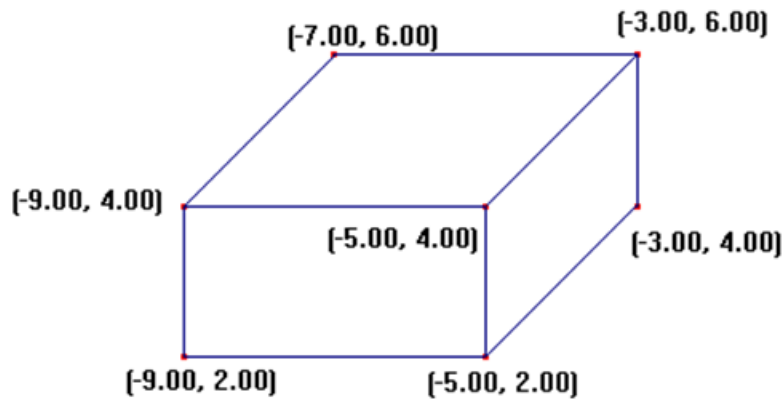
2.b



Latihan

- Jelaskan istilah-istilah dalam Grafika Komputer sebagai berikut:
 - Ukuran layar monitor
 - Resolusi layar
 - Dot Pitch
 - Interlace/Non-Interlace
 - Flat/Non Flat
 - RGB
 - LCD
 - CGA, EGA, VGA, XGA
- [10] Turunkan matriks transformasi umum (MTU) untuk rotasi dengan pusat rotasi pada sebuah titik sembarang (x, y) dan sudut rotasi sebesar θ° searah jarum jam (*clock wise*).

3. Diketahui sebuah objek grafis seperti di bawah ini.



Tentukan :

- Matriks Transformasi Umum jika objek di atas dirotasikan sebesar $\text{PHI}/4$ radian Clock Wise dengan pusat rotasi $(-1, -2)$
- Tentukan koordinat objek baru hasil transformasi dengan MTU pada soal a
- Gambarkan objek baru hasil transformasinya

4. **Tentukan:**

- Matriks Transformasi Umum jika objek pada soal 3 direfleksikan pada cermin yang berimpit dengan garis $y = 2x/5 + 4$
- Tentukan koordinat objek baru hasil transformasi dengan MTU pada soal a
- Gambarkan objek baru hasil transformasinya

5. Tuliskan algoritma perkalian matriks 2 dimensi menggunakan bahasa C atau Pascal !

Daftar Pustaka

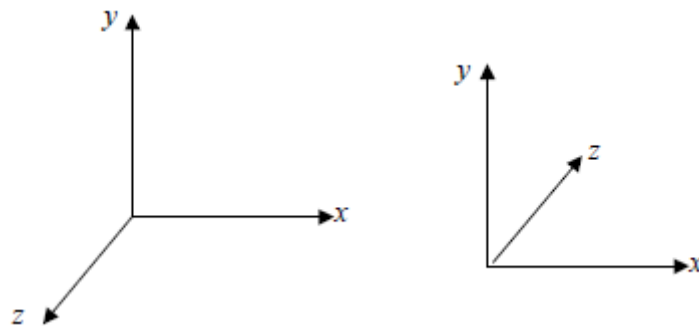
- David F. Rogers, Alan J. Adams , Mathematical Elements for Computer Graphics (2nd edition), McGraw-Hill, 1989
- John F. Hughes, Andries Van Dam, Morgan Mcguire, David F. Sklar, James D. Foley, Steven K. Feiner, Kurt Akeley, Computer Graphics: Principles and Practice (3rd edition), Addison-Wesley, 2014
- Ollie Cornes, Jay Glynn, Burton Harvey, Craig McQueen, Jerod Moemeka, Christian Nagel, Simon Robinson, Morgan Skinner, Karli Watson, Professional C# - Graphics with GDI+, Wrox, 2001

MODUL VI TRANSFORMASI 3 DIMENSI

Setelah membaca modul ini, mahasiswa akan memiliki pengetahuan dan mampu menjelaskan (i) apa yang dimaksud dengan transformasi 3 dimensi pada objek grafik (ii) proses transformasi dasar dan melakukan proses komputasi transformasi dasar (iii) Konsep sistem koordinat berganda dan transformasi majemuk secara global

VI.1 Pengertian Transformasi 3D

Dalam ruang dua dimensi suatu titik akan berada pada suatu posisi yang dinyatakan oleh dua sumbu. Umumnya kita sebut sumbu x dan sumbu y . Dalam ruang tiga dimensi terdapat sumbu ketiga yang biasanya kita sebut sumbu z . Terdapat dua konvensi dalam merepresentasikan suatu titik: kaidah tangan kanan dan kaidah tangan kiri. Dalam kaidah tangan kanan jika sumbu x positif mengarah ke kanan dan sumbu y positif mengarah ke atas maka sumbu z positif mengarah mendekati kita sementara dalam kaidah tangan kiri sumbu z positif mengarah menjauhi kita.



Gambar 6.1 Sumbu Koordinat 3 Dimensi

Transformasi-transformasi geometris yang dasar di ruang tiga dimensi sama halnya dengan di ruang dua dimensi kecuali

- rotasi kita perlu membedakan rotasi terhadap masing-masing sumbu
- refleksi adalah terhadap bidang-bidang xy , yz , atau zx , dan
- *shear* adalah terhadap dua sumbu, misalnya x dan z .

Demikian pula kita dapat memanfaatkan sistem koordinat homogen untuk suatu titik (x, y, z) dalam ruang tiga dimensi direpresentasikan sebagai matriks kolom $\begin{bmatrix} x & y & z & 1 \end{bmatrix}$. Selanjutnya setiap transformasi dasar dapat dinyatakan sebagai perkalian matriks:

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

VI.2 Operasi Dasar Transformasi 3D

Matriks transformasi umum 3D dinyatakan sebagai matriks 4x4 sebagai berikut:

$$\begin{bmatrix} a & b & c & p \\ d & e & f & q \\ g & h & i & r \\ l & m & n & s \end{bmatrix}$$

Dimana a, b, c, d, e, f, g, h, i adalah elemen yang berpengaruh terhadap transformasi linier; p, q, r adalah elemen yang untuk proyeksi dan perspektif l, m, n adalah elemen untuk translasi pada sumbu x, y dan z s adalah elemen untuk overall scaling.

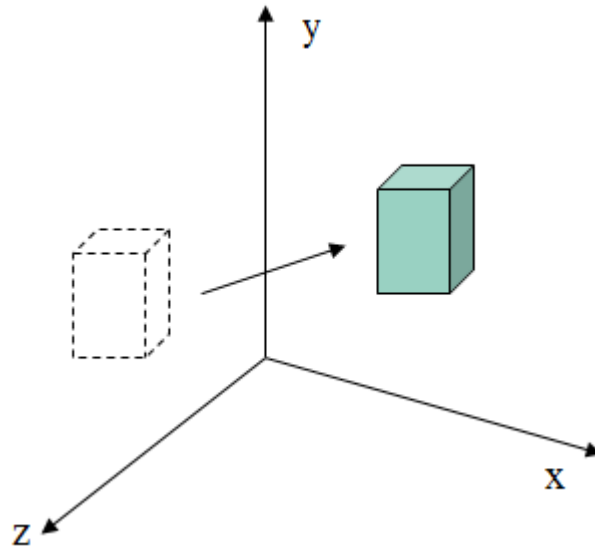
Translasi

Transformasi translasi 3 dimensi dinyatakan sebagai

$$x' = x + t_x, \quad y' = y + t_y, \quad z' = z + t_z$$

Komposisi perkalian matriksnya adalah sebagai berikut:

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



Gambar 6.2 Translasi 3 Dimensi

Penskalaan

Dalam domain 3 dimensi, terdapat dua jenis penskalaan atau *scaling* yaitu *local scaling* dan *overall scaling* atau *global scaling*. Dalam *local scaling* penskalaan bisa dilakukan terhadap salah satu atau semua sumbu (x, y dan z). Sedangkan dalam *overall scaling* dilakukan secara seragam untuk semua sumbu.

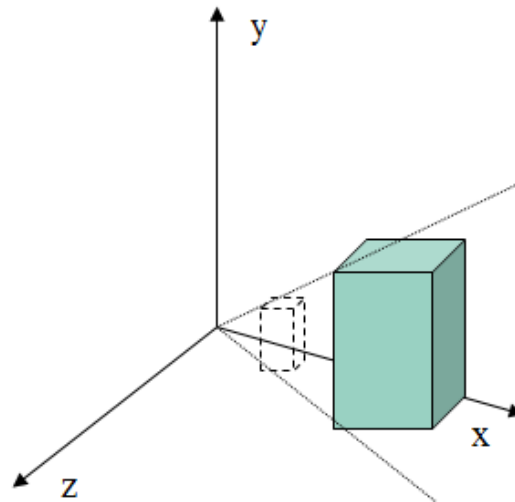
Elemen matriks transformasi umum yang mempengaruhi *local scaling* adalah elemen diagonal yaitu elemen a, e dan i. Sedangkan elemen yang mempengaruhi *overall scaling* adalah elemen s. Formulasi transformasi untuk *global scaling* dan *local scaling* adalah sebagai berikut:

$$[x' \ y' \ z' \ 1] = [x \ y \ z \ 1] \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \frac{1}{S_g} \end{bmatrix}$$

$$[x' \ y' \ z' \ 1] = [x \ y \ z \ 1] \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Operasi transformasi *scaling* pada elemen matriksnya adalah operasi perkalian sebagai berikut.

$$x' = x \cdot s_x, \quad y' = y \cdot s_y, \quad z' = z \cdot s_z$$



Gambar 6.3 Skala 3 Dimensi

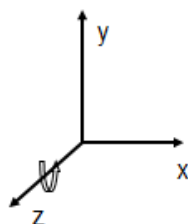
Rotasi

Berbeda dengan transformasi rotasi 2D dimana yang menjadi sumbu adalah sebuah titik, dalam transformasi 3D, yang menjadi sumbu adalah garis atau sumbu.

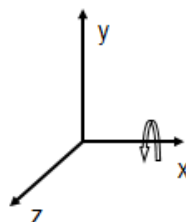
Formula untuk rotasi dapat dilihat pada tabel di bawah ini.

■ Z-Axis Rotation ■ X-Axis Rotation ■ Y-Axis Rotation

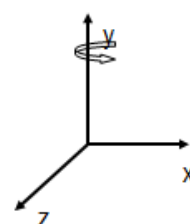
$$\begin{bmatrix} \cos \theta & \sin \theta & 0 & 0 \\ -\sin \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \theta & \sin \theta & 0 \\ 0 & -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

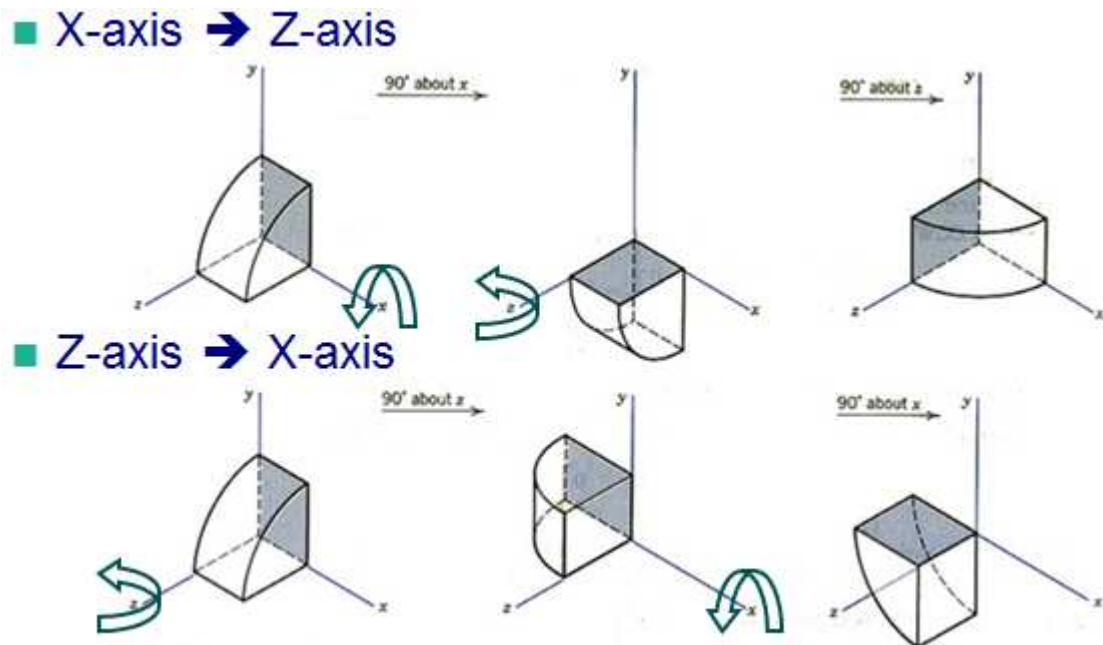


$$\begin{bmatrix} \cos \theta & 0 & -\sin \theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin \theta & 0 & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Gambar 6.4 Rotasi 3 Dimensi

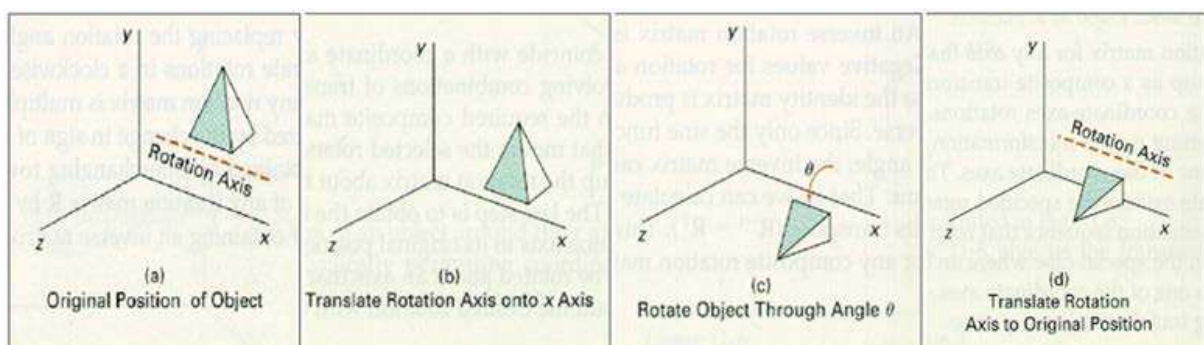
Dalam rotasi 3D perlu diperhatikan bahwa urutan proses rotasi tidaklah komutatif. Gambar berikut mengilustrasikan keadaan ini. Digambarkan bahwa hasil akhir antara urutan proses rotasi pada sumbu x dilanjutkan rotasi sumbu z TIDAK SAMA HASILNYA dengan rotasi pada sumbu z dilanjutkan dengan rotasi pada sumbu x.



Gambar 6.5 Rotasi 3 Dimensi Tidak Komutatif

Rotasi pada Sumbu yang Paralel dengan Sumbu Utama

Langkah-langkah yang harus dilalui adalah (i) translasikan objek sedemikian rupa sehingga berimpit dengan salah satu sumbu utama (ii) lakukan rotasi (iii) lakukan re-translasi. Ilustrasinya ditunjukkan pada gambar di bawah ini.

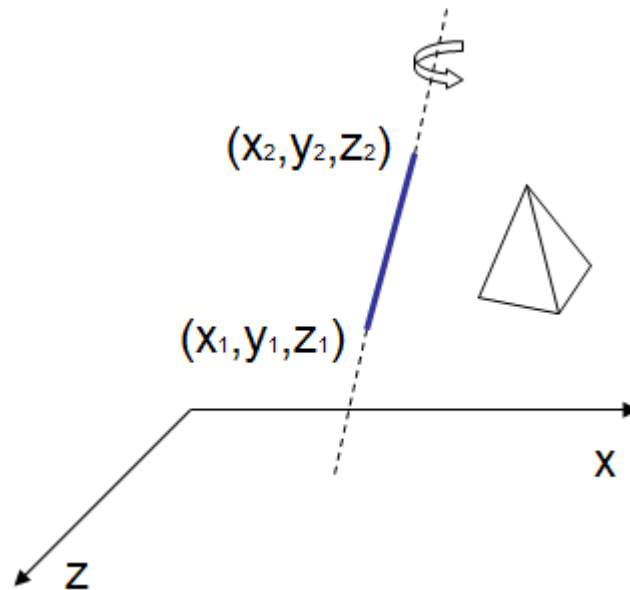


Gambar 6.6 Rotasi Pada Sumbu Sembarang yang Sejajar Sumbu Utama

Rotasi pada Sumbu Sembarang

Proses ini cukup kompleks dan dilakukan sebanyak 5 langkah. Formulasi perkalian matriksnya adalah sebagai berikut:

$$[T_R]_{ARB} = [T_{TR}]^{-1} [T_R]_x^{-\alpha} [T_R]_y^{-\phi} [T_R]_z^{\theta} [T_R]_y^{\phi} [T_R]_x^{\alpha} [T_{TR}]$$

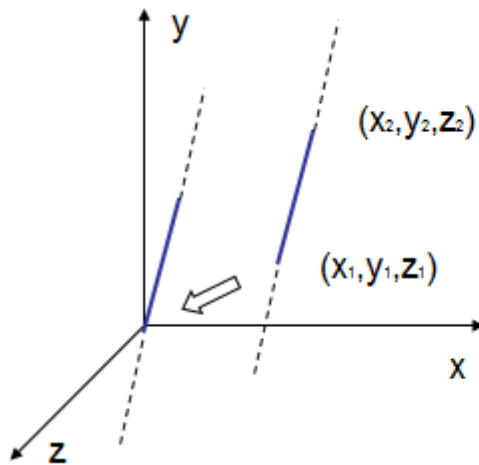


Gambar 6.7 Rotasi 3 Dimensi Pada Sumbu Sembarang

Langkah-langkah yang dilakukan adalah

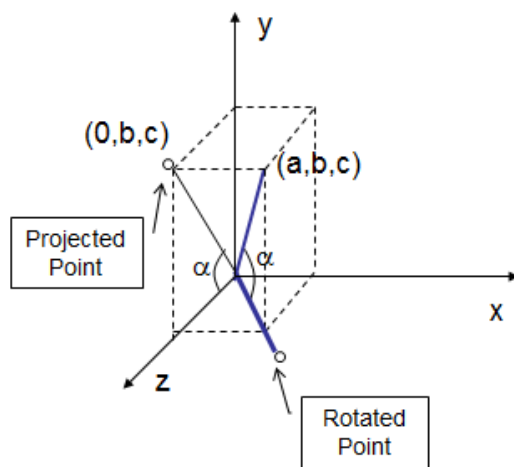
1. Translasikan (x_1, y_1, z_1) ke origin
2. Rotasikan (x'_2, y'_2, z'_2) pada sumbu Z
3. Rotasikan objek pada sumbu Z
4. Re-rotasi sumbu ke orientasi semula
5. Re-translasi

Langkah 1: Translasi



$$T_{TR} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -x_1 & -y_1 & -z_1 & 1 \end{bmatrix}$$

Langkah 2: Rotasi

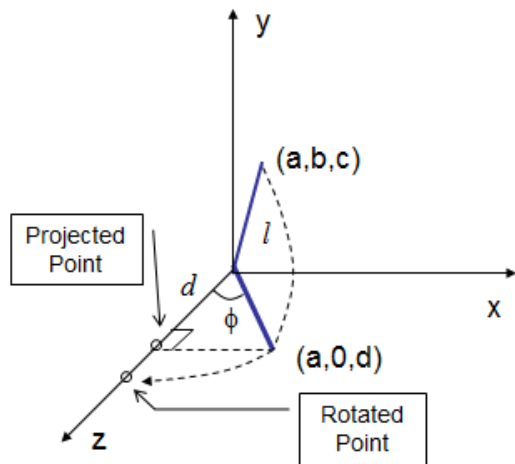


$$\sin \alpha = \frac{b}{\sqrt{b^2 + c^2}} = \frac{b}{d}$$

$$\cos \alpha = \frac{c}{\sqrt{b^2 + c^2}} = \frac{c}{d}$$

$$[T_R]_x^a = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \alpha & -\sin \alpha & 0 \\ 0 & \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c/d & -b/d & 0 \\ 0 & b/d & c/d & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Langkah 3: Rotasi



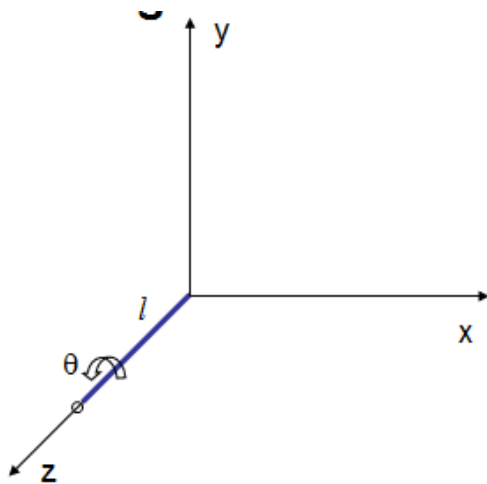
$$\sin \phi = \frac{a}{l}, \quad \cos \phi = \frac{d}{l}$$

$$l^2 = a^2 + b^2 + c^2 = a^2 + d^2$$

$$d = \sqrt{b^2 + c^2}$$

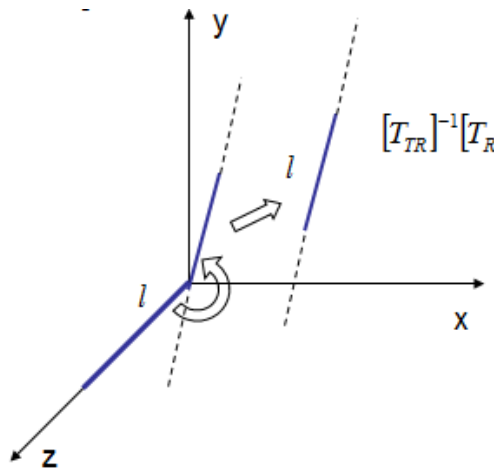
$$[T_R]_y^\phi = \begin{bmatrix} \cos \phi & 0 & \sin \phi & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \phi & 0 & \cos \phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} d/l & 0 & a/l & 0 \\ 0 & 1 & 0 & 0 \\ -a/l & 0 & d/l & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Langkah 4: Re-Rotasi



$$[T_R]_x^\theta = \begin{bmatrix} \cos \theta & \sin \theta & 0 & 0 \\ -\sin \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

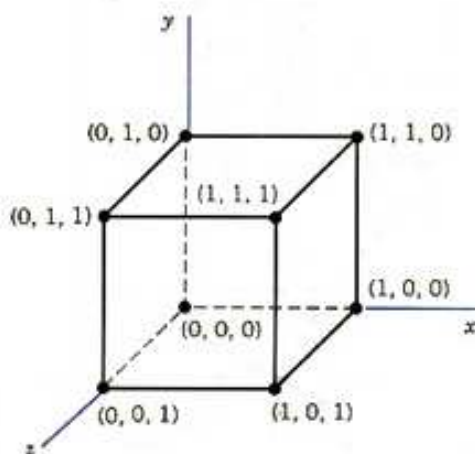
Langkah 5: Re-translasi



$$[T_{TR}]^{-1} [T_R]_x^\alpha [T_R]_y^\phi = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ x_1 & y_1 & z_1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \alpha & \sin \alpha & 0 \\ 0 & -\sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \phi & 0 & -\sin \phi & 0 \\ 0 & 1 & 0 & 0 \\ \sin \phi & 0 & \cos \phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

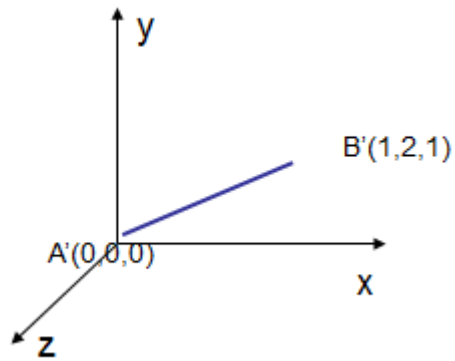
Contoh

Tentukan koordinat baru dari kuus yang dirotasikan 90° pada sebuah sumbu putar yang dibentuk oleh garis yang melalui A(2,1,0) dan B(3,3,1).



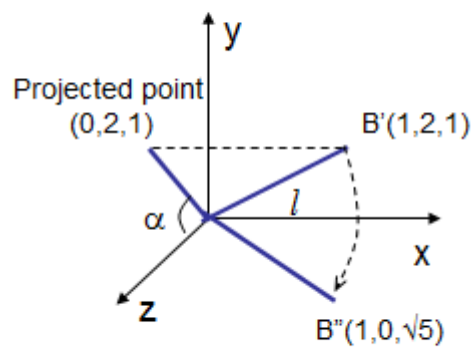
Penyelesaian:

Langkah 1: Translasi ke origin



$$[T_{TR}] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -2 & -1 & 0 & 1 \end{bmatrix}$$

Langkah 2: Rotasi



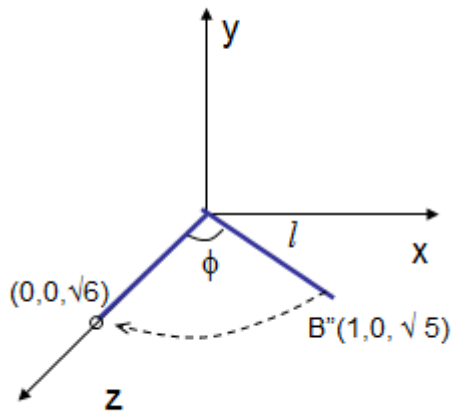
$$\sin \alpha = \frac{2}{\sqrt{2^2 + 1^2}} = \frac{2}{\sqrt{5}} = \frac{2\sqrt{5}}{5}$$

$$\cos \alpha = \frac{1}{\sqrt{5}} = \frac{\sqrt{5}}{5}$$

$$l = \sqrt{1^2 + 2^2 + 1^2} = \sqrt{6}$$

$$[T_R]_x^a = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{\sqrt{5}}{5} & -\frac{2\sqrt{5}}{5} & 0 \\ 0 & \frac{2\sqrt{5}}{5} & \frac{\sqrt{5}}{5} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Langkah 3: Rotasi Inti



$$\sin \phi = \frac{1}{\sqrt{6}} = \frac{\sqrt{6}}{6}$$

$$\cos \phi = \frac{\sqrt{5}}{\sqrt{6}} = \frac{\sqrt{30}}{6}$$

$$[T_R]_y^\phi = \begin{bmatrix} \frac{\sqrt{30}}{6} & 0 & -\frac{\sqrt{6}}{6} & 0 \\ 0 & 1 & 0 & 0 \\ \frac{\sqrt{6}}{6} & 0 & \frac{\sqrt{30}}{6} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Langkah 4 dan 5: Lakukan Re-rotasi dan Re-translasi

$$[T_R]_{ARB} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -2 & -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{\sqrt{5}}{5} & -\frac{2\sqrt{5}}{5} & 0 \\ 0 & \frac{2\sqrt{5}}{5} & \frac{\sqrt{5}}{5} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{\sqrt{30}}{6} & 0 & -\frac{\sqrt{6}}{6} & 0 \\ 0 & 1 & 0 & 0 \\ \frac{\sqrt{6}}{6} & 0 & \frac{\sqrt{30}}{6} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{\sqrt{30}}{6} & 0 & \frac{\sqrt{6}}{6} & 0 \\ 0 & 1 & 0 & 0 \\ -\frac{\sqrt{6}}{6} & 0 & \frac{\sqrt{30}}{6} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{\sqrt{5}}{5} & \frac{2\sqrt{5}}{5} & 0 \\ 0 & -\frac{2\sqrt{5}}{5} & \frac{\sqrt{5}}{5} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 2 & 1 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0.166 & -0.075 & 0.983 & 1.742 \\ 0.742 & 0.667 & 0.075 & -1.151 \\ -0.650 & 0.741 & 0.167 & 0.560 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

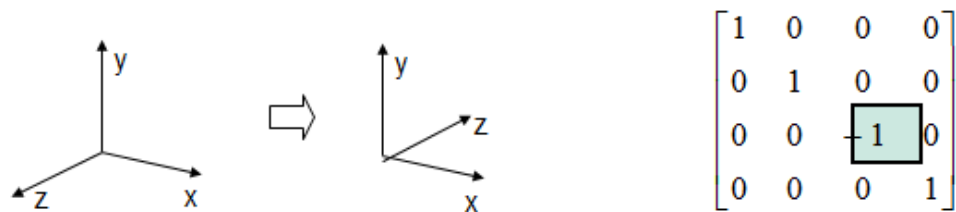
$$[P^*] = [T_R]_{ARB} \cdot [P]$$

$$[P^*] = \begin{bmatrix} 0.166 & -0.075 & 0.983 & 1.742 \\ 0.742 & 0.667 & 0.075 & -1.151 \\ -0.650 & 0.741 & 0.167 & 0.560 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 2.650 & 1.667 & 1.834 & 2.816 & 2.725 & 1.742 & 1.909 & 2.891 \\ -0.558 & -0.484 & 0.258 & 0.184 & -1.225 & -1.151 & -0.409 & -0.483 \\ 1.467 & 1.301 & 0.650 & 0.817 & 0.726 & 0.560 & -0.091 & 0.076 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Refleksi

■ Reflection the xy Plane

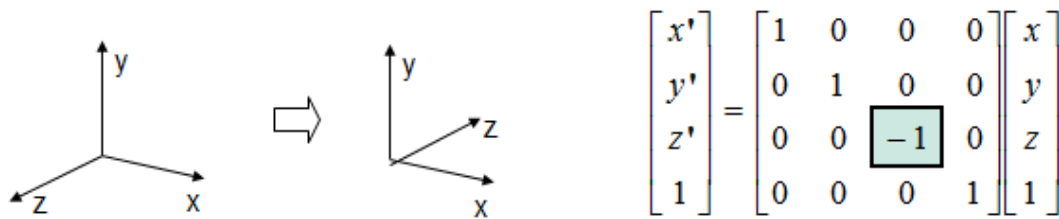


■ Reflection the yz Plane

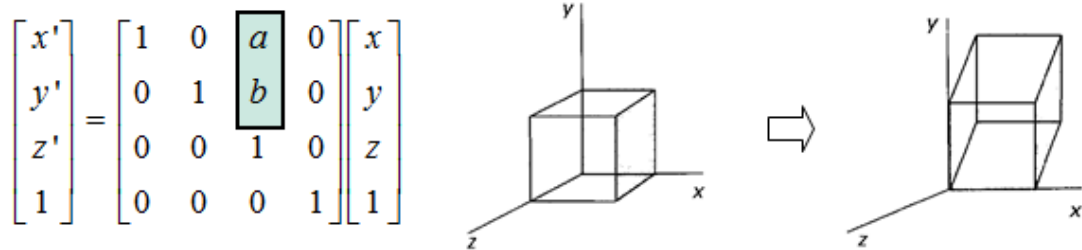
■ Reflection the xz Plane

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

■ Reflection Relative to the xy Plane



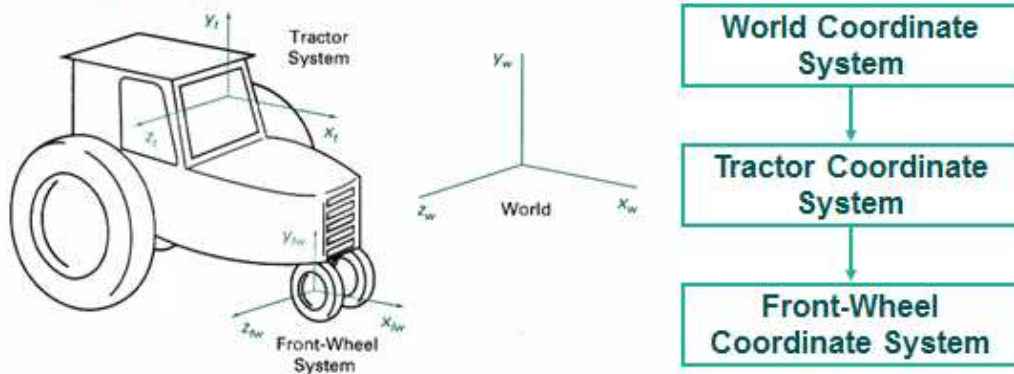
■ Z-axis Shear



VI.3 Sistem Koordinat Berganda

■ Multiple Coordinate System

■ Hierarchical Modeling

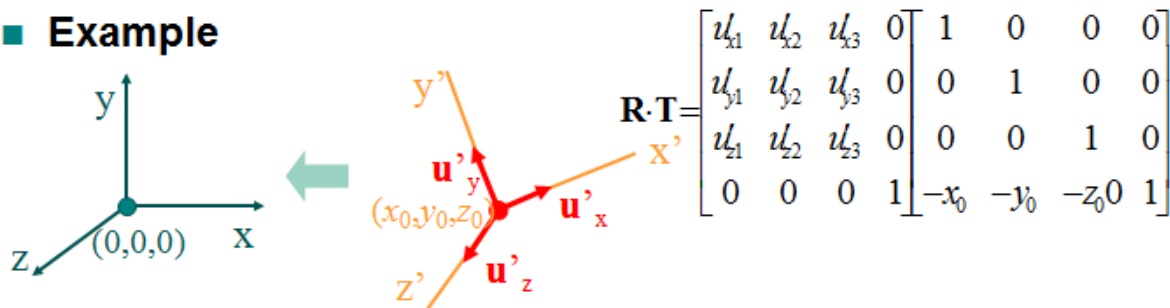


- As tractor moves, **tractor coordinate system** and **front-wheel coordinate system** move in world coordinate system
- **front wheels** rotate in wheel coordinate system
- When tractor turns, **wheel coordinate system** rotates in tractor system

■ Transformation of an Object Description from One Coordinate System to Another

- Set up a **translation** that brings the new coordinate origin to the position of the other coordinate origin
- **Rotations** that corresponding coordinate axes
- **Scaling** transformation, if different scales are used in the two coordinates systems

■ Example



Latihan Penskalaan

Diketahui sebuah objek P dengan koordinat sebagai berikut : $\{(0,0,1,1), (2,0,1,1), (2,3,1,1), (0,3,1,1), (0,0,0,1), (2,0,0,1), (2,3,0,1), (0,3,0,1)\}$.

