
Grafika Komputer
Irfan Maliki, S.T

DAFTAR ISI

1 Pendahuluan 1

1.1 Pengertian Grafika Komputer 1

1.2 Sejarah Grafika Komputer 1

1.3 Peranan dan Penggunaan Grafika Komputer 2

1.4 Desain 2

1.5 Grafik Presentasi 3

1.6 Computer Art 3

1.7 Film 4

1.8 Televisi 5

1.9 Video musik 5

1.10 Game 6

1.11 Pendidikan 6

1.12 Visualisasi 7

1.13 Image Processing 8

1.14 Graphical User Interface (GUI) 9

2 SISTEM GRAFIKA KOMPUTER 10

2.1 Teknologi Display 10

2.2 Raster-scan Display 11

2.3 Random-scan Display 12

2.4 Monitor Color CRT 12

2.5 Flat Panel Display 13

2.6 Peralatan Input Interaktif 14

2.7 Peralatan Hardcopy 15

2.8 Perangkat Lunak Grafika 16

3 OUTPUT PRIMITIF 17

3.1 Titik dan Garis 17

3.2 Algoritma pembentukan garis 17

3.2.1 Algoritma garis DDA 18

3.2.2 Algoritma Garis Bressenhem 20

3.3 Algoritma Pembentukan Lingkaran 22
4 ATRIBUT OUTPUT PRIMITIF 25
4.1 Atribut Garis 25
4.1.1 Tipe Garis 25
4.1.2 Tebal Garis 25
4.1.3 Pilihan Pen dan Brush 26
4.1.4 Warna Garis 26
4.2 Fill Area Primitif 26
4.2.1 Algoritma Boundary-Fill 26
4.2.2 Algoritma Flood-Fill 27
4.3 Pembentukan Karakter 27
4.4 Antialiasing 27
4.4.1 Supersampling atau Postfiltering 28
4.4.2 Area Sampling 29
4.4.3 Pixel Phasing 30
5 Transformasi 2 Dimensi 31
5.1 Translasi 31
5.2 Penskalaan 32
5.3 Perputaran (Rotasi) 33
5.4 Refleksi 35
5.5 Shear 36
6 Clipping 2 Dimensi 38
6.1 Clipping Garis 39
6.1.1 Algoritma Cohen-Sutherland (CS) 39
6.2 Kliping Poligon 41
6.2.1 Algoritma Sutherland-Hodgeman (SH) 41
7 3 DIMENSI 44
7.1 Konsep Dasar 3 Dimensi 44
7.2 Sistem Koordinat 44
7.3 Benda Tiga Dimensi 45
7.4 Struktur Data Benda 3 Dimensi 46
8 Proyeksi 49

8.1 Proyeksi Paralel 50
8.1.1 Proyeksi Orthographic 50
8.1.2 Proyeksi Oblique 51
8.2 Proyeksi Perspektif 52
9 Rendering 53
9.1 Warna 53
9.2 Pencahayaan (Lighting) 54
9.2.1 Cahaya Tersebar (Diffuse) 55
9.3 Metode *Shading* 55
9.3.1 Metode Flat Shading 56
9.3.2 Metode Gouraud Shading 57
9.4 Texture Mapping 60
10 Transformasi 3 Dimensi 62
10.1 Translasi 63
10.2 Penskalaan 64
10.3 Rotasi 64
11 Konsep Dasar OpenGL 67
11.1 Sintaks Perintah OpenGL 67
11.2 Library yang Berhubungan dengan OpenGL 68
11.3 Menggambar Objek Geometri 68
11.3.1 Membersihkan Windows 68
11.3.2 Spesifikasi Warna 69
11.3.3 Memaksa Proses Menggambar Sampai Selesai 69
11.3.4 Menggambar di Bidang Tiga Dimensi 69
11.3.5 Metode Hidden_Surface Removal 71
11.3.6 Teknik Viewing pada OpenGL 71
11.3.7 Transformasi Modelling 73
11.3.8 Transformasi Viewing 73
11.3.9 Transformasi Proyeksi 74
11.3.10 Pewarnaan pada OpenGL 75
11.3.11 Pencahayaan 75

DAFTAR GAMBAR

Gambar 1-1 Rancangan Pesawat Terbang 2
Gambar 1-2 Rancangan Mobil 3
Gambar 1-3 Contoh Penggunaan 3D 3
Gambar 1-4 Grafik Presentasi 3
Gambar 1-5 Contoh Software Computer Art 4
Gambar 1-6 Pemanfaatan Software Computer Art 4
Gambar 1-7 Film Animasi 4
Gambar 1-8 Film 3 D 5
Gambar 1-9 Special Effect 5
Gambar 1-10 Pembuatan Iklan 5
Gambar 1-11 Grafik audio 6
Gambar 1-12 Game 2D 6
Gambar 1-13 Game 3D 6
Gambar 1-14 Aplikasi Bidang Pengajaran 7
Gambar 1-15 Visualisasi Data Penelitian 8
Gambar 1-16 Visualisasi Data Penelitian 8
Gambar 1-17 Bidang Kedokteran 8
Gambar 1-18 Citra Sebelum dan Sesudah di proses 9
Gambar 1-19 Graphical User Interface (GUI) 9
Gambar 2-1 Monitor CRT 11
Gambar 2-2 Rangkaian Monitor CRT 11
Gambar 2-3 Monitor Color CRT 13
Gambar 2-4 Monitor Plasma 14
Gambar 2-5 Monitor LCD 14
Gambar 2-6 Monitor Touch Screen 14
Gambar 2-7 Scanner 15
Gambar 2-8 Barcode Scanner 15
Gambar 2-9 Fingerprint Scanner 15
Gambar 2-10 Printer 15

Gambar 2-11 Plotter 16
Gambar 2-12 Software Open GL 16
Gambar 6-1 Kode Cohen Sutherland 40
Gambar 7-1 Sistem koordinat kartesian 3 dimensi 44
Gambar 7-2 Permukaan tiga dimensi 45
Gambar 7-3 Polygon 45
Gambar 7-4 Arah vertex : (a) berlawanan jarum jam; (b) searah jarum jam 46
Gambar 7-5 Kubus 46
Gambar 8-1 Proyeksi Planar 49
Gambar 8-2 transformasi untuk memperoleh proyeksi orthographic 50
Gambar 7-3 proyeksi axonometric 51
Gambar 8-4 proyeksi Oblique dari titik p(x,y,z) ke titik q(xp,yp) 51
Gambar 9-1 Model RGB 53
Gambar 9-2 Perjalanan cahaya dari sumber cahaya 54
Gambar 9-3Cahaya Pantulan Tersebar (Diffuse) 55
Gambar 9-4 Flat Shading 57
Gambar 9-5 Vektor normal pada vertex V 58
Gambar 9-6 Interpolasi linier 58
Gambar 9-7 Interpolasi incremental dari nilai intensitas sepanjang tepi Polygon untuk garis pindai (scanline) yang berurutan 59
Gambar 9-8 Gouraud shading 60
Gambar 9-9 Permukaan pada flat shading dan smooth shading 60
Gambar 1-19 Tahap texture mapping 61
Gambar 10-1 Translasi Objek 63
Gambar 10-2 Skala Objek 64
Gambar 10-3 Rotasi dan sumbu rotasi. 65
Gambar 10-4 Rotasi pada sumbu z 65
Gambar 10-5 Rotasi pada sumbu x 66
Gambar 10-6 Rotasi pada sumbu y 66

1 Pendahuluan

1.1 Pengertian Grafika Komputer

Grafika komputer (*Computer Graphic*) dapat diartikan sebagai seperangkat alat yang terdiri dari hardware dan software untuk membuat gambar, grafik atau citra realistik untuk seni, game komputer, foto dan animasi komputer.

1.2 Sejarah Grafika Komputer

Sejarah grafika komputer telah dimulai sejak jaman dahulu kala yaitu ketika bangsa Mesir, Roma dan Yunani berkomunikasi secara grafik. Beberapa lukisan terdapat pada batu nisan orang Mesir dapat dikatakan sebagai lukisan teknik.

Perkembangan grafika komputer secara sederhana dapat dibagi menjadi empat fase, yaitu :

1. **Fase Pertama (1950) era grafika komputer interaktif**
    - Tidak begitu cepat karena teknologi, jumlah dan harga komputer tidak mendukung.
    - MIT berhasil mengembangkan komputer whirlwind dengan tabung sinar katode (Cathode Ray Tube-CRT).
    - Sudah menggunakan pena cahaya (light pen) yaitu sebuah alat input bentuknya seperti pensil yang digunakan untuk memilih posisi, menunjuk sesuatu dan menggambar pada layar dengan pendeteksian cahaya yang datang dari titik-titik pada layar CRT.
    - Telah ada alat pemrograman otomatis (Automatic Programming Tool)
2. **Fase Kedua (1960) Jaman Penelitian/Riset Grafika Komputer Interaktif**
    - Grafika interaktif modern telah ditemukan oleh Ivan Sutherland.

- Mengembangkan teknik interaktif dengan sarana keyboard dan pena cahaya.
- Sejumlah projek penelitian dan produk Computer Aided Design/Manufacturing (CAD/CAM) telah muncul.

3. Fase Ketiga (1970)

Grafika komputer interaktif telah digunakan oleh sektor industri, pemerintah dan ilmuawan untuk memperbaiki kualitas desain produk secara cepat dan mudah.

4. Fase Keempat (1980-1990)

- Penelitian pada dekade ini bertumpu pada penggabungan dan pengotomasasian pelbagai unsur desain dan pemodelan pejal (solid modelling).
- Teknologi hibrid mulai diperkenalkan. Teknologi ini berguna untuk penggabungan objek pejal dengan permukaan.

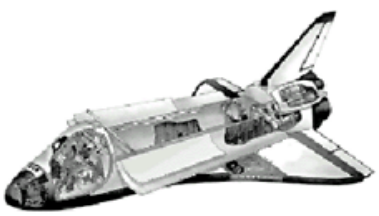
1.3 Peranan dan Penggunaan Grafika Komputer

Grafika komputer telah menunjukkan kemajuan yang pesat dalam pengembangan berbagai aplikasi untuk menghasilkan gambar. Walaupun pada awalnya aplikasi dalam sains dan engineering memerlukan peralatan yang mahal, perkembangan teknologi komputer memberikan kemudahan penggunaan komputer sebagai alat bantu aplikasi grafik komputer interaktif. Pada saat ini grafika komputer digunakan secara rutin dibidang ilmu pengetahuan, teknik, kedokteran, bisnis, industri, pemerintahan, seni, hiburan, pendidikan, periklanan, dan lain sebagainya.

1.4 Desain

Dalam proses desain grafika komputer terutama digunakan pada sistem engineering dan arsitektur. Pada umumnya Computer Aided Design (CAD) digunakan untuk pembuatan desain mobil, bangunan, pesawat terbang, kapal, komputer, tekstil, dan lain-lain.

Pada beberapa aplikasi desain, objek ditampilkan dalam bentuk wireframe, dimana diperlihatkan keseluruhan bentuk, dengan bentuk internal dari objek tersebut. Penggunaan wireframe bermanfaat bagi designer untuk melihat isi dari objek tersebut.



Gambar 1-1 Rancangan Pesawat Terbang



Gambar 1-2 Rancangan Mobil

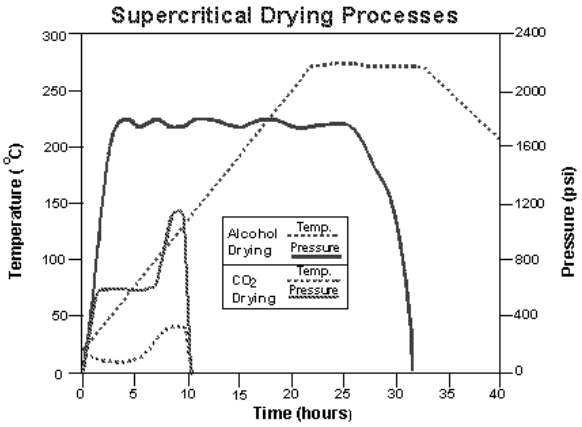
Contoh perangkat lunak yang digunakan yaitu AutoCAD, 3D Studio Max, dan Maya.



Gambar 1-3 Contoh Penggunaan 3D

1.5 Grafik Presentasi

Bidang lain yang berhubungan dengan grafika komputer adalah grafik presentasi yang dapat berupa cetakan, slide, dan transparansi. Grafik presentasi biasanya digunakan untuk melengkapi laporan keuangan, sains, data ekonomi, dan lain-lain. Bentuk grafik presentasi tersebut adalah chart, bar chart, pie chart, dan lain-lain.

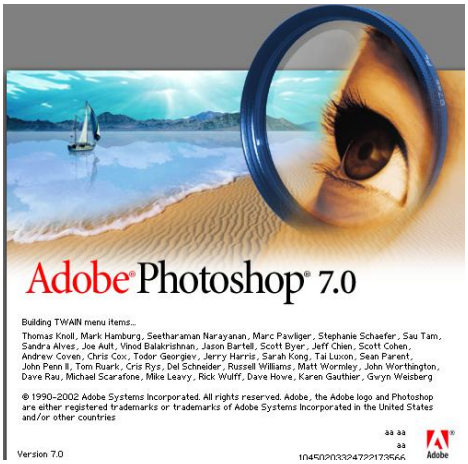


Gambar 1-4 Grafik Presentasi

1.6 Computer Art

Metode grafika komputer digunakan dalam aplikasi commercial art dan fine art. Seniman menggunakan bermacam-macam perangkat lunak grafik, dan kadang dilengkapi dengan perangkat keras khusus.

Contoh perangkat lunak yang digunakan yaitu Corel Draw, Adobe Photoshop, Adobe Ilustrator, Macromedia, dan sebagainya.