1. Gambarkan objek tersebut !
2. Lakukan local scaling terhadap objek P dengan faktor skala $xyz = \{1/2, 1/3 \text{ dan } 1\}$.
 - a. Tentukan koordinat baru
 - b. Gambarkan hasilnya
3. Lakukan *overal scaling* terhadap objek asli dengan faktor 2.
 - a. Tentukan koordinat baru
 - b. Gambarkan hasilnya

Latihan Rotasi

Diketahui sebuah objek Q dengan koordinat sebagai berikut : $\{(0,0,1,1), (3,0,1,1), (3,2,1,1), (0,2,1,1), (0,0,0,1), (3,0,0,1), (3,2,0,1), (0,2,0,1)\}$.

1. Gambarkan objek tersebut !
 - a. Lakukan rotasi terhadap Q sebesar $\theta = -90^\circ$ pada x
 - b. Tentukan koordinat baru

- c. Gambarkan hasilnya
2. Lakukan rotasi terhadap objek Q sebesar $\phi = 90^\circ$ pada sumbu y
 - a. Tentukan koordinat baru
 - b. Gambarkan hasilnya

Latihan Refleksi

Diketahui sebuah objek Q dengan koordinat sebagai berikut : $\{(1,0,-1,1), (2,0,-1,1), (2,1,-1,1), (1,1,-1,1), (1,0,-2,1), (2,0,-2,1), (2,1,-2,1), (1,1,-2,1)\}$.

1. Gambarkan objek tersebut !
2. Lakukan refleksi pada bidang xy
 - a. Tentukan koordinat baru
 - b. Gambarkan hasilnya

Latihan Transformasi Gabungan

1. Tentukan Matriks Transformasi Umum untuk transformasi berurutan berikut ini:
 1. Translasi sebesar -1, -1, -1 pada sumbu x, y, z
 2. Rotasi sebesar $+30^\circ$ pada sumbu x
 3. Rotasi sebesar $+45^\circ$ pada sumbu y
2. Tentukan koordinat objek baru untuk vektor posisi homogen $(3 \ 2 \ 1 \ 1)$ yang ditransformasikan dengan MTU yang dihasilkan

Daftar Pustaka

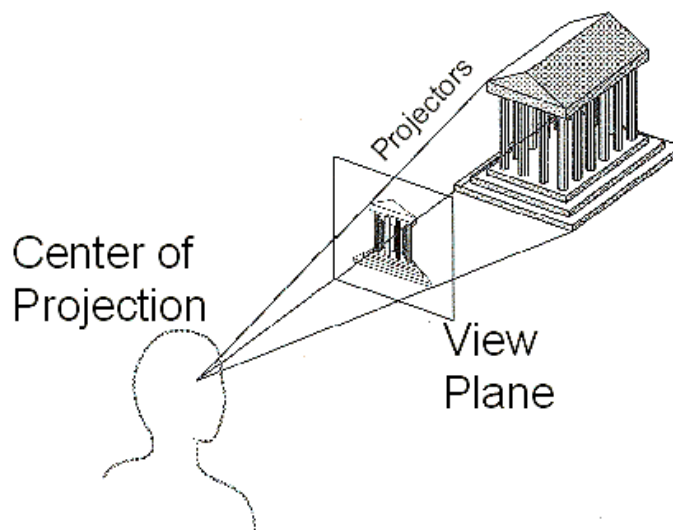
1. David F. Rogers, Alan J. Adams , Mathematical Elements for Computer Graphics (2nd edition), McGraw-Hill, 1989
2. John F. Hughes, Andries Van Dam, Morgan Mcguire, David F. Sklar, James D. Foley, Steven K. Feiner, Kurt Akeley, Computer Graphics: Principles and Practice (3rd edition), Addison-Wesley, 2014
3. Ollie Cornes, Jay Glynn, Burton Harvey, Craig McQueen, Jerod Moemeka, Christian Nagel, Simon Robinson, Morgan Skinner, Karli Watson, Professional C# - Graphics with GDI+, Wrox, 2001

MODUL VII PROYEKSI GEOMETRI BIDANG

Setelah membaca modul ini, mahasiswa akan memiliki pengetahuan dan mampu menjelaskan (i) apa yang dimaksud proyeksi geometri bidang (ii) taksonomi proyeksi geometri bidang (iii) proyeksi paralel dan proyeksi perspektif (iv) titik hilang atau *vanishing points*.

VII.1 Pengertian Proyeksi Geometri Bidang

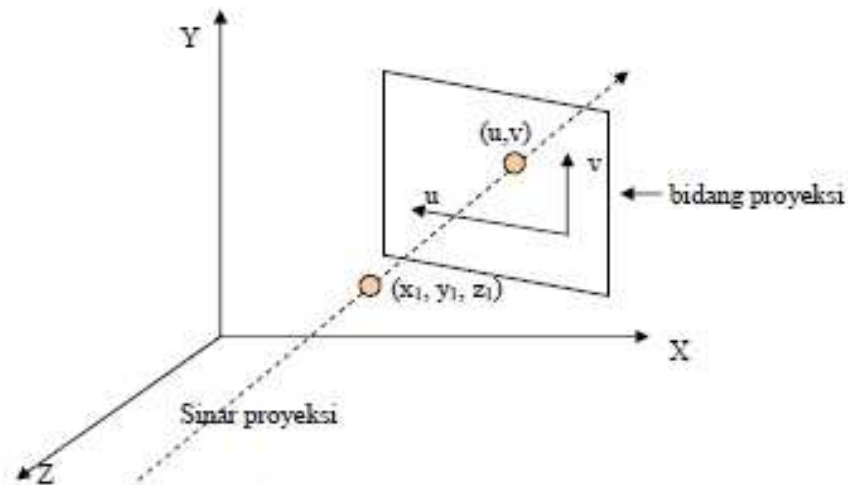
Proyeksi merupakan salah satu jenis transformasi, yaitu transformasi koordinat. Proyeksi merupakan proses dimana informasi tentang titik disebuah sistem koordinat berdimensi n dipindahkan ke sistem koordinat berdimensi kurang dari n . Sebagai contoh titik (x,y,z) yang berada di sistem koordinat 3D dipindahkan ke sistem koordinat 2D sehingga menjadi (x,y) , transformasi tersebut tentunya harus memperhitungkan pengaruh z terhadap titik (x,y) . Gambar di bawah ini memberikan ilustrasi tentang proyeksi.



Gambar 7.1 Proyeksi Geometri Bidang

Proyeksi dapat dilakukan terhadap bidang datar (*planar*) atau bidang kurva. Materi ini akan membahas proyeksi pada bidang datar atau disebut. *planar geometric projection* dilakukan melalui sinar proyeksi yang muncul dari titik pusat proyeksi melewati setiap titik

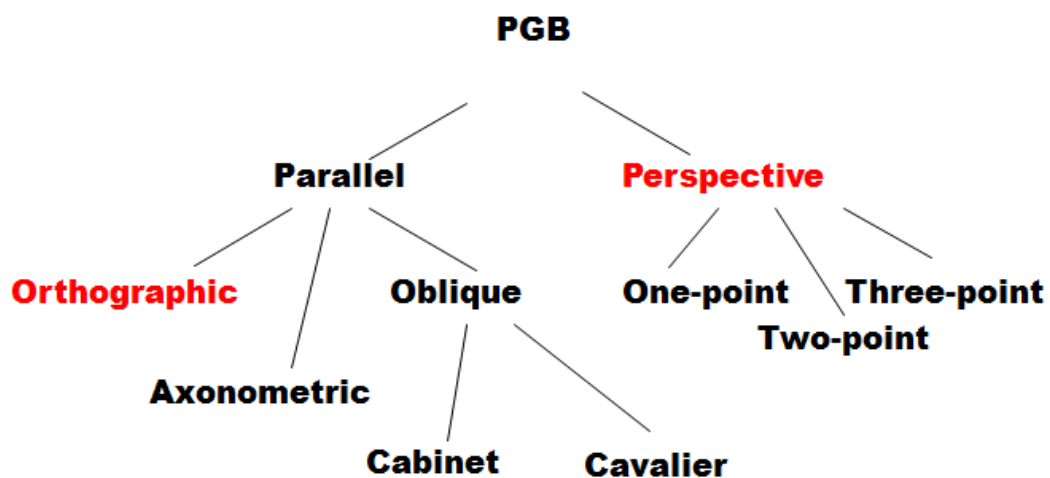
dari benda dan memotong bidang proyeksi (*projection plane*) untuk mendapatkan benda hasil proyeksi. Secara matematis proyeksi dapat digambarkan sebagai berikut:



Gambar 7.2 Proyeksi dan Bidang Proyeksi

VII.2 Taksonomi Proyeksi Geometri Bidang

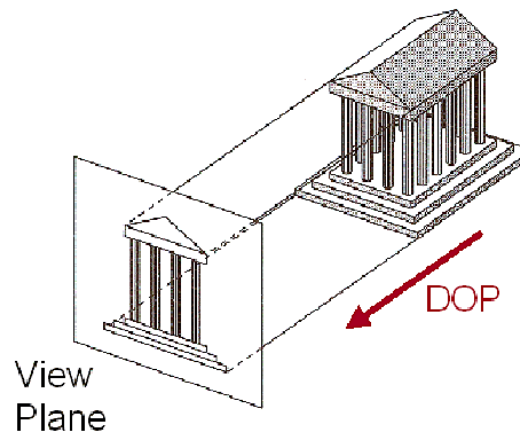
Proyeksi geometri bidang (PGB) dapat dibagi menjadi dua macam, yaitu: proyeksi paralel dan proyeksi perspektif. Perbedaan antara kedua proyeksi ini adalah: pada proyeksi perspektif jarak antara titik pusat proyeksi ke bidang proyeksi bersifat finite (tertentu) sedangkan pada proyeksi paralel jarak antara titik pusat proyeksi ke bidang proyeksi tidak terhingga.



Gambar 7.3 Taksonomi Proyeksi Geometri Bidang

VII.3 Proyeksi Paralel

Proyeksi paralel adalah jenis proyeksi dimana pusat proyeksi pada titik tak hingga (*infinity*). Dengan demikian arah proyeksi (*Direction of Projection-DOP*) adalah sama untuk semua titik. Gambar berikut mengilustrasikan proyeksi paralel.

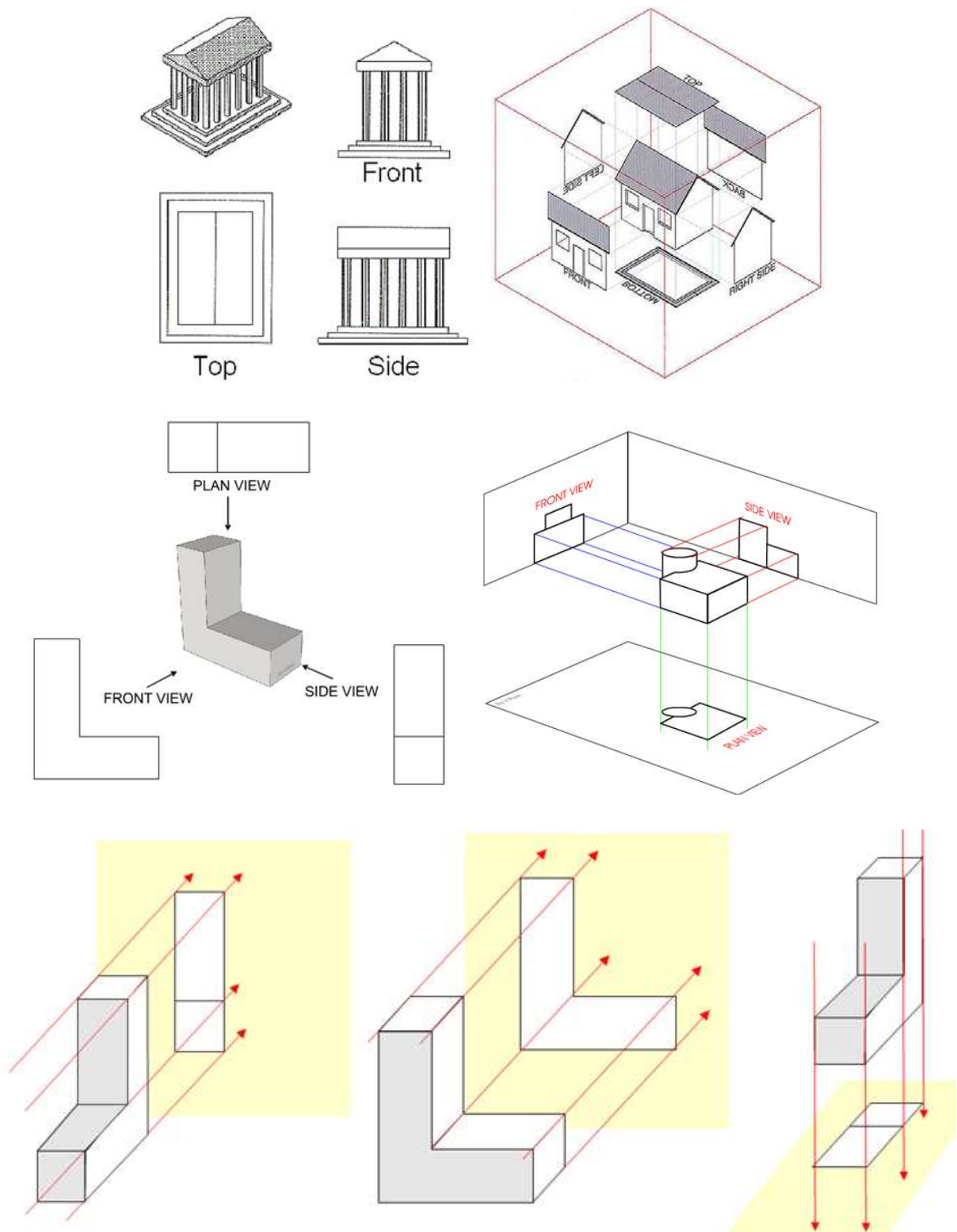


Gambar 7.4 Proyeksi Paralel

Proyeksi paralel dapat dikategorikan menurut hubungan antara arah proyeksi dengan vektor normal dari bidang proyeksi, ke dalam dua macam proyeksi yaitu *orthographic* dan *oblique*.

Proyeksi *Orthographic*

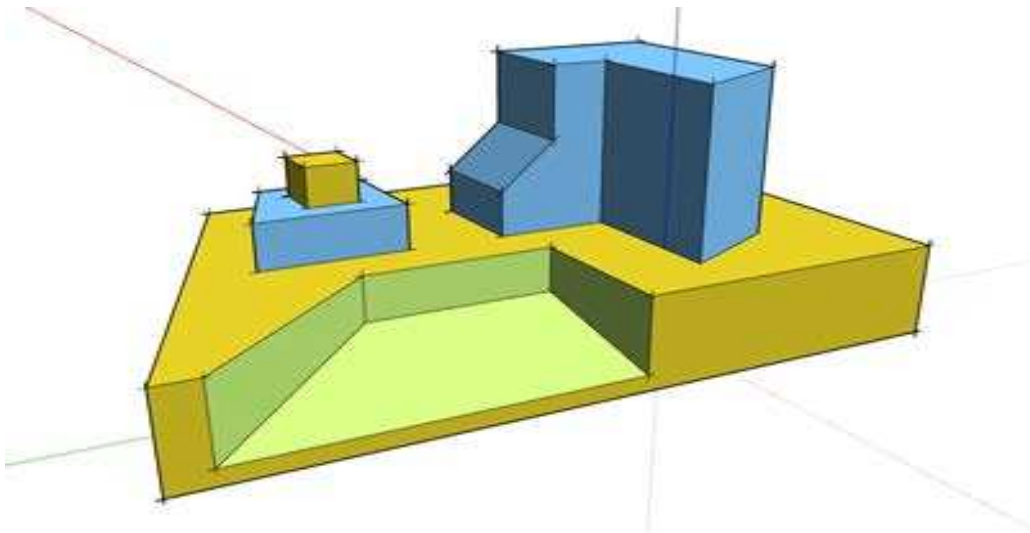
Proyeksi *orthographic* diperoleh apabila sinar proyeksi tegak lurus dengan bidang proyeksi. Proyeksi *orthographic* sering digunakan untuk menghasilkan tampak depan, tampak atas dari sebuah benda atau disebut sebagai *multiview orthographic*. Tampak atas, tampak belakang dan tampak samping dari sebuah benda sering disebut sebagai *elevation*. Sedangkan tampak atas disebut sebagai *plan view*. Gambar-gambar berikut ini mengilustrasikan proyeksi *orthographic*.



Gambar 7.5 Berbagai-macam Proyeksi Orthographic

Latihan

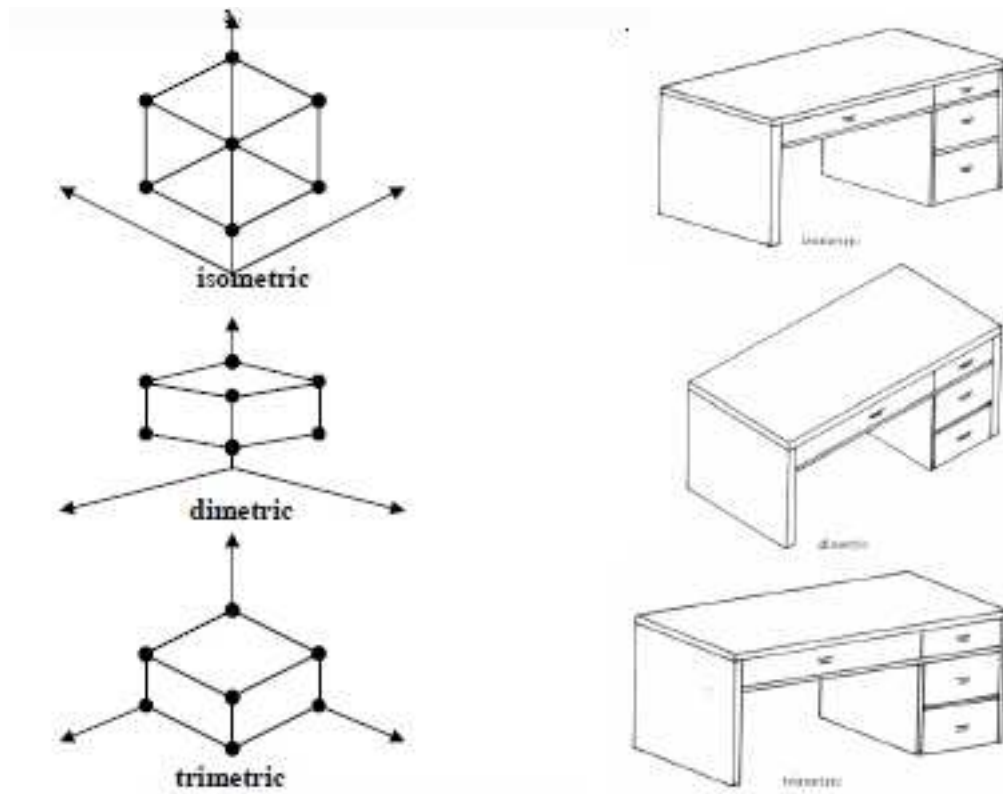
Tentukan secara sketsa proyeksi ortografik dari objek berikut ini.



Proyeksi Axonometric

Proyeksi *orthographic* yang menampilkan lebih dari satu permukaan benda disebut sebagai proyeksi *axonometric*. Apabila proyeksi *axonometric* dilakukan dengan mengatur agar bidang proyeksi berpotongan dengan ketiga sumbu koordinat (*principal axes*) pada sudut yang sama maka kita akan memperoleh proyeksi *isometric*.

Jenis lain dari proyeksi *axonometric* adalah proyeksi *dimetric* yaitu proyeksi yang diperoleh dengan mengatur agar bidang proyeksi berpotongan dengan dua sumbu utama pada sudut yang sama, sedangkan proyeksi *trimetric* diperoleh apabila ketiga sumbu utama berpotongan dengan bidang proyeksi pada sudut yang berbeda. Gambar berikut memperlihatkan proyeksi *isometric*, *dimetric* dan *trimetric*.

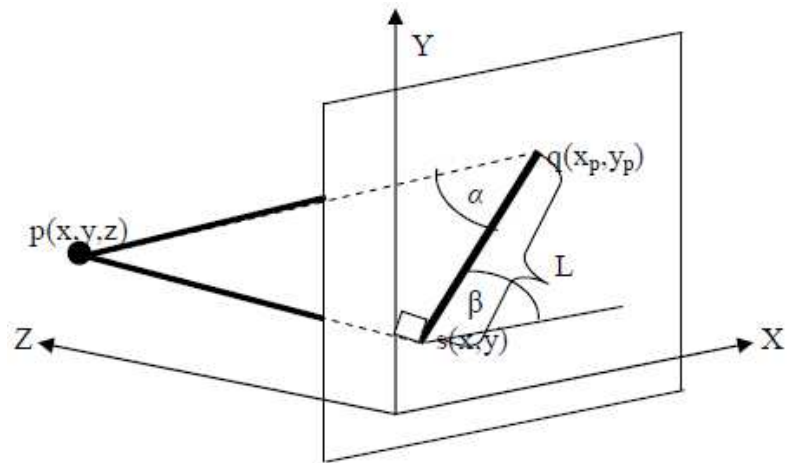


Gambar 7.6 Proyeksi *Isometric*, *Dimetric* dan *Trimetric*

Proyeksi *Oblique*

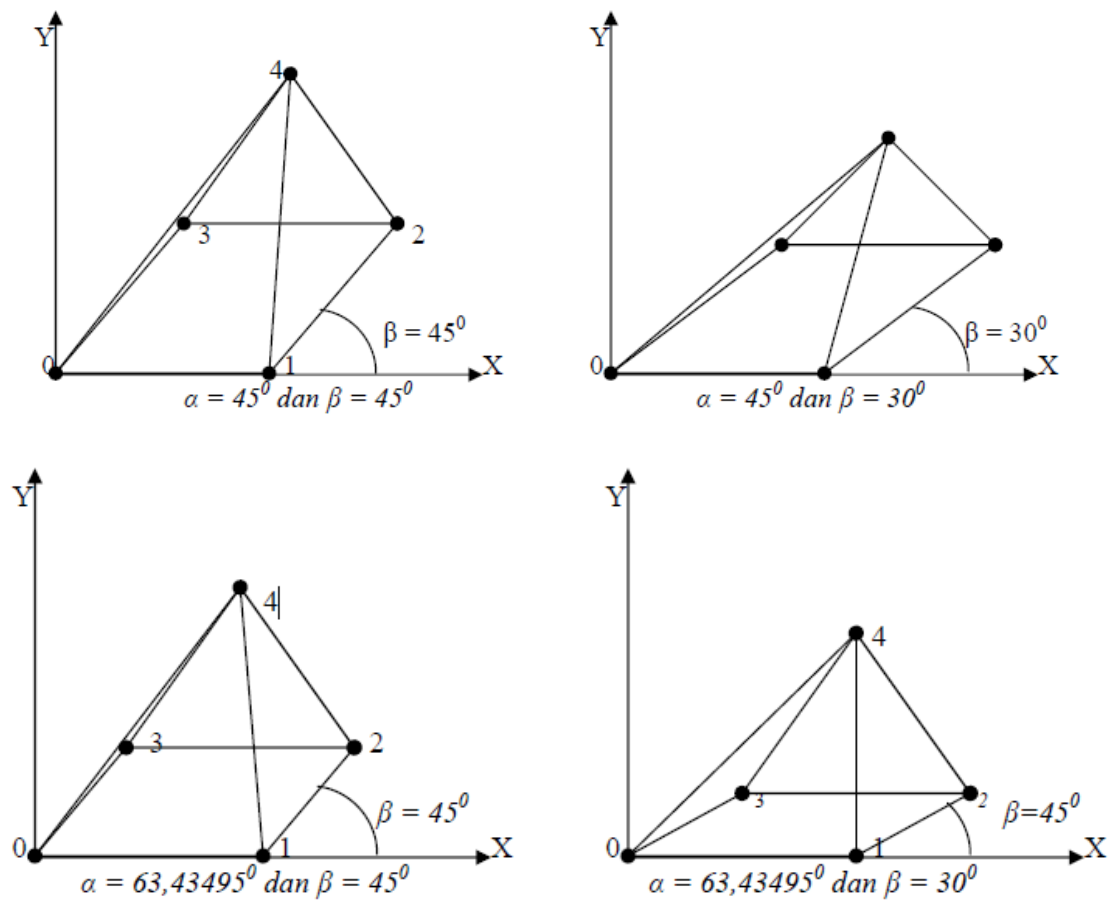
Proyeksi *oblique* diperoleh dengan cara membuat sinar proyeksi tidak tegak lurus terhadap bidang proyeksi. Proyeksi *oblique* membutuhkan dua buah sudut yaitu α dan β seperti diilustrasikan pada Gambar 7.7.

Titik (x,y,z) diproyeksikan menjadi titik $q(x_p, y_p)$ di bidang proyeksi. Titik hasil proyeksi orthographic terletak di $s(x,y)$. Sinar proyeksi membuat sudut α terhadap garis $q-s$ yang terletak di bidang proyeksi. Garis $q-s$ dengan panjang L membentuk sudut terhadap arah mendatar dari bidang proyeksi.



Gambar 7.7 Proyeksi Oblique

Gambar berikut ini mengilustrasikan hasil proyeksi *oblique jenis cavalier* dan *cabinet*.



Gambar 7.8 Proyeksi Oblique Jenis Cavalier dan Cabinet

Operasi Transformasi Proyeksi Ortografik

Matriks transformasi umum dari operasi proyeksi ortografik pada bidang yz ($x=0$), xz ($y=0$) dan zy ($z=0$) dapat dituliskan sebagai berikut:

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Contoh Soal

Tentukan koordinat objek hasil proyeksi ortografik terhadap koordinat objek

$$[X] = \begin{bmatrix} -0.5 & -0.5 & 0.5 & 1 \\ 0.5 & -0.5 & 0.5 & 1 \\ 0.5 & 0.5 & 0.5 & 1 \\ -0.5 & 0.5 & 0.5 & 1 \\ -0.5 & -0.5 & -0.5 & 1 \\ 0.5 & -0.5 & -0.5 & 1 \\ 0.5 & 0.5 & -0.5 & 1 \\ -0.5 & 0.5 & -0.5 & 1 \end{bmatrix}$$

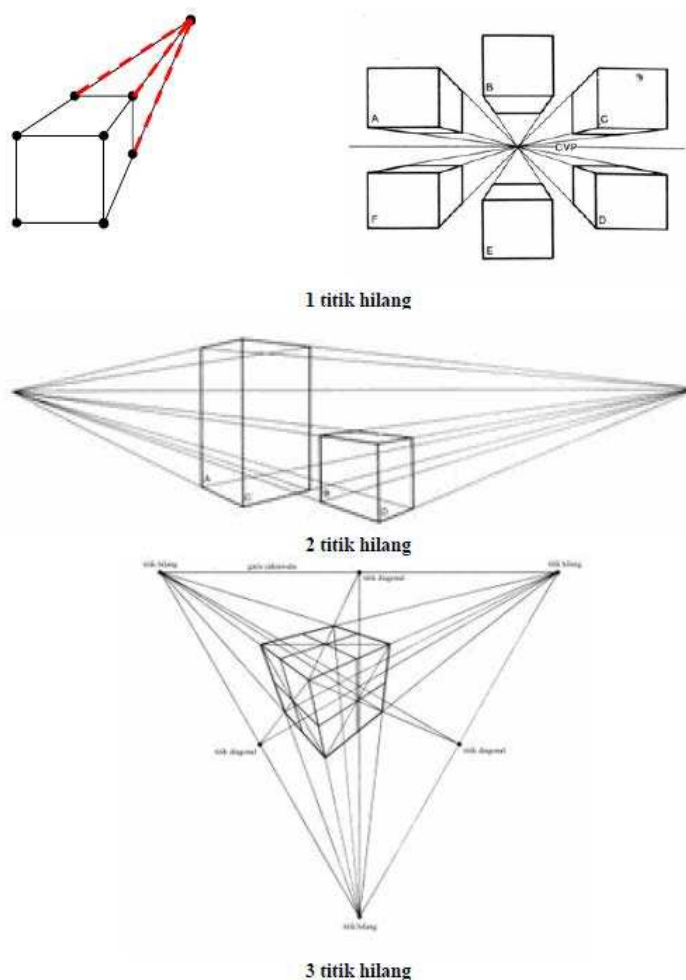
Jawab: Untuk mendapatkan koordinat objek baru maka kalikan matrix objek (X) dengan Matriks Transformasi Umum untuk masing-masing proyeksi ortografik. Contoh, untuk proyeksi pada bidang xy , maka koordinat objek barunya adalah:

$$[X1] = \begin{bmatrix} -0.5 & -0.5 & 0 & 1 \\ 0.5 & -0.5 & 0 & 1 \\ 0.5 & 0.5 & 0 & 1 \\ -0.5 & 0.5 & 0 & 1 \\ -0.5 & -0.5 & 0 & 1 \\ 0.5 & -0.5 & 0 & 1 \\ 0.5 & 0.5 & 0 & 1 \\ -0.5 & 0.5 & 0 & 1 \end{bmatrix}$$

VII.4 Proyeksi Perspektif

Proyeksi perspektif memberikan sudut pandang yang lebih realistis dibandingkan proyeksi *orthographic*. Proyeksi perseprktif memberikan tampilan yang sama dengan apa yang kita lihat sehari-hari karena pada kenyataanya jarak benda terhadap kita akan mempengaruhi bagaimana benda tersebut terlihat. Benda yang terlihat jauh akan kelihatan kecil sedangkan benda yang dekat akan terlihat lebih besar. Efek ini disebut sebagai *shortening* (pemendekan).

Pada proyeksi perspektif semua garis menghilang pada satu atau lebih titik yang sama atau disebut titik hilang (*vanishing point*). Hal ini mengakibatkan gari sejajar akan tampak tidak sejajar ketika diproyeksikan perspektif. Bergantung kepada lokasi dimana kita melihat benda maka kita akan memperoleh efek: 1 titik hilang, 2 titik hilang dan 3 titik hilang. Gambar berikut memperlihatkan benda berdasarkan banyaknya titik hilang.



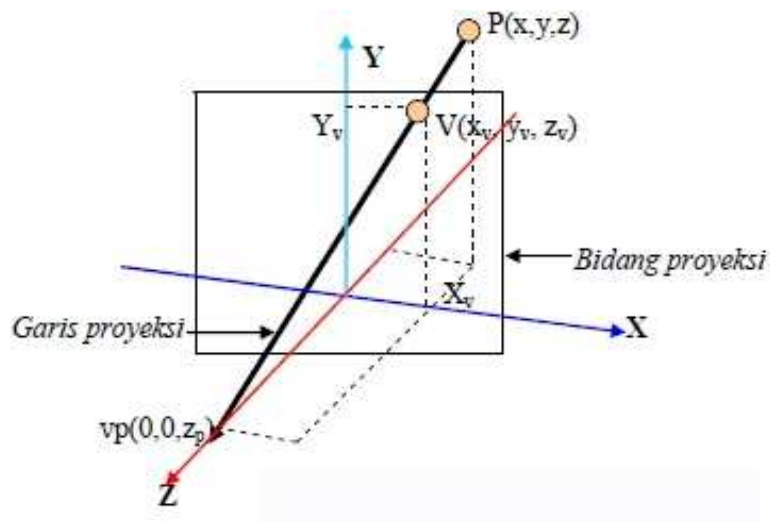
Gambar 7.9 Proyeksi Perspektif

Perspektif 1 titik hilang akan diperoleh apabila ketinggian pengamat relatif sama dengan ketinggian benda yang dilihat dan berada pada jarak relatif dekat, dan perspektif 2 titik hilang akan diperoleh apabila pemirsa berada sedikit lebih tinggi atau lebih rendah dan agak jauh dari benda, sedangkan perspektif 3 titik hilang akan diperoleh apabila lokasi pemirsa jauh lebih tinggi atau lebih rendah dibandingkan benda yang dilihat.