Gambar 1-5 Contoh Software Computer Art



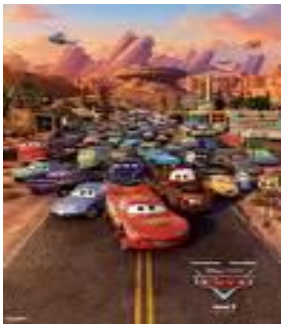
Gambar 1-6 Pemanfaatan Software Computer Art

1.7 Film

pada pembuatan film layar lebar, komputer banyak digunakan untuk menunjang proses pemodelan, visualisasi, dan editing. Misalnya dalam proses special effect, film animasi.



Gambar 1-7 Film Animasi



Gambar 1-8 Film 3 D



Gambar 1-9 Special Effect

1.8 Televisi

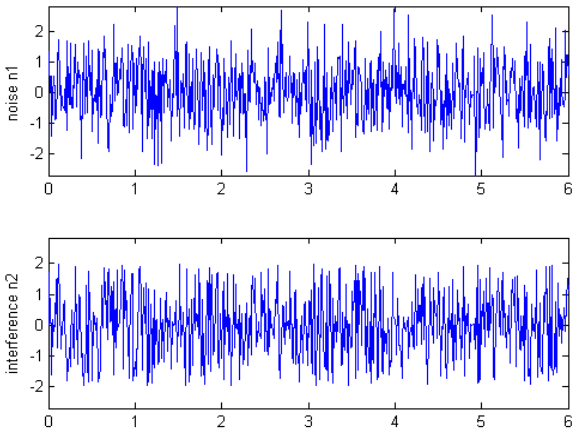
Grafika komputer dalam tayangan televisi juga dapat berupa iklan, tampilan tiap acara, dan lainnya.



Gambar 1-10 Pembuatan Iklan

1.9 Video musik

Produksi video musik tidak terlepas dari grafika komputer, diantaranya pembuatan promosi, cover atau kemasan video, serta animasi yang mengiringi setiap lagu. Proses editing video dan audio dilakukan dengan menggunakan komputer.



Gambar 1-11 Grafik audio

1.10 Game

Berbagai game dapat dijalankan pada komputer PC, video player dengan monitor TV, dan ada yang menggunakan perangkat keras khusus. Alat input interaktif seperti mouse dan joystick diperlukan untuk aplikasi game.



Gambar 1-12 Game 2D



Gambar 1-13 Game 3D

1.11 Pendidikan

Komputer sebagai alat bantu pendidikan (Computer Assisted Instruction) sudah cukup dikenal. Komputer juga digunakan pada aplikasi-aplikasi bukan pengajaran untuk

menunjang sistem pendidikan, seperti mengolah data, mencatat kehadiran, dan sebagainya.

Aplikasi bidang pengajaran dengan komputer sebagai alat bantunya, diantaranya :

Drill and Practice (latih dan praktek) –CAI menggantikan pengajar untuk memberikan latihan kepada siswa.

Tutorial (penjelasan) –sistem komputer digunakan untuk menyampaikan materi ajaran.

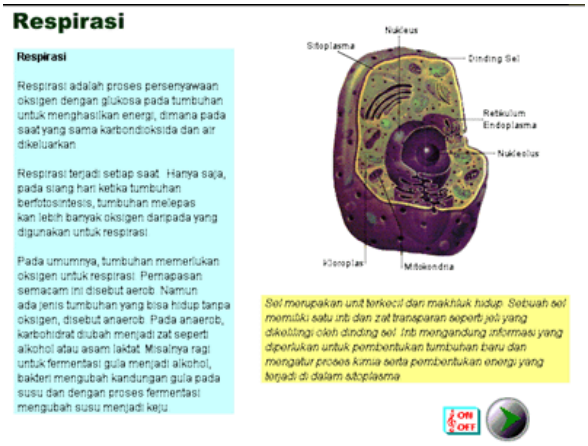
Simulasi – digunakan untuk mengkaji permasalahan yang rumit pada bidang biologi, transportasi, ekonomi, dan lain-lain.

Aplikasi bidang bukan pengajaran dengan alat bantu komputer, diantaranya :

Computer Assisted Testing (Ujian Berbantuan Komputer) – komputer digunakan untuk sarana ujian.

Computer Assisted Guidance (Pengarahan Berbantuan Komputer) – komputer digunakan sebagai sarana untuk mencari informasi yang diperlukan.

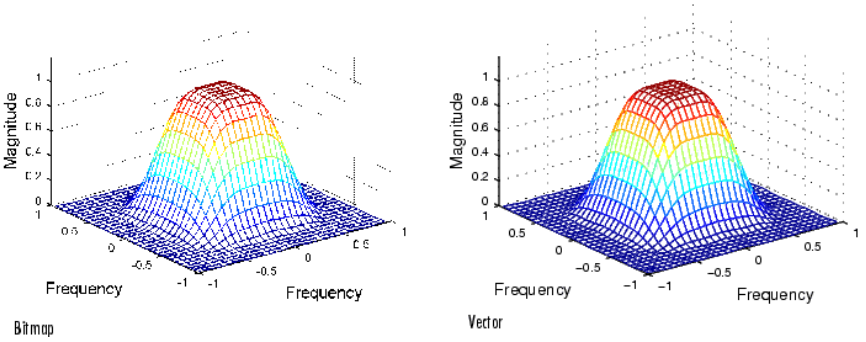
Computer Managed Instruction – komputer digunakan untuk merencanakan pelajaran, evaluasi belajar, serta memantau prestasi siswa.



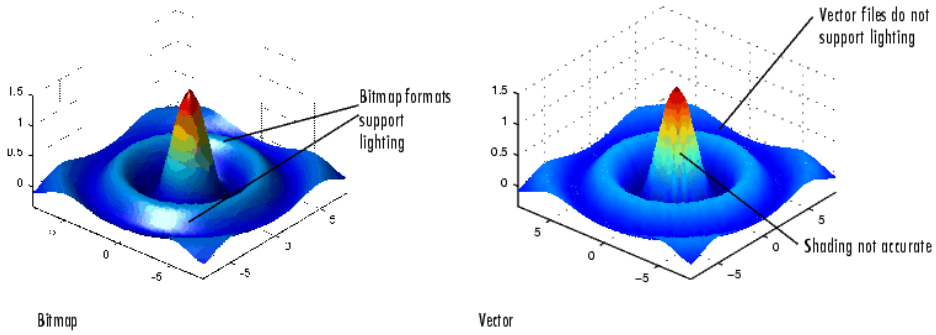
Gambar 1-14 Aplikasi Bidang Pengajaran

1.12 Visualisasi

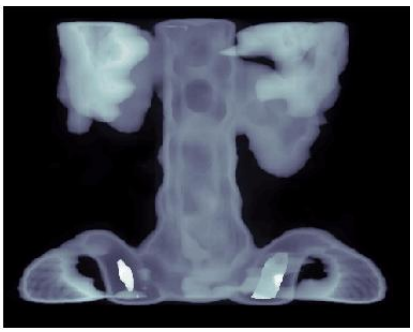
Ilmuwan, ahli kedokteran, analis bisnis, dan lain-lain sering menggunakan banyak informasi suatu masalah dalam mempelajari perilaku proses tertentu. Informasi tersebut berisi ribuan data untuk memberikan gambaran hasil suatu evaluasi. Data tersebut diproses sehingga mendapatkan hasil dalam bentuk visual.



Gambar 1-15 Visualisasi Data Penelitian



Gambar 1-16 Visualisasi Data Penelitian



Gambar 1-17 Bidang Kedokteran

1.13 Image Processing

Image processing (pengolahan citra) merupakan teknik untuk memodifikasi atau menginterpretasi gambar yang ada, seperti foto dan rangkaian gambar film. Dua macam prinsip pengolahan citra adalah :

- Meningkatkan kualitas gambar.
- Memberikan persepsi dari informasi visual, seperti pada robotic.

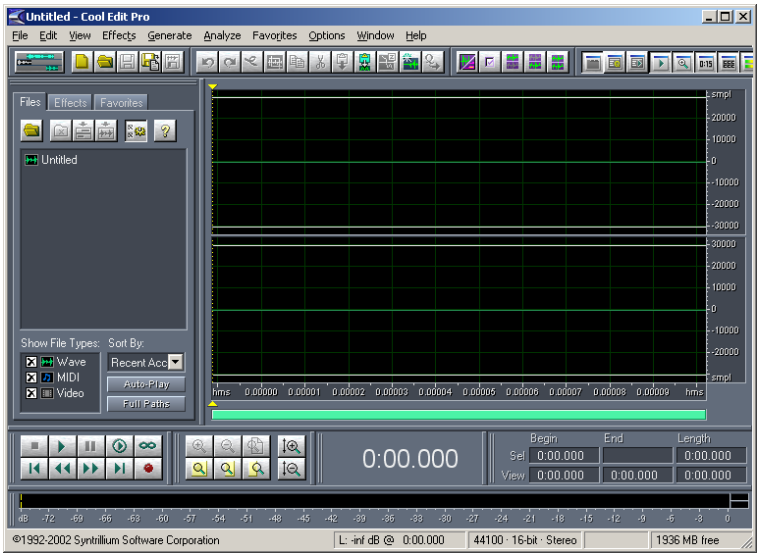


Gambar 1-18 Citra Sebelum dan Sesudah di proses

Untuk melakukan pengolahan citra, pertama-tama membuat digitasi dari foto atau membuat foto menjadi file image. Selanjutnya metode digital dapat digunakan untuk memodifikasi gambar sehingga mendapatkan kualitas yang baik.

1.14 Graphical User Interface (GUI)

Graphical interface (antarmuka grafik) banyak digunakan dalam setiap aplikasi. Komponen utamanya adalah window manager, dimana pengguna dapat mengatur tampilan dari window. Interface juga menampilkan menu dan icon untuk mempercepat pemilihan yang dilakukan oleh pengguna.



Gambar 1-19 Graphical User Interface (GUI)

2

SISTEM GRAFIKA KOMPUTER

Pada saat ini perangkat keras dan perangkat lunak telah tersedia untuk kelengkapan berbagai aplikasi grafika. Kemampuan untuk menyajikan bentuk dua dimensi dan tiga dimensi merupakan hal yang sudah biasa dalam aplikasi grafika pada umumnya. Dengan *Personal Computer* (PC), dapat digunakan berbagai macam alat input interaktif dan aplikasi grafika.

2.1 Teknologi Display

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 :

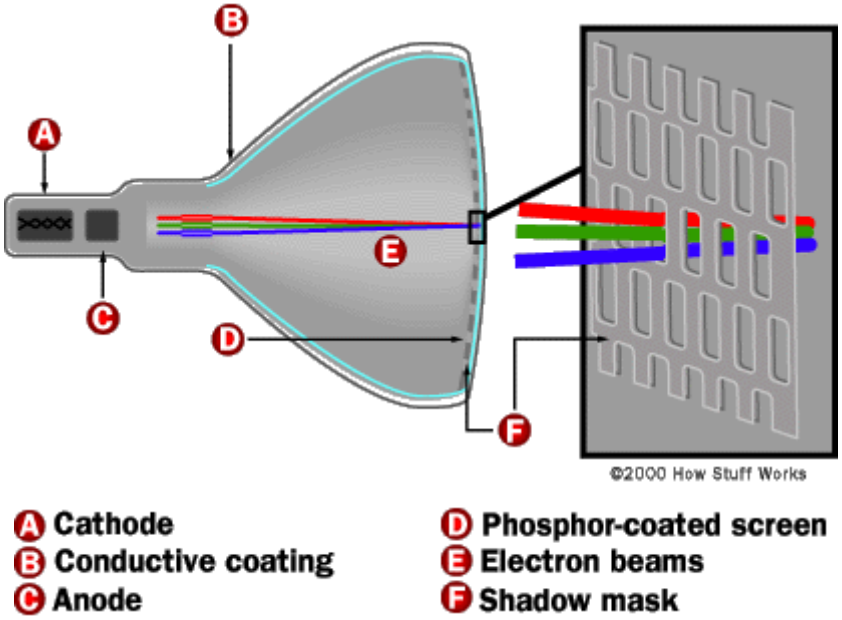
Sebuah electron gun memancarkan elektron, melalui *focusing system* (sistem untuk menentukan fokus), dan *deflection system* (sistem untuk mengatur pembelokan) sehingga pancaran elektron mencapai posisi tertentu dari lapisan fosfor pada layar. Kemudian, fosfor memancarkan sinar kecil pada setiap posisi 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*.



Gambar 2-1 Monitor CRT



Gambar 2-2 Rangkaian Monitor CRT

2.2 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 scane 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*.

2.3 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.

2.4 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.

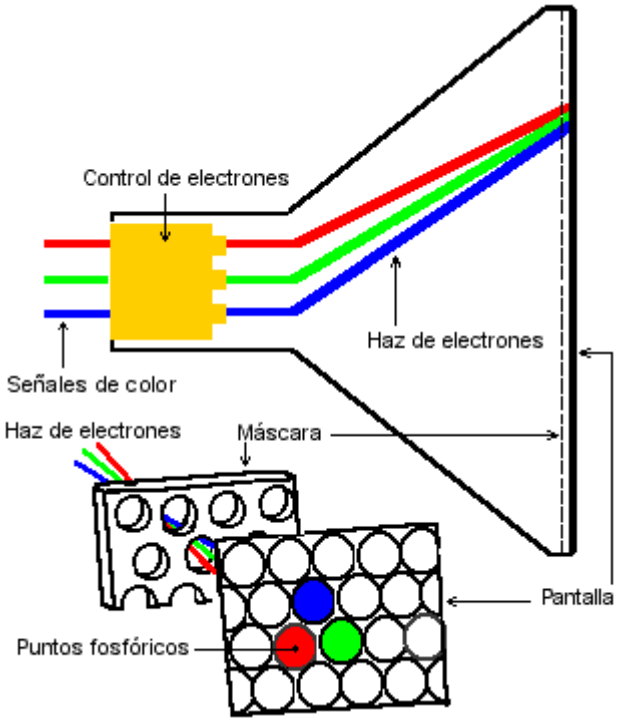
Beam penetration digunakan untuk menampilkan gambar berwarna dengan random-scan monitor. Dua lapisan fosfor, biasanya merah dan hijau, dilapiskan pada bagian dalam dan warna yang dihasilkan tergantung dari seberapa besar pancaran elektron menembus lapisan fosfor. Pancaran yang lemah hanya mencapai bagian 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 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 eletron dapat menga

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 2-3 Monitor Color CRT

2.5 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 Chrystal Display* (LCD).