Proyeksi Paralel VS Proyeksi Perspektif

Proyeksi Perspektif	Proyeksi Paralel
<ul style="list-style-type: none"> • Ukuran berdasarkan jarak – lebih realistik • Jarak dan sudut tidak selalu <i>preserved</i> • Garis paralel tidak selalu sejajar 	<ul style="list-style-type: none"> • Baik untuk pengukuran yang membutuhkan ketelitian/presisi • Garis paralel tetap sejajar • Sudut tidak <i>preserved</i> • Kurang realistik

Bagaimana proyeksi perspektif terjadi, ditunjukkan pada gambar berikut ini.



Gambar 7.10 Mekanisme Proyeksi Perspektif

Titik $p(x,y,z)$ diproyeksikan ke bidang $x-y$ melalui garis proyeksi yang memotong sumbu z pada jarak z_p . Garis proyeksi akan memotong bidang proyeksi di titik $v(x_v,y_v,z_v)$.

Operasi Proyeksi Perspektif

Proses komputasi untuk operasi proyeksi perspektif ditentukan oleh elemen p , q , r dari matriks transformasi umum 3 dimensi yang telah diberikan pada bab sebelumnya.

$$\begin{bmatrix} a & b & c & p \\ d & e & f & q \\ g & h & i & r \\ l & m & n & s \end{bmatrix}$$

p , q , r adalah nilai proyeksi yang besarannya dihitung sebagai

- $p = -1/x$ untuk pusat proyeksi pada sumbu x
- $q = -1/y$ untuk pusat proyeksi pada sumbu y
- $r = -1/z$ untuk pusat proyeksi pada sumbu z

Latihan

1. Tentukan MTU Proyeksi 2-titik dengan pusat proyeksi pada $x = -10$ dan $y = -10$ diproyeksikan pada bidang $z=0$

Jawab:
$$\begin{bmatrix} 1 & 0 & 0 & 0.1 \\ 0 & 1 & 0 & 0.1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2. Tentukan MTU Proyeksi 1-titik dengan pusat proyeksi pada $z=10$ setelah objek ditranslasikan sebesar $\frac{1}{2}$ unit pada sumbu x dan y

Jawab:
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -0.5 & -0.5 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -0.1 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -0.1 \\ -0.5 & -0.5 & 0 & 1 \end{bmatrix}$$

3. Consider an origin-centered unit cube with position vectors given by

$$[X] = \begin{bmatrix} -0.5 & -0.5 & 0.5 & 1 \\ 0.5 & -0.5 & 0.5 & 1 \\ 0.5 & 0.5 & 0.5 & 1 \\ -0.5 & 0.5 & 0.5 & 1 \\ -0.5 & -0.5 & -0.5 & 1 \\ 0.5 & -0.5 & -0.5 & 1 \\ 0.5 & 0.5 & -0.5 & 1 \\ -0.5 & 0.5 & -0.5 & 1 \end{bmatrix}$$

Translate the cube -5 units in the x and y directions and perform a single-point perspective projection onto the $z=0$ plane from the center of projection at $z=z_c=10$

4. Consider a cube with position vectors given by

$$[X] = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

Rotate the cube about the y-axis by $\phi=60^\circ$ and translated 2 units into y then projected onto the $z=0$ plane from the center of Projection at $z=z_c=2.5$

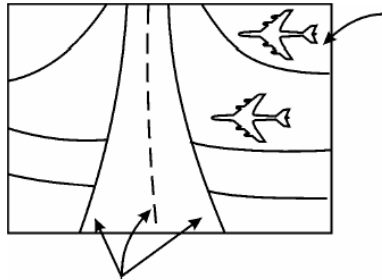
5. Consider a cube with position vectors given by

$$[X] = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

Rotate the cube about the y-axis by $\phi = -30^\circ$, about the x-axis by $\Theta = 45^\circ$ and projected onto the $z=0$ plane a center of projection at $z=z_c=2.5$

VII.5 Titik Hilang (*Vanishing Points*)

Titik hilang adalah sebuah titik dimana beberapa garis paralel bertemu. Tentu definisi ini apabila dilihat secara matematis adalah hal yang mustahil. Mana mungkin garis-garis yang paralel atau sejajar dapat bertemu di satu titik.



Gambar 7.11 Titik Hilang atau *Vanishing Point*

Formula matematika untuk mencari titik hilang adalah sebagai berikut:

$$[TH] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} [MTU]$$

Latihan

Mengacu kepada soal sebelumnya, hasil MTU nya adalah

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -0.1 \\ -0.5 & -0.5 & 0 & 1 \end{bmatrix}$$

Dengan menerapkan formula pencarian titik hilang di atas maka diperoleh hasil

$$\begin{bmatrix} 0.866 & -0.354 & 0 & -0.141 \\ 0 & 0.707 & 0 & -0.283 \\ -0.5 & -0.612 & 0 & -0.245 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Dari hasil tersebut maka titik hilangnya ada 3 yaitu

$$\begin{bmatrix} -6.142 & 2.5 & 0 & 1 \\ 0 & -2.5 & 0 & 1 \\ 2.04 & 2.5 & 0 & 1 \end{bmatrix}$$

Atau apabila dituliskan dengan koordinat, maka titik hilangnya adalah (-6.142, 2.5) , (0, -2.5) dan (2.04, 2.5). Ingat bahwa proyeksi perspektif mengeliminasi sumbu Z atau membuat sumbu Z bernilai nol.

Catatan: Untuk menghasilkan titik hilang, kolom ke 4 harus bernilai 1.

Daftar Pustaka

1. David F. Rogers, Alan J. Adams , Mathematical Elements for Computer Graphics (2nd edition), McGraw-Hill, 1989
2. John F. Hughes, Andries Van Dam, Morgan Mcguire, David F. Sklar, James D. Foley, Steven K. Feiner, Kurt Akeley, Computer Graphics: Principles and Practice (3rd edition), Addison-Wesley, 2014
3. Ollie Cornes, Jay Glynn, Burton Harvey, Craig McQueen, Jerod Moemeka, Christian Nagel, Simon Robinson, Morgan Skinner, Karli Watson, Professional C# - Graphics with GDI+, Wrox, 2001

MODUL VIII KOMPONEN PENDUKUNG PEMROGRAMAN GRAFIS

Setelah membaca modul ini, mahasiswa akan memiliki pengetahuan dan mampu menjelaskan (i) apakah yang dimaksud dengan GDI+ (ii) bagaimana definisi dari GDI+ (iii) bagaimana menggunakan GDI+ dalam aplikasi (iv) hal-hal baru apa yang terdapat dalam GDI+ (v) *namespace* utama mana pada .NET *framework* yang menunjukkan fungsionalisasi dari DGI+ (vi) Prosedur operasi matriks menggunakan C#

VIII.1 Pemahaman GDI+

GDI+ adalah *library* pada .NET yang digunakan untuk membuat aplikasi grafis berbasis *Windows* dan yang dapat berinteraksi dengan perangkat grafis berupa printer atau monitor. *Graphical User Interface* (GUI) berinteraksi dengan perangkat *hardware* seperti monitor, printer, atau scanner untuk menampilkan format yang dapat dibaca oleh manusia, tetapi tidak ada program yang dapat mengakses perangkat *hardware* tersebut secara langsung, maka dibuat *user interface* agar pengguna dapat berinteraksi dengan perangkat-perangkat tersebut.

Untuk membuat *user interface* tersebut harus digunakan *third component* sebagai jembatan antara program dan perangkat keras, maka dibuatlah komponen GDI+ *library*, dengan komponen tersebut anda dapat menampilkan keluaran berupa teks dan gambar ke perangkat *hardware*.

Pada aplikasi .NET anda dapat menggunakan GDI+ untuk membuat aplikasi grafis baik untuk aplikasi berbasis *Windows* maupun aplikasi web. *Library* GDI+ sudah terinstall secara *default* pada sistem operasi *Windows* yang berlokasi di *class library* dengan nama *Gdiplus.dll*.



Gambar 8.1 GDI+ Sebagai *Interface* antara Aplikasi dan *Hardware*

VIII.2 Eksplorasi Fungsionalitas GDI+

Apa saja fitur GDI+?

Fitur GDI+ dapat kategorikan dalam 3 fungsi utama yaitu:

- Grafik vektor 2D
- Imaging
- Typograhya

Pemrograman Grafik Vektor 2D

Grafik vektor merupakan komponen pembentuk bentuk gambar mis (kotak, lingkaran) yang dibentuk dari kumpulan titik pada sistem koordinat.

Pada .NET *Framework library* 2D vector graphic programming dibagi menjadi dua kategori yaitu *general* dan *advanced*. General 2D vector graphic didefinisikan dalam *library System.Drawing namespace* dan *advance function* didefinisikan dalam *library System.Drawing.Drawing2D namespace*.

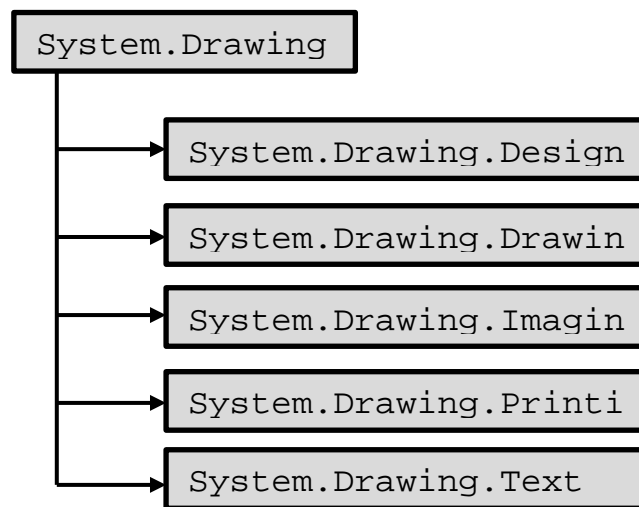
Imaging

Imaging digunakan untuk memanipulasi *image*. *Image file* yang dapat dimanipulasi misalnya .bmp, .jpg, .gif, dan .png. Fungsi-fungsi untuk memanipulasi *images* ada dalam *Image class* yang digunakan untuk *create, save, dan load images*.

Typography

Typography digunakan untuk mendesign dan merepresentasikan teks. GDI+ menyediakan *class* untuk membuat dan menggunakan *font*.

VIII.3 GDI+ Namespaces dan Classes dalam .NET



Gambar 8.2 Struktur *namespace* `System.Drawing`

Kelas-kelas yang ada pada `System.Drawing`

Class	Description
Bitmap	<i>An abstract base class that cannot be instantiated directly. The Brush class provides functionality used by its derived brush classes and represents a brush graphics object. A brush is used to fill the interior of a graphical shape with a specified color.</i>
Brush	<i>Represents brushes with all the standard colors. This class has a static member for each standard color. For example, <code>Brushes.Blue</code> represents a blue brush.</i>
ColorConverter	<i>Provides methods and properties to convert colors from one type to another.</i>
ColorTranslator	<i>Provides various methods to translate colors from one type to another.</i>
Font	<i>Provides members to define the format of font text, name, face, size, and styles. The Font class also provides methods to create a Font object from a window handle to a device context or window handle.</i>
FontConverter	<i>Provides members that convert fonts from one type to another.</i>
FontFamily es.	<i>Defines a group of typefaces having a similar basic design and certain variations in styl</i>

Graphics	<i>A key class that encapsulates drawing surfaces. Among many other things, the Graphics class provides members to draw and fill graphical objects</i>
Icon	<i>Represents a Windows icon. The Icon class provides members to define the size, width, and height of an icon.</i>
	<i>Provides members to convert an Icon object from one type to another.</i>
IconConverter	<i>Provides members to convert an Icon object from one type to another.</i>
Image	<i>Provides members to define the size, height, width, and format of an image. The Image class also provides methods to create Image objects from a file, a window handle, or a stream; and to save,</i>
ImageAnimator	<i>Provides methods to start and stop animation, and to update frames for an image that has timebased frames.</i>
ImageConverter	<i>Provides members to convert Image objects from one type to another.</i>
ImageFormatConverter	<i>Defines members that can be used to convert images from one format to another.</i>
Pen	<i>Defines a pen with a specified color and width. A pen is used to draw graphical objects such as a line, a rectangle, a curve, or an ellipse</i>
ImageAnimator	<i>Provides methods to start and stop animation, and to update frames for an image that has timebased frames.</i>
Pens	<i>Provides static members for all the standard colors. For example, Pens.Red represents a red pen.</i>
PointConverter.	<i>Defines members that can be used to convert Point objects from one type to another</i>
RectangleConverter	<i>Defines members that can be used to convert Rectangle objects from one type to another.</i>
Region	<i>Represents a region in GDI+, which describes the interior of a graphics shape.</i>
SizeConverter	<i>Defines members that can be used to convert size from one type to another.</i>
SolidBrush	<i>Inherited from the Brush class. This class defines a solid brush of a single color.</i>

StringFormat	<i>Provides members to define text format, including alignment, trimming and line spacing, display manipulations, and OpenType features</i>
SystemBrushes	<i>Defines static properties. Each property is a SolidBrush object with a Windows display element such as Highlight, HighlightText, or ActiveBorder.</i>
SystemColors	<i>Defines static properties of a Color structure.</i>
SystemIcons	<i>Defines static properties for Windows systemwide icons.</i>
SystemPens	<i>Defines static properties. Each property is a Pen object with the color of a Windows display</i>
TextureBrush	<i>Inherited from the Brush class. This class defines a brush that has an image as its texture.</i>
ToolboxBitmapAttribute	<i>Defines the images associated with a specified component.</i>

Kelas-kelas pada System.Drawing.Design

Class	Description
BitmapEditor	<i>User interface (UI) for selecting bitmaps using a Properties window.</i>
CategoryNameCollection	<i>Collection of categories.</i>
FontEditor	<i>UI for selecting and configuring fonts.</i>
ImageEditor	<i>UI for selecting images in a Properties window.</i>
PaintValueEventArgs	<i>Provides data for the PaintValue event.</i>
PropertyValueUIItem	<i>Provides information about the property value UI for a property.</i>
ToolboxComponentsCreatedEventArgs	<i>Provides data for the ComponentsCreated event, which occurs when components are added to the toolbox.</i>
ToolboxComponentsCreatingEventArgs	<i>Provides data for the ComponentsCreating event, which occurs when components are added to the toolbox.</i>
ToolboxItem	<i>Provides a base implementation of a toolbox item.</i>

ToolboxItemCollection	Collection of toolbox items.
UITypeEditor	Provides a base class that can be used to design value editors.
BitmapEditor	User interface (UI) for selecting bitmaps using a Properties window.

Kelas-kelas pada System.Drawing.Drawing2D

Class	Description
AdjustableArrowCap	<i>Represents an adjustable arrow-shaped line cap. Provides members to define the properties to fill, and to set the height and width of an arrow cap.</i>
Blend	<i>Gradient blends are used to provide smoothness and shading to the interiors of shapes. A blend pattern contains factor and pattern arrays, which define the position and percentage of color of the starting and ending colors. The Blend class defines a blend pattern, which uses LinearGradientBrush to fill the shapes. The Factors and Positions properties represent the array of blend factors and array of positions for the gradient, respectively.</i>
ColorBlend	<i>Defines color blending in multicolor gradients. The Color and Position properties represent the color array and position array, respectively.</i>
CustomLineCap	<i>Encapsulates a custom, user-defined line cap.</i>
GraphicsContainer	<i>Represents the data of a graphics container. A graphics container is created by Graphics.BeginContainer followed by a call to Graphics.EndContainer.</i>
GraphicsPath	<i>In GDI+, a path is a series of connected lines and curves. This class provides properties to define the path's fill mode and other properties. This class also defines methods to add graphics shapes to a path. For instance, the AddArc and AddCurve methods add an arc and a curve, respectively, to the path. Wrap, Transform, Reverse, and Reset are some of the associated methods.</i>
GraphicsPathIterator	<i>A path can contain subpaths. This class provides the ability to find the number of subpaths and iterate through them. Count and SubpathCount return the number of points and the number of subpaths in a path, respectively.</i>

GraphicsState	Represents the state of a Graphics object.
HatchBrush	Hatch brushes are brushes with a hatch style, a foreground color, and a background color. This class represents a hatch brush in GDI+.
LinearGradientBrush	Represents a brush with a linear gradient.
Matrix	Encapsulates a 3x3 matrix that represents a geometric transformation. This class defines methods for inverting, multiplying, resetting, rotating, scaling, shearing, and translating matrices.
PathData	Contains the data in the form of points and types that makes up a path. The Points property of the class represents an array of points, and the Types property represents the types of the points in a path.
PathGradientBrush	Represents a brush with a graphics path. PathGradientBrush contains methods and properties for blending, wrapping, scaling, and transformation. This class encapsulates a Brush object that fills the interior of a GraphicsPath object with a gradient.
RegionData	Represents the data stored by a Region object. The Data property of this class represents the data in the form of an array of bytes.

Kelas-kelas pada System.Drawing.Imaging

Class	Description
BitmapData	<i>Often we don't want to load and refresh all data of a bitmap because rendering each pixel is not only a slow process, but also consumes system resources. With the help of the BitmapData class and its LockBits and UnlockBits methods, we can lock the required data of a bitmap in memory and work with that instead of working with all the data.</i>
ColorMap	<i>Defines a map for converting colors. ColorMap is used by the ImageAttributes class.</i>
ColorMatrix	<i>Defines a 5x5 matrix that contains coordinates for the ARGB space. ColorMatrix is used by the ImageAttributes class.</i>
ColorPalette	<i>Defines an array of colors that make up a color palette. ColorPalette is used by the ImageAttributes class.</i>

Encoder	<i>Represents an encoder, which represents a globally unique identifier (GUID) that identifies the category of an image encoder parameter. Encoder is used by the EncoderParameter class.</i>
EncoderParameter	<i>An encoder parameter, which sets values for a particular category of an image. This class is used in the Save method with the help of EncoderParameters.</i>
EncoderParameters	<i>An array of EncoderParameter objects.</i>
FrameDimension	<i>Provides properties to get the frame dimensions of an image.</i>
ImageAttributes	<i>Contains information about how image colors are manipulated during rendering</i>
ImageCodecInfo	<i>Retrieves information about the installed image codecs.</i>
ImageFormat	<i>Specifies the format of an image.</i>
Metafile	<i>Defines a graphic metafile, which contains graphics operations in the form of records that can be recorded (constructed) and played back (displayed).</i>
MetafileHeader	<i>Stores information about a metafile.</i>
MetaHeader	<i>Contains information about a Windows-format (WMF) metafile.</i>
PropertyItem	<i>Encapsulates a metadata property to be included in an image file.</i>
WmfPlaceableFileHeader	<i>Defines a placeable metafile.</i>

Kelas-kelas pada System.Drawing.Printing

Class	Description
Margins	<i>Specifies the margins of a printed page. The Bottom, Left, Right, and Top properties are used to get and set the bottom, left, right, and top margins, respectively, of a page in hundredths of an inch.</i>
MarginsConverter	<i>Provides methods to convert margins, including CanConvertFrom, CanConvertTo, ConvertFrom, and ConvertTo</i>
PageSettings	<i>Specifies settings of a page, including properties such as Bounds, Color, Landscape, Margins, PaperSize,</i>

	<i>PaperSource, PrinterResolution, and PrinterSettings.</i>
PaperSize	<i>Specifies the paper size. Its properties include Height, Width, PaperName, and Kind. The Kind property is the type of paper, represented by the PaperKind enumeration, which has members that represent A3, envelopes, sheets, ledgers, and so on.</i>
PaperSource	<i>Specifies the paper tray from which the printer gets paper, with properties Kind and</i>
SourceName.	<i>SourceName is a type of PaperSource enumeration, which defines members based on the Kind property.</i>
PreviewPageInfo	<i>Provides print preview information for a single page. The Image property returns the image of the printed page, and the PhysicalSize property returns the size of the printed page in 1/1000 inch.</i>
PreviewPrintController	<i>Displays a document on a screen as a series of images for each page. The UseAntiAlias property gets and sets the anti-aliasing when displaying the print preview.</i>
PrintController	<i>Controls how a document is printed. The class provides four methods: OnStartPage, OnStartPrint, OnEndPage, and OnEndPrint.</i>
PrintDocument	<i>Starts the printing process. Creates an instance of this class, sets the printing properties that describe how to print, and calls the Print method to start the process.</i>
PrinterResolution	<i>Provides properties to return a printer resolution. The Kind, X, and Y properties return the printer resolution, horizontal resolution in dots per inch (dpi), and vertical printer resolution in dpi, respectively.</i>
PrinterSettings	<i>Provides methods and properties for setting how a document is printed, including the printer that prints it. Some of the common properties are MinimumPage, MaximumPage, Copies, MaximumCopies, PrinterName, and so on.</i>
PrinterSettings.PaperSizeCollection	<i>Collection of PaperSize objects.</i>
PrinterSettings.PaperSourceCollection	<i>Collection of PaperSource objects.</i>
PrinterSettings.PrinterResolutionCollection	<i>Collection of PrinterResolution objects.</i>

PrinterUnitConvert	<i>Specifies a series of conversion methods that are useful when interoperating with the Win32</i>
PrintEventArgs	<i>Provides data for the BeginPrint and EndPrint events.</i>
PrintingPermission	<i>Controls access to printers.</i>
PrintingPermissionAttribute	<i>Allows declarative printing permission checks.</i>
PrintPageEventArgs	<i>Provides data for the PrintPage event.</i>
QueryPageSettingsEventArgs	<i>Provides data for the QueryPageSettings event.</i>
StandardPrintController	<i>Specifies a print controller that sends information to a printer</i>

Kelas-kelas pada System.Drawing.Text

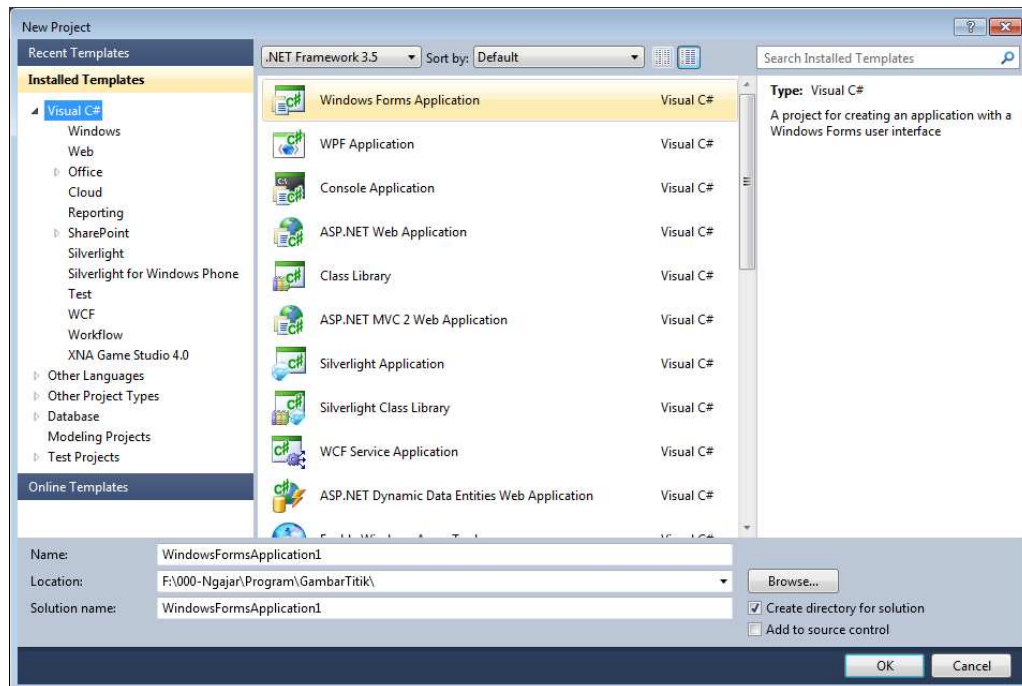
Class	Description
FontCollection	<i>Abstract base class for installed and private font collections. It provides a method to get a list of the font families contained in the collection. Two derived classes from the FontCollection class are InstalledFontCollection and PrivateFontCollection</i>
InstalledFontCollection	<i>Represents the fonts installed on the system.</i>
PrivateFontCollection	<i>Represents a collection of font families built from font files that are provided by the client application</i>

VIII.4 Mempersiapkan Penggunaan Grafis pada Visual Studio

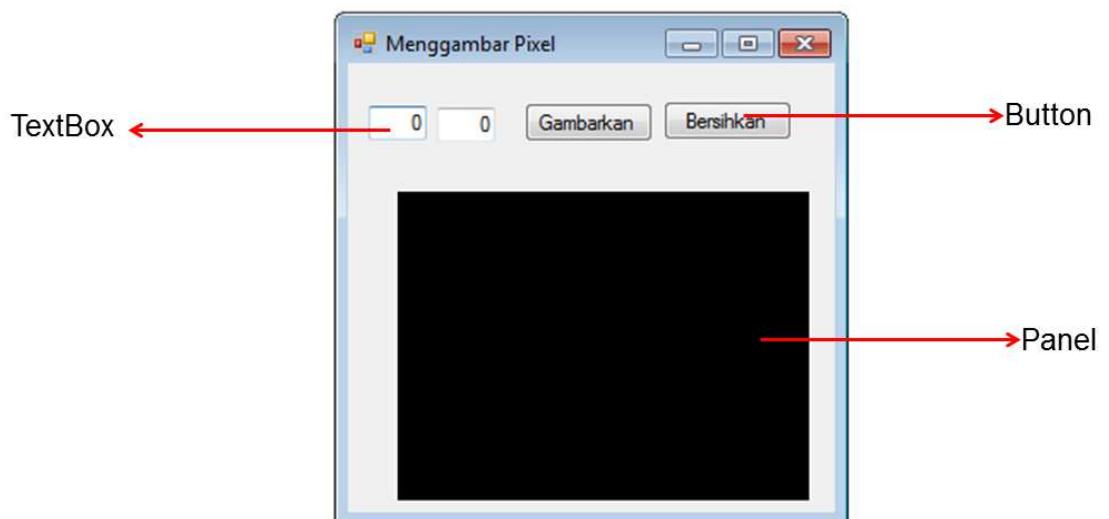
Sebelum kita bisa melakukan pemrograman, harus dipersiapkan dulu perangkat lunak Visual Studio. Saat ini versi yang terakhir dan terlengkap dari visual studio adalah Visual Studio Ultimate 2013 Update 3. Setelah Visual Studio di-install, jalankan Visual Studio dan lakukan langkah-langkah sebagai berikut. Dalam contoh ini akan digunakan Visual Studio 2010 Ultimate. Diasumsikan proses instalasi telah sukses dilakukan.

1. Jalankan Visual Studio
2. Buat Proyek Baru (*File-New-Project*)

3. Pilih Windows Form Application, yakinkan bahwa *template* yang digunakan adalah Visual C#



4. Tentukan *Name*, *Location* dan *Solution Name* sesuai dengan keinginan pemrogram
5. Jika sudah lengkap maka pada layar akan ditampilkan form kosong.
6. Dengan memilih komponen dalam ToolBox, buatlah *form* sebagai berikut:



7. Ketikkan program berikut pada tab Form1.cs

```
Graphics g;
int x, y;
Brush aBrush = (Brush)Brushes.White;

private void Form1_Load(object sender, EventArgs e)
{
    g = canvas.CreateGraphics();
}

private void DrawPixel_Click(object sender, EventArgs e)
{
    x=Convert.ToInt16(PointX.Text);
    y =Convert.ToInt16(PointY.Text);
    g.FillRectangle(aBrush, x, y, 1, 1);
}

private void ClearScreen_Click(object sender, EventArgs e)
{
    canvas.Refresh();
}
```

8. Jalankan program tersebut dan lihat hasilnya.

Catatan: Pastikan bahwa komposnens istem grafika dituliskan dalam program yaitu :

```
↓
using System.Drawing;
↑
```

VIII.5 Operasi Matriks Menggunakan C#

Menginputkan data ke dalam Matriks

```
int[,] intArr1 = new int[2, 2];
intArr1[0, 0] = 12;
intArr1[0, 1] = 23;
intArr1[1, 0] = 14;
intArr1[1, 1] = -9;
for (int i = 0; i < 2; i++)
{
    for (int j = 0; j < 2; j++)
    {
        Console.Write(intArr1[i, j] + " ");
    }
    Console.WriteLine();
}
```

Operasi Penjumlahan Matriks

```
int intBaris = 0, intKolom = 0;
Console.Write("Masukan Baris Matrix :");
intBaris = Convert.ToInt32(Console.ReadLine());
Console.Write("Masukan Banyak Kolom :");
intKolom = Convert.ToInt32(Console.ReadLine());
int[,] intMatrix1 = new int[intBaris, intKolom];
int[,] intMatrix2 = new int[intBaris, intKolom];
int[,] intMatrixHasil = new int[intBaris, intKolom];
Console.WriteLine("Input Matrix1");
for (int b = 0; b < intBaris; b++)
{
    for (int k = 0; k < intKolom; k++)
    {
        Console.Write
            ("Masukan Matrix1[" + b + "," + k + "] : ");
        intMatrix1[b, k] =
            Convert.ToInt32(Console.ReadLine());
    }
}
Console.WriteLine("Input Matrix2");
for (int b = 0; b < intBaris; b++)
{
    for (int k = 0; k < intKolom; k++)
    {
        Console.Write
            ("Masukan Matrix2[" + b + "," + k + "] : ");
        intMatrix2[b, k] =
            Convert.ToInt32(Console.ReadLine());
    }
}
```

```

}
for (int b = 0; b < intBaris; b++)
{
    for (int k = 0; k < intKolom; k++)
    {
        intMatrixHasil[b, k] =
            intMatrix1[b, k] + intMatrix2[b, k];
    }
}
Console.WriteLine("Hasil Penjumlahan Matrix :");
for (int b = 0; b < intBaris; b++)
{
    for (int k = 0; k < intKolom; k++)
    {
        Console.Write(intMatrixHasil[b, k] + " ");
    }
}
Console.WriteLine();
}

```

Operasi Perkalian Matriks

```

int intBMat1, intKmlBm2, intKMat2;
Console.Write("Masukan baris matrix1 : ");
intBMat1 = Convert.ToInt32(Console.ReadLine());
Console.Write("Masukan kolom matrix1 dan baris matrix2 : ");
intKmlBm2 = Convert.ToInt32(Console.ReadLine());
Console.Write("Masukan kolom matrix2 : ");
intKMat2 = Convert.ToInt32(Console.ReadLine());
int[,] matrix1 = new int[intBMat1, intKmlBm2];
int[,] matrix2 = new int[intKmlBm2, intKMat2];
int[,] matrixHasil = new int[intBMat1, intKMat2];
for (int x = 0; x < intBMat1; x++)
{
    for (int y = 0; y < intKmlBm2; y++)
    {
        Console.Write
            ("Masukan matrix1[" + x + "," + y + "] : ");
        matrix1[x, y] =
            Convert.ToInt32(Console.ReadLine());
    }
}
for (int x = 0; x < intKmlBm2; x++)
{
    for (int y = 0; y < intKMat2; y++)
    {
        Console.Write
            ("Masukan matrix2[" + x + "," + y + "] : ");
        matrix2[x, y] = Convert.ToInt32(Console.ReadLine());
    }
}

```



```

}
Console.WriteLine();
for (int x = 0; x < intBMat1; x++)
{
    for (int y = 0; y < intKMat2; y++)
    {
        matrixHasil[x, y] = 0;
        for (int k = 0; k < intKmlBm2; k++)
        {
            matrixHasil[x, y] =
            matrixHasil[x, y] + matrix1[x, k] *
            matrix2[k, y];
        }
    }
}
Console.WriteLine();
for (int x = 0; x < intBMat1; x++)
{
    for (int y = 0; y < intKMat2; y++)
    {
        Console.Write(matrixHasil[x, y] + " ");
    }
    Console.WriteLine();
}

```

Latihan

1. Sebutkan komponen pendukung dari pemrograman grafis yang anda pelajari !
2. Sebutkan kepanjangand dari GDI+
3. Jelaskan fitur-fitur dari GDI+
4. Deskripsikan *namespace* dan class yang ada di dalam komponen grafis !

Daftar Pustaka

1. Edward Angel, Interactive Computer Graphics: A Top-Down Approach with OpenGL 2nd, Addison Wesley, 2005
2. Mahesh Chand, Graphics Programming with GDI+, Addison-Wesley, 2003
3. Ollie Cornes, Jay Glynn, Burton Harvey, Craig McQueen, Jerod Moemeka, Christian Nagel, Simon Robinson, Morgan Skinner, Karli Watson, Professional C# - Graphics with GDI+, Wrox, 2001

MODUL IX DASAR-DASAR PEMROGRAMAN GRAFIS

Setelah membaca modul ini, mahasiswa akan memiliki pengetahuan dan mampu menjelaskan (i) apa yang dimaksud area gambar atau *drawing area* dalam pemrograman grafis (ii) apa yang dimaksud dengan sistem koordinat dan implementasinya dalam sistem grafika (iii) bagaimana membuat program pertama aplikasi grafika komputer (iv) menggunakan struktur paling sederhana yaitu Point dan PointF dalam aplikasi grafika komputer (v) membuat program aplikasi grafis untuk menggambar objek-objek standar seperti garis, persegi, lingkaran dan sebagainya.