1. 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 2-4 Monitor Plasma

2. Liquid Chrystal 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-chrystal. Liquid-chrystal 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 2-5 Monitor LCD



Gambar 2-6 Monitor Touch Screen

2.6 Peralatan Input Interaktif

Beberapa macam input interaktif melengkapi sistem grafika, yaitu :

- Keyboard, untuk memasukan karakter atau string
- Locator, untuk mengenali posisi atau orientasi
- Contoh mouse, joystick, glove, light pen.

- Pick, untuk menyeleksi entity suatu tampilan
- Choice, untuk menyeleksi dari suatu action atau pilihan yang tersedia.

  Misal peralatan button pada tablet dan mouse, peralatan choice digunakan untuk memasukan perintah atau pilihan menu pada suatu porgram grafik.



Gambar 2-7 Scanner



Gambar 2-8 Barcode Scanner



Gambar 2-9 Fingerprint Scanner

2.7 Peralatan Hardcopy

Format output dari harcopy dapat bermacam-macam diantaranya kertas, film, transparansi, dan lain-lain. Kualitas gambar yang dihasilkan tergantung dari ukuran dot per size dan dot per inch yang ditampilkan.

Contoh peralatan hardcopy : printer, plotter



Gambar 2-10 Printer



Gambar 2-11 Plotter

2.8 Perangkat Lunak Grafika

Perangkat lunak grafika terdiri dari dua macam, yaitu perangkat lunak untuk pemrograman dan paket aplikasi khusus. Paket pemrograman grafika dilengkapi dengan fungsi grafik yang dapat digunakan pada bahasa pemrograman tingkat tinggi misal C, Pascal, Fortran. Pada paket aplikasi khusus misalnya GL (Graphic Library).



Gambar 2-12 Software Open GL

Tugas

1. Teknologi Display (cara kerja)
   Monitor touch screen
   Monitor wide screen
   Monitor plasma
   Monitor LCD
   Monitor Flat System
2. Scanner
   Paper scanner
   Bio scanner
   Handy scanner
   Barcode scanner
3. Digital Kamera
   Lensa

3
OUTPUT 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.

3.1 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.

3.2 Algoritma pembentukan garis

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 : $\Delta y = m. \Delta x$

3.2.1 Algoritma garis DDA

DDA adalah algoritma pembentukan garis berdasarkan perhitungan Δx dan Δy, menggunakan rumus $\Delta y = m. \Delta x$. 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 :
   - bila nilai $|\Delta y| > |\Delta x|$ maka step = nilai $|\Delta y|$.
   - bila tidak maka step = $|\Delta x|$.
5. Hitung penambahan koordinat pixel yaitu x_increment = Δx / step dan y_increment = Δy / step.
6. Koordinat selanjutnya (x+x_incerement, y+y_increment).
7. Posisi pixel pada layer ditentukan dengan pembulatan nilai koordinasi tersebut.
8. 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.

$\Delta x = x_1 - x_0 = 17\text{-}10 = 7$

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

selanjutnya hitung dan bandingkan nilai absolutnya.

$|\Delta x| = 7$

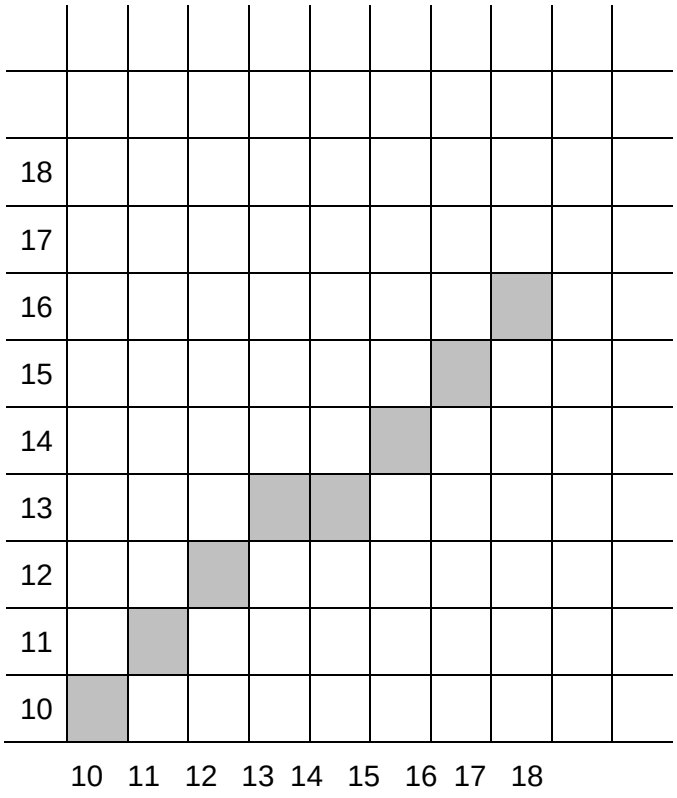
$|\Delta y| = 6$

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)



3.2.2 Algoritma Garis Bressenhem

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. Bressenham 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 Bressenhem disebut juga midpoint line algorithm adalah algoritma konversi penambahan nilai integer yang juga dapat diadaptasi untuk menggambar sebuah lingkaran.

Langkah-langkah untuk membentuk garis menurut algoritma ini adalah :

1. Tentukan dua titik yang akan dihubungkan dalam pembentukan garis.
2. Tetukan 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
   - bila $p_k < 0$ maka titik selanjutnya (x_k+1, y_k) dan $p_{k+1} = p_k + 2\Delta y$
   - bila tidak maka titik selanjutnya adalah (x_k+1, y_k+1)dan

     $p_{k+1} = p_k + 2\Delta y - 2\Delta x$.
6. Ulangi langkah nomor 5 untuk menentukan posisi pixel selanjutnya, sampai $x = x_1$ dan $y = y_k$.

Contoh :

Untuk menggambarkan algoritma Bressenham 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 :

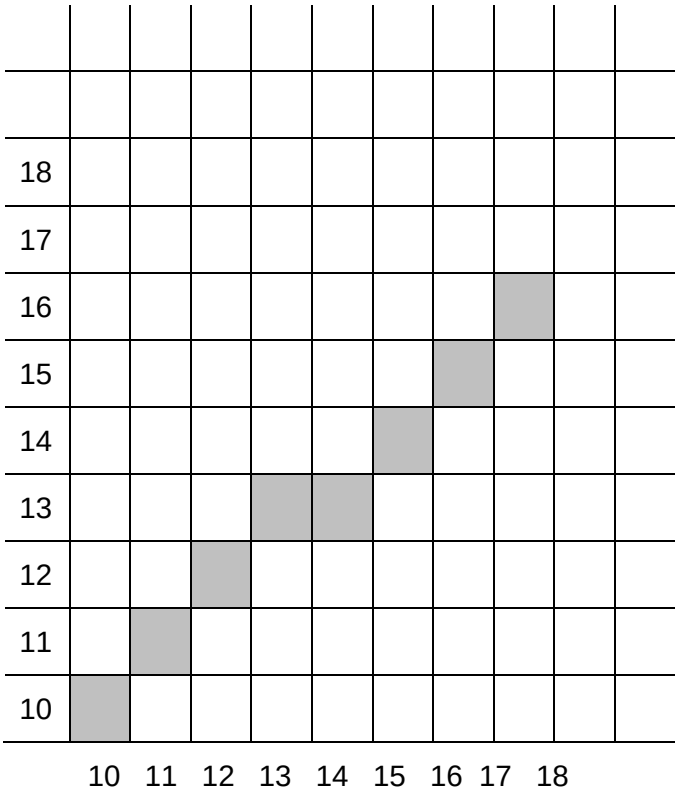
- $\Delta x = x_1 - x_0$ dan $\Delta y = y_1 - y_0$

  $\Delta x = 7$ dan $\Delta y = 6$
- parameter $p_0 = 2\Delta y - \Delta x$

  $p_0 = 5$
- 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)



Latihan

- Buat sebuah garis yang menghubungkan dari titik (20,10) sampai dengan titik (30,18) dengan menggunakan algoritma DDA.
- Buat sebuah garis yang menghubungkan dari titik (20,10) sampai dengan titik (30,18) dengan menggunakan algoritma Bressenham.

3.3 Algoritma Pembentukan Lingkaran

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

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 ± y untuk setiap step.

Simetris delapan titik

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 45^o 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)

Algoritma lingkaran midpoint disebut juga algoritma lingkaran Bressenham. Algoritma yang digunakan membentuk semua titik berdasarkan titik pusat dengan penambahan semau jalur disekeliling lingkaran. Dalam hal ini hanya diperhatikan bagian 45^o dari suatu lingkaran, yaitu oktan kedua dari x = 0 ke x = R/√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

   $$P_0 = 5/4 - r \approx 1-r$$

3. Tentukan nilai awal k = 0, untuk setiap posisi x_k berlaku sbb :
   - Bila p_k <0, maka titik selanjutnya adalah (x_k+1,y_k)

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

   - Bila p_k >0, maka titik selanjutnya adalah (x_k+1,y_k-1)

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

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

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>=y.

Contoh :

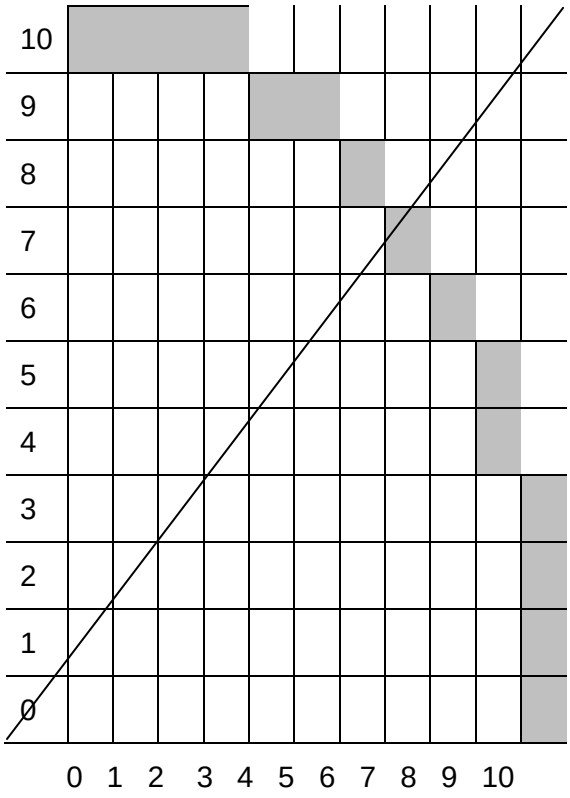
Untuk menggambarkan algoritma Bressenham dalam pembentukan suatu lingkaran dengan titik pusat (0,0) dan radius 10, perhitungan berdasarkan pada oktan dari kuadran pertama dimana x =0 sampai x =y.

Penyelesaian :

- (x_0,y_0) =(0,0) r = 10
- (x_0,y_0) = (0,10) $2x_0 = 0$, $2y_0 = 20$
- parameter $p_0 = 1-r$

  $p_0 = -9$

k	**p_k**	**(x_{k+1},y_{k+1})**	**$2\ x_{k+1}$**	**$2\ y_{k+1}$**
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



Latihan

- Buat sebuah lingkaran dengan r = 9 terhadap titik pusat (0,0)

4

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 vetical 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.

4.1 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.

4.1.1 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.

4.1.2 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 pixel pada tiap posisi, seperti algoritma Bressenham. Garis dengan ketebalan didapatkan dengan perkalian integer positif dari garis standar, dan menempatkan tambahan pixel pada posisi sejajar. Untuk garis dengan slope kurang dari 1, routine 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.

4.1.3 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.

4.1.4 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. Routine polyline membuat garis pada warna tertentu dengan mengatur nilai warna pada frame buffer untuk setiap posisi pixel, menggunakan prosedur set pixel. Jumlah warna tergantung pada jumlah bit yang akan digunakan untuk menyimpan informasi warna.

4.2 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 raster sistem.

- 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.

4.2.1 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.

4.2.2 Algoritma Flood-Fill

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

4.3 Pembentukan Karakter

Huruf, angka dan karakter lain dapat ditampilkan dalam berbagai ukuran (size) dan style. Jenis huruf dibagi menjadi 4 macam, yaitu serif, sanserif, agyptian 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.

- Agyptian

  Huruf dalam kategori agyptian mempunyai kait dengan bentuk segi empat yang mempunyai karakter kokoh.

- Dekoratif

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

Dua macam metode dapat digunakan untuk menyimpan jenis huruf dalam komputer. 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 pixel pada monitor.

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

4.4 Antialiasing

Seperti yang telah dikatakan sebelumnya bahwa konversi raster-scan adalah pengisian harga-harga elemen suatu "matriks" (yaitu frame buffer) sedemikian rupa sehingga secara visual "tergambarkan" primitif- rimitif grafik yang bersangkutan. Jadi pada dasarnya adalah semacam diskretisasi obyek tsb. 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 (effek 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 pixel-pixel "batas" obyek dengan latar atau obyek lainnya. Modifikasi tsb. akan memper-"halus" batas-batas tsb. sehingga mengurangi penampakan yang "jaggies" tsb. Ada tiga pendekatan:

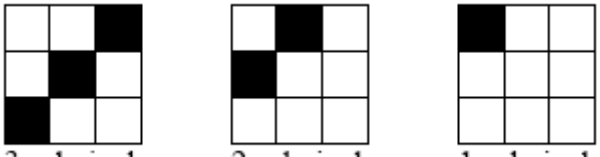
- Supersampling (postfiltering)
- Area sampling
- pixel phasing

4.4.1 Supersampling atau Postfiltering

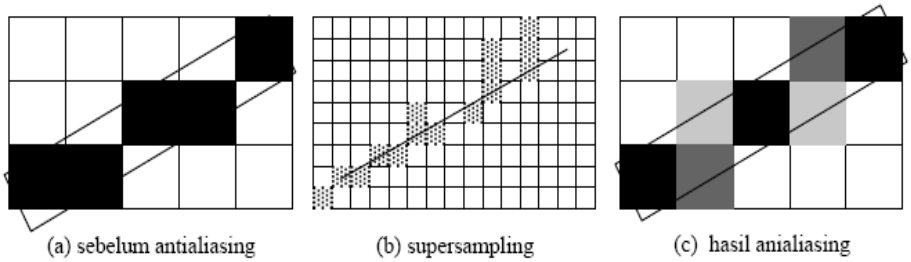
Secara lojik metoda ini "memperhalus" ukuran pixel ke dalam subpixel-subpixel dan "menggambarkan" garis pada grid subpixel tsb. lalu harga intensitas suatu pixel ditentukan sesuai dengan berapa banyak subpixelnya dikenai "garis" tersebut. Relasi: intensitas pixel ~ jumlah subpixel pada garis.

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 subpixel, 2 subpixel, 1 subpixel, dan tidak ada. Pemberian intensitas sesuai dengan keempat tingkat tersebut.

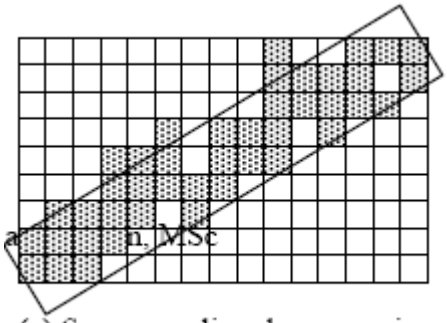


Contoh :

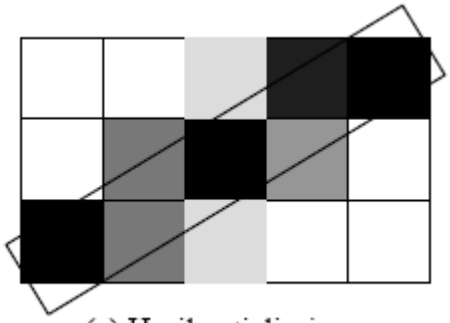


- Menganggap garis adalah garis dengan tebal tetap yaitu 1 pixel (yaitu suatu segiempat dengan lebar 1 pixel) dan intensitas dihitung sesuai dengan jumlah subpixel yang "tertutupi" oleh segi empat ini (Perlu diambil acuan bahwa suatu subpixel "tertutupi", misalnya jika sudut kiri bawah subpixel ada di dalam segi empat).Yang paling sederhana adalah menggunakan harga rasio jumlah subpixel

terhadap total subpixel pada pixel sebagai fungsi intensitas. Untuk subsampling 3x3 total subpixel adalah 9 sehingga ada 10 tingkat intensitas yang bisa diberikan. Khusus titik ujung yang berharga bilangan bulat (karena bisa untuk koordinat bilangan real) Akan diberi harga penuh



(a) Supersampling dengan rasio



(a) Hasil antialiasing

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

¨ Pixel-weighting Masks

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

Contohnya:

- box mask (berefek averaging)

- gaussian mask

Kadang-kadang mask meliputi juga subpixel di pixel tetangganya untuk mendapatkan hasil yang lebih smooth.

¨ Continuous Filtering

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

4.4.2 Area Sampling

Pada Unweighted Area Sampling suatu garis diangap sebagai segiempat dengan lebar 1 pixel seperti halnya pada supersampling cara kedua di atas. Yang dihitung adalah luas bagian pixel 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 pixel yang tertutup area (garis atau fill-area) dan dari rasio luas tsb. terhadap luas pixel dapat ditentukan bobot foreground terhadap background untuk mendapatkan intensitas pixel. Cara penghitungannya?

Pitteway & Watkinson: untuk fill-area dengan memodifikasi midpoint algorithm untuk garis sehingga fungsi diskriminan p menentukan juga persentasi tsb. Dalam algoritma ini pada persamaan garis

$y = m\,x + b,\ m \pounds 1$

digunakan fungsi keputusan:

$p = m\,(x\mathrm{i} + 1) + b - (y\mathrm{i} + ½)$

Sementara bagian pixel yang tertutup area di bawah garis tersebut adalah suatu trapesium dengan ketinggian kiri

$y = m\,(x\mathrm{i} - ½) + b - (y\mathrm{i} – ½)$ dan ketinggian kanan $y = m\,(x\mathrm{i} + ½) + b - (y\mathrm{i} – ½)$ serta lebar 1 (satuan pixel). Luas

trapesium ini adalah $= m\,x\mathrm{i} + b - (y\mathrm{i} - 0.5) = p - (1 - m)$

4.4.3 Pixel Phasing

Pergeseran mikro (microposition) yang dilakukan oleh deflektor elektron sebesar 1/4, 1/2 atau 3/4 diameter

pixel.

- Kompensasi Perbedaan Intensitas Garis

Secara normal garis diagonal (tanpa antialiasing) lebih tipis dari garis horisontal/vertikal karena pada garis tsb. pixel-pixel lebih spanned dari pada pixel-pixel pada garis hosrisontal/diagonal. Jadi secara visual efek ini dapat juga dikurangi dengan menaikkan intensitas garis yang mengarah diagonal sesuai dengan sudut dan mencapai maksimum pada 450 dengan faktor Ö2 dari garis horisontal/vertikal.

5 Transformasi 2 Dimensi

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.

5.1 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 (tx,ty), dimana tx adalah translasi menurut sumbu x dan ty adalah translasi menurut sumbu y. Koorinat baru titik yang ditranslasi dapat diperoleh dengan menggunakan rumus :

$x' = x + t_x$ (x,y) = titik asal sebelum translasi

$y' = y + t_y$ (x',y') = 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 tx,ty(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)

5.2 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 (scalling factor) yaitu (sx,sy) yang diberikan. sx adalah faktor penskalaan menurut sumbu x dan sy faktor penskalaan menurut sumbu y. Koordinat baru diperoleh dengan

x’ = x + sx (x,y) = titik asal sebelum diskala

y‘= y + sy (x’,y’) = 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 scalling.

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 ?

A : x’=10*3=30

y’=10*2=20

A’=(30,20)

B : x’=30*3=90

y’=10*2=20

B’=(90,20)

C : x’=10*3=30

y’=30*2=60

C’=(30,60)

5.3 Perputaran (Rotasi)

Putaran adalah suatu operasi yang menyebabkan objek bergerak berputar pada titik pusat atau pada sumbu putar yang dipilih berdasarkan sudut putaran tertentu. Untu melakukan rotasi diperlukan sudut rotasi θ 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 :

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

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

sedangkan di ketahui

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

lakukan subtitusi, 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 (x_p,y_p) :

$x'= x_p+(x - x_p) \cos \theta - (y - y_p) \sin \theta$

$y'= y_p+(x - x_p) \sin \theta + (y - y_p) \cos \theta$

Contoh

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

Titik A

$x'= x_p+(x - x_p) \cos \theta - (y - y_p) \sin \theta$

$=10+(10-10)*0.9 – (10-10)*0.5 = 10$

$y'= y_p+(x - x_p) \sin \theta + (y - y_p) \cos \theta$

$= 10+(10-10)*0.5 – (10-10)*0.9 = 10$

Titik A'(10,10)

Titik B

$x'= x_p+(x - x_p) \cos \theta - (y - y_p) \sin \theta$

$=10+(30-10)*0.9 – (10-10)*0.5 = 28$

$y'= y_p+(x - x_p) \sin \theta + (y - y_p) \cos \theta$

$= 10+(30-10)*0.5 – (10-10)*0.9 = 20$

Titik B'(28,20)

Titik C

$x' = x_p + (x - x_p) \cos \theta - (y - y_p) \sin \theta$

$= 10 + (10-10)*0.9 - (30-10)*0.5 = 0$

$y' = y_p + (x - x_p) \sin \theta + (y - y_p) \cos \theta$

$= 10 + (10-10)*0.5 - (30-10)*0.9 = 28$

Titik A'(0,28)

5.4 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^o terhadap refleksi. Sumbu refleksi dapat dipilih pada bidang x,y. Refleksi terhadap garis y=0, yaitu sumbu x dinyatakan dengan matriks

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

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

- Objek diangkat
- Putar 180^o 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 kemudia direfleksi terhadap sumbu y. Transformasi ini dinyatakan dengan :

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

Refleksi ini sama dengan rotasi 180^o pada bidang xy dengan koordinat menggunakan titik pusat koordinat sebagai pivot point.

Refleksi suatu objek terhadap garis y=x dinyatakan dengan bentuk matriks

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

Matriks dapat diturunkan dengan menggabungkan suatu sekuen rotasi dari sumbu koordinat merefleksi matriks. Pertama-tama dilakukan rotasi searah jarum jam dengan sudut 45^o yang memutar garis y=x terhadap sumbu x. Kemudian objek direfleksi 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^o.

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

- Rotasi 45^o searah jarum jam
- Refleksi terhadap axis y
- Rotasi 90^o 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

5.5 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

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

Dengan koordinat transformasi

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

parameter sh_x dinyatakan dengan sembarang bilangan. Posisi kemudian digeser menurut arah horizontal.

- Shear terhadap sumbu y

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

Dengan koordinat transformasi

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

parameter sh_y dinyatakan dengan sembaran bilangan. Posisi koordinat kemudian menurut arah vertikal.

Latihan soal

1. 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)
2. dari soal (1) lakukan penskalaan dengan faktor skala (3,2)
3. dari soal (2) lakukan rotasi terhadap titik pusat (A) dengan sudut rotasi 30^o.
4. Transformasi shear dengan nilai $sh_x = 2$ dengan koordinat A(0,0), B(1,0), C(1,1), dan D(0,1)
5. Transformasi shear dengan nilai $sh_y = 2$ dengan koordinat A(0,0), B(1,0), C(1,1), dan D(0,1)

6
Clipping 2 Dimensi

Dalam peragaan obyek (atau obyek-obyek) pada windownya maka tidak semua bagian dari obyek tersebut perlu diperagakan akibat keterbatasan ukuran viewport itu sendiri. Jadi akan ada sejumlah primitif grafika yang diperagakan karena sepenuhnya ada dalam window, ada sejumlah lainnya yang tidak perlu diperagakan karena sepenuhnya di luaw window, dan sisanya adalah primitif-primitif yang terpotong oleh window sehingga sebagian berada di dalam window dan sebagian lain di luar.

Kita perlu menangani hal yang terakhir tersebut secara khusus karena dalam sejumlah lingkungan grafika hal ini bisa menghasilkan kekacauan peragaan, misalnya: bagian yang seharusnya tdak tampak, muncul di bagian ujung lain pada screen (*wrap-around*), atau menyebabkan program error karena akses keluar batas memory, atau minimal adalah ketidak-efisienan komputasi akibat komputasi pada data yang ternyata tidak perlu dimunculkan.

Selama ini untuk menangani masalah tersebut dapat dilakukan sejumlah metoda sbb.

- Metoda penggunaan kanvas bitmap yang diperluas: teknik ini sederhana karena melakukan penggambaran pada suatu bitmap yang amat besar mencakup semua penggambaran primitif, kemudian mengambil bagian yang sesuai (cropping) dengan bagian window dengan operasi transfer blok memori. Masalah teknik ini jelas perlunya memory space yang amat besar.
- Melakukan “scissoring” yaitu memodifikasi algoritma penggambaran piksel dengan menambahkan pemeriksaan batas-batas window; piksel baru digambari jika berada dalam batas window. Masalah teknik ini adalah kliping hanya dapat dilakukan pada level operasi piksel demi piksel dan komputasi keseluruhan primitif grafika tetap dilakukan walaupun ternyata hanya sebagaian kecil saja yang perlu ditampilkan. Masalah selanjutnya adalah konsep kliping hanya berlaku di level bawah (peragaan) dan tidak bisa ditarik ke level konseptual (kliping seara umum).
- Melakukan usaha analitis untuk menemukan titik-titik perpotongannya lalu mendapatkan potongan-potongan garis untuk diperagakan. Kliping dapat digunakan di level konseptual karena garis dan window dinyatakan dalam besaran-

besaran real. Masalahnya, tidak semua primitif grafika dapat dengan mudah dianalisis secara geometris demikian.

6.1 Clipping Garis

Kita mengharapkan dari suatu garis akan diketahui apakah suatu garis itu sepenuhnya berada dalam window, sepenuhnya diluat window, diperolehnya suatu garis (atau garis-garis) baru hasil kliping dalam batas-batas window. Garis itu sendiri (baik yang sebelum maupun setelah kliping) dinyatakan dalam koordinat titik-titik ujungnya. Secara umum bentuk window adalah suatu poligon. Namun dalam kebanyakan metoda window adalah persegi panjang dengan batas-batasnya sejajar dengan sumbusumbu sistem koordinat. Hal ini dibedakan dari window dengan bentuk poligon yang umum karena tingkat kerumitan algoritmisnya berbeda jauh. Lebih lanjut lagi, window dengan poligon konveks jauh lebih sederhana dari window poligon konkaf karena jumlah titik perpotongan suatu garis dengan suatu poligon konveks maksimum hanya dua, sementara dengan poligon konkaf bisa lebih dari dua.