IX.1 Area Gambar

Setiap aplikasi *drawing* paling tidak terdiri dari tiga komponen yaitu *canvas*, *brush*, *pen*, dan *process* sebagai berikut:

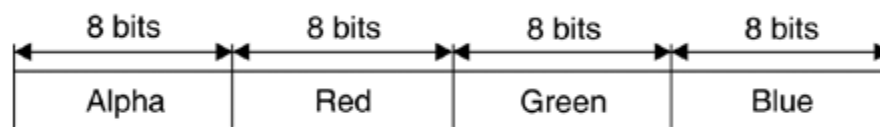
- *Canvas*: adalah area dimana object akan digambar, misal pada aplikasi Windows, Windows Form adalah canvasnya.
- *Brush* atau *Pen* merepresentasikan tekstur, warna, dan ukuran dari objek yang akan digambar pada canvas.
- *Process* mendefinisikan bagaimana object tersebut akan digambar kedalam canvas.

Setiap *area drawing* memiliki empat *property* utama yaitu: *weight*, *height*, *resolution* dan *color depth* sebagai berikut:

- *Width* dan *height property* digunakan untuk medeskripsikan ukuran *area drawing* secara vertikal dan horizontal
- *Resolution* adalah satuan untuk mengukur *output quality* dari *graphic object* atau *images* dalam satuan *dot per inch* (dpi). Sebagai contoh resolusi 72 dpi berarti 1 *inch* dari area tersebut terdiri dari 72 *horizontal* dan 72 *vertical pixel*. Untuk monitor resolusi 1280x1024 berarti pada area monitor tersebut terdiri dari 1280 *horizontal pixel* dan 1024 *vertical pixel*.
- *Color depth* adalah jumlah warna yang digunakan untuk merepresentasikan setiap *pixel*.

Definisi dari *pixel* adalah: elemen terkecil pada proses drawing untuk menampilkan *graphic object* atau *images* pada layar. *Pixel density* sering direpresentasikan dengan nilai dpi (*dot per inch*).

Struktur *color* pada GDI+ mempunyai empat komponen warna yaitu: *alpha*, *red*, *green*, dan *blue*. Ketiga komponen dikenal sebagai RGB mewakili kombinasi warna yang akan muncul. Setiap komponen dalam RGB mempunyai 256 *color combination*, sehingga kombinasi tiga komponen RGB tersebut mempunyai kemungkinan warna 256x256x256. Alpha komponen mewakili aspek transparan dari warna, yang akan terlihat ketika beberapa warna digabungkan.



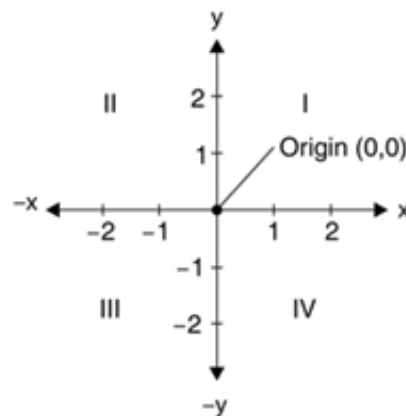
Gambar 9.1 Struktur Warna dalam GDI+

Dari gambar di atas dapat diasumsikan bahwa *area drawing* paling tidak harus mendukung 24-bit *color system* (8-bit untuk setiap R, G, dan B komponen).

IX.2 Sistem Koordinat

Memahami sistem koordinat sangat penting untuk pemrograman grafika, sistem koordinat merepresentasikan letak *graphic object* pada perangkat tampilan seperti monitor dan *printer*.

Sistem Koordinat Kartesian merupakan sistem koordinat standar yang umum digunakan.



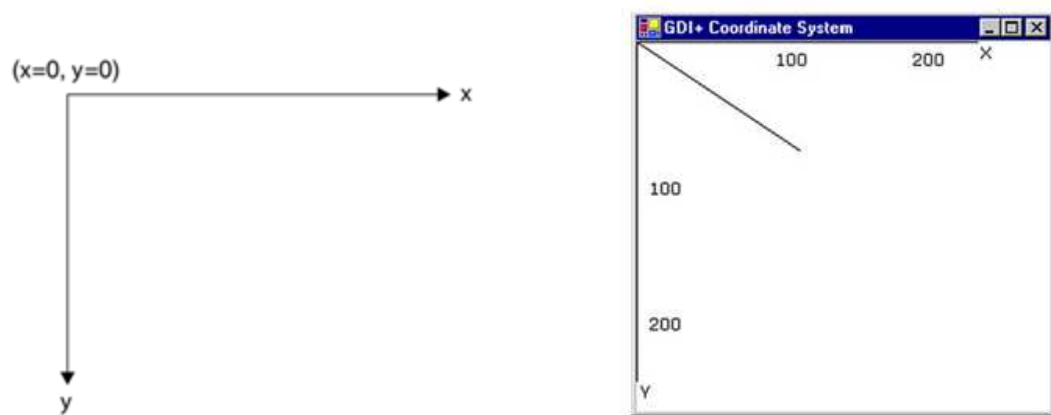
Gambar 9.2 Sistem Koordinat Biasa/Umum

Posisi sistem koordinat kartesian dibagi menjadi beberapa kuadran:

- Kuadran I: $x > 0$ dan $y > 0$
- Kuadran II: $x < 0$ dan $y > 0$
- Kuadran III: $x < 0$ dan $y < 0$
- Kuadran IV: $x > 0$ dan $y < 0$
-

Sistem Koordinat Standar GDI+

Tidak seperti sistem kartesian, pada perangkat tampilan seperti monitor sumbu $(0,0)$ terletak pada pojok kiri atas.



Gambar 9.3 Sistem Koordinat GDI+

IX.3 Membuat Program Pertama Aplikasi Grafika Komputer

Pada bagian ini akan dijelaskan salah satu cara bagaimana membuat aplikasi menggunakan library GDI+ menggunakan .NET Framework.

1. Pertama buat aplikasi windows form baru pada *visual studio*, beri nama projectnya program1
2. Untuk menggunakan GDI+ maka anda harus menambahkan referensi *library* kedalam program sebagai berikut

```
using System.Drawing;  
using System.Drawing.Drawing2D;
```

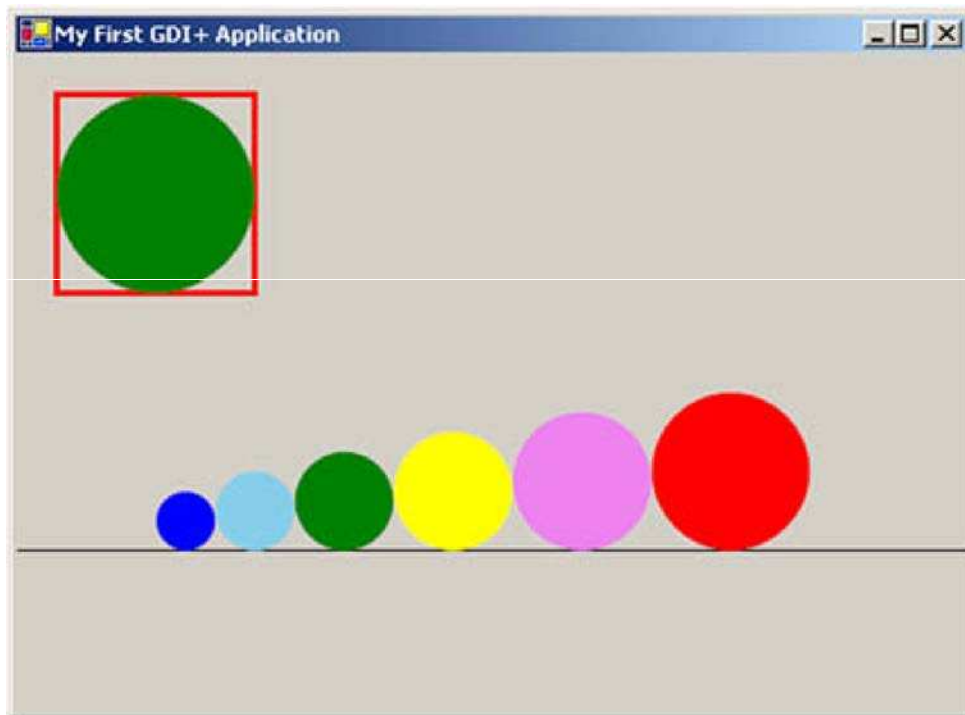
3. Untuk menggambar di form anda dapat menambahkan kode penambahan instant object dari class graphic dalam method OnPaint sebagai berikut:

```
private void Form1_Paint(object sender, PaintEventArgs e)  
{  
    Graphics g = e.Graphics;  
}
```

4. Untuk menambahkan object Pen, Brush, serta Drawing Shape kedalam form tambahkan kode sebagai berikut:

```
private void Form1_Paint(object sender, PaintEventArgs e)  
{  
    Graphics g = e.Graphics;  
    g.SmoothingMode = SmoothingMode.AntiAlias;  
    Rectangle rect = new Rectangle(20, 20, 100, 100);  
    Pen redPen = new Pen(Color.Red, 3);  
    Pen blackPen = Pens.Black;  
    SolidBrush greenBrush = new SolidBrush(Color.Green);  
    g.DrawRectangle(redPen, rect);  
    g.FillEllipse(greenBrush, rect);  
    g.DrawLine(blackPen, 0, 250, this.Width, 250);  
    g.FillEllipse(Brushes.Blue, 70, 220, 30, 30);  
    g.FillEllipse(Brushes.SkyBlue, 100, 210, 40, 40);  
    g.FillEllipse(Brushes.Green, 140, 200, 50, 50);  
    g.FillEllipse(Brushes.Yellow, 190, 190, 60, 60);  
    g.FillEllipse(Brushes.Violet, 250, 180, 70, 70);  
    g.FillEllipse(Brushes.Red, 320, 170, 80, 80);  
}
```

5. Jalankan program dengan menekan tombol F5 untuk melihat hasilnya sebagai berikut:



Objek Dasar GDI+

Berikut ini adalah daftar property dari Color structure

Property	Description
Red, Blue, Green, Aqua, Azure,	A specified color static property for almost every color.
A	Returns the alpha component value in a Color structure.
R	Returns the red component value in a Color structure.
G	Returns the green component value in a Color structure.
B	Returns the blue component value in a Color structure.
IsEmpty	Indicates whether a Color structure is uninitialized.
IsKnownColor	Indicates whether a color is predefined.
IsNamedColor	Indicates whether a color is predefined.
IsSystemColor	Indicates whether a color is a system color.
Name.	Returns the name of the color.

Berikut adalah daftar method dari Color structure

Method	Description
FromArgb	Creates a Color structure from the four 8-bit ARGB component (alpha-red-green-blue) values.
FromKnownColor	Creates a Color structure from the specified predefined color.
FromName	Creates a Color structure from the specified name of a predefined color.
GetBrightness	Returns the hue-saturation-brightness (HSB) brightness value of this Color structure.
GetHue	Returns the HSB hue value, in degrees, of this Color structure.
GetSaturation	Returns the HSB saturation value of this Color structure.
ToArgb	Returns the 32-bit ARGB value of this Color structure.
ToKnownColor	Returns the KnownColor value of this Color structure.

IX.4 Menggunakan Struktur Point dan PointF

Pada GDI+ point digunakan untuk mendefinisikan koordinat titik dua dimensi yang direpresentasikan sebagai matrix (x dan y koordinat). Terdapat tiga macam konstruktor yang dapat digunakan untuk membuat object point yaitu:

```
Point pt1 = new Point(10);  
  
Point pt2 = new Point( new Size(20, 20) );  
  
Point pt3 = new Point(30, 30);
```

Struktur PointF hampir sama dengan Point hanya saja nilai sumbu x dan y-nya tidak berupa integer melainkan floating point.

```
PointF pt3 = new PointF(30.0f, 30.0f);
```

Program di bawah menunjukkan contoh penggunaan struktur Point.

```
private void Form1_Paint(object sender, PaintEventArgs e)
{
    Graphics g = e.Graphics;
    Point pt = new Point(50, 50);
    Point newPoint = Point.Empty;
    newPoint.X = 100;
    newPoint.Y = 200;
    Pen pn = new Pen(Color.Blue, 4);
    g.DrawLine(pn, pt, newPoint);
    pn.Dispose();
    g.Dispose();
}
```

Selain struktur Point dan PointF terdapat struktur lain yaitu Rectangle. Program di bawah ini menggambarkan penggunaan struktur tersebut.

```
private void Form1_Paint(object sender, PaintEventArgs e)
{
    Graphics g = this.CreateGraphics();
    int x = 40;
    int y = 40;
    int height = 120;
    int width = 120;
    Point pt = new Point(80, 80);
    Size sz = new Size(100, 100);
    Rectangle rect1 = new Rectangle(pt, sz);
    Rectangle rect2 =
    new Rectangle(x, y, width, height);
    Rectangle rect3 =
    new Rectangle(10, 10, 180, 180);
    Pen redPen = new Pen(Color.Red, 2);
    SolidBrush greenBrush =
    new SolidBrush(Color.Blue);
    SolidBrush blueBrush =
    new SolidBrush(Color.Green);
    g.DrawRectangle(redPen, rect3);
    g.FillRectangle(blueBrush, rect2);
    g.FillRectangle(greenBrush, rect1);
    redPen.Dispose();
    blueBrush.Dispose();
    greenBrush.Dispose();
    g.Dispose();
}
```

Struktur Rectangle memiliki beberapa method yang bermanfaat dalam pengoperasiannya. Method-method itu adalah sebagai berikut:

Method	Description
Ceiling	Converts a <code>RectangleF</code> object to a <code>Rectangle</code> object by rounding the <code>RectangleF</code> values to the next higher integer values.
Contains	Determines if the specified point is contained within the rectangular region of a rectangle.
FromLTRB.	Creates a rectangle with the specified edge locations.
Inflate	Creates and returns an inflated copy of a rectangle.
Intersect	Replaces a rectangle with the intersection of itself and another rectangle.
IntersectsWith	Determines if a specified rectangle intersects with <code>rect</code> .
Offset	Adjusts the location of a specified rectangle by the specified amount.
Round	Converts a <code>RectangleF</code> object to a <code>Rectangle</code> object by rounding the <code>RectangleF</code> values to the nearest integer values.
Truncate	Truncate Converts a <code>RectangleF</code> object to a <code>Rectangle</code> object by truncating the <code>RectangleF</code> values.
Union	Union Returns a rectangle that contains the union of two <code>Rectangle</code> structures.

IX.5 Menggambar Objek-objek Grafis Lainnya

Untuk menggambar objek-objek grafis lain, digunakan method yang ada di dalam kelas grafik. Metode-metode itu adalah sebagai berikut:

Method	Description
DrawArc	<code>DrawArc</code> Draws an arc (a portion of an ellipse specified by a pair of coordinates, a width, a height, and start and end angles).
DrawBezier.	Draws a Bézier curve defined by four <code>Point</code> structures.

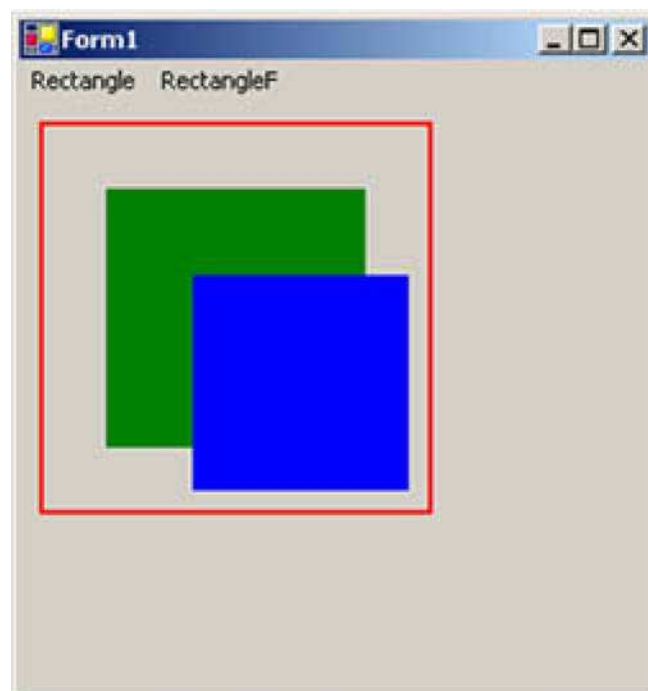
<code>DrawBeziers</code>	Draws a series of Bézier splines from an array of <code>Point</code> structures.
<code>DrawClosedCurve.</code>	Draws a closed cardinal spline defined by an array of <code>Point</code> structures.
<code>DrawCurve</code>	Draws a cardinal spline through a specified array of <code>Point</code> structures.
<code>DrawEllipse</code>	Draws an ellipse defined by a bounding rectangle specified by a pair of coordinates, a height, and a width.
<code>DrawIcon</code>	Draws an image represented by the specified <code>Icon</code> object at the specified coordinates.
<code>DrawIconUnstretched</code>	<code>DrawIconUnstretched</code> Draws an image represented by the specified <code>Icon</code> object without scaling the image.
<code>DrawImage</code>	Draws the specified <code>Image</code> object at the specified location and with the original size.
<code>DrawImageUnscaled</code>	<code>DrawImageUnscaled</code> Draws the specified <code>Image</code> object with its original size at the location specified by a coordinate pair.
<code>DrawLine</code>	Draws a line connecting two points specified by coordinate pairs.
<code>DrawLines</code>	Draws a series of line segments that connect an array of <code>Point</code> structures.
<code>DrawPath.</code>	Draws a <code>GraphicsPath</code> object.
<code>DrawPie</code>	<code>DrawPie</code> Draws a pie shape specified by a coordinate pair, a width, a height, and two radial lines.
<code>DrawPolygon</code>	Draws a polygon defined by an array of <code>Point</code> structures.
<code>DrawRectangle</code>	Draws a rectangle specified by a coordinate pair, a width, and a height.
<code>DrawRectangles</code>	Draws a series of rectangles specified by an array of <code>Rectangle</code> structures.
<code>DrawString</code>	Draws the specified text string at the specified location using the specified <code>Brush</code> and <code>Font</code> objects.

Program Menggambar Garis

```
private void Form1_Paint(object sender, PaintEventArgs e)
{
    Graphics g = e.Graphics;
    Pen redPen = new Pen(Color.Red, 1);
    g.DrawRectangle(bluePen,
    x, y, width, height);
    g.DrawRectangle(redPen, 60, 80, 140, 50);
    g.DrawRectangle(greenPen, rect);
    redPen.Dispose();
    bluePen.Dispose();
    greenPen.Dispose();
}
```

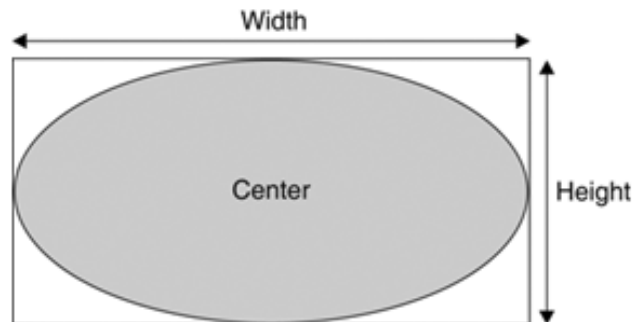
Menggambar Beberapa Rectangle Sekaligus

```
private void Form1_Paint(object sender, PaintEventArgs e)
{
    Graphics g = e.Graphics;
    Pen greenPen = new Pen(Color.Green, 4);
    RectangleF[] rectArray =
    {
        new RectangleF( 5.0F, 5.0F, 100.0F, 200.0F),
        new RectangleF(20.0F, 20.0F, 80.0F, 40.0F),
        new RectangleF(60.0F, 80.0F, 140.0F, 50.0F)
    };
    g.DrawRectangles(greenPen, rectArray);
    greenPen.Dispose();
}
```



Menggambar Elips dan Lingkaran

Bentuk elips dan lingkaran memiliki karakteristik yang serupa. Perbedaannya hanya terletak pada ukuran lebar (width) dan tinggi (height). Untuk lingkaran, lebar sama dengan tinggi, sedangkan untuk elips lebar tidak sama dengan tinggi. Dalam bahasa Matematika, lebar dan tinggi adalah sumbu semi-mayor dan sumbu semi-minor.



Gambar 9.4 Anatomi Ellips

Program berikut menunjukkan penggambaran elips.

```
private void Form1_Paint(object sender, PaintEventArgs e)
{
    Graphics g = e.Graphics;
    Pen redPen = new Pen(Color.Red, 6);
    Pen bluePen = new Pen(Color.Blue, 4);
    Pen greenPen = new Pen(Color.Green, 2);
    Rectangle rect = new Rectangle(80, 80, 50, 50);
    g.DrawEllipse(greenPen, 100.0F, 100.0F, 10.0F, 10.0F);
    g.DrawEllipse(redPen, rect);
    g.DrawEllipse(bluePen, 60, 60, 90, 90);
    g.DrawEllipse(greenPen, 40.0F, 40.0F, 130.0F, 130.0F);
    redPen.Dispose();
    greenPen.Dispose();
    bluePen.Dispose();
}
```

Menggambar atau menuliskan teks dalam mode grafis

Teks atau string dapat dituliskan dalam mode grafis. Umumnya penulisan ini dilakukan pada objek grafis lainnya. Program berikut menunjukkan penulisan teks secara grafis.

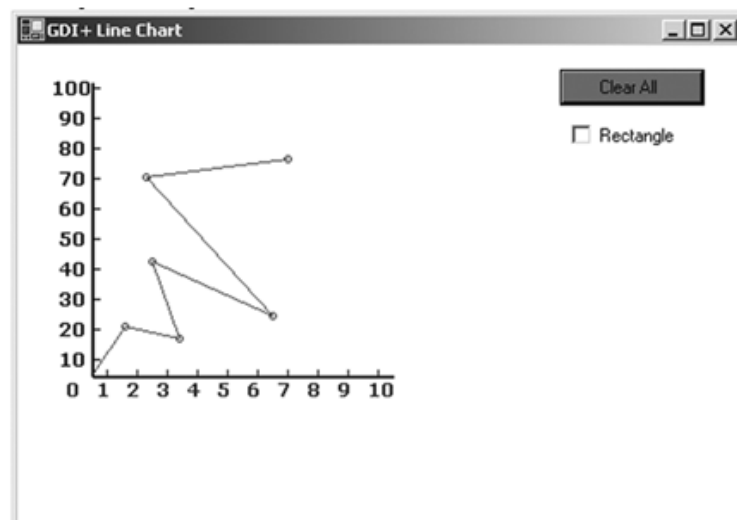
```

private void Form1_Paint(object sender, PaintEventArgs e)
{
    Graphics g = e.Graphics;
    SolidBrush blueBrush = new SolidBrush(Color.Blue);
    SolidBrush redBrush = new SolidBrush(Color.Red);
    SolidBrush greenBrush = new SolidBrush(Color.Green);
    Rectangle rect = new Rectangle(20, 20, 200, 100);
    String drawString = "Menulis teks dalam mode grafis";
    Font drawFont = new Font("Verdana", 14);
    float x = 100.0F;
    float y = 100.0F;
    StringFormat drawFormat = new StringFormat();
    drawFormat.FormatFlags =
        StringFormatFlags.DirectionVertical;
    g.DrawString("Drawing text",
        new Font("Tahoma", 14), greenBrush, rect);
    g.DrawString(drawString,
        new Font("Arial", 12), redBrush, 120, 140);
    g.DrawString(drawString, drawFont,
        blueBrush, x, y, drawFormat);
    blueBrush.Dispose();
    redBrush.Dispose();
    greenBrush.Dispose();
    drawFont.Dispose();
}

```

Aplikasi Diagram Garis

Diagram garis adalah salah satu cara merepresentasikan data numerik dalam bentuk visual. Pada paket-paket aplikasi seperti Excel, SPSS, Matlab, fasilitas ini sudah tersedia dan tinggal memakainya. Pada bagian ini akan dibahas bagaimana membuat programnya.



1. Deklarasikan dua variabel untuk start dan end point

```
private Point startPoint = new Point(50, 217);  
private Point endPoint = new Point(50, 217);
```

2. Buat sumbu koordinat secara manual dengan menggambar line dan string

```
private void Form1_Paint(object sender, PaintEventArgs e)  
{  
    Graphics g = e.Graphics;  
    Font vertFont =  
        new Font("Verdana", 10, FontStyle.Bold);  
    Font horzFont =  
        new Font("Verdana", 10, FontStyle.Bold);  
    SolidBrush vertBrush = new SolidBrush(Color.Black);  
    SolidBrush horzBrush = new SolidBrush(Color.Blue);  
    Pen blackPen = new Pen(Color.Black, 2);  
    Pen bluePen = new Pen(Color.Blue, 2);  
    g.DrawLine(blackPen, 50, 220, 50, 25);  
    g.DrawLine(bluePen, 50, 220, 250, 220);  
    g.DrawString("0", horzFont, horzBrush, 30, 220);  
    g.DrawString("1", horzFont, horzBrush, 50, 220);  
    g.DrawString("2", horzFont, horzBrush, 70, 220);  
    g.DrawString("3", horzFont, horzBrush, 90, 220);  
    g.DrawString("4", horzFont, horzBrush, 110, 220);  
    g.DrawString("5", horzFont, horzBrush, 130, 220);  
    g.DrawString("6", horzFont, horzBrush, 150, 220);  
    g.DrawString("7", horzFont, horzBrush, 170, 220);  
    g.DrawString("8", horzFont, horzBrush, 190, 220);  
    g.DrawString("9", horzFont, horzBrush, 210, 220);  
    g.DrawString("10", horzFont, horzBrush, 230, 220);  
    StringFormat vertStrFormat = new StringFormat();  
    vertStrFormat.FormatFlags =  
        StringFormatFlags.DirectionVertical;  
    g.DrawString("-", horzFont, horzBrush,  
        50, 212, vertStrFormat);  
    g.DrawString("-", horzFont, horzBrush,  
        70, 212, vertStrFormat);  
    g.DrawString("-", horzFont, horzBrush,  
        90, 212, vertStrFormat);  
    g.DrawString("-", horzFont, horzBrush,  
        110, 212, vertStrFormat);  
    g.DrawString("-", horzFont, horzBrush,  
        130, 212, vertStrFormat);  
    g.DrawString("-", horzFont, horzBrush,  
        150, 212, vertStrFormat);  
    g.DrawString("-", horzFont, horzBrush,  
        170, 212, vertStrFormat);  
    g.DrawString("-", horzFont, horzBrush,  
        190, 212, vertStrFormat);
```

```

g.DrawString("-", horzFont, horzBrush,
    210, 212, vertStrFormat);
g.DrawString("-", horzFont, horzBrush,
g.DrawString("100-", vertFont, vertBrush, 20, 20);
g.DrawString("90 -", vertFont, vertBrush, 25, 40);
g.DrawString("80 -", vertFont, vertBrush, 25, 60);
g.DrawString("70 -", vertFont, vertBrush, 25, 80);
g.DrawString("60 -", vertFont, vertBrush, 25, 100);
g.DrawString("50 -", vertFont, vertBrush, 25, 120);
g.DrawString("40 -", vertFont, vertBrush, 25, 140);
g.DrawString("30 -", vertFont, vertBrush, 25, 160);
g.DrawString("20 -", vertFont, vertBrush, 25, 180);
g.DrawString("10 -", vertFont, vertBrush, 25, 200);
vertFont.Dispose();
horzFont.Dispose();
vertBrush.Dispose();
horzBrush.Dispose();
blackPen.Dispose();
bluePen.Dispose();
}

```

3. Pada saat event mouse down akan digambar line yang berawal dari titik (0,0)

```

if (e.Button == MouseButton.Left)
{
    Graphics g1 = this.CreateGraphics();
    Pen linePen = new Pen(Color.Green, 1);
    Pen ellipsePen = new Pen(Color.Red, 1);
    startPoint = endPoint;
    endPoint = new Point(e.X, e.Y);
    g1.DrawLine(linePen, startPoint, endPoint);
    if (checkBox1.Checked)
    {
        g1.DrawRectangle(ellipsePen,
            e.X - 2, e.Y - 2, 4, 4);
    }
    else
    {
        g1.DrawEllipse(ellipsePen,
            e.X - 2, e.Y - 2, 4, 4);
    }
    linePen.Dispose();
    ellipsePen.Dispose();
    g1.Dispose();
}

```

4. Tombol “Clear All” digunakan untuk menghapus gambar line chart, kodenya adalah sebagai berikut:

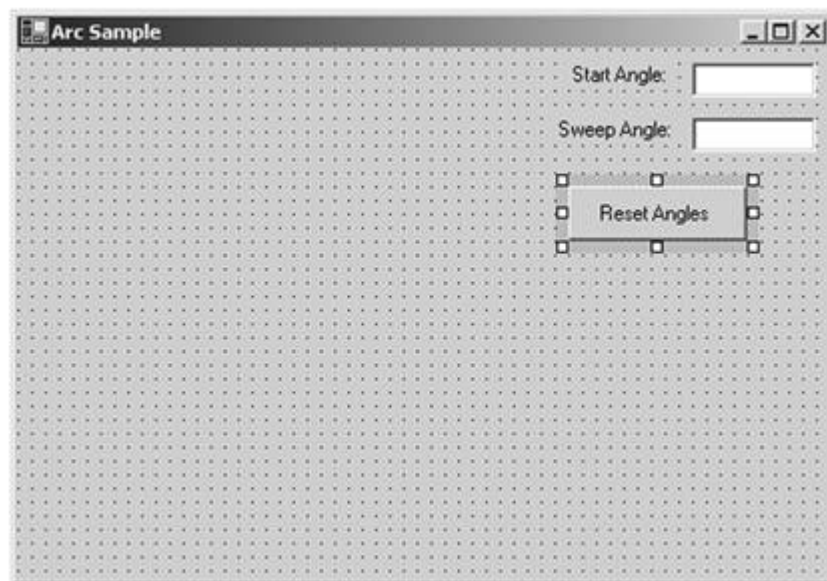
```

private void btnAll_Click(object sender, EventArgs e)
{
    startPoint.X = 50;
    startPoint.Y = 217;
    endPoint.X = 50;
    endPoint.Y = 217;
    this.Invalidate(this.ClientRectangle);
}

```

Menggambar Lengkungan (arc)

Arc adalah bagian dari elips, yang dibentuk dari area rectangle, start angle, dan sweep angle. Cara penggunaan Arc adalah sebagai berikut:



```

private void Form1_Paint(object sender, PaintEventArgs e)
{
    Graphics g = e.Graphics;
    Pen redPen = new Pen(Color.Red, 3);
    Rectangle rect = new Rectangle(20, 20, 200, 200);
    g.DrawArc(redPen, rect, startAngle, sweepAngle);
    redPen.Dispose();
}
private void btnReset_Click(object sender, EventArgs e)
{
    startAngle = (float)Convert.ToDouble(txtStart.Text);
    sweepAngle = (float)Convert.ToDouble(txtSweep.Text);
    Invalidate();
}

```


Menggambar Kurva

Terdapat dua jenis kurva yaitu kurva terbuka dan kurva tertutup.



```
private void Form1_Paint(object sender, PaintEventArgs e)
{
    Graphics g = e.Graphics;
    Pen bluePen = new Pen(Color.Blue, 1);
    PointF pt1 = new PointF(40.0F, 50.0F);
    PointF pt2 = new PointF(50.0F, 75.0F);
    PointF pt3 = new PointF(100.0F, 115.0F);
    PointF pt4 = new PointF(200.0F, 180.0F);
    PointF pt5 = new PointF(200.0F, 90.0F);
    PointF[] ptsArray =
    {
        pt1, pt2, pt3, pt4, pt5
    };
    g.DrawCurve(bluePen, ptsArray);
    bluePen.Dispose();
}
```

Menggambar Kurva dengan Tensi

```
private float tension = 0.5F;
private void Form1_Paint(object sender, PaintEventArgs e)
{
    Graphics g = e.Graphics;
    Pen bluePen = new Pen(Color.Blue, 1);
    PointF pt1 = new PointF(40.0F, 50.0F);
    PointF pt2 = new PointF(50.0F, 75.0F);
    PointF pt3 = new PointF(100.0F, 115.0F);
    PointF pt4 = new PointF(200.0F, 180.0F);
    PointF pt5 = new PointF(200.0F, 90.0F);
    PointF[] ptsArray =
    {
        pt1, pt2, pt3, pt4, pt5
    };
    g.DrawCurve(bluePen, ptsArray, tension);
    bluePen.Dispose();
}
```

```
private void btnApply_Click(object sender, EventArgs e)
{
    tension = (float)Convert.ToDouble(txtTension.Text);
    Invalidate();
}

```

Menggambar Kurva Tertutup

```
private void Form1_Paint(object sender, PaintEventArgs e)
{
    Graphics g = e.Graphics;
    Pen bluePen = new Pen(Color.Blue, 1);
    PointF pt1 = new PointF(40.0F, 50.0F);
    PointF pt2 = new PointF(50.0F, 75.0F);
    PointF pt3 = new PointF(100.0F, 115.0F);
    PointF pt4 = new PointF(200.0F, 180.0F);
    PointF pt5 = new PointF(200.0F, 90.0F);
    PointF[] ptsArray =
    {
        pt1, pt2, pt3, pt4, pt5
    };
    g.DrawClosedCurve(bluePen, ptsArray);
    bluePen.Dispose();
}

```

Menggambar Poligon

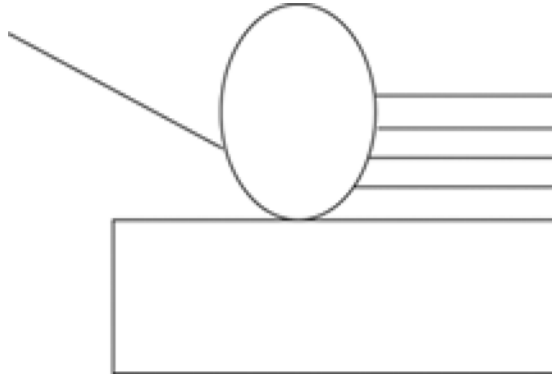
Poligon adalah bentuk (shape) yang terdiri dari tiga atau lebih garis, misalnya segitiga dan persegi.

```
private void Form1_Paint(object sender, PaintEventArgs e)
{
    Graphics g = e.Graphics;
    Pen greenPen = new Pen(Color.Green, 2);
    Pen redPen = new Pen(Color.Red, 2);
    PointF p1 = new PointF(40.0F, 50.0F);
    PointF p2 = new PointF(60.0F, 70.0F);
    PointF p3 = new PointF(80.0F, 34.0F);
    PointF p4 = new PointF(120.0F, 180.0F);
    PointF p5 = new PointF(200.0F, 150.0F);
    PointF[] ptsArray =
    {
        p1, p2, p3, p4, p5
    };
    g.DrawPolygon(greenPen, ptsArray);
    greenPen.Dispose();
    redPen.Dispose();
}

```

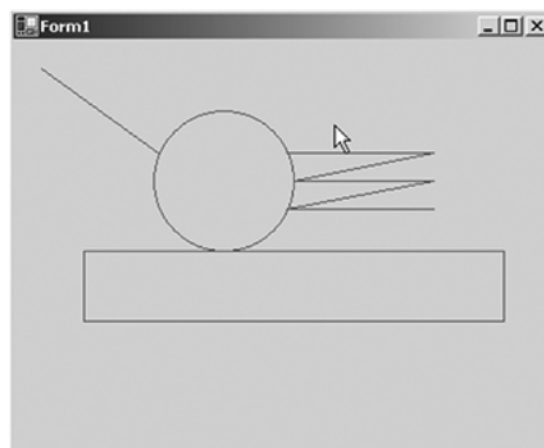
Menggambar Graphic Path

Graphic Path menghubungkan beberapa objek drawing seperti line, rectable, circle dan sebagainya. Gambar di bawah ini mengilustrasikan Graphics Path.