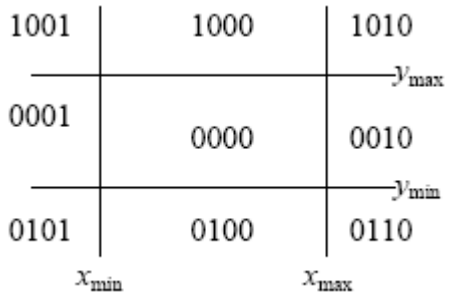
Berikut ini akan dibahas algoritma-algoritma dalam bentuknya yang baku. Terdapat banyak varian dari algoritma-algoritma tersebut yang dibuat orang demi mendapatkan peningkatan efisiensinya.

6.1.1 Algoritma Cohen-Sutherland (CS)

Algoritma ini terbatas pada window yang berbentuk segi empat dengan sisi-sisinya sejajar sumbusumbu koordinat. Ide dasarnya adalah sebagai berikut. Jika window dinyatakan dengan titik-titik ujung kiri bawah (xmin, ymin) dan kanan atas (xmax, ymax) maka ruang dua dimensi penggambaran dibagi ke dalam sembilan ruangan oleh garis-garis perpanjangan tepi window. Jadi ruang yang ditengah adalah window kliping itu sendiri. Titik-titik (x, y) yang berada pada masing-masign ruangan tersebut dapat diberi kode empat bit *b1b2b3b4* dengan aturan pemberian kode-kode tersebut adalah sbb.

jika y > ymax maka b1 = 1, dan jika y ≤ymax maka b1 = 0
jika y < ymin maka b2 = 1, dan jika y ≥ymin maka b2 = 0
jika x > xmax maka b3 = 1, dan jika x □xmax maka b3 = 0
jika x < xmin maka b4 = 1, dan jika x □xmin maka b4 = 0

Sehingga dapat digambarkan pembagian ruangan dan pengkodeannya adalah sebagai berikut.



Gambar 6-1 Kode Cohen Sutherland

Apakah suatu garis diluar, atau di dalam window, atau memotongnya, dapat diketahui berdasarkan operasi lojik pada kode-kode dari kedua titik ujung garis tersebut. Misalkan garis dinyatakan dengan titik-titik ujung *P*0 dan *P*1 dengan pengkodean *C*0 dan *C*1. Maka dapat diketahui sbb.

- Jika (*C*0 or *C*1) != 0000 maka garis berada di luar window
- Jika (*C*0 and *C*1) == 0000 maka garis berada di dalam window
- Yang lainnya berarti memotong garis batas window atau hanya perpanjangannya (dalam hal ini mungkin saja tidak melintasi ruang window).

Untuk kasus ketiga tersebut perlu dilakukan pemeriksaan lebih lanjut dengan memotong secara bertahap terhadap garis batas yang dilintasinya.

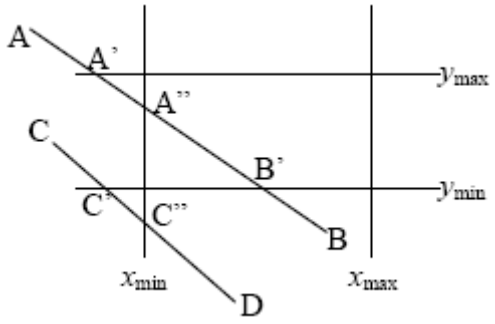
Jika *C*1 == 0000 maka periksa *P*0, jika tidak maka *P*1 yang diperiksa, (misalkan yang diperiksa *P*0, jika *P*1 menjadi kebalikannya) sbb.

- Jika (*C*0 and 1000) != 0000 maka cari perpotongan dengan garis *y*=*y*max
- Jika tidak maka jika (*C*0 and 0100) != 0000 maka cari perpotongan dengan garis *y*=*y*min
- Jika tidak maka jika (*C*0 and 0010) != 0000 maka cari perpotongan dengan garis *x*=*x*max
- Jika tidak maka pasti (*C*0 and 0001) != 0000 dan cari perpotongan dengan garis *x*=*x*min

Jika *P*0 adalah titik perpotongannya maka selanjutnya ulangi algoritma ini untuk ruas garis *P*0'*P*j

Sampai akhirnya di peroleh potongan garis dengan titik-titik ujung *P*0* dan *P*1* yang bisa dipastikan keberadaannya di dalam window atau di luar window. Urutan pemeriksaan bisa diubah dan menghasilkan tahapan pemotongan yang berbeda tetapi hasilnya tetap sama.

Contoh pada gambar berikut garis dari A ke B akan mengalami pemotongan menjadi A'B, kemudian menjadiA"B dan kemudian menjadi A"B' yang berada dalam window. Sementara garis dari C ke D akan mengalami pemotongan menjadi C'D kemudian menjadi C"D yang berada di luar window.



Penghitungan untuk mencari perpotongan dapat disederhanakan berdasarkan persamaan garis

$y = y1 + m\,x$

$x = x1 + 1/m\,y$

dengan m = ($y2$ - $y1$)/($x2$ - $x1$).

Titik perpotongan garis tsb dengan $y = yt$ adalah (x, yt) dengan $x = x1 + yt\,/m$. Dan, titik perpotongan dengan $x = xt$ adalah (xt, y) dengan $y = y1 + m\,xt$.

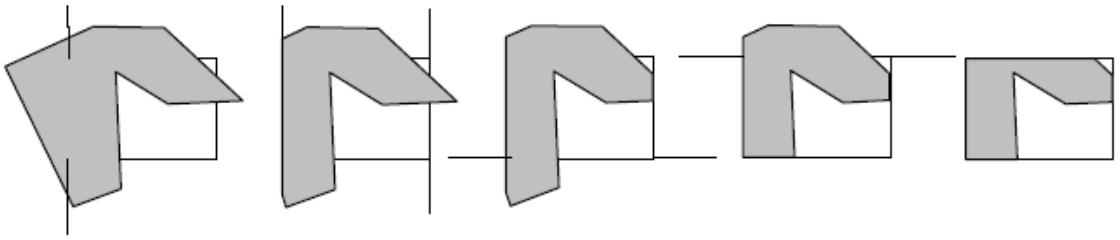
Karena adanya pemotongan berulang maka jika koordinat direpresentasikan dengan bilangan integer maka setiap pemotongan menyebabkan pembulatan harga dan selanjutnya bentuk geometrisnya berubah. Untuk menghindari hal ini maka koordinat direpresentasikan dalam bilangan real hingga saat penggambaran potongan garis tsb.

6.2 Kliping Poligon

Suatu poligon dinyatakan dengan deretan koordinat titik-titik verteksnya. Poligon bisa konveks atau konkaf. Diharapkan dari kliping poligon terhadap suatu window maka akan diperoleh poligon (atau poligon-poligon) baru irisan dari poligon asal dengan window. Window sendiri seperti halnya pada masalah kliping garis yang paling sederhana bisa berbentuk segi empat, atau poligon konveks atau poligon konkaf yaitu yang paling sulit.

6.2.1 Algoritma Sutherland-Hodgeman (SH)

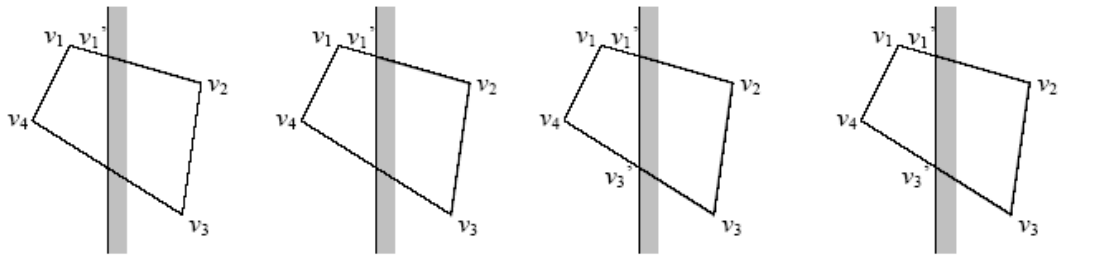
Algoritma ini adalah untuk kliping poligon konkaf/konveks terhadap suatu poligon konveks. Idenya adalah melakukan pemotongan terhadap batas demi batas window secara terpisah. Pemotongan terhadap suatu batas (dan perpanjangan batas itu) menghasilkan suatu poligon lain yang akan dipotongkan terhadap batas selanjutnya (dan perpanjangannya). Perhatikan contoh pada gambar berikut ini di mana suatu poligon dipotong terhadap suatu window berbentuk persegi panjang. Pemotongan dilakukan pertama terhadap sisi kiri, kemudian terhadap sisi kanan, bawah, dan terakhir atas.



Pertanyaan selanjutnya adalah bagaimana caranya pemotongan terhadap suatu garis batas? Algoritma ini memiliki aturan-aturan sebagai berikut jika poligon dinyatakan oleh verteks-verteks *v*1, *v*2, …, *v*n.

- Sisi demi sisi diperiksa terhadap batas window mulai dari sisi *v*1*v*2, *v*1*v*3, …, *v*n-1*v*n, dan *v*n*v*1, untuk mendapatkan verteks-verteks membentuk poligon baru hasil pemotongan tersebut. Pada tahap inisialisasi poligon hasil berisikan 0 verteks.
- Bila suatu sisi *vivi*+1 **berpotongan** dengan batas window dengan *vi* berada di **luar** mengarah dan *vi*+1 berada di **dalam** batas window maka dilakukan komputasi untuk mendapatkan titik perpotongannya yaitu *vi*', dan verteks-verteks *vi*' dan *vi*+1 dicatat sebagai verteks berikutnya di poligon hasil pemotongan.
- Bila suatu sisi *vivi*+1 **berpotongan** dengan batas window dengan *vi* berada di **dalam** mengarah dan *vi*+1 berada di **luar** batas window maka dilakukan komputasi untuk mendapatkan titik perpotongannya yaitu *vi*', dan verteks *vi*' dicatat sebagai verteks berikutnya di poligon hasil pemotongan.
- Bila suatu sisi *vivi*+1 **tidak berpotongan** dengan batas window dan berada di **sebelah dalam** batas window maka verteks *vi*+1 dicatat sebagai verteks berikutnya di poligon hasil pemotongan.
- Bila suatu sisi *vivi*+1 **tidak berpotongan** dengan batas window dan berada di sebelah luar batas window maka tidak ada yang dicatat.

Contoh berikut adalah pemotongan poligon terhadap sisi kiri window persegi empat.

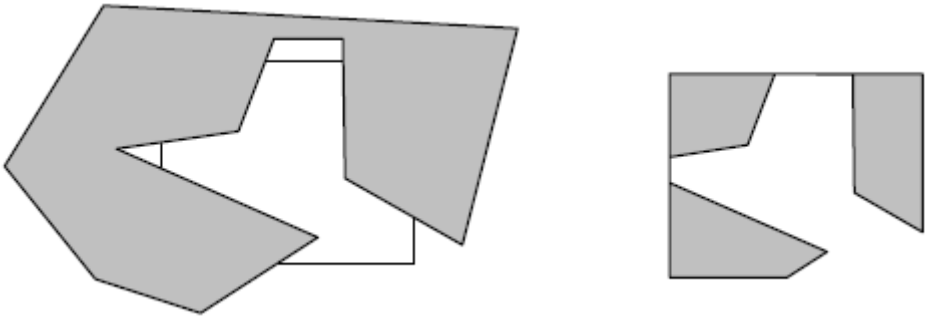


Poligon yang dihasilkan adalah dengan verteks-verteks $v_1'v_2v_3v_3'$.

Masalah yang muncul pada algoritma ini adalah apabila terjadi lebih dari satu kali keluar-masuk window maka akan terbentuk suatu poligon yang sebenarnya adalah

beberapa area terpisah tapi dihubungkan oleh garis-garis. Ini mungkin terjadi pada poligon konkaf dan tidak terjadi pada poligon konveks.

Perhatikan gambar berikut yang menggambarkan sebelum dan setelah kliping suatu poligon.



Jika diharapkan bahwa untuk kasus ini akan terbentuk bukan hanya satu poligon tetapi sejumlah piligon untuk setiap area maka perlu modifikasi pada algoritma dengan menambahkan pemeriksaan akhir ada tidaknya sisi-sisi poligon yang berimpit dan jika ada melakukan pemotongan pada tempat tersebut.

7

3 DIMENSI

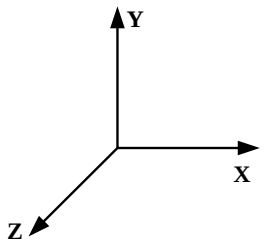
7.1 Konsep Dasar 3 Dimensi

Yang membedakan 2 dimensi dan 3 dimensi adalah kedalaman. Kedalaman didefinisikan sebagai jarak antara *viewer* terhadap benda yang dia lihat. Ini berarti berbeda dengan 2 dimensi yang hanya menggunakan 2 ukuran, yaitu panjang dan lebar, maka 3 dimensi menggunakan 3 ukuran, yaitu panjang, lebar dan kedalaman. Secara geometri ketiga ukuran tersebut disimbolkan dengan sumbu x, y, dan z.

7.2 Sistem Koordinat

Salah satu sistem koordinat yang dikenal adalah sistem koordinat kartesian yang digunakan untuk membedakan lokasi atau posisi sembarang titik atau objek dengan titik atau objek yang lain. Sistem koordinat kartesian terdiri atas sistem koordinat kartesian 2 dimensi dan sistem koordinat kartesian 3 dimensi.

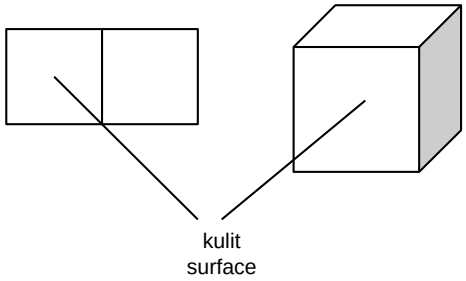
Dalam sistem koordinat 3 dimensi terdapat satu sumbu lain selain sumbu x dan sumbu y, yaitu sumbu z yang arahnya tegak lurus terhadap sumbu x dan sumbu y. Hadirnya sumbu z menyebabkan sistem koordinat ini menjadi lebih hidup karena efek jauh dekat menjadi terlihat. Untuk lebih jelasnya, lihat pada gambar di bawah ini :



Gambar 7-1 Sistem koordinat kartesian 3 dimensi

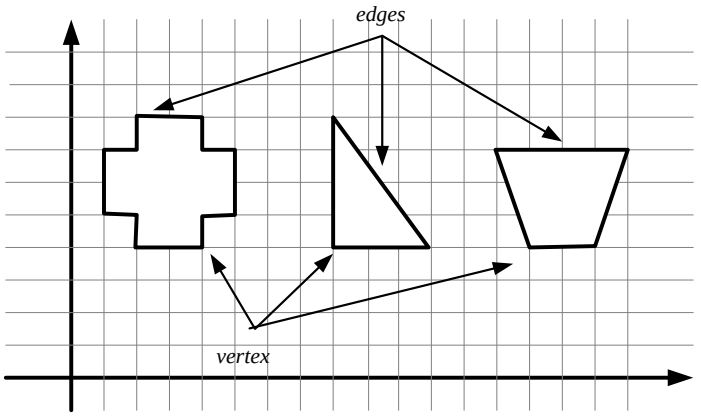
7.3 Benda Tiga Dimensi

Benda tiga dimensi disusun dari sekumpulan *surface*. Gambar di bawah memberikan contoh bagaimana *surface* digunakan untuk menyusun benda tiga dimensi.



Gambar 7-2 Permukaan tiga dimensi

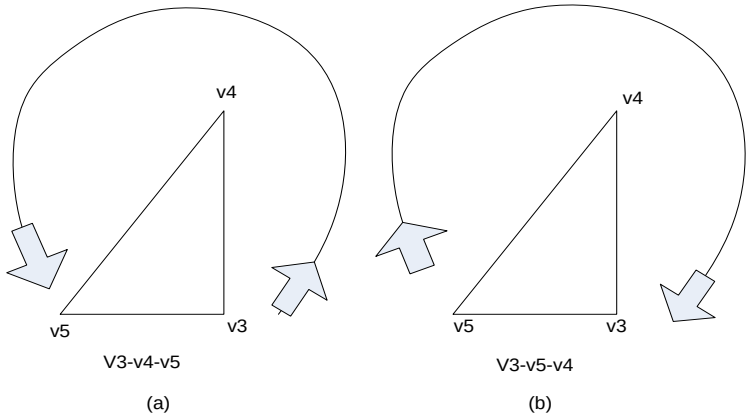
Surface dapat dibuat dari rangkaian Polygon. Polygon adalah bentuk yang disusun dari serangkaian garis yang terhubung satu dengan yang lain dan berbentuk kurva tertutup sehingga membentuk sebuah objek gambar. Titik sudut dari Polygon disebut *vertex* sedangkan garis penyusun Polygon disebut *edge*.



Gambar 7-3 Polygon

Polygon digambar dengan menggambar masing-masing *edge* dengan setiap *edge* merupakan pasangan dari $vertex_i$ – $vertex_{i+1}$ kecuali untuk *edge* terakhir merupakan pasangan dari $vertex_n$ – $vertex_1$.

Bentuk polygon yang paling tepat digunakan untuk membuat permukaan benda tiga dimensii adalah polygon segitiga (*triangle*). Hal ini disebabkan polygon segitiga akan selalu berada dalam keadaan *planar* (datar) sementara polygon dengan *vertex* lebih dari 3 dapat berada dalam kondisi non-planar karena salah satu *vertex* tidak berada di lokasi yang sejajar. Satu hal yang harus diperhatikan pada saat menggunakan polygon segitiga adalah winding. Winding adalah urutan dan arah *vertex-vertex* penyusun polygon. Winding menganggap bahwa arah *vertex* berlawanan jarum jam sebagai tampak depan sedangkan arah *vertex* searah jarum jam dianggap sebagai bagian belakang benda.

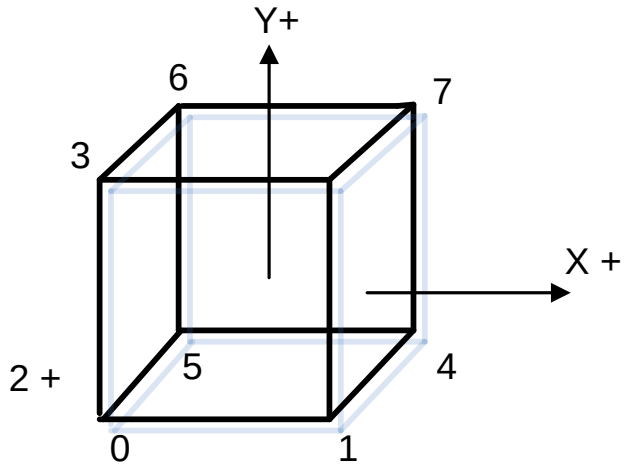


Gambar 7-4 Arah* vertex *: (a) berlawanan jarum jam; (b) searah jarum jam

Pada gambar 1-4.a polygon menggunakan winding berlawanan jarum jam karena didefinisikan sebagai urutan *vertex* v3-v4-v5 sedangkan pada gambar 1-4.b menggunakan winding searah jarum jam karena urutan *vertex* yang digunakan v3-v5-v4.

7.4 Struktur Data Benda 3 Dimensi

Sebuah benda tiga dimensi dapat disusun dengan menghubungkan sejumlah *vertex*. Kumpulan *vertex* yang saling terhubung tersebut disebut sebagai *wireframe* atau kerangka, sedangkan benda yang disusun dari kumpulan titik dan permukaan tersebut disebut sebagai *mesh*. Perhatikan Gambar 1.5, kubus yang ada di gambar tersebut dapat disusun berdasarkan data yang ada pada tabel 1.1 dan tabel 1.2



Gambar 7-5 Kubus

Tabel 7-1* vertex *penyusun kubus

Vertex	X	Y	Z
0	-1	-1	+1
1	+1	-1	+1
2	+1	+!	+1
3	-1	+1	+1
4	+1	-1	-1
5	-1	-1	-1
6	-1	+1	-1
7	+1	+1	-1

Tabel 7-2 Permukaan (face) kubus

Face	V1	V2	V3	keterangan
F1.a	0	1	2	Depan
F1.b	0	2	3	Depan
F2.a	1	4	7	Kanan
F2.b	1	7	2	Kanan
F3.a	4	5	6	Belakang
F3.b	4	6	7	Belakang
F4.a	5	3	0	Kiri
F4.b	5	6	3	Kiri
F6.a	3	2	7	Atas
F6.b	3	7	6	Atas
F6.a	0	1	4	Bawah

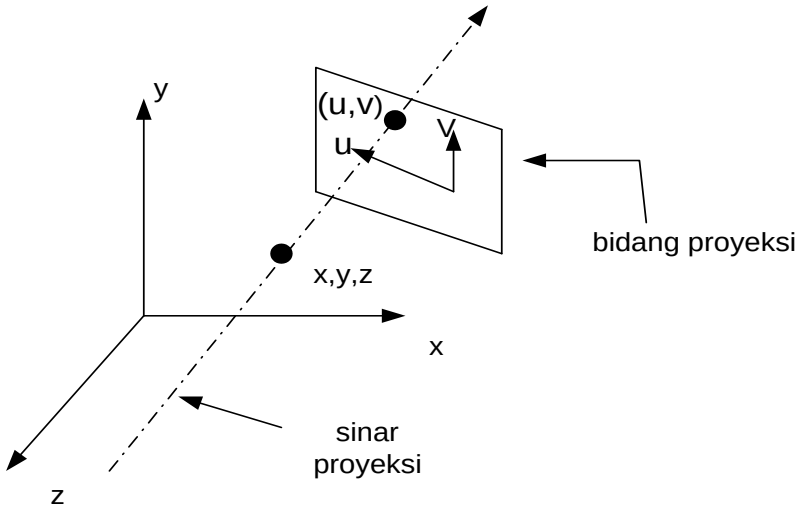
F6.b	0	4	5	bawah

Tabel 1.2 Menyatakan urutan *vertex* yang menyusun sebuah permukaan. Untuk setiap permukaan didefinisikan sebagai polygon segitiga (*triangle*), sebagai contoh permukaan kubus bagian depan didefinisikan melalui dua buah segitiga, yaitu F1.a dan F1.b yang masing-masing disusun dari *vertex* 0, *vertex* 1, dan *vertex* 2, serta *vertex* 0, vertek 2 dan *vertex* 3.

8
Proyeksi

Proyeksi merupakan salah satu jenis transformasi , yaitu transformasi koordinat. Proyeksi merupakan proses dimana informasi tentang titik di sebuah sistem koordinat berdimensi n dipindahkan ke system koordinat berdimensi kurang dari n. sebagai contoh, titik (x,y,z) yang berada di sistem koordinat berdimensi 3 dipindahkan ke sistem koordinat yang berdimensi 2 sehingga menjadi (x,y), transformasi tersebut tentunya harus memperhitungkan pengaruh z terhadap titik (x,y). Proyeksi dapat dilakukan terhadap bidang datar (*planar*) atau kebidang kurva. Bab ini hanya akan membahas proyeksi ke bidang *planar* atau disebut sebagai *planar geometric projections.*

Planar geometric projections 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.



Gambar 8-1 Proyeksi Planar

Proyeksi *planar* dapat dibagi menjadi dua macam, yaitu: proyeksi ***parallel*** dan proyeksi ***perspektif***. Perbedaan utama antara kedua proyeksi tersebut adalah, pada proyeksi *perspektif* jarak antara titik pusat proyeksi ke bidang proyeksi bersifat *infinite* (tertentu) sedangkan pada proyeksi parallel jarak antara titik pusat proyeksi ke bidang proyeksi tidak terhingga.

8.1 Proyeksi Paralel

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

8.1.1 Proyeksi Orthographic

Proyeksi *Orthographic* diperoleh apabila sinar proyeksi tegak lurus dengan bidang proyeksi. Proyeksi *orthographic* sering digunakan untuk menghasilkan tampak depan, tampak belakang, tampak samping dan tampak atas dari sebuah benda atau disebut sebagai *Multiview orthographic*. Tampak atas, tampak belakang dan tampak dari samping sebuah benda sering disebut sebagai *elevation.* Sedangkan tampak dari atas disebut sebagai *plan view* (Hearn dan Baker, 2002).

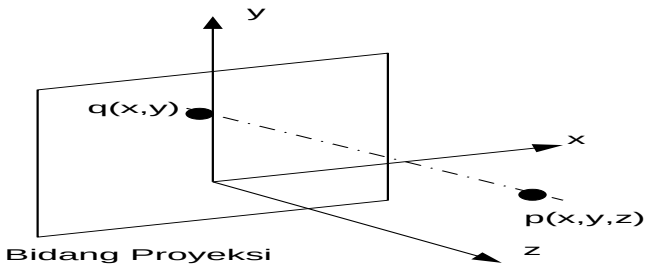
Transformasi untuk proyeksi *multiview orthographic* dapat diperoleh dengan rumus

Proyeksi terhadap bidang x-z:qx = px,qy =pz

Proyeksi terhadap bidang y-z:qx = px,qy =pz

Proyeksi terhadap bidang x-y:qx = px,qy =py

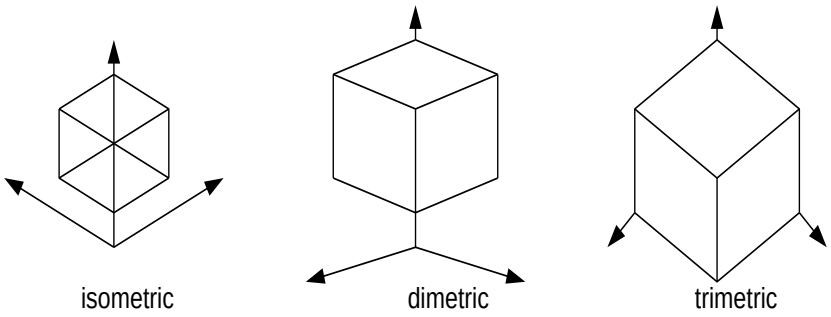
Dimana q(x,y) merupakan titik hasil proyeksi dari p(x,y,z) seperti digambarkan di bawah ini



Gambar 8-2 transformasi untuk memperoleh proyeksi orthographic

Proyeksi *orthographic* yang menampakan 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 a*xonometric* adalah proyeksi *dimetric* yaitu proyeksi yang diperoleh

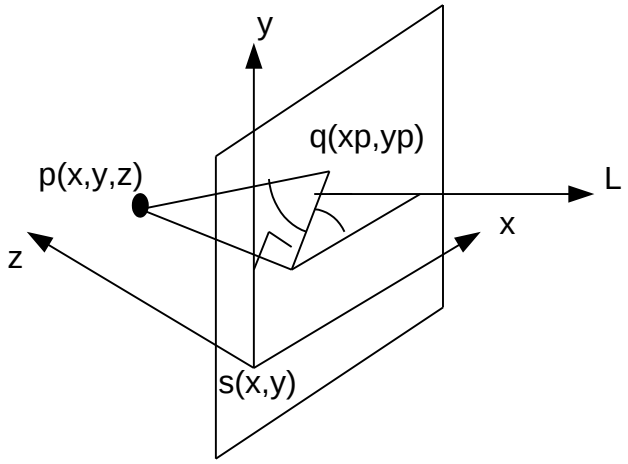
dengan mengatur agar bidang proyeksi berpotongan dengan dua sumbu utama pada sudut yang sama, sedangkan proyeksi *trimetric* diperoleh apabila ketiga sumbu utama berpotongan dengan bidang proyeksi pada sudut yang berbeda. Gambar 1.8 memperlihatkan proyeksi *isometric*, *diametric* dan *trimetric*.