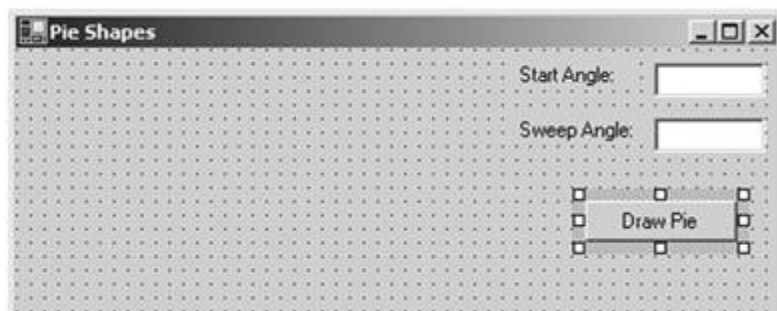
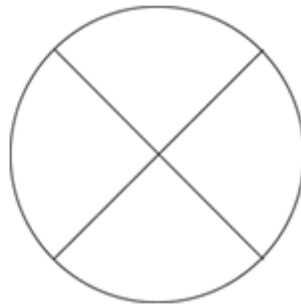


```
using System.Drawing;
using System.Drawing.Drawing2D;
private void Form1_Paint(object sender, PaintEventArgs e)
{
    Graphics g = e.Graphics;
    Pen greenPen = new Pen(Color.Green, 1);
    GraphicsPath path = new GraphicsPath();
    path.AddLine(20, 20, 103, 80);
    path.AddEllipse(100, 50, 100, 100);
    path.AddLine(195, 80, 300, 80);
    path.AddLine(200, 100, 300, 100);
    path.AddLine(195, 120, 300, 120);
    Rectangle rect =
        new Rectangle(50, 150, 300, 50);
    path.AddRectangle(rect);
    g.DrawPath(greenPen, path);
    greenPen.Dispose();
}
```

Output dari program ini adalah sebagai berikut:

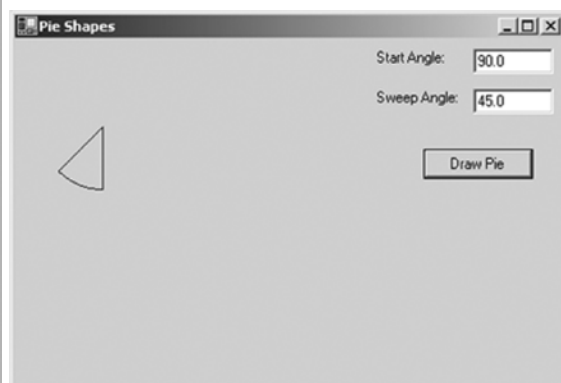
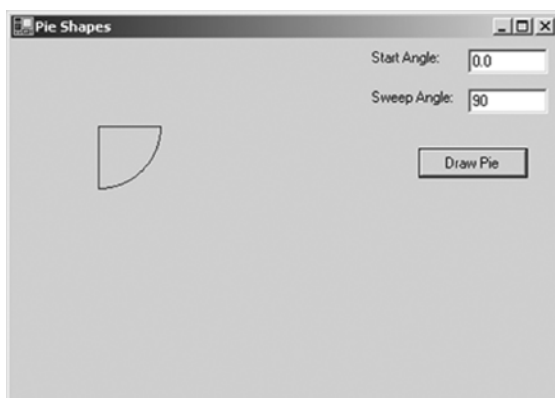


Menggambar Bentuk Pie



```
private void btnDraw_Click(object sender, EventArgs e)
{
    Graphics g = this.CreateGraphics();
    g.Clear(this.BackColor);
    float startAngle =
        (float)Convert.ToDouble(textBox1.Text);
    float sweepAngle =
        (float)Convert.ToDouble(textBox2.Text);
    Pen bluePen = new Pen(Color.Blue, 1);
    g.DrawPie(bluePen, 20, 20, 100, 100,
        startAngle, sweepAngle);

    bluePen.Dispose();
    g.Dispose();
}
```



Filling Objek-objek Grafis

Objek-objek grafis yang dibuat sebelumnya adalah objek grafis berongga dimana yang muncul adalah kerangka grafisnya. Untuk mengisi rongga interior, maka dilakukan proses filling menggunakan metode-metode sebagai berikut. Sama seperti pembuatan objek lainnya, method-method ini juga memiliki property atribut yang sama.

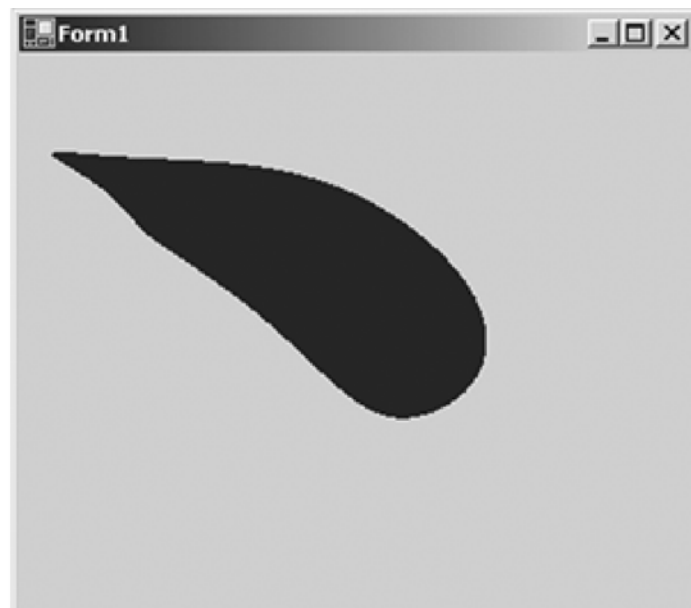
Method	Description
FillClosedCurve.	Fills the interior of a closed cardinal spline curve defined by an array of Point structures.
FillEllipse	Fills the interior of an ellipse defined by a bounding rectangle specified by a pair of coordinates, a width, and a height.
FillPath	Fills the interior of a GraphicsPath object.
FillPie	Fills the interior of a pie section defined by an ellipse specified by a pair of coordinates, a width, a height, and two radial lines.
FillPolygon	Fills the interior of a polygon defined by an array of points specified by Point structures.
FillRectangle	Fills the interior of a rectangle specified by a pair of coordinates, a width, and a height.
FillRectangles	Fills the interiors of a series of rectangles specified by Rectangle structures.
FillRegion	Fills the interior of a Region object.
FillClosedCurve	Fills the interior of a closed cardinal spline curve defined by an array of Point structures.
FillEllipse	Fills the interior of an ellipse defined by a bounding rectangle specified by a pair of coordinates, a width, and a height.
FillPath	Fills the interior of a GraphicsPath object.
FillPie	FillPie Fills the interior of a pie section defined by an ellipse specified by a pair of coordinates, a width, a height, and two radial line
FillPolygon	Fills the interior of a polygon defined by an array of points specified by Point structures.

FillRectangle	Fills the interior of a rectangle specified by a pair of coordinates, a width, and a height.
FillRectangles	Fills the interiors of a series of rectangles specified by Rectangle structures.
FillRegion	Fills the interior of a Region object.

Berikut ini beberapa contoh program yang menggunakan filling disertai dengan hasilnya.

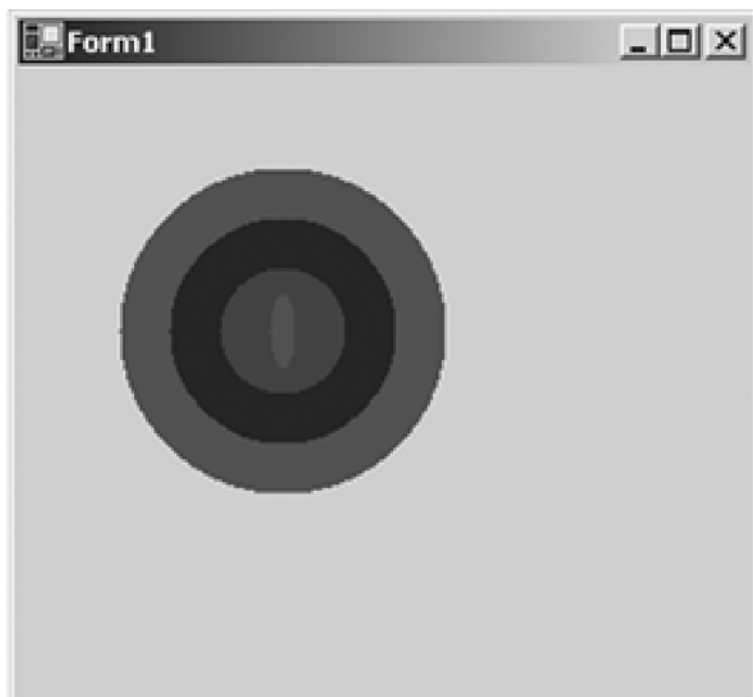
Filling Kurva Tertutup

```
private void Form1_Paint(object sender,
    System.Windows.Forms.PaintEventArgs e)
{
    PointF pt1 = new PointF( 40.0F, 50.0F);
    PointF pt2 = new PointF(50.0F, 75.0F);
    PointF pt3 = new PointF(100.0F, 115.0F);
    PointF pt4 = new PointF(200.0F, 180.0F);
    PointF pt5 = new PointF(200.0F, 90.0F);
    PointF[] ptsArray =
    {
        pt1, pt2, pt3, pt4, pt5
    };
    float tension = 1.0F;
    FillMode flMode = FillMode.Alternate;
    SolidBrush blueBrush = new SolidBrush(Color.Blue);
    e.Graphics.FillClosedCurve(blueBrush, ptsArray,
    flMode, tension);
    blueBrush.Dispose();
}
```



Filling Ellips

```
private void Form1_Paint(object sender,  
    System.Windows.Forms.PaintEventArgs e)  
{  
    Graphics g = e.Graphics ;  
    SolidBrush redBrush = new SolidBrush(Color.Red);  
    SolidBrush blueBrush = new SolidBrush(Color.Blue);  
    SolidBrush greenBrush = new SolidBrush(Color.Green);  
    Rectangle rect =  
    new Rectangle(80, 80, 50, 50);  
    g.FillEllipse(greenBrush, 40.0F, 40.0F, 130.0F, 130.0F );  
    g.FillEllipse(blueBrush, 60, 60, 90, 90);  
    g.FillEllipse(redBrush, rect );  
    g.FillEllipse(greenBrush, 100.0F, 90.0F, 10.0F, 30.0F );  
    blueBrush.Dispose();  
    redBrush.Dispose();  
    greenBrush.Dispose();  
}
```



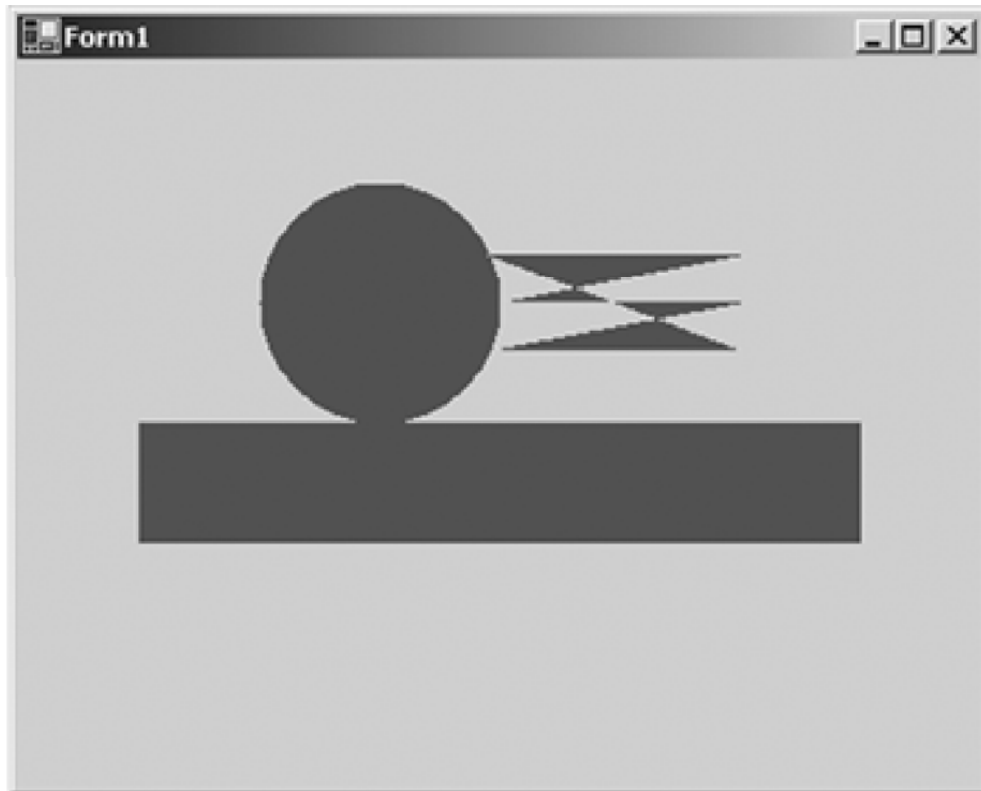
Filling sebuah graphics path

```
private void Form1_Paint(object sender,  
    System.Windows.Forms.PaintEventArgs e)  
{  
    SolidBrush greenBrush = new SolidBrush(Color.Green);  
    GraphicsPath path = new GraphicsPath();  
    path.AddLine(20, 20, 103, 80);  
    path.AddEllipse(100, 50, 100, 100);
```

```

path.AddLine(195, 80, 300, 80);
path.AddLine(200, 100, 300, 100);
path.AddLine(195, 120, 300, 120);
Rectangle rect = new Rectangle(50, 150, 300, 50);
path.AddRectangle(rect);
e.Graphics.FillPath(greenBrush, path);
greenBrush.Dispose();
}

```



Filling Polygon

```

private void Form1_Paint(object sender,
    System.Windows.Forms.PaintEventArgs e)
{
    Graphics g = e.Graphics ;
    SolidBrush greenBrush = new SolidBrush(Color.Green);
    PointF p1 = new PointF(40.0F, 50.0F);
    PointF p2 = new PointF(60.0F, 70.0F);
    PointF p3 = new PointF(80.0F, 34.0F);
    PointF p4 = new PointF(120.0F, 180.0F);
    PointF p5 = new PointF(200.0F, 150.0F);
    PointF[] ptsArray =
    {
        p1, p2, p3, p4, p5
    };
    e.Graphics.FillPolygon(greenBrush, ptsArray);
}

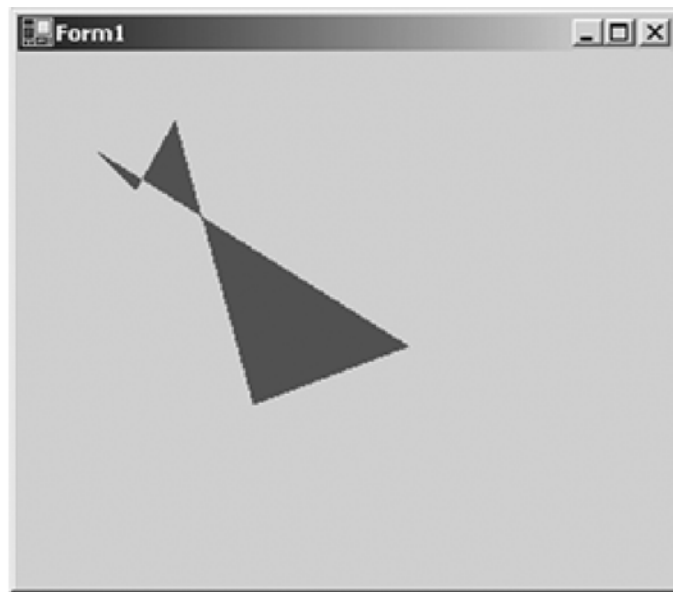
```



```

    greenBrush.Dispose();
}

```



Filling Rectangle

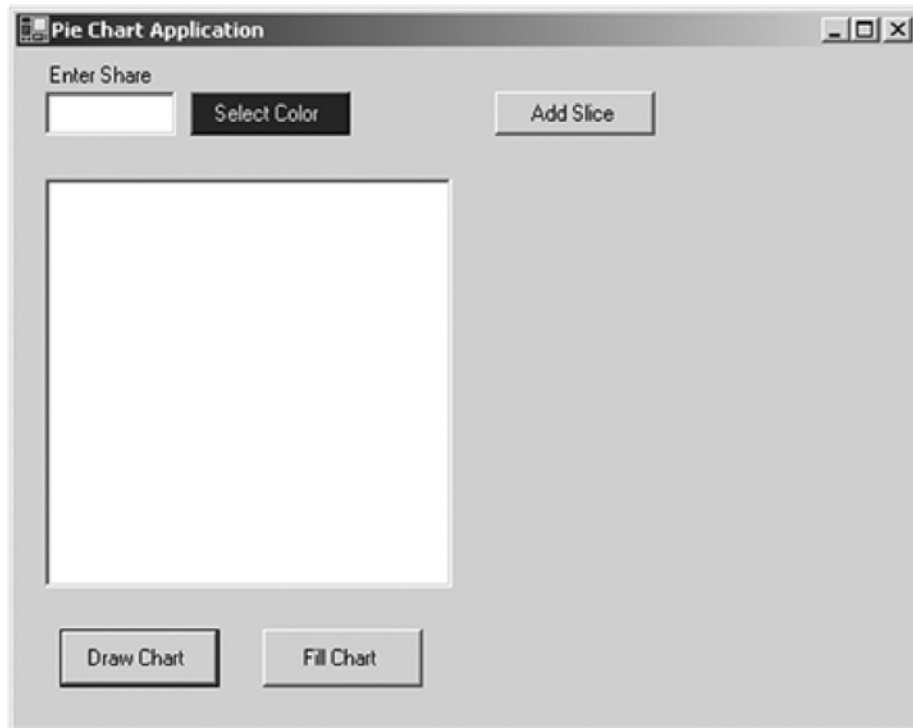
```

private void Form1_Paint(object sender,
    System.Windows.Forms.PaintEventArgs e)
{
    SolidBrush blueBrush = new SolidBrush(Color.Blue);
    Rectangle rect = new Rectangle(10, 20, 100, 50);
    e.Graphics.FillRectangle(new HatchBrush
        (HatchStyle.BackwardDiagonal, Color.Yellow, Color.Black),
        rect);
    e.Graphics.FillRectangle(blueBrush,
        new Rectangle(150, 20, 50, 100));
    blueBrush.Dispose();
}

```

Aplikasi Nyata Menggambar PieChart

Pie Chart adalah salah satu bentuk visualiasi data dalam bentuk pie/kue bulat, yang umumnya berupa lingkaran dengan potongan-potongan (slice). Aplikasi penggambaran grafis ini sudah banyak tersedia, namun bagaimana membuat programnya akan dipelajari disini. Bentuk tampilan programnya adalah sebagai berikut:



Data pie chart diambil dari textbox share, yang apabila button AddSlice di-click, maka akan muncul listbox. Programnya dan struktur datanya adalah sebagai berikut:

```
private Rectangle rect =
    new Rectangle(250, 150, 200, 200);
public ArrayList sliceList = new ArrayList();
struct sliceData
{
    public int share;
    public Color clr;
};
private Color curClr = Color.Black;
int shareTotal = 0;

private void button1_Click(object sender, System.EventArgs e)
{
    int slice = Convert.ToInt32(textBox1.Text);
    shareTotal += slice;
    sliceData dt;
    dt.clr = curClr;
    dt.share = slice;
    sliceList.Add(dt);
    listBox1.Items.Add("Share:" + slice.ToString() + " , " +
        curClr.ToString());
}
```

Pemilihan warna potongan dilakukan dengan menekan tombol Select Color yang programnya adalah sebagai berikut:

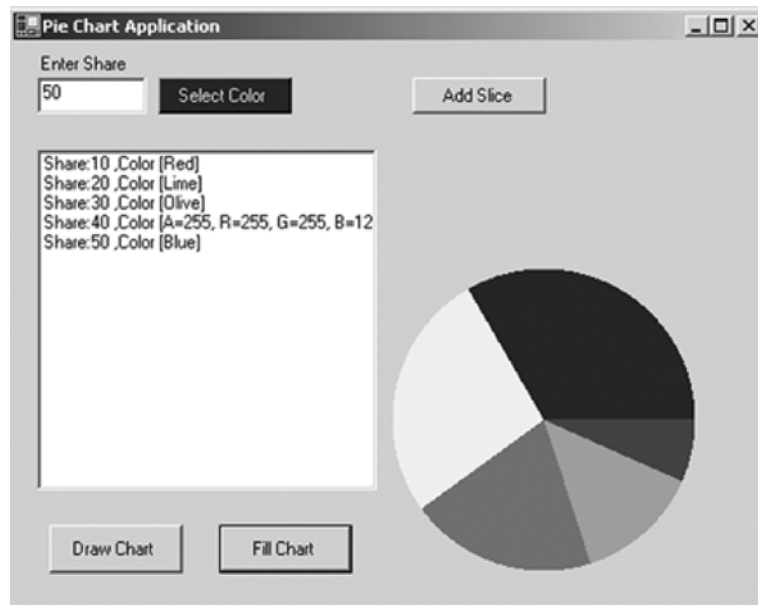
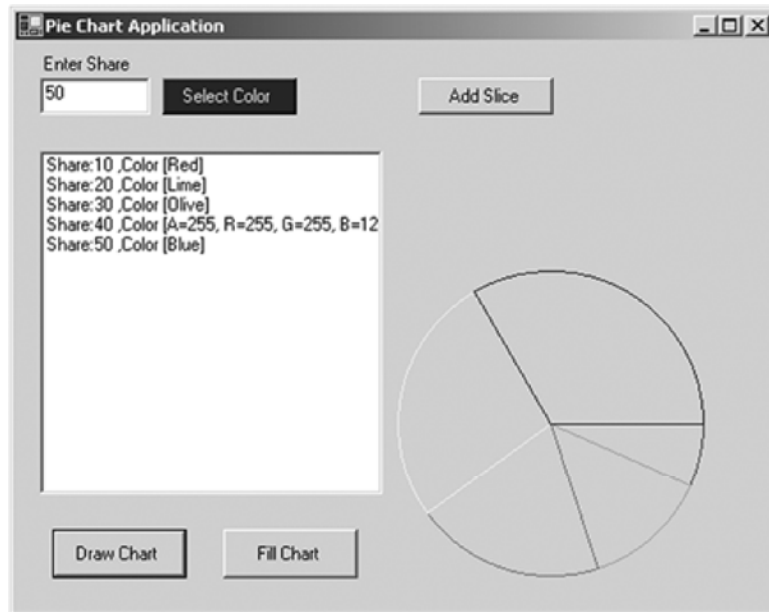
```
private void ColorBtn_Click(object sender, System.EventArgs e)
{
    ColorDialog clrDlg = new ColorDialog();
    if (clrDlg.ShowDialog() == DialogResult.OK)
    {
        curClr = clrDlg.Color;
    }
}
```

Penggambaran dan pengisian (filling) pie dilakukan dengan program berikut:

```
private void DrawPie_Click(object sender, System.EventArgs e)
{
    DrawPieChart(false);
}
private void FillChart_Click(object sender,
    System.EventArgs e)
{
    DrawPieChart(true);
}

private void DrawPieChart(bool flMode)
{
    Graphics g = this.CreateGraphics();
    g.Clear(this.BackColor);
    Rectangle rect = new Rectangle(250, 150, 200, 200);
    float angle = 0;
    float sweep = 0;
    foreach(sliceData dt in sliceList)
    {
        sweep = 360f * dt.share / shareTotal;
        if(flMode)
            g.FillPie(new SolidBrush(dt.clr), rect,
                angle, sweep);
        else
            g.DrawPie(new Pen(dt.clr), rect, angle, sweep);
        angle += sweep;
    }
    g.Dispose();
}
```

Hasil atau output dari program berupa Draw Chart atau Fill Chart adalah sebagai berikut:



Latihan

1. Apakah perbedaan sistem koordinat Cartesian dengan sistem koordinat GDI+?
2. Jelaskan struktur warna (color) dalam GDI+ dan jelaskan masing-masing komponennya
3. Apakah yang dimaksud dengan Graphics Path?
4. Adakah perbedaan antara struktur Point dan PointF? Jika ada jelaskan secara komprehensif!
5. Masih banyak method yang belum dibahas dalam buku ini. Eksplorasilah dengan method-method yang ada di dalam kelas Graphic dan tuliskan pengalaman anda berikut dengan penjelasannya.

Daftar Pustaka

1. Edward Angel, Interactive Computer Graphics: A Top-Down Approach with OpenGL 2nd, Addison Wesley, 2005
2. Mahesh Chand, Graphics Programming with GDI+, Addison-Wesley, 2003
3. Nick Symmonds, GDI+ Programming in C# and VB .NET, Apress, 2002
4. Ollie Cornes, Jay Glynn, Burton Harvey, Craig McQueen, Jerod Moemeka, Christian Nagel, Simon Robinson, Morgan Skinner, Karli Watson, Professional C# - Graphics with GDI+, Wrox, 2001

MODUL X TOPIK-TOPIK GRAFIKA KOMPUTER LANJUT

Setelah membaca modul ini, mahasiswa akan memiliki pengetahuan dan mampu menjelaskan (i) topik-topik grafika komputer lanjut (*advance*) (ii) pencahayaan dalam grafika komputer (iii) fraktal secara umum (iv) Konsep *ray tracing* (v) Pemrograman Grafika dengan OpenGL

X.1 Topik Grafika Komputer Lanjut

Grafika komputer adalah topik keilmuan yang senantiasa berkembang dari tahun ke tahun. Berkembangnya perangkat keras, sekaligus juga dengan perangkat lunak, membuat konsep grafika komputer yang dahulu belum memungkinkan diimplementasikan, sekarang dapat diimplementasikan. Selain itu grafika komputer sudah merupakan kebutuhan tidak hanya sebagai bidang ilmu informatika, namun sudah merupakan kebutuhan dasar manusia khususnya yang berkaitan dengan penguasaan teknologi.

Selain grafika komputer, animasi dan visualisasi juga merupakan bidang yang menyertai grafika komputer dan membuat grafika komputer semakin *solid* dan kuat menjadi tiang fondasi bagi bidang ilmu lainnya. Perkembangan lanjut yang lain adalah *konsep virtual reality* dan *augmented reality*, menambah ketergantungan kita kepada grafika komputer.

Apa yang disampaikan dalam buku ini adalah topik-topik yang masih klasik dan *fundamental*, walaupun dalam beberapa sisi ada sentuhan modern. Penguasaan topik dasar menjadi suatu keharusan sebelum kita melangkah ke topik-topik yang lanjut. Dengan kata lain masih banyak topik-topik yang belum dibahas dalam buku ini dan menjadi kewajiban kita apabila kita mau berkembang pada jalur keilmuan grafika komputer.

Dalam buku *Computer Graphics : Principles and Practises* dituliskan topik-topik lanjut yang bisa dibahas dalam grafika komputer yaitu :

- *Modeling non-rigid objects*
- *Modeling natural phenomena*
- *Grammar-based modeling*
- *Evolutionary procedural modeling*
- *Particle systems*

- *Finite element methods in graphics*
- *Visualizing multivariate data*
- *Antialiasing*
- *Surface/light interactions*
- *Forward or backward ray tracing*
- *Coherence in ray tracing*
- *Flow visualization*
- *Volume visualization*
- *Graphics architectures*
- *Graphics languages*
- *Object-oriented graphics*
- *Topics in computational geometry*
- *Efficiency and complexity issues in graphics algorithms*
- *Graphics and human perception*
- *Molecular graphics*
- *Fractals and chaos*
- *Turbulence*
- *Radiosity*
- *Interactions in virtual reality*
- *Texture mapping*
- *Animating position, speed, or orientation*
- *Animating articulated structures*
- *Shadows*
- *Morphing*
- *Multimedia*

Pada modul ini akan dibahas 4 topik grafika komputer lanjut yaitu pencahayaan, fraktal, *ray tracing* dan OpenGL

X.2 Pencahayaan dan Warna

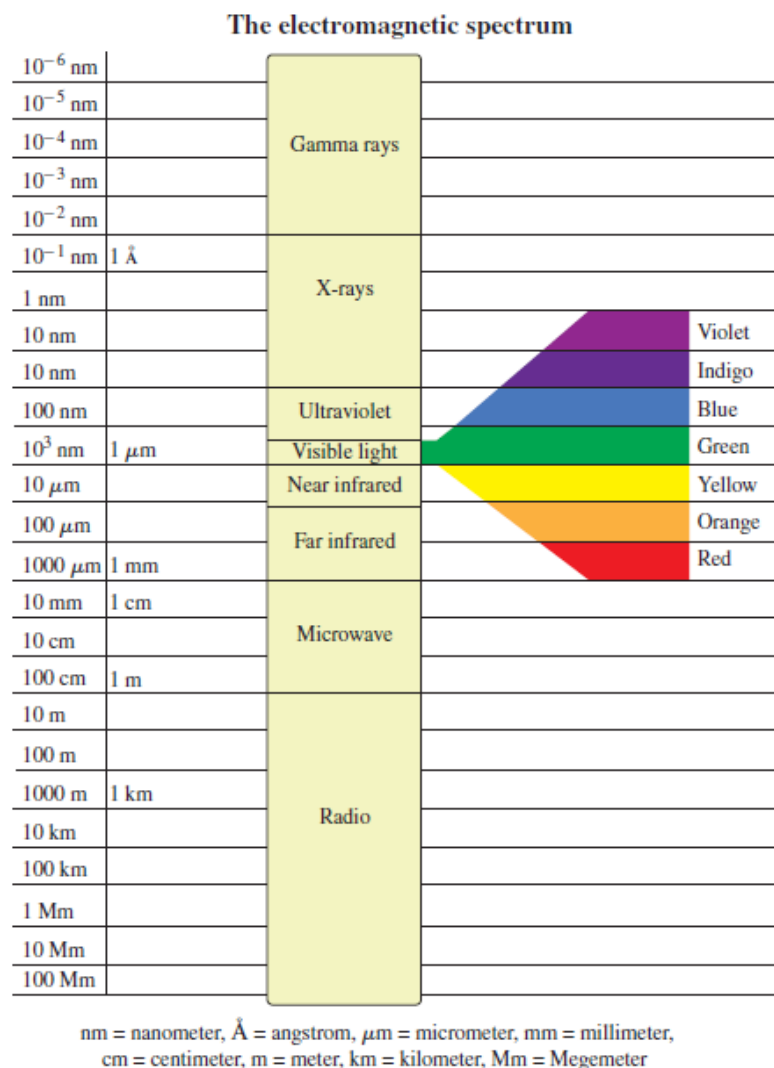
Hasil nyata dari grafika komputer adalah gambar yang secara alamiah adalah objek yang pada langkah akhirnya harus bisa oleh mata manusia. Secara teoritis penglihatan

manusia sangat dipengaruhi oleh banyak hal, salah satunya adalah adanya cahaya. Tanpa adanya cahaya maka manusia tidak bisa melihat objek. Kuat lemahnya pencahayaan juga mempengaruhi objek yang dilihat. Selanjutnya objek yang dilihat dipersepsikan oleh otak.