Gambar 8-3 proyeksi axonometric

8.1.2 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 seperti ditunjuakn pada Gambar 1.9 titik p(x,y,z) diproyeksikan menjadi titik q(xp,yp) 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-sudut terhadap arah mendatar dari bidang proyeksi.



Gambar 8-4 proyeksi Oblique dari titik p(x,y,z) ke titik q(xp,yp)

Koordinat hasil proyeksi dapat dituliskan sebagai berikut :

$Xp = X + L \cos\beta$

$Yp = Y + L \sin\beta$

Panjang L merupakan fungsi dari koordinat z dan dapat dihitung sebagai berikut :

Tan α = $\frac{z}{L} = \frac{1}{L1}$

Dengan L1 merupakan panjang dari q-s saat z = 1.

$\frac{z}{L} = \frac{1}{L1}$

L = zL1

Sehingga rumus tersebut dapat ditulis ulang sebagai berikut :

Xp = x + z (L1 cosβ)

Yp = y + z (L1 sinβ)

Apabila α = 90^0 maka L1 = 0 sehingga dari rumus di atas kita memperoleh proyeksi orthographic, tetapi apabila L1 tidak sama dengan 0 maka kita akan memperoleh proyeksi oblique. Proyeksi oblique dengan α = 45^0 disebut sebgai proyeksi cavalier, apabila α = $63{,}43495^0$ maka kita akan memperoleh proyeksi cabinet.

8.2 Proyeksi Perspektif

Proyeksi perspektif memberikan sudut pandang yang lebih realistis dibandingkan proyeksi orthographic. Proyeksi perspektif memberikan tampilan yang sama dengan apa yang kita lihat sehari-hari karena pada kenyataannya 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*.

Pada *perspektif* semua garis menghilang pada satu atau lebih titik yang sama atau disebut titik hilang (*vanishing point*). Hal ini mengakibatkan garis 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.

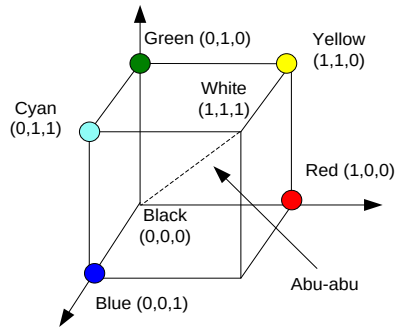
9 Rendering

Rendering merupakan salah satu teknik pencitraan objek 3 dimensi untuk mendapatkan *image* yang realistis dengan penambahan beberapa efek, seperti pencahayaan dan *shading*. Kebanyakan manusia menggemari sesuatu yang hidup, bukan yang kaku dan statik.

9.1 Warna

Warna sebenarnya merupakan persepsi kita terhadap pantulan cahaya dari benda-benda di depan mata. Tidak ada ketentuan jumlah warna dasar tetapi dalam implementasi dengan komputer hanya dibutuhkan tiga warna dasar. Ada berbagai model untuk menyatakan warna dasar serta rentang warna yang dihasilkan, salah satu diantaranya adalah RGB.

Model warna RGB dapat digambarkan sebagai sebuah kotak yang mempunyai tiga sumbu yaitu : R(*red*), G(*green*), B(*blue*), dimana nilai masing-masing sumbu berkisar dari 0 sampaii 1. Warna ditentukan berdasarkan lokasi warna tersebut terhadap Sumbu RGB. Gambar 1.10 menunjukkan model RGB.



Gambar 9-1 Model RGB

9.2 Pencahayaan (Lighting)

Salah satu tujuan dari grafika komputer adalah menghasilkan tampilan yang senyata mungkin, dan karena pengaruh cahaya sangat besar terhadap hasil nyata maka dalam membuat tampilan akhir faktor pencahayaan harus diperhitungkan pula. Tetapi mengingat bahwa grafika komputer adalah model matematika dari kehidupan nyata maka pencahayaan juga harus diubah menjadi model matematika. Model matematika itu harus memenuhi persyaratan sebagai berikut :

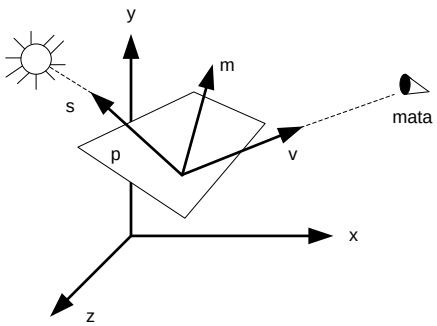
- Menghasilkan efek seperti cahaya sungguhan
- Dapat dihitung dengan cepat

Model pencahayaan tiga dimensi menyangkut yang realistik menyangkut dua elemen penting yang sangat berkaitan erat dengan *shading model*, yaitu :

- Keakuratan dalam menggambarkan objek.
- Teknik pencahayaan yang baik.

Saat cahaya menimpa permukaan benda maka sebagian cahaya akan dipantulkan dan sebagian lain diserap. Bergantung kepada frekuensi atau panjang gelombang yang dipantulkan dan diserap maka kita akan melihat warna. Mata kita selain sensitif terhadap warna juga sensitif terhadap intensitas cahaya (*brightness*). Secara awam kita menyebut intensitas cahaya sebagai kecerahan.

Sifat materi penyusun benda menentukan bagaimana cahaya bereaksi terhadap materi penyusun benda. Secara umum, cahaya yang menimpa sebuah permukaan akan dipantulkan oleh permukaan seperti diperlihatkan pada gambar 1.11



Gambar 9-2 Perjalanan cahaya dari sumber cahaya

Keterangan :

- Vektor m = vektor normal dari permukaan p.
- Vektor s = menunjukkan arah yang ditempuh oleh cahaya dari sumber cahaya menuju ke permukaan p.
- Vektor v = menunjukkan arah pantulan cahaya dari permukaan p menuju ke mata.

Bergantung kepada materi penyusun benda maka ada tiga kemungkinan arah pantulan cahaya ketika cahaya menimpa permukaan benda : *diffuse*, *specular*, *translucent.*

9.2.1 Cahaya Tersebar (Diffuse)

Suatu objek yang mempunyai permukaan yang kasar maka cahaya yang dipantulkan cenderung akan menyebar ke segala arah, cahaya yang menyebar ini disebut cahaya tersebar. Beberapa cahaya menembus permukaan dan diradiasi kembali secara seragam ke dalam semua arah.

Penghitungan cahaya tersebar menggunakan m, v dan s.

- Sebagaimana cahaya tersebar disebarkan secara seragam dalam semua arah, lokasi mata, v, tidak penting kecuali kalau v.m < 0 jika diinginkan intensitas cahaya I = 0
- Hubungan antara kecerahan permukaan dan orientasinya cahaya didasarkan pada cos(θ).

  $I_d = I_s P_d \cos(\theta)\ atau$

  $I_d = I_s P_d (s.m/|s||m|)$

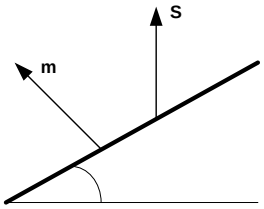
  Keterangan :

  $I_d =$ kuat cahaya tersebar (*diffuse*)

  $I_s =$ kuat cahaya di sumber cahaya

  $p_d =$ koefisien pantulan tersebar (*diffuse*)

Untuk lebih jelasnya, lihat pada gambar 1.12



Gambar 9-3Cahaya Pantulan Tersebar (Diffuse)

9.3 Metode *Shading*

Pada penggambaran objek 3 dimensi, efek yang paling mempengaruhi benda sehingga akan terlihat nyata atau tidak adalah efek *shading*, yaitu efek pewarnaan tingkat kecerahan setiap *face* yang terdapat pada objek gambar. Dalam bagian ini, kita mempertimbangkan aplikasi dari suatu model iluminasi untuk membuat objek grafik baku yang dibentuk oleh permukaan Polygon. Masing-masing Polygon dapat dibuat

dengan intensitas tunggal, atau intensitas dapat dibuat pada titik masing-masing permukaan dengan menggunakan suatu rencana interpolasi.

Metode *shading* yang digunakan antara lain :

9.3.1 Metode Flat Shading

Flat shading adalah salah satu teknik *shading* dimana satu *face* mempunyai warna yang sama. Pada metode ini sebuah intensitas tunggal dihitung untuk masing-masing Polygon, semua titik pada permukaan Polygon dipaparkan dengan nilai intensitas yang sama.

Karakteristik *flat shading* diantaranya :

- Pemberian *tone* yang sama untuk setiap Polygon
- Penghitungan jumlah cahaya mulai dari titik tunggal pada permukaan.
- Penggunaan satu normal untuk seluruhnya.

Secara umum *flat shading* dapat menghasilkan *shading* yang akurat dengan ketentuan sebagi berikut :

- Objek berbentuk polihendra (segi banyak), yaitu jaring yang mempunyai ruang terhingga dan tertutup.

Semua sumber cahaya jauh dari permukaan objek, maka $N \cdot L$ adalah tetap untuk semua permukaan Polygon.

- Posisi penglihatan yang cukup jauh dari permukaan sehingga $N \cdot V$ adalah tetap untuk semua permukaan Polygon.

Definisi matematik :

$\overline{L}$: Vektor sumber cahaya (arah) membentur permukaan (yang disingkat untuk ' L', tetapi ingat ini adalah suatu garis vektor).

$\overline{V}$: Vektor sudut pandang (*eyepoint*) pemandangan di permukaan (yang disingkat untuk ' V', tetapi ingat ini adalah suatu garis vektor).

$\overline{N}$: Vektor normal (tegak lurus) pada permukaan (yang disingkat untuk ' N', tetapi ingat ini adalah suatu garis vektor).

Sekalipun semua kondisi-kondisi ini tidak benar, kita masih bisa memperkirakan efek cahaya permukaan dengan menggunakan permukaan Polygon kecil dengan menggunakan *flat shading* dan menghitung intensitas untuk setiap permukaan, khususnya pada pusat Polygon.



Gambar 9-4 Flat Shading

9.3.2 Metode Gouraud Shading

Metode ini merender sebuah permukaan Polygon dengan interpolasi linier yaitu nilai intensitas yang mengenai setiap permukaan berbeda. Warna yang dipantulkan dihitung tiap *vertex* kemudian secara halus diinterpolasikan. Membuat suatu permukaan Polygon dengan menginterpolasikan nilai intensitas secara linier ke seberang permukaan. Nilai intensitas untuk masing-masing Polygon dipasangkan dengan nilai dari Polygon yang bersebelahan sepanjang tepi (*edges*) yang umum, dengan begitu menghentikan penghapusan intensitas yang dapat terjadi di *flat shading*.

Karakteristik *shading* yang dihasilkan :

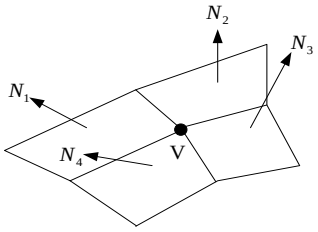
Shading yang dihasilkan halus (tampak nyata)

Penggunaan level abu-abu yang berbeda disepanjang Polygon diinterpolasikan di antara titik.

Masing-masing permukaan Polygon dibuat menggunakan *gouraud shading* dengan melakukan perhitungan sebagai berikut :

- Tentukan satuan vektor normal rata-rata pada setiap titik ujung Polygon.
- Pakaikan model iluminasi untuk setiap titik untuk menghitung intensitas titik.
- Interpolasikan secara linier intensitas titik pada permukaan Polygon.

Pada setiap vertex Polygon, kita memperoleh suatu vektor normal dengan menambahkan permukaan normal dari semua Polygon dibagi vertex itu, seperti digambarkan dalam gambar 1.14. Vektor normal pada vertex V dihitung dengan jumlah rata-rata permukaan normal dari masing-masing Polygon dibagi vertex tersebut.



Gambar 9-5 Vektor normal pada vertex V

Kemudian untuk memposisikan vertex V, kita memperoleh unit vertex normal dengan perhitungan :

$$N_V = \frac{\sum_{k=1}^{n} N_k}{\left|\sum_{k=1}^{n} N_k\right|}$$

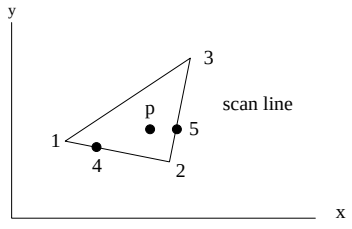
Dimana :

N_v : Vektor normal vertex V

N_k : Vektor normal vertex k, dengan k adalah konstanta

Setelah kita memiliki vertex normal, kita dapat menentukan intensitas di vertex itu dengan menggunakan model pencahayaan.