Cahaya dapat dilihat sebagai gelombang energi atau artikel (photon). Cahaya yang dipandang sebagai gelombang energi dapat dibagi menjadi dua bagian besar yaitu:

1. Cahaya terlihat / tampak (visible light).
2. Cahaya tidak tampak (invisible light).

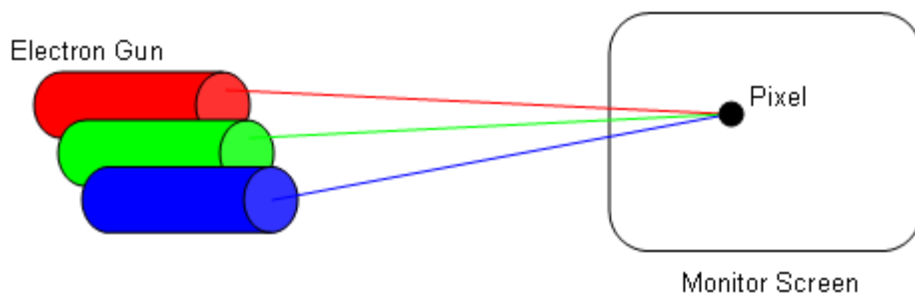
Cahaya tampak mempunyai panjang gelombang 390 s/d 720nm(nano meter). Mata kita hanya peka terhadap panjang gelombang 400 – 700nm. Cahaya tak tampak mempunyai panjang gelombang < 390nm atau panjang gelombang > 720nm. Gambar berikut memperlihatkan spektrum cahaya tampak dan tak tampak.



Gambar 10.1 Spektrum Elektromagnetik

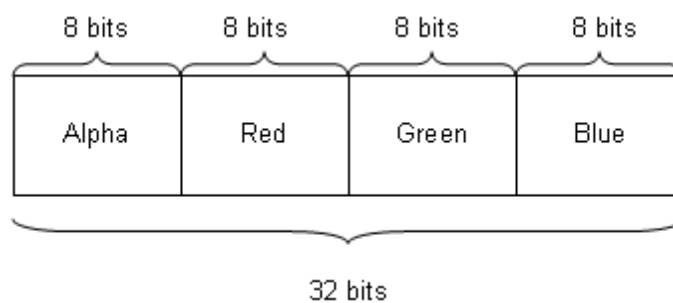
Representasi Warna

Gambar yang tampil pada layar monitor pada dasarnya terdiri dari elemen-elemen gambar yang dikenal dengan sebutan piksel. Pada layar monitor berwarna, piksel merupakan hasil dari perpaduan sinar elektron yang dipancarkan oleh *electron gun*. Ketiga sinar itu berwarna merah, hijau dan biru, atau dikenal dengan *Red, Green, Blue* (RGB). Perpaduan RGB akan menghasilkan variasi warna yang unik dan berbeda. berikut menunjukkan ilustrasinya.



Gambar 10.2 Representasi Warna

Perpaduan sinar dari *electron guns* merupakan perpaduan nilai intensitas dari masing-masing warna. Dalam model RGB, nilai masing-masing warna adalah antara 0 sampai dengan 255. Jadi dalam hal ini, jika nilai R, G, B adalah semuanya 0, maka warna piksel yang dihasilkan adalah warna hitam. Jika semua nilai RGB adalah 255, maka warna yang dihasilkan adalah warna putih. Pada gambar digital ada satu elemen lain dari RGB yang disebut dengan elemen alpha. Elemen ini menentukan nilai transparansi warna. Jadi pada dasarnya warna dasar diwakili sebesar 8 bit dan memiliki struktur seperti pada gambar di bawah. Model RGB adalah salah satu dari model-model warna yang ada. Model warna yang lain adalah HSI (HSB atau HSV), YCrCb, TSL, CMY, CIE-Lab, YIQ serta varian-variananya.

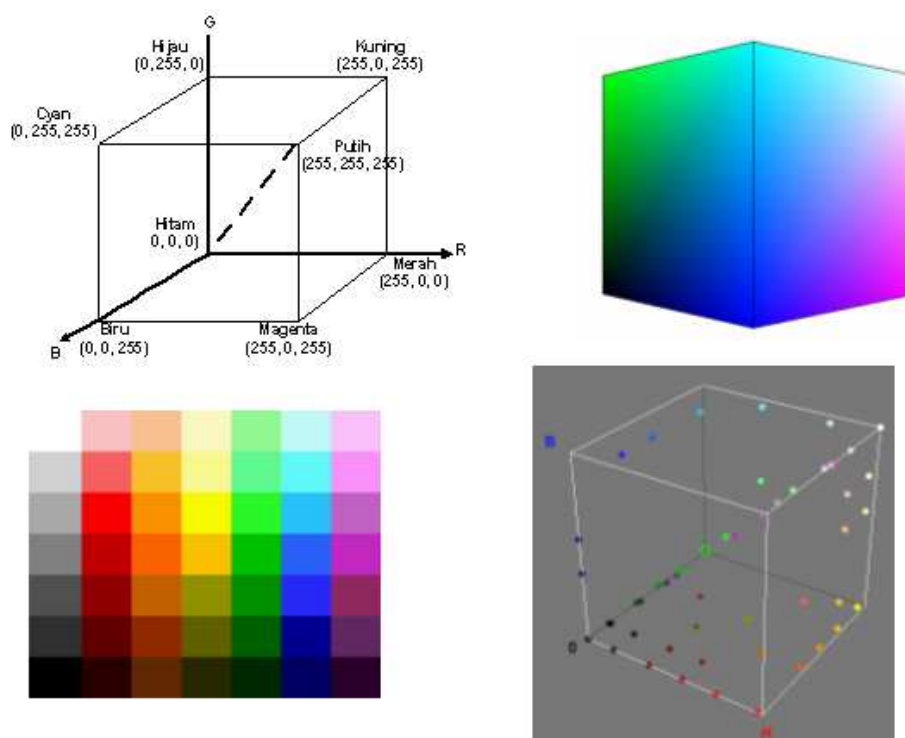


Gambar 10.3 Elemen PembangunWarna

Ruang Warna RGB

RGB adalah salah satu ruang warna yang paling banyak digunakan untuk pengolahan dan penyimpanan data gambar. Namun ruang warna RGB memiliki kelemahan apabila digunakan untuk analisis warna dan pengenalan objek berdasarkan warna. Kelemahan itu adalah tingginya tingkat ketergantungan terhadap peralatan (*device dependent*), korelasi warna yang erat antara kanal dan bersifat *semi-intuitive*, serta tidak terlihatnya perbedaan antara elemen *chrominance* (warna-warni) dan *luminance* (kecerahan). Sebagai tambahan model warna

RGB bersifat *perceptually uniform* yang artinya sifat RGB tidak menggambarkan kesensitivitasan sistem penglihatan manusia. Karena kekurangan tersebut, model warna RGB kurang tepat digunakan untuk pendeteksian warna kulit, walaupun beberapa penelitian telah dilakukan dan dilaporkan memberikan hasil yang relatif baik. Gambar berikut mengilustrasikan ruang warna RGB.

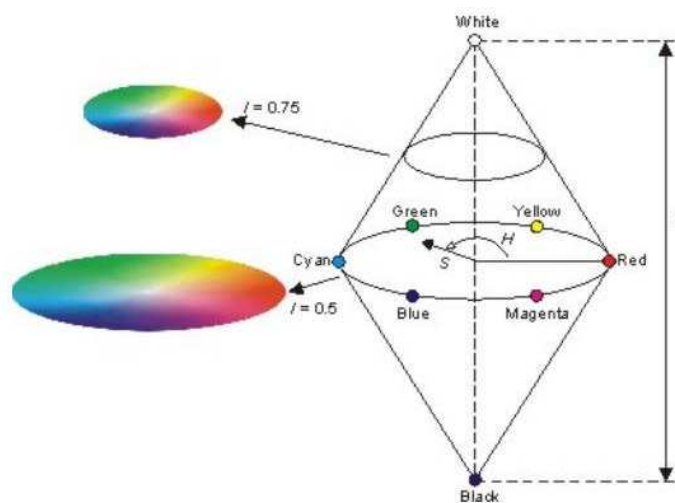


Gambar 10.4 Ruang Warna RGB dan Representasinya

Ruang Warna HSI

Model warna HSI mengandung tiga elemen yaitu *Hue*, *Saturation* dan *Intensity*. *Hue* adalah warna yang dominan, misalnya merah, hijau, ungu dan kuning, pada sebuah area, saturation berkaitan dengan *colorfulness* pada sebuah area, misalnya gradasi warna merah, dan intensity berkaitan dengan *luminance*, yaitu kecerahan (terang gelap). Model warna ini menarik para peneliti dalam bidang pendeteksian warna kulit karena sifatnya yang secara eksplisit dan intuitif membedakan antara *chrominance* dan *luminance*. Selain itu, model warna ini juga invarian terhadap sumber cahaya putih dan permukaan yang redup (*matte*).

Sistem warna HSI bersifat *non-linier* dan menggunakan koordinat polar sehingga memiliki karakteristik siklis, dimana nilai *Hue* berada pada interval 0 - 360. Nilai Hue 0 menunjukkan warna merah, 60 menunjuk pada warna kuning, 120 berarti warna hijau, 240 menunjuk pada warna biru dan 300 berarti magenta. Komponen *Saturation* menunjuk pada seberapa jauh sebuah warna dipengaruhi oleh warna putih. Interval Saturation adalah bilangan real $[0,1]$. Komponen Intensitas menunjuk pada interval $[0,1]$, dimana 0 adalah hitam dan 1 adalah putih. Gambar 10.5 menunjukkan visualisasi model warna HSI. Dari gambar tersebut dapat dilihat bahwa komponen Hue lebih memiliki arti jika *Saturation* mendekati 1 (jika 0 tidak, berapapun nilai Hue tidak akan berpengaruh) dan kurang berarti jika *Saturation* mendekati 0 atau Intensitas mendekati 0 atau 1. Komponen Intensitas membatasi *Saturation*, *Value*), HSB (*Hue*, *Saturation*, *Brightness*) dan HSL (*Hue*, *Saturation*, *Lightness/Luminance*). Pada dasarnya varian-varian tersebut sama dengan HSI. Visualiasi model warna HSI disajikan pada gambar di bawah ini.



Gambar 10.6 Visualisasi Ruang Warna HSI

Model Pencahayaan

Salah satu tujuan dari grafik komputer adalah menghasilkan tampilan yang senyata mungkin. Untuk mewujudkan keinginan tersebut harus memperhatikan efek pencahayaan. Komputer grafik sebenarnya adalah model matematik dari kehidupan nyata maka pencahayaan harus dapat diubah menjadi model matematika. Model matematik tersebut harus memenuhi persyaratan sebagai berikut:

1. Menghasilkan efek seperti cahaya sesungguhnya.
2. Dapat dihitung dengan cepat.

Pencahayan Global dan Lokal

Ada berbagai model matematika yang diusulkan, namun kita dapat mengelompokkan sebagai berikut:

1. Model pencahayaan global.
2. Model pencahayaan lokal.

Model pencahayaan global

Model pencahayaan global merupakan model matematika yang memperhitngkan pengaruh interaksi cahaya terhadap berbagai objek, seperti (Pantulan, Serapan cahaya, Bayangan). Model pencahayaan global dapat dikelompokkan sebagai berikut:

Ray tracing

Ray tracing cahaya menyebar ke berbagai arah, kemudian menghitung kuat cahaya pada saat cahaya mengenai mata.

Radiocity

Radiocity mengasumsikan sembarang permukaan benda yang tidak berwarna hitam diasumsikan menjadi sumber cahaya. Cahaya yang dikeluarkan oleh benda tersebut dipengaruhi oleh cahaya yang berasal dari sumber cahaya dan pantulan dari benda lain.

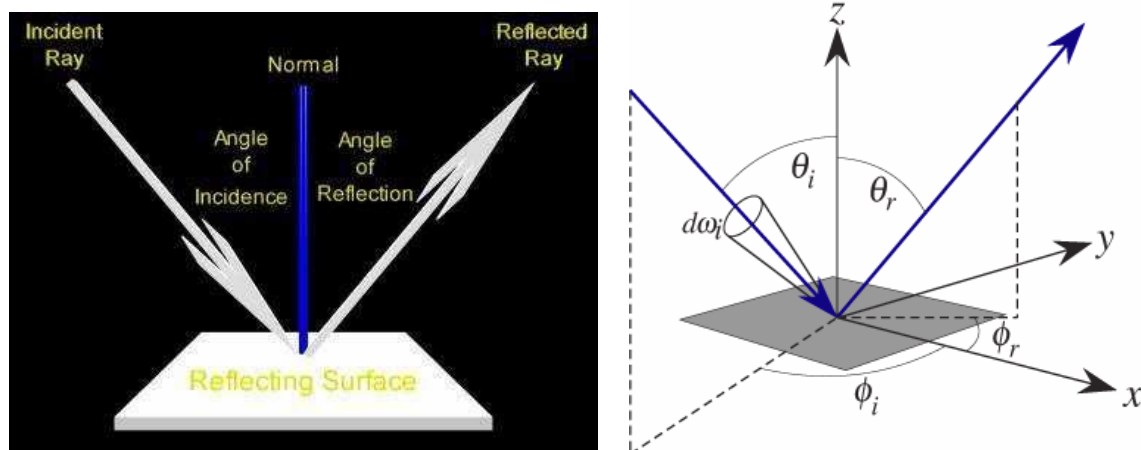
Model Pencahayaan Lokal

Model pencahayaan lokal tidak memperhitungkan pengaruh cahaya yang dihasilkan oleh benda lain disekitarnya. Model pencahayaan lokal hanya membutuhkan:

1. Sifat materi penyusun benda.
2. Sumber cahaya.
3. Geometri permukaan benda.
4. Posisi mata.

Sifat Materi Penyusun Benda

Sifat materi penyusun benda menentukan bagaimana cahaya bereaksi terhadap materi penyusun benda. Secara umum cahaya yang mengenai permukaan suatu benda akan dipantulkan oleh permukaan benda tersebut. Gambar berikut mengilustrasikan perjalanan cahaya dari sumber cahaya.



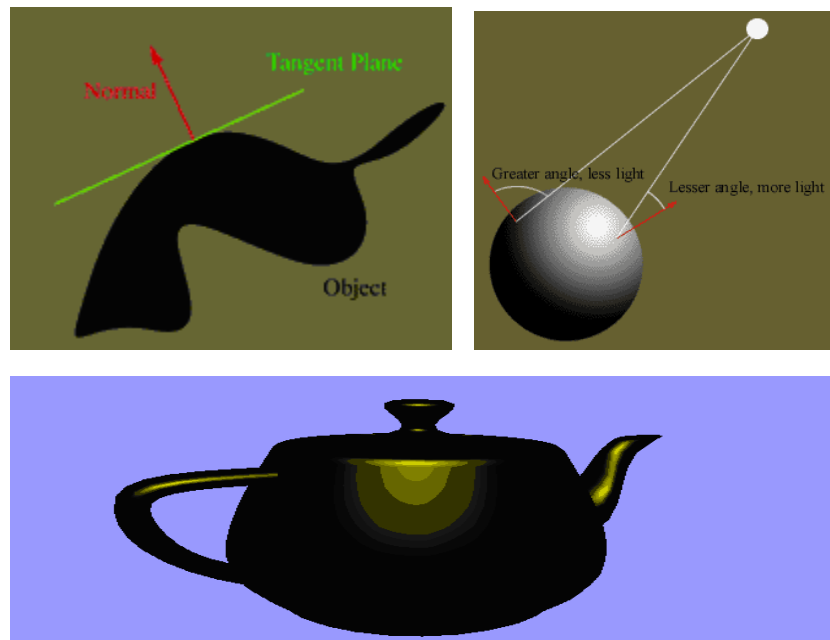
Gambar 10.7 Model Pencahayaan

Berdasarkan kepada materi penyusun benda maka ada tiga kemungkinan arah pantulan cahaya ketika cahaya menimpa permukaan benda, yaitu:

1. Specular.
2. Diffuse.
3. Translucent.

Specular

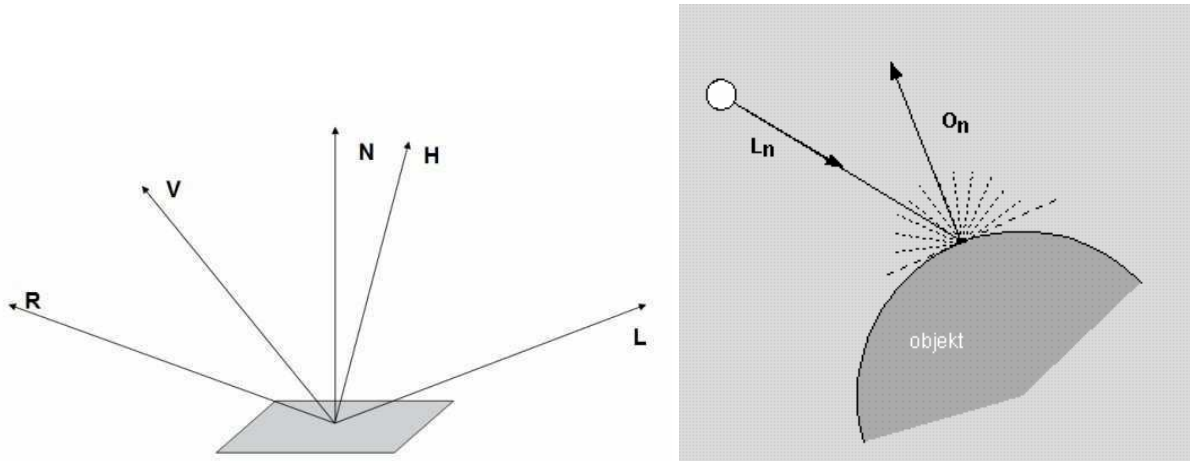
Specular merupakan permukaan yang licin dan halus, misalnya cermin, benda-benda dari plastik. Cahaya yang jatuh pada permukaan pada benda-benda seperti ini akan dipantulkan kembali. Dan apabila kita melihat dari sumber datanya cahaya maka kita melihat satu area yang relatif paling terang, area tersebut disebut dengan *specular highlight*. Gambar berikut mengilustrasikan sumber cahaya *specular*.



Gambar 10.8 Model Pencahayaan Spekular

Diffuse

Diffuse merupakan sifat permukaan dimana cahaya yang datang dipantulkan ke segala arah, benda bersifat diffuse misalnya: batu, meja, tembok. Karena cahaya dipantulkan ke segala arah maka permukaan benda terlihat lebih kasar.



Gambar 10.9 Model Pencahayaan *Diffuse*

Translucent

Benda yang mempunyai permukaan *translucent* akan meneruskan cahaya yang datang dan sekaligus memantulkan cahaya tersebut.



Gambar 10.8 Model Pencahayaan *Translucent*

Model Sumber Cahaya

Sumber cahaya dapat dikelompokkan menjadi dua macam, yaitu:

1. Cahaya lingkungan (*Ambient Light*).
2. Cahaya Titik (*Point Light*)

Cahaya Lingkungan (*Ambient Light*)

Didalam dunia nyata semua benda memantulkan cahaya meskipun sedikit, cahaya lingkungan digunakan untuk memodelkan cahaya yang berasal dari berbagai sumber tersebut. Cahaya ambient tidak mempunyai arah dan lokasi.

Cahaya Titik (*Point Light*)

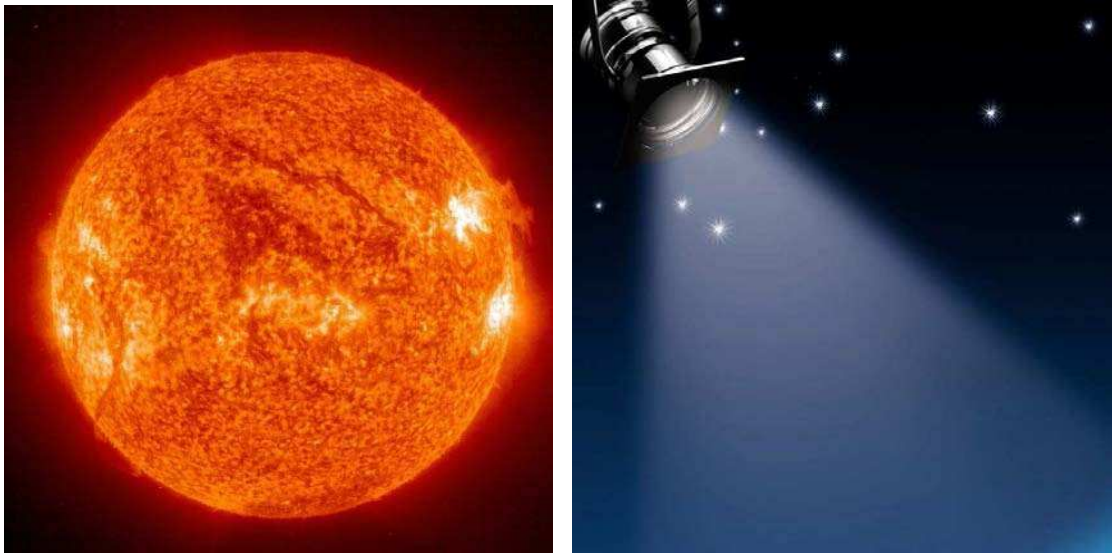
Sumber cahaya ini mempunyai lokasi dan arah, dengan demikian jarak antara sumber cahaya terhadap benda akan berpengaruh terhadap kuat cahaya yang diterima oleh benda. Model cahaya ini dibedakan menjadi 2 bagian, yaitu:

a. Directional

Directional mempunyai karakteristik energi dari sumber tersebut menyebar ke semua arah dengan kekuatan yang sama. Contoh sumber cahaya ini adalah cahaya matahari.

b. Positional

Sumber cahaya ini mempunyai sifat dimana energi dari sumber cahaya tersebut akan melemah sebanding dengan jarak dan sudut terhadap sumber cahaya. Melemahnya kuat cahaya karena pengaruh jarak disebut *attenuation*. Apabila cahaya yang keluar dari sumber cahaya potensial dibatasi sudut penyebarannya maka akan memperoleh efek lampu sorot.



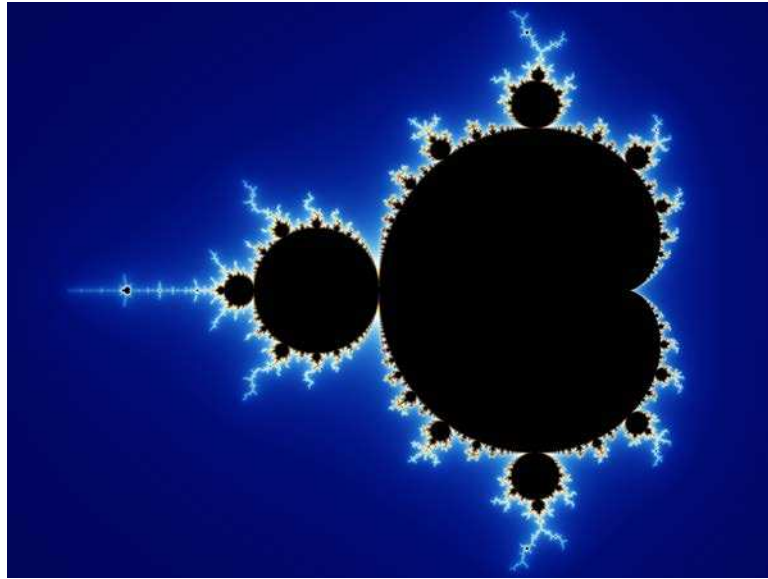
Gambar 10.8 Model Pencahayaan *Directional* dan *Positional*

X.3 Tinjauan Fraktal Secara Umum

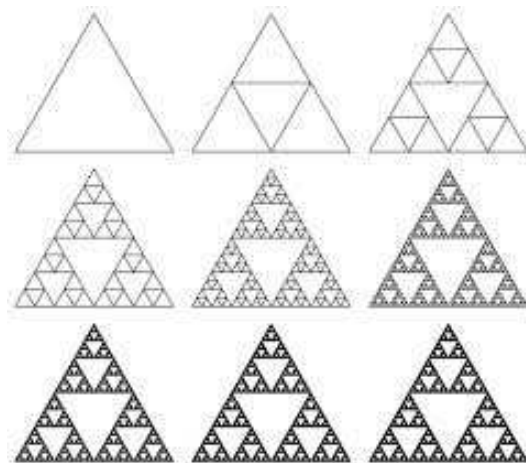
Fraktal adalah benda geometris yang kasar pada segala skala, dan terlihat dapat "dibagibagi" dengan cara yang radikal. Beberapa fraktal bisa dipecah menjadi beberapa bagian yang semuanya mirip dengan fraktal aslinya. Fraktal dikatakan memiliki detil yang tak hingga dan dapat memiliki struktur serupa diri pada tingkat perbesaran yang berbeda. Pada banyak kasus, sebuah fraktal bisa dihasilkan dengan cara mengulang suatu pola, biasanya dalam proses rekursif atau iteratif.

Bahasa Inggris dari fraktal adalah fractal. Istilah fractal dibuat oleh Benoît Mandelbrot pada tahun 1975 dari kata Latin fractus yang artinya "patah", "rusak", atau "tidak teratur". Sebelum Mandelbrot memperkenalkan istilah tersebut, nama umum untuk struktur semacamnya (misalnya bunga salju Koch) adalah kurva *monster*.

Berbagai jenis fraktal pada awalnya dipelajari sebagai benda-benda matematis. Geometri fraktal adalah cabang matematika yang mempelajari sifat-sifat dan perilaku fraktal. Fraktal bisa membantu menjelaskan banyak situasi yang sulit dideskripsikan menggunakan geometri klasik, dan sudah cukup banyak diaplikasikan dalam sains, teknologi, dan seni karya komputer. Dulu ide-ide konseptual fraktal muncul saat definisi-definisi tradisional geometri Euclid dan kalkulus gagal melakukan berbagai pengukuran pada benda-benda *monster* tersebut.



Fractal Mandelbrot



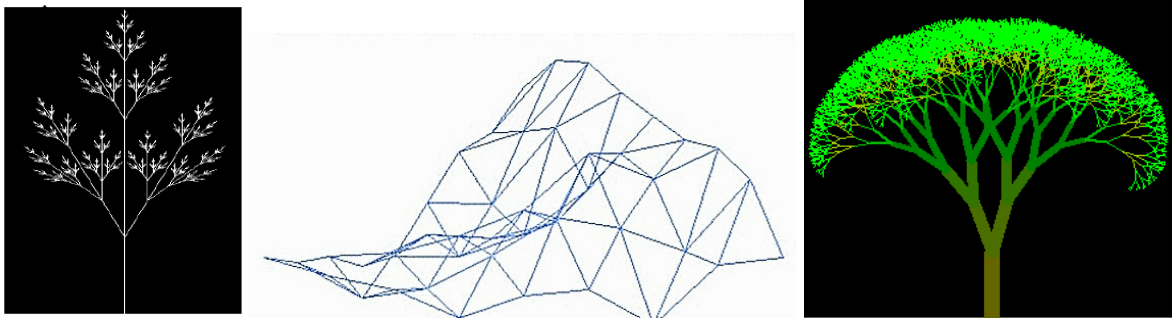
Segitiga Sierpinski

Gambar 10.9 Contoh Fraktal

Beberapa contoh fraktal yang umum adalah himpunan Mandelbrot, fraktal Lyapunov, himpunan Cantor, segitiga Sierpinski, karpet Sierpinski, spons Menger, kurva naga, kurva Peano, dan kurva Koch. Fraktal bisa deterministik maupun stokastik. Sistem dinamikal chaotik sering (bahkan mungkin selalu) dihubungkan dengan fraktal. Benda-benda yang mendekati fraktal bisa ditemukan dengan mudah di alam. Benda-benda tersebut menunjukkan struktur fraktal yang kompleks pada skala tertentu.

Contohnya adalah awan, gunung, jaringan sungai, dan sistem pembuluh darah. Harrison meluaskan kalkulus Newtonian ke domain fraktal, termasuk teorema Gauss, Green, dan Stokes. Fraktal biasanya digambar oleh komputer dengan perangkat lunak fraktal. Fraktal acak memiliki kegunaan praktis yang terbesar sebab dapat digunakan untuk mendeskripsikan banyak benda di alam. Contohnya adalah awan, gunung, turbulensi, garis pantai, dan pohon. Teknik-teknik fraktal juga telah digunakan pada kompresi gambar fraktal dan berbagai disiplin sains.

Berbagai Objek Hasil dari Fraktal

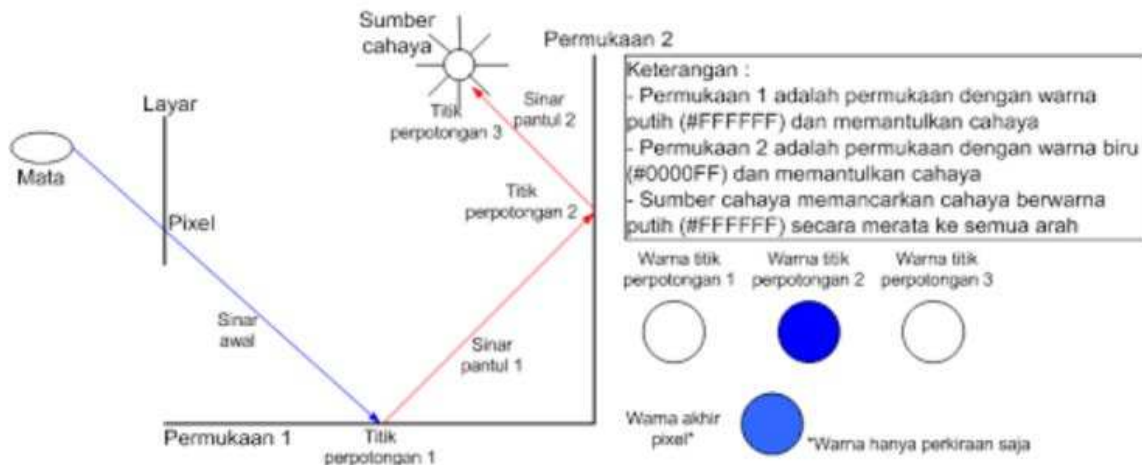


Gambar 10.10 Contoh Fraktal Alamiah

X.4 Ray Tracing

Ray tracing adalah suatu metode untuk menghasilkan gambar yang dibuat dalam lingkungan komputer 3D. Cara kerjanya adalah dengan mengikuti jejak (*tracing*) suatu sinar (*ray*) dari suatu mata imajiner yang melalui sebuah *pixel* di layar virtual dan mengakumulasi kontribusi setiap sinar dalam *scene* di *pixel* tersebut. Setiap sinar yang berasal dari mata tersebut diperiksa apakah berpotongan/bertabrakan dengan objek-objek di dalam *scene*. *Scene* adalah kumpulan objek-objek dan sumber cahaya yang akan dilihat oleh pengamat. Setiap terjadi tabrakan antara sinar dan objek, warna *pixel* di-*update*, lalu tergantung dari jenis material objek dan algoritma yang dipakai, sinar tersebut dapat diteruskan atau dihilangkan.

Dengan metode *ray tracing* ini, kita dapat membuat berbagai efek yang sulit atau bahkan tidak mungkin dengan metode lain. Diantara efek-efek tersebut adalah pemantulan, tembus cahaya, dan bayangan.



Gambar 10.11 Perjalanan Cahaya

Dalam menggunakan *ray tracing*, umumnya dibuat suatu batasan agar lebih jelas. Contohnya, *pixel* akan di-*update* jika sinar telah memantul n kali atau telah bergerak sejauh m tanpa berpotongan dengan apapun. n dan m adalah nilai pembatas. Intensitas cahaya dan warna dari *pixel* yang bersangkutan dihitung berdasarkan sejumlah algoritma, baik dengan algoritma klasik atau dengan teknik *radiosity*.

Salah satu metode turunan dari *ray tracing* adalah *photon mapping*. Pada *photon mapping* ini, sinar dibuat dari sumber cahaya dan dari mata pengamat secara independen. Sinar yang dibuat akan terus bergerak dalam *scene* sampai habis diserap oleh sebuah permukaan, bergerak ke arah yang tidak akan terjadi perpotongan, atau jaraknya terlalu jauh dari pengamat. Inti dari *photon mapping* adalah melakukan simulasi pencahayaan, namun hal ini jauh lebih lambat daripada *ray tracing* biasa.

Konsep Ray Tracing

Ada dua konsep yang menjadi dasar teori untuk *ray tracing*, yaitu :

1. Kita dapat melihat sebuah benda karena benda tersebut memantulkan cahaya. Cahaya yang dipantulkan tersebut lalu akan ditangkap oleh retina mata dan diterjemahkan oleh otak menjadi apa yang kita lihat.
2. Dalam perjalanan sebuah sinar, jika sinar tersebut menabrak suatu permukaan, dapat terjadi tiga hal tergantung pada jenis permukaan yang ditabrak, yaitu penyerapan, pemantulan, dan pembiasan. Sebuah permukaan dapat memantulkan semua atau sebagian dari sinar, baik ke satu atau banyak arah. Permukaan tersebut juga dapat

menyerap sebagian dari sinar, mengurangi intensitas sinar yang terpantul atau terbias. Jika permukaan tersebut memiliki sifat tembus cahaya (*transparency/translucent*) maka permukaan itu akan membiaskan sebagian sinar dan menyerap sebagian atau semua spektrum sinar, sehingga dapat mengubah warna sinar.

Namun perlu diperhatikan bahwa ada perbedaan mendasar antara konsep diatas dengan *ray tracing*. Pada *ray tracing*, umumnya sinar berasal dari mata pengamat, sedangkan pada kenyataannya sinar selalu berasal dari sumber cahaya. Karena itu ada dua jenis *ray tracing*, *eye-based* dan *light-based*. *Eye-based* adalah *ray tracing* dimana sinar berasal dari mata pengamat, sedangkan pada *light-based ray tracing*, sinar berasal dari sumber cahaya.

Algoritma Ray Tracing

Sebelum membahas algoritma *ray tracing*, ada beberapa hal penting yang harus kita perhatikan :

1. Tiga efek umum dalam *ray tracing* adalah pemantulan, tembus cahaya, dan bayangan.
2. *Ray tracing* adalah fungsi rekursif.

Setiap sebuah sinar berpotongan dengan sebuah permukaan (disebut juga tabrakan), terjadi rekursi. Dari titik perpotongan tersebut, satu atau lebih sinar dibuat untuk menentukan objek apa yang terpantul di titik itu (jika memantulkan cahaya), objek apa yang terlihat melalui titik itu (jika tembus cahaya), sumber cahaya mana saja yang dapat terlihat dari titik itu (untuk menentukan bayangan), dan lain-lain.

Basis dari fungsi *ray tracing* ini adalah batasan dimana sinar berhenti bergerak. Basis-basis yang umum digunakan diantaranya :

- a. Jika tidak berpotongan dengan objek maka warna yang dihasilkan adalah warna latar belakang
- b. Jika objek yang berpotongan terdekat adalah sumber cahaya maka warna yang dihasilkan sesuai warna cahaya
- c. Jika jumlah pantulan melewati batas pemantulan maka sinar berhenti bergerak
- d. Jika jarak sinar dari layar melewati jarak maksimum maka sinar berhenti bergerak

3. Sinar

Sinar yang digunakan dalam *ray tracing* adalah sebuah vektor. Persamaan sebuah sinar dapat ditulis sebagai :

$$S + tD$$

Dimana

S = titik sumber sinar (x,y,z) ,

t = panjang sinar

D = arah sinar (dalam vektor satuan)

4. Fungsi Perpotongan

Untuk menentukan apakah sebuah sinar berpotongan dengan objek, diperlukan sebuah fungsi perpotongan. Fungsi ini dibuat spesifik untuk setiap jenis objek, misalnya bola atau poligon. Sebagai contoh, kita akan membuat fungsi perpotongan dengan bola.

Persamaan bola pada ruang 3D :

$$(x-a)^2 + (y-b)^2 + (z-c)^2 = r^2$$

Dimana (x,y,z) adalah titik pada bola

(a,b,c) adalah pusat bola

Substitusi x,y , dan z pada persamaan bola dengan persamaan sinar :

$$((S.x+t(D.x))-a)^2 + ((S.y+t(D.y))-b)^2 + ((S.z+t(D.z))-c)^2 = r^2$$

Agar lebih mudah membaca rumus diatas maka kita lakukan substitusi :

$$A = (D.x)^2 + (D.y)^2 + (D.z)^2$$

$$B = 2 \cdot ((D.x)(S.x-a) + (D.y)(S.y-b) + (D.z)(S.z-c))$$

$$C = ((S.x-a)^2 + (S.y-b)^2 + (S.z-c)^2) - r^2$$

Didapat persamaan kuadrat :

$$At^2 + Bt + C=0$$

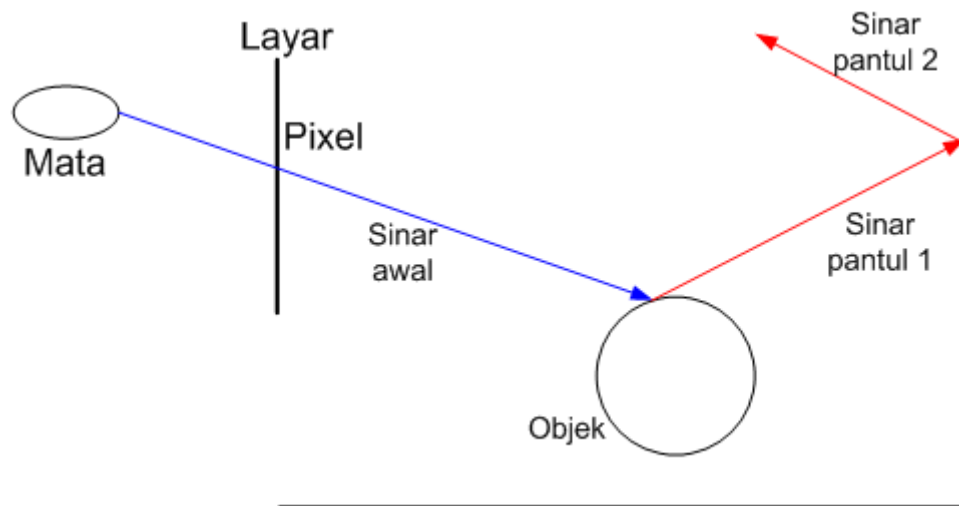
t bisa didapat dengan menyelesaikan persamaan kuadrat diatas:

$$t = (-B \pm \sqrt{B^2 - 4AC}) / 2A$$

Jika t adalah bilangan real, maka sinar berpotongan dengan bola. Jika tidak, maka sinar tidak berpotongan dengan bola.

5. Pemantulan

Jika permukaan yang ditabrak sinar adalah permukaan yang memantulkan cahaya seperti cermin, *ray tracer* harus menentukan warna titik perpotongan tersebut dengan memperhitungkan warna permukaan dan warna yang terpantul pada titik tersebut. Hal itu dilakukan dengan menentukan arah sinar pantulan dan membuat sinar baru yang bergerak sesuai arah tersebut.



Gambar 10.12 Efek Pemantulan Cahaya

Persamaan untuk menentukan arah sinar pantul :

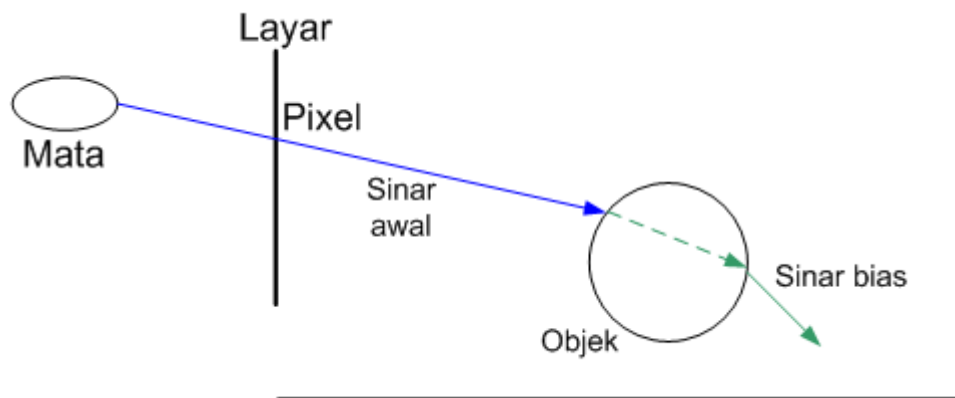
$$P = 2 * (N \cdot (-D)) * N + D$$

Dimana N adalah vektor normal permukaan

 D adalah arah sinar

Tembus cahaya

Tembus cahaya memiliki sifat yang mirip dengan pemantulan, tapi sinar tidak dipantulkan pada permukaan, melainkan dibiaskan di dalam objek yang bersangkutan. Arah sinar bias ditentukan berdasarkan indeks bias benda tersebut, jumlahnya bisa lebih dari satu atau tidak ada sama sekali. Sinar baru akan dibuat dengan arah sinar bias



Gambar 10.13 Efek Tembus Cahaya

Persamaan untuk menentukan arah sinar bias :

$$T = ((n1/n2)(N.I) - \text{sqrt}(1 - (n1/n2)^2 \cdot (1 - (N.I)^2))) \cdot N - (n1/n2).I$$

Dimana

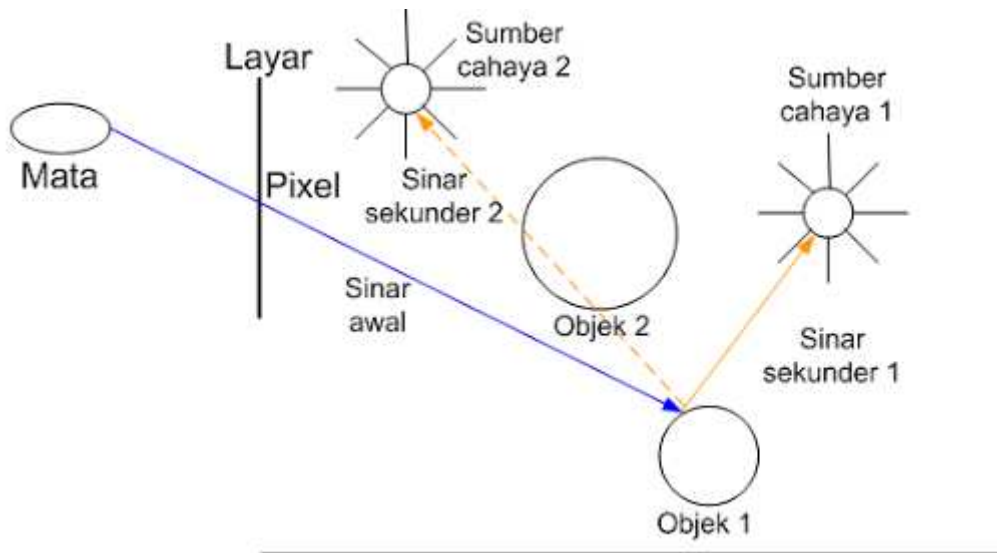
T adalah arah sinar bias

n1 adalah indeks bias material awal

n2 adalah indeks bias material objek

I adalah sinar awal

N adalah vektor normal permukaan



Gambar 10.14 Efek Pantul dan Tembus dengan Berbagai Sumber Cahaya

Untuk menentukan apakah titik perpotongan dengan permukaan berada dalam suatu wilayah bayangan dilakukan pemeriksaan antara titik tersebut dengan semua sumber cahaya. Hal ini dilakukan dengan membuat sinar-sinar baru dari titik ke sumber cahaya. Jika ada objek diantara titik dan sumber cahaya, maka titik tersebut tidak akan mendapat cahaya dari sumber yang bersangkutan, dengan kata lain, titik tersebut berada dalam bayangan. Contoh dapat dilihat pada gambar 4 diatas, titik pada objek 1 mendapat bayangan dari sumber cahaya 2, dan mendapat cahaya dari sumber cahaya 1.

Pseudocode untuk Raytracing

```
for(semua pixel dalam layar){
    buat sinar dari mata pengamat melalui pixel

    raytrace(sinar_asal, warna_saat_ini, kedalaman_rekursi, jarak_dari_layar){
        inisialisasi objekTerdekat dan perpotonganTerdekat dengan NULL

        for(semua objek dalam scene){
            if(sinar berpotongan dengan objek){
                if(jarak perpotongan t lebih kecil dari perpotonganTerdekat){
                    set perpotonganTerdekat dengan t
                    set objekTerdekat dengan objek yang sedang diperiksa
                }
            }
        }

        if(objekTerdekat == NULL){
            isi pixel dengan warna latar belakang
        }else if(objekTerdekat adalah sumber cahaya){
            isi pixel dengan warna cahaya yang dipancarkan
        }else{
            /*buat sinar ke semua sumber cahaya untuk memeriksa bayangan*/
            for(semua sumber cahaya){
                buat sinar ke sumber cahaya
                for(semua objek dalam scene){
                    if(sinar berpotongan dengan objek){
                        titik ini berada dalam bayangan, intensitas cahaya dikurangi
                    }
                }
            }
            akumulasikan warna yang didapat dari bayangan
            /*jika permukaan memantulkan cahaya, buat sinar pantulan (rekursi)*/
            buat sinar pantulan
            if(kedalaman_rekursi < kedalaman_max dan jarak_dari_layar < jarak_max){
                raytrace(sinar_pantulan, warna_permukaan, kedalaman_rekursi+1,
                jarak_dari_layar)
            }
            akumulasikan warna yang didapat dari pantulan
            /*jika permukaan tembus cahaya, buat sinar-sinar bias (rekursi)*/
            perhitungkan pembiasan yang terjadi dan buat semua sinar bias
            for(semua sinar bias){
                if(kedalaman_rekursi < kedalaman_max dan jarak_dari_layar < jarak_max){
                    raytrace(sinar_bias, warna_permukaan, kedalaman_rekursi+1,
                    jarak_dari_layar)
                }
            }
            akumulasikan warna yang didapat dari pembiasan
            jika tidak terjadi bayangan, pemantulan, atau pembiasan, ambil warna permukaan
            perhitungkan warna berdasarkan semua warna yang didapat sebelumnya
            isi pixel dengan warna hasil perhitungan
        }
    }
}
```

X.5 Pemrograman Grafika Komputer dengan OpenGL

OpenGL adalah sebuah program aplikasi interface yang digunakan untuk mendefinisikan komputer grafis 2D dan 3D. Program lintas-*platform* API ini umumnya dianggap ketetapan standar dalam industri komputer dalam interaksi dengan komputer grafis 2D dan juga telah menjadi alat yang biasa untuk digunakan dengan grafis 3D. Singkatnya, *Open Graphics Library*, OpenGL menghilangkan kebutuhan untuk pemrogram untuk menulis ulang bagian grafis dari sistem operasi setiap kali sebuah bisnis akan *diupgrade* ke versi baru dari sistem.

Fungsi dasar dari OpenGL adalah untuk mengeluarkan koleksi perintah khusus atau *executable* ke sistem operasi. Dengan demikian, program ini bekerja dengan perangkat keras grafis yang ada yang berada pada *hard drive* atau sumber tertentu lainnya. Setiap perintah dalam dirancang untuk melakukan tindakan tertentu, atau memulai efek khusus tertentu yang terkait dengan grafis.

Membuat perintah dalam OpenGL dapat terjadi dalam dua cara yang berbeda. Pertama, adalah mungkin bagi *programmer* untuk membuat dan menyimpan daftar perintah yang dapat dieksekusi secara berulang. Ini adalah salah satu cara yang lebih rutin untuk program *interface* yang digunakan. Seiring dengan berkembangnya kelompok perintah yang kurang lebih permanen, maka memungkinkan untuk membuat dan menjalankan salah satu perintah dalam batas-batas waktu dari komputer grafis.

Seiring dengan kemampuan *interface* dari sistem operasi, OpenGL juga menyediakan beberapa *built-in* protokol yang mungkin berguna bagi pengguna akhir. Di antaranya fitur alat seperti *alpha blending*, pemetaan tekstur, dan efek atmosfer. Alat ini dapat berinteraksi dengan sistem operasi yang sedang digunakan.

Awalnya dikembangkan oleh Silicon Graphics, OpenGL kini dianggap standar industri. *Interface* program aplikasi yang aktif didukung oleh Microsoft ini, menawarkan *download* gratis daftar perintah OpenGL untuk digunakan pada sistem Windows. OpenGL juga bekerja sangat baik dengan Inventor Open, sebuah pemrograman berorientasi obyek alat juga diciptakan oleh Silicon Graphics.

Langkah-langkah OpenGL

- a. Install Microsoft Visual Studio.NET pada komputer anda
- b. Siapkan file OpenGL95.exe dan glut-3.7.6.zip pada direktori sementara
- c. Masukkan
 1. GL.H, GLAUX.H, GLU.H dan glut.h ke drive:\Program Files\Microsoft Visual Studio .NET 2003\Vc7\include\gl
 2. GLAUX32.LIB, GLU32.LIB, OPENG32.LIB dan glut32.lib ke drive:\Program Files\Microsoft Visual Studio .NET 2003\Vc7\lib
 3. OPENG32.DLL, GLU32.DLL dan glut32.dll ke drive:\Windows\System

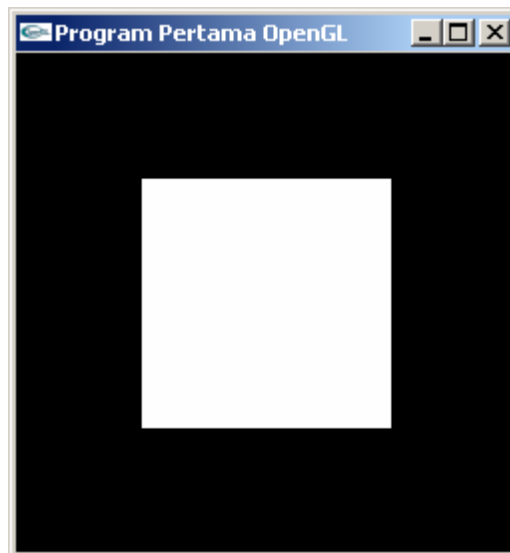
Membuat Program OpenGL Sederhana

- a. Jalankan Visual Studio .NET, buatlah sebuah proyek dengan tipe Visual C++ Projects dan template Win32 Console Project. Beri nama sesuai selera anda.
- b. Pada Application Settings pilih Console Application dan Empty Project
- c. Pada Solution Explorer, klik kanan Source Files lalu Add - Add New Item,lalu pilih template C++ File (.cpp), beri nama menurut selera anda, klik Open. Pada layar akan terlihat halaman kosong
- d. Masukkan program yang berikut ini:

```
#include <windows.h>
#include <GL\glut.h>
void display(void)
{
    glClear (GL_COLOR_BUFFER_BIT);
    glColor3f (1.0, 1.0, 1.0);
    glBegin(GL_POLYGON);
    glVertex3f (0.25, 0.25, 0.0);
    glVertex3f (0.75, 0.25, 0.0);
    glVertex3f (0.75, 0.75, 0.0);
    glVertex3f (0.25, 0.75, 0.0);
    glEnd();
    glFlush ();
}
int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize (250, 250);
    glutInitWindowPosition (100, 100);
    glutCreateWindow ("Program Pertama OpenGL");
    glClearColor (0.0, 0.0, 0.0, 0.0);
```

```
glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
glutDisplayFunc(display);
glutMainLoop();
return 0;
}
```

- e. Modifikasi Project Properties dengan memilih Project – Properties – All Configurations – Linker, masukkan opengl32.lib glu32.lib glut32.lib pada textbox Additional Dependencies
- f. Jika diperlukan, agar console windows tidak dibuka pada saat menjalankan program (console output akan disabled), masukan /SUBSYSTEM:WINDOWS /ENTRY:mainCRTStartup pada Linker – Command Line – Additional Options
- g. Lakukan kompilasi terhadap program tersebut, hasilnya adalah sebagai berikut



Latihan

Cobalah program-program contoh yang ada pada file opengl_x.zip. Berikan komentar dari hasil yang anda peroleh. Program-program yang patut dicoba adalah:

1. bteapot.c
2. contour2.c
3. cube.c
4. cubev.c
5. cubeview.c
6. cubevs.c
7. dynamic.c
8. earth.c

9. figure.c
10. figuretr.c
11. gasket2.c
12. gasket3d.c
13. gasket.c
14. mandelbrot.c
15. newpaint.c
16. robot.c
17. shadow.c
18. single_double.c
19. sphere.c
20. square.c
21. teapot.c
22. tetra.c
23. trackball.c

PROGRAM-PROGRAM CONTOH DARI REDBOOK OPENGL

Pengantar: Program-program di bawah ini diambil dari buku OpenGL Programming Guide (Redbook) Addison-Wesley Publishing Company). Versi e-booknya serta file programnya dapat didownload pada situs kuliah ini.

01: Program Menampilkan Poligon (Example 1-2)

```
/* Prog-01: Menampilkan POLIGON */
#include <gl/glut.h>
void display(void)
{
    glClear (GL_COLOR_BUFFER_BIT);
    glColor3f (1.0, 1.0, 1.0);
    glBegin(GL_POLYGON);
        glVertex3f (0.25, 0.25, 0.0);
        glVertex3f (0.75, 0.25, 0.0);
        glVertex3f (0.75, 0.75, 0.0);
        glVertex3f (0.25, 0.75, 0.0);
    glEnd();
    glFlush ();
}
int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize (250, 250);
    glutInitWindowPosition (100, 100);
    glutCreateWindow ("Program Pertama OpenGL");
    glClearColor (0.0, 0.0, 0.0, 0.0);
    glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
    glutDisplayFunc(display);
    glutMainLoop();
    return 0;
}
```

02: Program Double-buffer untuk Animasi Poligon (Example 1-3)

```
#include <windows.h>
#include <GL/glut.h>
#include <stdlib.h>
static GLfloat spin = 0.0;
void init(void)
{
    glClearColor (0.0, 0.0, 0.0, 0.0);
    glShadeModel (GL_FLAT);
}

void display(void)
{
    glClear(GL_COLOR_BUFFER_BIT);
    glPushMatrix();
    glRotatef(spin, 0.0, 0.0, 1.0);
    glColor3f(1.0, 1.0, 1.0);
    glRectf(-25.0, -25.0, 25.0, 25.0);
    glPopMatrix();
    glutSwapBuffers();
}

void spinDisplay(void)
{
    spin = spin + 2.0;
    if (spin > 360.0)
        spin = spin - 360.0;
    glutPostRedisplay();
}
```

```

}
void reshape(int w, int h)
{
    glViewport (0, 0, (GLsizei) w, (GLsizei) h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-50.0, 50.0, -50.0, 50.0, -1.0, 1.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}
void mouse(int button, int state, int x, int y)
{
    switch (button) {
        case GLUT_LEFT_BUTTON:
            if (state == GLUT_DOWN)
                glutIdleFunc(spinDisplay);
            break;
        case GLUT_MIDDLE_BUTTON:
            if (state == GLUT_DOWN)
                glutIdleFunc(NULL);
            break;
        default:
            break;
    }
}
int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode (GLUT_DOUBLE | GLUT_RGB);
    glutInitWindowSize (250, 250);
    glutInitWindowPosition (100, 100);
    glutCreateWindow (argv[0]);
    init ();
    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
    glutMouseFunc(mouse);
    glutMainLoop();
    return 0;
}

```

03: Menggambar Kubus (Example 3-1)

```

/* Prog-03: Menggambar Kubus */
#include <windows.h>
#include <GL/glut.h>
void init(void)
{
    glClearColor (0.0, 0.0, 0.0, 0.0);
    glShadeModel (GL_FLAT);
}
void display(void)
{
    glClear (GL_COLOR_BUFFER_BIT);
    glColor3f (1.0, 1.0, 1.0);
    glLoadIdentity (); /* clear the matrix */
    /* viewing transformation */
    gluLookAt (0.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
    glScalef (1.0, 2.0, 1.0); /* modeling transformation */
    glutWireCube (1.0);
    glFlush ();
}
void reshape (int w, int h)
{
    glViewport (0, 0, (GLsizei) w, (GLsizei) h);
    glMatrixMode (GL_PROJECTION);
    glLoadIdentity ();
    glFrustum (-1.0, 1.0, -1.0, 1.0, 1.5, 20.0);
    glMatrixMode (GL_MODELVIEW);
}

```



```

}
int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize (500, 500);
    glutInitWindowPosition (100, 100);
    glutCreateWindow (argv[0]);
    init ();
    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
    glutMainLoop();
    return 0;
}

```

04: Line Patterns (Example 2-5)

```

#include <windows.h>
#include <GL/glut.h>
#define drawOneLine(x1,y1,x2,y2) glBegin(GL_LINES); \
glVertex2f ((x1),(y1)); glVertex2f ((x2),(y2)); glEnd();
void init(void)
{
    glClearColor (0.0, 0.0, 0.0, 0.0);
    glShadeModel (GL_FLAT);
}
void display(void)
{
    int i;
    glClear (GL_COLOR_BUFFER_BIT);
    /* select white for all lines */
    glColor3f (1.0, 1.0, 1.0);
    /* in 1st row, 3 lines, each with a different stipple */
    glEnable (GL_LINE_STIPPLE);
    glLineStipple (1, 0x0101); /* dotted */
    drawOneLine (50.0, 125.0, 150.0, 125.0);
    glLineStipple (1, 0x00FF); /* dashed */
    drawOneLine (150.0, 125.0, 250.0, 125.0);
    glLineStipple (1, 0x1C47); /* dash/dot/dash */
    drawOneLine (250.0, 125.0, 350.0, 125.0);
    /* in 2nd row, 3 wide lines, each with different stipple */
    glLineWidth (5.0);
    glLineStipple (1, 0x0101); /* dotted */
    drawOneLine (50.0, 100.0, 150.0, 100.0);
    glLineStipple (1, 0x00FF); /* dashed */
    drawOneLine (150.0, 100.0, 250.0, 100.0);
    glLineStipple (1, 0x1C47); /* dash/dot/dash */
    drawOneLine (250.0, 100.0, 350.0, 100.0);
    glLineWidth (1.0);
    /* in 3rd row, 6 lines, with dash/dot/dash stipple */
    /* as part of a single connected line strip */
    glLineStipple (1, 0x1C47); /* dash/dot/dash */
    glBegin (GL_LINE_STRIP);
        for (i = 0; i < 7; i++)
            glVertex2f (50.0 + ((GLfloat) i * 50.0), 75.0);
    glEnd ();
    /* in 4th row, 6 independent lines with same stipple */
    for (i = 0; i < 6; i++) {
        drawOneLine (50.0 + ((GLfloat) i * 50.0), 50.0,
            50.0 + ((GLfloat)(i+1) * 50.0), 50.0);
    }

    /* in 5th row, 1 line, with dash/dot/dash stipple */
    /* and a stipple repeat factor of 5 */
    glLineStipple (5, 0x1C47); /* dash/dot/dash */
    drawOneLine (50.0, 25.0, 350.0, 25.0);
    glDisable (GL_LINE_STIPPLE);
    glFlush ();
}

```

```

void reshape (int w, int h)
{
    glViewport (0, 0, (GLsizei) w, (GLsizei) h);
    glMatrixMode (GL_PROJECTION);
    glLoadIdentity ();
    gluOrtho2D (0.0, (GLdouble) w, 0.0, (GLdouble) h);
}
int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize (400, 150);
    glutInitWindowPosition (100, 100);
    glutCreateWindow (argv[0]);
    init ();
    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
    glutMainLoop();
    return 0;
}

```

05: Polygon Patterns (Example 2-6)

```

#include <windows.h>
#include <GL/glut.h>
void display(void)
{
    GLubyte fly[] = {
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
        0x03, 0x80, 0x01, 0xC0, 0x06, 0xC0, 0x03, 0x60,
        0x04, 0x60, 0x06, 0x20, 0x04, 0x30, 0x0C, 0x20,
        0x04, 0x18, 0x18, 0x20, 0x04, 0x0C, 0x30, 0x20,
        0x04, 0x06, 0x60, 0x20, 0x44, 0x03, 0xC0, 0x22,
        0x44, 0x01, 0x80, 0x22, 0x44, 0x01, 0x80, 0x22,
        0x44, 0x01, 0x80, 0x22, 0x44, 0x01, 0x80, 0x22,
        0x44, 0x01, 0x80, 0x22, 0x44, 0x01, 0x80, 0x22,
        0x66, 0x01, 0x80, 0x66, 0x33, 0x01, 0x80, 0xCC,
        0x19, 0x81, 0x81, 0x98, 0x0C, 0xC1, 0x83, 0x30,
        0x07, 0xe1, 0x87, 0xe0, 0x03, 0x3f, 0xfc, 0xc0,
        0x03, 0x31, 0x8c, 0xc0, 0x03, 0x33, 0xcc, 0xc0,
        0x06, 0x64, 0x26, 0x60, 0x0c, 0xcc, 0x33, 0x30,
        0x18, 0xcc, 0x33, 0x18, 0x10, 0xc4, 0x23, 0x08,
        0x10, 0x63, 0xC6, 0x08, 0x10, 0x30, 0x0c, 0x08,
        0x10, 0x18, 0x18, 0x08, 0x10, 0x00, 0x00, 0x08};
    GLubyte halftone[] = {
        0xAA, 0xAA, 0xAA, 0xAA, 0x55, 0x55, 0x55, 0x55,
        0xAA, 0xAA, 0xAA, 0xAA, 0x55, 0x55, 0x55, 0x55,
        0xAA, 0xAA, 0xAA, 0xAA, 0x55, 0x55, 0x55, 0x55,
        0xAA, 0xAA, 0xAA, 0xAA, 0x55, 0x55, 0x55, 0x55,
        0xAA, 0xAA, 0xAA, 0xAA, 0x55, 0x55, 0x55, 0x55,
        0xAA, 0xAA, 0xAA, 0xAA, 0x55, 0x55, 0x55, 0x55,
        0xAA, 0xAA, 0xAA, 0xAA, 0x55, 0x55, 0x55, 0x55,
        0xAA, 0xAA, 0xAA, 0xAA, 0x55, 0x55, 0x55, 0x55,
        0xAA, 0xAA, 0xAA, 0xAA, 0x55, 0x55, 0x55, 0x55,
        0xAA, 0xAA, 0xAA, 0xAA, 0x55, 0x55, 0x55, 0x55,
        0xAA, 0xAA, 0xAA, 0xAA, 0x55, 0x55, 0x55, 0x55,
        0xAA, 0xAA, 0xAA, 0xAA, 0x55, 0x55, 0x55, 0x55,
        0xAA, 0xAA, 0xAA, 0xAA, 0x55, 0x55, 0x55, 0x55,
        0xAA, 0xAA, 0xAA, 0xAA, 0x55, 0x55, 0x55, 0x55,
        0xAA, 0xAA, 0xAA, 0xAA, 0x55, 0x55, 0x55, 0x55};

    glClear (GL_COLOR_BUFFER_BIT);
    glColor3f (1.0, 1.0, 1.0);
    /* draw one solid, unstippled rectangle, */
    /* then two stippled rectangles */
    glRectf (25.0, 25.0, 125.0, 125.0);
    glEnable (GL_POLYGON_STIPPLE);
}

```

```

        glPolygonStipple (fly);
        glRectf (125.0, 25.0, 225.0, 125.0);
        glPolygonStipple (halftone);
        glRectf (225.0, 25.0, 325.0, 125.0);
        glDisable (GL_POLYGON_STIPPLE);
        glFlush ();
    }
    void init (void)
    {
        glClearColor (0.0, 0.0, 0.0, 0.0);
        glShadeModel (GL_FLAT);
    }
    void reshape (int w, int h)
    {
        glViewport (0, 0, (GLsizei) w, (GLsizei) h);
        glMatrixMode (GL_PROJECTION);
        glLoadIdentity ();
        gluOrtho2D (0.0, (GLdouble) w, 0.0, (GLdouble) h);
    }
    int main(int argc, char** argv)
    {
        glutInit(&argc, argv);
        glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
        glutInitWindowSize (350, 150);
        glutCreateWindow (argv[0]);
        init ();
        glutDisplayFunc(display);
        glutReshapeFunc(reshape);
        glutMainLoop();
        return 0;
    }

```

06: Wireframe Sphere (Example 3-5)

```

#include <windows.h>
#include <GL/glut.h>
void init(void)
{
    glClearColor (0.0, 0.0, 0.0, 0.0);
    glShadeModel (GL_FLAT);
}
void display(void)
{
    GLdouble eqn[4] = {0.0, 1.0, 0.0, 0.0};
    GLdouble eqn2[4] = {1.0, 0.0, 0.0, 0.0};
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f (1.0, 1.0, 1.0);
    glPushMatrix();
    glTranslatef (0.0, 0.0, -5.0);
    /* clip lower half -- y < 0 */
    glClipPlane (GL_CLIP_PLANE0, eqn);
    glEnable (GL_CLIP_PLANE0);
    /* clip left half -- x < 0 */
    glClipPlane (GL_CLIP_PLANE1, eqn2);
    glEnable (GL_CLIP_PLANE1);
    glRotatef (90.0, 1.0, 0.0, 0.0);
    glutWireSphere(1.0, 20, 16);
    glPopMatrix();
    glFlush ();
}
void reshape (int w, int h)
{
    glViewport (0, 0, (GLsizei) w, (GLsizei) h);
    glMatrixMode (GL_PROJECTION);
    glLoadIdentity ();
    gluPerspective(60.0, (GLfloat) w/(GLfloat) h, 1.0, 20.0);
    glMatrixMode (GL_MODELVIEW);
}

```

```

int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize (500, 500);
    glutInitWindowPosition (100, 100);
    glutCreateWindow (argv[0]);
    init ();
    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
    glutMainLoop();
    return 0;
}

```

07: Planetary System (Example 3-6)

```

#include <windows.h>
#include <GL/glut.h>
static int year = 0, day = 0;
void init(void)
{
    glClearColor (0.0, 0.0, 0.0, 0.0);
    glShadeModel (GL_FLAT);
}
void display(void)
{
    glClear (GL_COLOR_BUFFER_BIT);
    glColor3f (1.0, 1.0, 1.0);
    glPushMatrix();
    glutWireSphere(1.0, 20, 16); /* draw sun */
    glRotatef ((GLfloat) year, 0.0, 1.0, 0.0);
    glTranslatef (2.0, 0.0, 0.0);
    glRotatef ((GLfloat) day, 0.0, 1.0, 0.0);
    glutWireSphere(0.2, 10, 8); /* draw smaller planet */
    glPopMatrix();
    glutSwapBuffers();
}
void reshape (int w, int h)
{
    glViewport (0, 0, (GLsizei) w, (GLsizei) h);
    glMatrixMode (GL_PROJECTION);
    glLoadIdentity ();
    gluPerspective(60.0, (GLfloat) w/(GLfloat) h, 1.0, 20.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt (0.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
}
void keyboard (unsigned char key, int x, int y)
{
    switch (key) {
        case 'd':
            day = (day + 10) % 360;
            glutPostRedisplay();
            break;
        case 'D':
            day = (day - 10) % 360;
            glutPostRedisplay();
            break;
        case 'y':
            year = (year + 5) % 360;
            glutPostRedisplay();
            break;
        case 'Y':
            year = (year - 5) % 360;
            glutPostRedisplay();
            break;
        default:
            break;
    }
}

```

```

    }
}
int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode (GLUT_DOUBLE | GLUT_RGB);
    glutInitWindowSize (500, 500);
    glutInitWindowPosition (100, 100);
    glutCreateWindow (argv[0]);
    init ();
    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
    glutKeyboardFunc(keyboard);
    glutMainLoop();
    return 0;
}

```

08: Tangan ROBOT

```

#include <windows.h>
#include <GL/glut.h>
static int shoulder = 0, elbow = 0;
void init(void)
{
    glClearColor (0.0, 0.0, 0.0, 0.0);
    glShadeModel (GL_FLAT);
}
void display(void)
{
    glClear (GL_COLOR_BUFFER_BIT);
    glPushMatrix();
    glTranslatef (-1.0, 0.0, 0.0);
    glRotatef ((GLfloat) shoulder, 0.0, 0.0, 1.0);
    glTranslatef (1.0, 0.0, 0.0);
    glPushMatrix();
    glScalef (2.0, 0.4, 1.0);
    glutWireCube (1.0);
    glPopMatrix();
    glTranslatef (1.0, 0.0, 0.0);
    glRotatef ((GLfloat) elbow, 0.0, 0.0, 1.0);
    glTranslatef (1.0, 0.0, 0.0);
    glPushMatrix();
    glScalef (2.0, 0.4, 1.0);
    glutWireCube (1.0);
    glPopMatrix();
    glPopMatrix();
    glutSwapBuffers();
}
void reshape (int w, int h)
{
    glViewport (0, 0, (GLsizei) w, (GLsizei) h);
    glMatrixMode (GL_PROJECTION);
    glLoadIdentity ();
    gluPerspective(65.0, (GLfloat) w/(GLfloat) h, 1.0, 20.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glTranslatef (0.0, 0.0, -5.0);
}
void keyboard (unsigned char key, int x, int y)
{
    switch (key) {
        case 's': /* s key rotates at shoulder */
            shoulder = (shoulder + 5) % 360;
            glutPostRedisplay();
            break;
        case 'S':
            shoulder = (shoulder - 5) % 360;
            glutPostRedisplay();

```

```

        break;
    case 'e': /* e key rotates at elbow */
        elbow = (elbow + 5) % 360;
        glutPostRedisplay();
        break;
    case 'E':
        elbow = (elbow - 5) % 360;
        glutPostRedisplay();
        break;
    default:
        break;
}
}
int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode (GLUT_DOUBLE | GLUT_RGB);
    glutInitWindowSize (500, 500);
    glutInitWindowPosition (100, 100);
    glutCreateWindow (argv[0]);
    init ();
    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
    glutKeyboardFunc(keyboard);
    glutMainLoop();
    return 0;
}

```

09: Geometric Processing Pipeline (Example 3-8)

```

#include <windows.h>
#include <GL/glut.h>
#include <stdlib.h>
#include <stdio.h>
void display(void)
{
    glClear(GL_COLOR_BUFFER_BIT);
    glFlush();
}
void reshape(int w, int h)
{
    glViewport (0, 0, (GLsizei) w, (GLsizei) h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective (45.0, (GLfloat) w/(GLfloat) h, 1.0, 100.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}
void mouse(int button, int state, int x, int y)
{
    GLint viewport[4];
    GLdouble mvmatrix[16], projmatrix[16];
    GLint really; /* OpenGL y coordinate position */
    GLdouble wx, wy, wz; /* returned world x, y, z coords */
    switch (button) {
    case GLUT_LEFT_BUTTON:
        if (state == GLUT_DOWN) {
            glGetIntegerv (GL_VIEWPORT, viewport);
            glGetDoublev (GL_MODELVIEW_MATRIX, mvmatrix);
            glGetDoublev (GL_PROJECTION_MATRIX, projmatrix);
            /* note viewport[3] is height of window in pixels */
            really = viewport[3] - (GLint) y - 1;
            printf ("Coordinates at cursor are (%4d, %4d)\n",
                x, really);
            gluUnProject ((GLdouble) x, (GLdouble) really, 0.0,
                mvmatrix, projmatrix, viewport, &wx, &wy, &wz);
            printf ("World coords at z=0.0 are (%f, %f, %f)\n",
                wx, wy, wz);
        }
    }
}

```

```

        gluUnProject ((GLdouble) x, (GLdouble) really, 1.0,
        mvmatrix, projmatrix, viewport, &wx, &wy, &wz);
        printf ("World coords at z=1.0 are (%f, %f, %f)\n",
        wx, wy, wz);
    }
    break;
case GLUT_RIGHT_BUTTON:
    if (state == GLUT_DOWN)
        exit(0);
    break;
default:
    break;
}
}
int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize (500, 500);
    glutInitWindowPosition (100, 100);
    glutCreateWindow (argv[0]);
    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
    glutMouseFunc(mouse);
    glutMainLoop();
    return 0;
}

```

10: Menggambar Segitiga Warna-warni Bergradasi (Example 4-1)

```

#include <windows.h>
#include <GL/glut.h>
void init(void)
{
    glClearColor (0.0, 0.0, 0.0, 0.0);
    glShadeModel (GL_SMOOTH);
}
void triangle(void)
{
    glBegin (GL_TRIANGLES);
    glColor3f (1.0, 0.0, 0.0);
    glVertex2f (5.0, 5.0);
    glColor3f (0.0, 1.0, 0.0);
    glVertex2f (25.0, 5.0);
    glColor3f (0.0, 0.0, 1.0);
    glVertex2f (5.0, 25.0);
    glEnd();
}
void display(void)
{
    glClear (GL_COLOR_BUFFER_BIT);
    triangle ();
    glFlush ();
}
void reshape (int w, int h)
{
    glViewport (0, 0, (GLsizei) w, (GLsizei) h);
    glMatrixMode (GL_PROJECTION);
    glLoadIdentity ();
    if (w <= h)
        gluOrtho2D (0.0, 30.0, 0.0, 30.0*(GLfloat) h/(GLfloat) w);
    else
        gluOrtho2D (0.0, 30.0*(GLfloat) w/(GLfloat) h, 0.0, 30.0);
    glMatrixMode(GL_MODELVIEW);
}
int main(int argc, char** argv)
{
    glutInit(&argc, argv);

```

```

    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize (500, 500);
    glutInitWindowPosition (100, 100);
    glutCreateWindow (argv[0]);
    init ();
    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
    glutMainLoop();
    return 0;
}

```

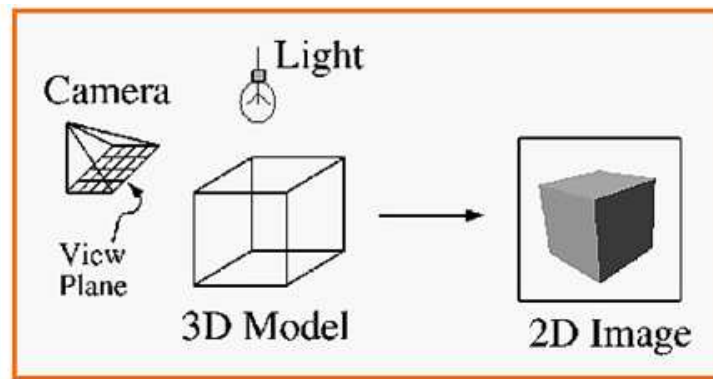
Daftar Pustaka

1. Donald D. Hearn, M. Pauline, Warren Carithers, Computer Graphics with Open GL (4th Edition), Prentice-Hall, 2011
2. Edward Angel, Interactive Computer Graphics: A Top-Down Approach with OpenGL 2nd, Addison Wesley, 2005
3. John F. Hughes, Andries Van Dam, Morgan Mcguire, David F. Sklar, James D. Foley, Steven K. Feiner, Kurt Akeley, Computer Graphics: Principles and Practice (3rd edition), Addison-Wesley, 2014

TUGAS GRAFIKA KOMPUTER

A. Pengetahuan Konseptual

1) Jelaskan gambar ini

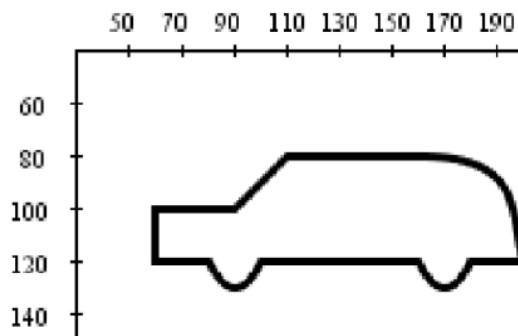


2) Jelaskan istilah ini, ilustrasikan dengan gambar

- a) Refresh rate
- b) Scan line
- c) Horizontal retrace
- d) Vertical retrace

B. Pengetahuan Programatikal

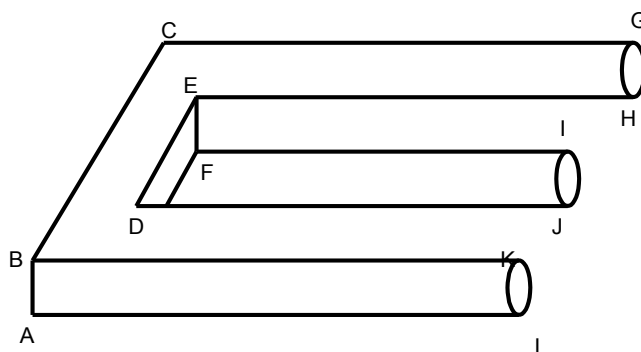
- 1) Tuliskan algoritma perkalian matriks secara umum
- 2) Tuliskan program dalam bahasa C# untuk menggambar objek mobil berikut ini



C. Kemampuan Numerikal

Diketahui sebuah objek seperti pada gambar di sebelah. Vektor posisinya adalah

$\{(1\ 1), (1\ 2), (5\ 6), (4\ 3), (6\ 5), (6\ 4), (13.5\ 6), (13.5\ 5), (12.5\ 5), (12.5\ 4), (11.5\ 2), (11.5\ 1)\}$.



Tentukan koordinat objek dan gambarkan untuk urutan proses transformasi

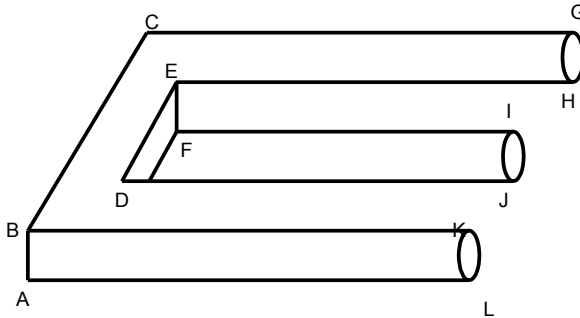
- 1) rotasi dengan sumbu rotasi adalah sumbu origin sebesar 60° ccw
- 2) Refleksi pada cermin yang terletak sejajar sumbu-x
- 3) local scaling sebesar 0.5

QUIZ GRAFIKA KOMPUTER

Waktu : 100 menit Close Book Kecuali Cheat Sheet Satu Lembar

Jawablah pertanyaan di bawah ini pada lembar jawaban secara berurutan.

1. Tiga bidang konsentrasi grafika komputer adalah : Modeling, _____ dan _____
2. Bentuk grafis paling sederhana dari sebuah model disebut _____
3. Dengan algoritma DDA, maka titik-titik yang dilalui oleh garis antara (2,1) dan (6,8) adalah (2,1) _____, (6,8).
4. An area on a display device to which a window is mapped is called _____
5. Koordinat viewport dari titik (7,5) yang terdapat pada window, dengan spesifikasi windows=(4,5,9,11) dan viewport=(3,4,5,7) adalah _____
6. Tuliskan method dalam VS C# 2010 untuk menggambar grafik SEGITIGA !
7. Diketahui objek berikut:



Koordinat ABCDEFGHIJKL adalah
(1 1), (1 2), (5 6), (4 3), (6 5), (6 4), (13 6), (13 5), (11 4), (11 4), (9 2), (9 1).

Tentukan koordinat objek baru untuk transformasi rotasi sebesar 30° CCW dengan pusat rotasi $(-1, -1)$.

8. Diketahui sebuah objek G dan objek PQR . Persamaan objek G adalah $y=(x+4)/2$ dan objek PQR memiliki vektor posisi $\{(2, 4), (4, 6), (2, 6)\}$, maka
 - a. Sudut yang dibentuk antara G dan sumbu X adalah _____
 - b. Matriks transformasi umum untuk refleksi objek PQR tersebut pada cermin yang berimpit dengan objek G adalah _____
 - c. Koordinat objek baru hasil transformasi di atas adalah _____
9. Ujilah apakah algoritma berikut ini adalah algoritma perkalian matriks . Jika bukan bagaimana seharusnya algoritma perkalian matriks yang benar?

```
for i=1 to m          // m adalah banyak baris
  for j=1 to n        // n adalah banyak kolom
    for k=1 to m
      Cij = Aik * Bkj;
```

Catatan nilai tiap soal:

Soal 1=Nilai 2; 2=1, 3=5; 4=1; 5=5; 6=6; 7=10; 8=15; 9=5. Total nilai adalah 50



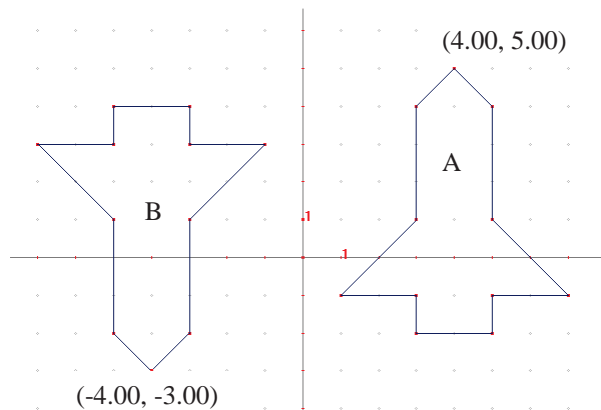
KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN
UNIVERSITAS PADJADJARAN
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
PROGRAM STUDI S-1 TEKNIK INFORMATIKA

Jl. Raya Bandung-Sumedang Km. 21 Jatinangor 45363 Telp./Fax. 022 7794696
<http://informatika.unpad.ac.id>, e-mail : informatika@unpad.ac.id

UJIAN TENGAH SEMESTER GANJIL 2014/2015

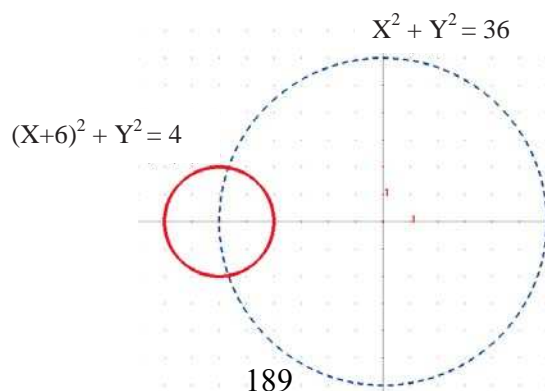
Mata Kuliah : Grafika Komputer
Dosen : Dr. Setiawan Hadi
Kelas : -
Waktu : Selasa 28 Oktober 2014, 13.15-14.45 (90 menit)
Sifat : Terbuka (Catatan dan Notebook diperbolehkan)

- (Nilai 50) Diketahui sebuah objek 2D dengan susunan koordinat $(-9, 5)$, $(-5, 5)$, $(-5, 2)$, $(-9, 2)$, $(-7, 6)$, $(-3, 6)$, $(-3, 3)$.
 - Gambarkan objek tersebut
 - Tentukan MTU
 - jika objek tersebut direfleksikan pada cermin yang berimpit pada garis $y = 4x/3 + 4$
 - selanjutnya dirotasikan sebesar 163.4° CCW dengan pusat rotasi $(6,0)$
 - Tentukan koordinat objek baru hasil refleksi dan hasil rotasi
 - Gambarkan objek baru hasil refleksi dan rotasi tersebut
- (Nilai 25) Diketahui objek A adalah objek asli dan objek B adalah objek hasil transformasi.



Tentukan:

- Jenis transformasi dan nilai komponen-komponen dari transformasinya
 - Matriks transformasi umumnya
- (Nilai (25) Diketahui gambar ilustrasi berikut ini: Lingkaran Kecil adalah sebuah objek dan Lingkaran Besar dengan garis putus-putus adalah lingkaran orbit. Tentukan **Matriks Transformasi Umum** untuk membuat lingkaran yang kecil berputar dalam orbit lingkaran yang besar.



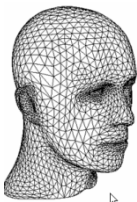
UJIAN AKHIR SEMESTER

Mata Kuliah : Grafika Komputer
Sifat Ujian : Close Book, Cheatsheet diperbolehkan !
Waktu / Durasi : 90 menit

SOAL PILIHAN BERGANDA

Jawaban -soal pilihan berganda bisa lebih dari satu atau tidak ada sama sekali !!!

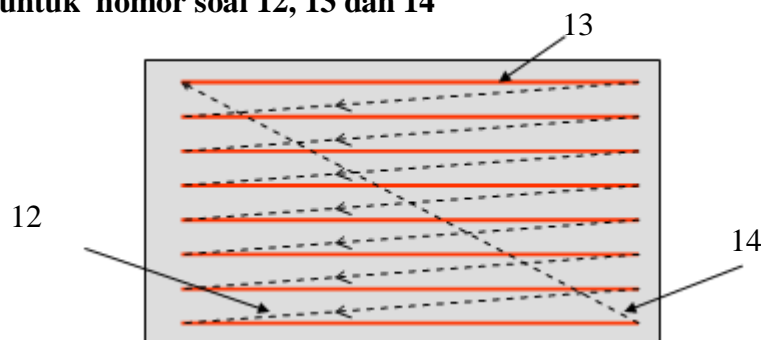
- Pada grafika komputer dikenal elemen-elemen yang menjadi konsentrasi utama yaitu
a. Transformation b. Modeling c. Rendering e. Morphing e. Animation
- Teknik grafika komputer yang merepresentasikan secara geometri objek nyata disebut
a. Pengolahan citra b. Pemodelan c. Animasi d. Presentasi e. Variasi
- Hasil dari sebuah pemodelan objek nyata adalah bentuk grafis paling sederhana yang disebut gambar
a. Wireframe b. Binary Image c. Wiremodel d. Dijital e. Mainframe
- Ilmu matematika yang erat kaitannya dengan grafika komputer adalah
a. Matriks b. Persamaan Garis & Lingkaran c. Seni d. Pengolahan Citra e. Visi Komputer
- Dilihat dari teknologi alat output, urutan teknologi yang menunjukkan kualitas pencetakan yang semakin berkualitas adalah
a. Laser b. Scanner c. Inkjet /Bubblejet
d. Digitizer e. Dot Matrix



- Gambar di bawah ini disebut dengan gambar atau model
a. Sederhana b. Wireless c. Monochrome
d. Wi-Fi e. Wireframe

- Warna RGB yang muncul pada layar komputer dipancarkan dari
a. Electromagnetic b. Tube Vacuum c. Machine Gun d. Phosphor e. Electron Gun
- Perangkat lunak grafis terkenal yang umum dan menggunakan format gambar vektor adalah
a. PaintShop b. Coreldraw c. 3D Studio Max d. Autocad e. Microsoft Paint
- Algoritma-algoritma persamaan garis dan lingkaran yang umum digunakan adalah
a. DDA b. Line Drawing c. Birmingham d. Image Drawing e. Bresenham Line
- Jika digunakan algoritma DDA, salah satu titik yang TIDAK dilalui oleh garis antara 1,1 ke 8,6 adalah
a. 1,1 b. 2,2 c. 3,3 d. 3, 5 e. 4,6
- Teknik untuk 'menghilangkan' efek jaggy adalah
a. Anti jaggy b. Straightforward c. Anti Aliasing d. Pixel Phasing e. Semua salah

Gambar di bawah ini untuk nomor soal 12, 13 dan 14



- a. Horizontal retrace b. Scan line c. Vertical retrace d. Resolution e. Monitor
- a. Horizontal retrace b. Scan line c. Vertical retrace d. Resolution e. Monitor

14. a. Horizontal retrace b. Scan line c. Vertical retrace d. Resolution e. Monitor
15. Jenis citra pada gambar yang bisa menimbulkan hilangnya ketajaman gambar karena menurunnya tingkat resolusi gambar adalah jenis citra
a. Raster b. Citra RGB c. Vektor d. Zoom-in e. Zoom-out
16. Istilah grafika komputer (**computer graphics**) diungkapkan pada tahun 1960 oleh
a. Stalling b. Apache c. Fetter d. Apostol e. Knuth
17. Matriks Transformasi Umum untuk rotasi sebesar θ° CW dengan pusat origin adalah
a. $\begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix}$ b. $\begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix}$ c. $\begin{bmatrix} \cos \beta & \sin \beta \\ -\sin \beta & \cos \beta \end{bmatrix}$ d. $\begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \sin \theta \end{bmatrix}$ e. $\begin{bmatrix} \cos \theta & -\sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix}$
18. Jika sebuah segitiga ABC dengan koordinat $\{(4,1), (5,2), (4,3)\}$ direfleksikan pada sumbu x , kemudian direfleksikan pada garis $y = -x$, maka hasilnya adalah segitiga $A^+B^+C^+$ dengan koordinat
a. $\{(4,-1), (5,-2), (4,-3)\}$ b. $\{(1,-4), (2,-5), (3,-4)\}$ c. $\{(4,-1), (5,2), (4,-3)\}$
d. $\{(1,4), (2,-5), (4,-3)\}$ e. Semua jawaban tidak ada yang benar
19. Kotak X dan Y pada gambar disebelah kiri adalah
a. Viewport dan Window
b. Window dan Crop
c. Window dan ViewMask
d. Clipping dan Windowing
e. Semua jawaban salah

Soal untuk no 20 s/d 22. Diketahui sebuah objek **G** dan objek **PQR**. Persamaan objek **G** adalah $y=0.5x+2$ dan objek **PQR** memiliki vektor posisi $\{(2, 4), (4, 6), (2, 6)\}$. Maka

20. Sudut yang dibentuk antara objek G dan sumbu horizontal X adalah
a. 26.56° b. 63.40° c. 52.66° d. 20.43° e. 35.56°
21. Matriks transformasi umum untuk refleksi objek **PQR** tersebut pada cermin yang berimpit dengan objek **G** adalah
a. $\begin{bmatrix} 0.6 & -0.8 & 0 \\ 0.8 & 0.6 & 0 \\ -1.6 & 3.2 & 1 \end{bmatrix}$ b. $\begin{bmatrix} 0.8 & 0.6 & 0 \\ 0.6 & -0.8 & 0 \\ -1.6 & 3.2 & 1 \end{bmatrix}$ c. $\begin{bmatrix} 0.6 & 0.8 & 0 \\ -0.8 & 0.6 & 0 \\ -1.6 & 3.2 & 1 \end{bmatrix}$ d. $\begin{bmatrix} 0.6 & 0.8 & 0 \\ 0.8 & -0.6 & 0 \\ 1.6 & -3.2 & 1 \end{bmatrix}$ e. $\begin{bmatrix} 0.6 & 0.8 & 0 \\ 0.8 & -0.6 & 0 \\ -1.6 & 3.2 & 1 \end{bmatrix}$
22. Koordinat objek baru hasil transformasi di atas adalah
a. $\{(2.8, 2.4), (5.6, 2.8), (4.4, 1.8)\}$ b. $\{(2.8, 2.4), (5.6, 2.8), (4.4, 1.2)\}$
c. $\{(2.4, 2.8), (5.6, 2.8), (4.4, 1.2)\}$ d. $\{(2.8, 2.4), (5.8, 2.6), (4.4, 1.8)\}$ e. Semua salah
23. Elemen skala pada matriks transformasi umum 3 dimensi terdapat pada
a. baris ke 4 kolom 1 b. baris ke 4 kolom 1 dan 2 c. baris ke 4 kolom 1,2 dan 3
d. baris ke 4 kolom 1,2,3 dan 4 e. Semua jawaban tidak ada yang benar

24. MTU Rotasi objek 3 dimensi pada sumbu Z adalah
a. $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \varphi & \sin \varphi & 0 \\ 0 & -\sin \varphi & \cos \varphi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ b. $\begin{bmatrix} \cos \varphi & \sin \varphi & 0 & 0 \\ \sin \varphi & \cos \varphi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ c. $\begin{bmatrix} \cos \varphi & 0 & -\sin \varphi & 0 \\ 0 & 1 & 0 & 0 \\ \sin \varphi & 0 & \cos \varphi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
d. $\begin{bmatrix} \cos \varphi & \sin \varphi & 0 & 0 \\ -\sin \varphi & \cos \varphi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ e. $\begin{bmatrix} \cos \varphi & 0 & -\sin \varphi & 0 \\ 0 & 0 & 0 & 0 \\ \sin \varphi & 0 & \cos \varphi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

25. Jika diketahui sebuah objek 3D dengan koordinat $\{(0,0,1), (1,0,1), (1,1,1), (0,1,1), (0,0,0), (1,0,0), (1,1,0), (0,1,0)\}$, maka Matriks Transformasi Umum apabila objek tersebut dirotasikan sebesar 60° pada sumbu y , kemudian ditranslasikan sebesar $y=2$ adalah

- a.

$$\begin{bmatrix} 0.5 & 0 & 0 & 0.35 \\ 0 & 1 & 0 & 0 \\ 0.87 & 0 & 0 & -0.2 \\ 0 & -2 & 0 & 1 \end{bmatrix}$$
- b.

$$\begin{bmatrix} 0.5 & 0 & -0.87 & 0 \\ 0 & 1 & 0 & 0 \\ 0.87 & 0 & 0.5 & 0 \\ 0 & -2 & 0 & 1 \end{bmatrix}$$
- c.

$$\begin{bmatrix} 0.35 & 0 & 0 & 0.5 \\ 0 & 1 & 0 & 0 \\ 0.87 & 0 & 0 & -0.2 \\ 0 & -2 & 0 & 1 \end{bmatrix}$$
- d.

$$\begin{bmatrix} 0.5 & 0 & 0 & 0.35 \\ 0 & 1 & 0 & 0 \\ 0.7 & 0 & 0 & 0.2 \\ 0 & -2 & 0 & 1 \end{bmatrix}$$

e. Semua jawaban tidak ada yang benar

26. Terdapat dua jenis penskalaan dalam transformasi 3 dimensi, yaitu penskalaan local dan penskalaan overall. Dalam matriks transformasi umum, penskalaan lokal dipengaruhi oleh elemen

- a. Baris ke 4
- b. Kolom ke 4
- c. Diagonal
- d. Vertikal
- e. Horizontal

27. Jika diketahui rumusan sebagai berikut:

$$[X][T] = \begin{bmatrix} x & y & z & 1 \end{bmatrix} \begin{bmatrix} 1 & b & c & 0 \\ d & 1 & f & 0 \\ g & i & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Maka transformasi $[T]$ disebut dengan

- a. Refleksi
- b. Rotasi.
- c. Proyeksi
- d. Shearing
- e. Scaling

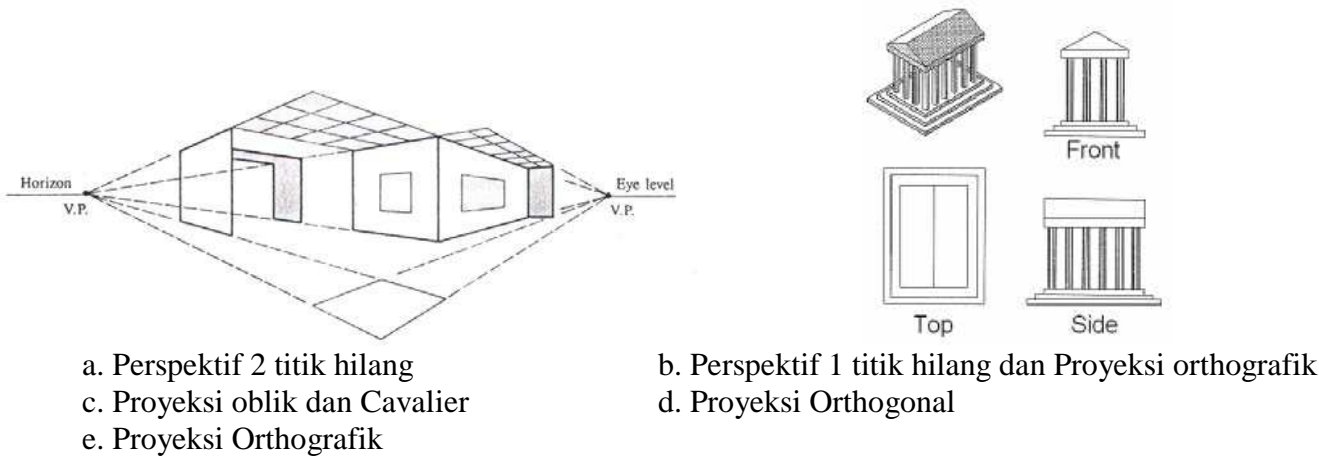
28. Transformasi umum untuk Rotasi 3D dengan pusat rotasi sembarang adalah

- a. $[M] = [T]^{-1}[R_x]^{-1}[R_y]^{-1}[R_\delta][R_y][R_x][T]$
- b. $[M] = [R_y]^{-1}[R_x]^{-1}[T]^{-1}[R_\delta][R_y][R_x][T]$
- c. $[M] = [T][R_x][R_y][R_\delta][R_y]^{-1}[R_x]^{-1}[T]^{-1}$
- d. $[M] = [T][R_x][R_y][R_\delta][R_\delta]^{-1}[R_y]^{-1}[R_x]^{-1}[T]^{-1}$
- e. $[M] = [T][R_x][R_y][R_\delta][R_y][R_x][T]$

29. Proyeksi geometri bidang secara umum dikategorikan menjadi dua bagian besar yaitu

- a. Proyeksi paralel dan proyeksi seri
- b. Proyeksi kavalier dan proyeksi kabinet
- c. Proyeksi ortografik dan proyeksi oblik
- d. Proyeksi perspektif dan orthografik
- e. Proyeksi paralel dan proyeksi perspektif

30. Gambar-gambar di bawah ini adalah



SOAL ESSAY (Point 50%)

Diketahui sebuah objek 3Dimensi dengan Vektor Posisi Homogen sebagai berikut $\{(0\ 0\ 1), (1\ 0\ 1), (1\ 1\ 1), (0\ 1\ 1), (0\ 0\ 0), (1\ 0\ 0), (1\ 1\ 0), (0\ 1\ 0)\}$.

Tentukan :

- a. Matriks Transformasi Umum apabila objek tersebut dirotasikan sebesar 45° pada sumbu- y , kemudian ditranslasikan sebesar $y = 2$ dan kemudian diproyeksikan kepada bidang $z = 0$ dengan pusat proyeksi pada $z = z_c = 2.5$
- b. Koordinat objek baru hasil transformasi dengan MTU di atas
- c. Tentukan koordinat titik (-titik) hilangnya
- d. Gambarkan (opsional) Bonus 10%