Gambar 1.15 menunjukkan langkah berikutnya : menyisipkan intensitas sepanjang tepi (*edges*) Polygon. Untuk setiap *scanline*, intensitas persimpangan *scanline* dengan tepi (*edge*) Polygon diinterpolasi secara linier dari intensitas di tepi titik akhir (*endpoints*). Contoh, dalam gambar 1.15, tepi Polygon dengan titik akhir vertex pada posisi 1 dan 2 dipotong oleh *scanline* pada titik 4. Suatu metoda cepat untuk memperoleh intensitas pada titik 4 adalah dengan menyisipkan intensitas antara I_1 dan I_2 hanya dengan memindahkan *scanline* secara vertikal. *Scanline* adalah garis dengan x = 0 dan bergerak dari y_{min} menuju y_{max} dengan y_{min} menunjukkan koordinat y paling kecil dari vertex-vertex polygon dan y_{max} menunjukkan y terbesar dari vertex-vertex Polygon..



Gambar 9-6 Interpolasi linier

Untuk *gouraud shading*, intensitas pada titik 4 secara linier disisipkan dari intensitas pada vertex 1 dan 2. Intensitas pada titik 5 secara linier disisipkan dari intensitas pada vertex 2 dan 3. Suatu titik bagian dalam **p** adalah menandakan suatu nilai intensitas yang secara linier disisipkan dari intensitas pada posisi 4 dan 5.

$$I_4 = \frac{y_4 - y_1}{y_1 - y_2} I_1 + \frac{y_1 - y_4}{y_1 - y_2} I_2$$

Dimana I adalah intensitas linier dan y adalah koordinat sumbu y dari vertex-vertex Polygon.

Dengan cara yang sama, intensitas pada perpotongan *scanline* yang tepat (titik 5) disisipkan dari nilai intensitas pada vertex 2 dan 3. Setelah membatasi intensitas dibentuk untuk sebuah *scanline*, sebuah titik pada bagian dalam disisipkan dari pembatasan intensitas pada titik 4 dan 5 seperti :

$$I_p = \frac{x_3 - x_p}{x_5 - x_4} I_4 + \frac{x_p - x_4}{x_5 - x_4} I_5$$

Dimana I adalah intensitas linier dan x adalah koordinat sumbu x dari vertex-vertex Polygon.

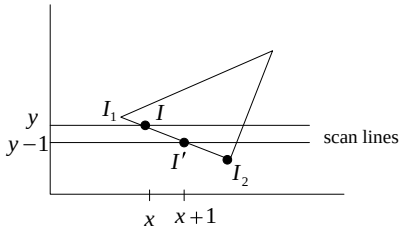
Perhitungan *Incremental* digunakan untuk memperoleh urutan nilai intensitas tepi (*edge*) antara *scanline* bentuk dan untuk memperoleh intensitas berurutan sepanjang *scanline*. Seperti ditunjukkan pada gambar 1.16, jika intensitas pada posisi *edge* (x, y) disisipkan, seperti :

$$I = \frac{y - y_2}{y_1 - y_2} I_1 + \frac{y_1 - y}{y_1 - y_2} I_2$$

Kemudian kita dapat memperoleh intensitas sepanjang tepi (*edge*) untuk *scanline* yang berikutnya, y-1, seperti :

$$I' = I + \frac{I_2 - I_1}{y_1 - y_2}$$

Dimana I' adalah intensitas sepanjang *edge* untuk *scanline* berikutnya.



Gambar 9-7 Interpolasi incremental dari nilai intensitas sepanjang tepi Polygon untuk garis pindai (scanline) yang berurutan

Perhitungan serupa digunakan untuk memperoleh intensitas pada urutan posisi pixel horizontal sepanjang setiap *scanline*.

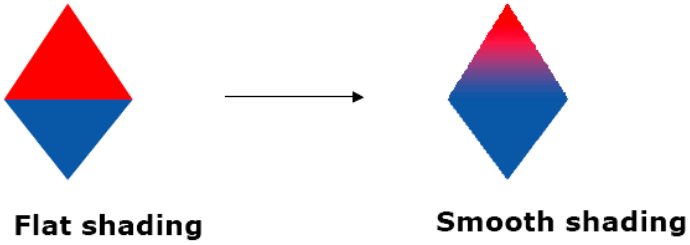
Ketika permukaan akan dibuat dengan menggunakan warna, intensitas dari tiap komponen warna dihitung pada vertex. *Gouraud shading* menghentikan pemindahan

intensitas dihubungkan dengan model *flat shading*, tetapi juga mempunyai beberapa kekurangan yang berbeda. Sorotan pada permukaan kadang-kadang ditunjukkan dengan bentuk ganjil, dan interpolasi intensitas yang linier dapat menyebabkan lapisan intensitas gelap atau terang disebut *mach bands*, yang nampak pada permukaan. Efek ini dapat dikurangi dengan pembagian permukaan ke dalam suatu permukaan Polygon yang lebih besar.



Gambar 9-8 Gouraud shading

Gambar 1.18 berikut menunjukkan perbedaan permukaan hasil *flat shading* dengan *smooth shading* atau dikenal dengan *gouraud shading*.



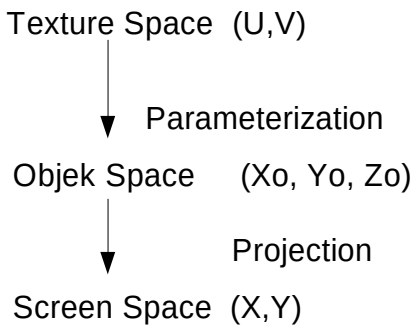
Gambar 9-9 Permukaan pada flat shading dan smooth shading

9.4 Texture Mapping

Texture mapping dapat dikatakan sebagai primitif grafika komputer seperti halnya titik dan garis. Aplikasi *texture mapping* tidak dapat dipisahkan dalam pemodelan tiga dimensi karena *texture mapping* selalu dipakai dalam semua pemodelan tiga dimensi. *Texture mapping* adalah teknik *shading* untuk pengolahan gambar yang memetakan sebuah fungsi pada permukaan tiga dimensi dalam *scene.*

Fungsi yang dipetakan mencakup satu dimensi, dua dimensi, dan tiga dimensi dan dapat digambarkan sebagai *array* atau fungsi matematika atau gambar. Sebagai contoh , tekstur satu dimensi diwakili oleh gambar lapisan, batuan, tekstur dua dimensi diwakili oleh gelombang atau permukaan yang bergelombang, dan tekstur tiga dimensi diwakili oleh awan, kayu atau marmer. Namun, dalam tugas akhir ini kita memakai tekstur dua dimensi yang akan diubah menjadi tekstur tiga dimensi. *Texture mapping*

dapat digunakan untuk menggambarkan banyak parameter permukaan disamping warna seperti *bump mapping* untuk memperjelas karakteristik permukaan yang bergelombang, *Transparency mapping* untuk mengatur intensitas cahaya permukaan tembus pandang, *Specularity mapping* untuk mengubah kehalusan permukaan dan *Illumination maaping* untuk memodelkan distribusi cahaya yang datang dari berbagai arah. Namun dari sumua itu yang paling penting adalah *Geometrical mapping, geometrical mapping* secara keseluruhan ditentukan dengan dengan transformasi tiga dimensi terhadap kamera, tansformasi model yang menggambarkan geometri *scene* dan pemberian parameter pada permukaan dengan tujuan memetakan tekstur ke permukaan, ada 2 tahap yang harus dilakukan untuk memetakan teksture dari *space* ke *screen space* seperti pada gambar dibawah ini.



Gambar 9-10 Tahap texture mapping

Sistem koordinat pada masing-masing *space* adalah pada *texture space koordinat* yang dipakai adalah (U,V), objek *space* (*scene*) memakai koordinat (Xo, Yo, Zo) dan *screen space* memakai koordinat (X,Y)

10

Transformasi 3 Dimensi

Salah satu sub bagian dari grafika komputer adalah pemodelan objek (*object modeling*). Pada dasarnya transformasi 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.

Transformasi pada dunia 3 dimensi, pada dasarnya sama dengan transformasi pada 2 dimensi, hanya pada 3 dimensi kita juga memperhitungkan sumbu Z. Sama seperti pada 2 dimensi, ada tiga transformasi dasar yang dapat dilakukan yaitu translasi, penskalaan, rotasi. Perbedaannya adalah pada objek 3 dimensi proses transformasinya dilakukan dengan mempertimbangkan koordinat yang merupakan besarnya kedalaman dari objek.

Transformasi pada 3 dimensi pada dasarnya sama dengan transformasi pada 2 dimensi, hanya pada 3 dimensi kita juga memperhitungkan sumbu z. sama seperti pada 2 dimensi, ada 3 transformasi dasar yang dilakukan terhadap *vertex*, yaitu: translasi, penskalaan , rotasi

Titik hasil transformasi dapat diperoleh melalui rumus dibawah ini disebut sebagai *Affine Transformation*.

$$Q = P * M + tr$$

Dimana :

Q = (Qx,Qy,Qz) menyatakan matriks 1x3 yang berisi titik hasil transformasi.

P = (Px, Py, Pz) menyatakan matrik 1x3 yang berisi titik yang akan ditransformasi.

Tr = (trx, try, trz) menyatakan matrik 1x3 yang berisi banyaknya pergeseran pada sumbu x, y dan z.

M = Matriks mentransformasi berukuran 3x3 seperti di bawah ini :

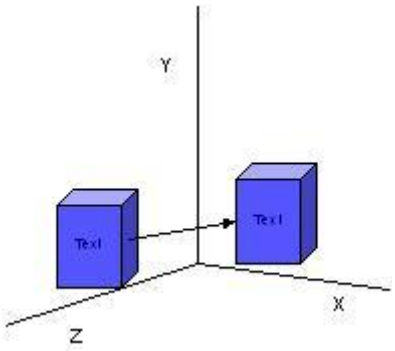
$$\begin{bmatrix} moo & mo1 & mo2 \\ m1o & m11 & m12 \\ m20 & m21 & m22 \end{bmatrix}$$

10.1 Translasi

Transformasi translasi merupakan operasi yang menyebabkan perpindahan objek tiga dimensi dari satu tempat ke tempat yang lainnya. Perubahan ini berlaku dalam arah yang sejajar dengan sumbu x, y, z. dalam operasi translasi, setiap titik pada suatu entitas yang ditranslasi bergerak dalam jarak yang sama. Pergerakan tersebut dapat berlaku dalam arah sumbu x, y, z.

Untuk mentranslasikan suatu titik (x,y,z) dengan pergeseran sebesar (tx, ty, tz) menjadi titik (x',y',z') adalah:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & tx \\ 0 & 1 & 0 & ty \\ 0 & 0 & 4 & tz \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



Gambar 10-1 Translasi Objek

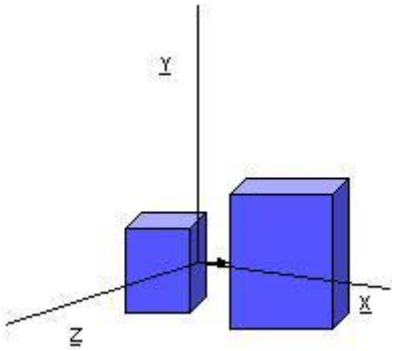
Untuk invers dari translasi dapat dilakuan dengan mengubah nilai vektor translasi menjadi negatif

10.2 Penskalaan

Transformasi skala merupakan operasi yang menyebabkan ukuran objek berubah. Perubahan ini berlaku dalam arah yang sejajar dengan sumbu x, y, z.

Untuk melakukan skala pada objek suatu titik (x,y,z) dengan faktor skala (sx,sy,sz) menjadi titik (x',y',z') adalah :

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} sx & 0 & 0 & 1 \\ 0 & sy & 0 & 0 \\ 0 & 0 & sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



Gambar 10-2 Skala Objek

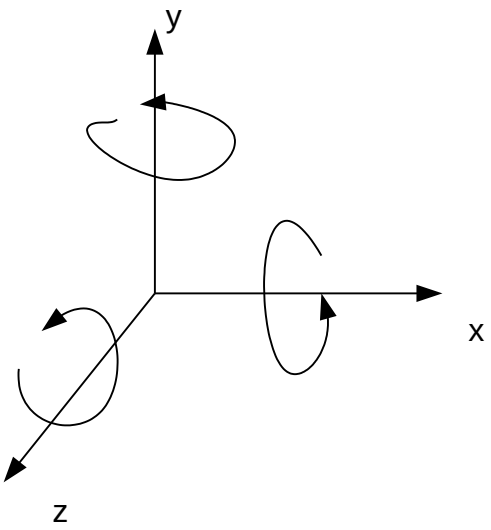
Untuk invers dari skala dapat dilakukan dengan mengubah nilai faktor skala menjadi (1/sx, 1/sy, 1/sy)

10.3 Rotasi

Berbeda dengan rotasi 2 dimensi yang menggunakan titik pusat (0,0)sebagai pusat perputaran, rotasi 3 dimensi menggunakan sumbu koordinat sebagai pusat perputaran. Dengan demikian ada 3 macam rotasi yang dapat dilakukan, yaitu:

- Rotasi sumbu x
- Rotasi sumbu y
- Rotasi sumbu z

Gambar 1-8 memperlihatkan bagaimana hubungan antara rotasi 3 dimensi dan sumbu rotasi.

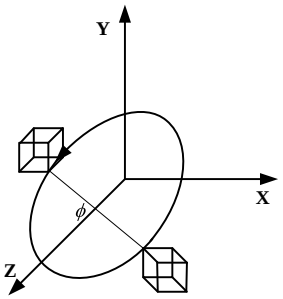


Gambar 10-3 Rotasi dan sumbu rotasi.

Mengingat ada 3 buah sumbu rotasi maka matriks transformasi yang digunakan juga bergantung kepada sumbu putar. Adapun isi matriks transformasi sesuai dengan sumbu putar didefinisikan sebagai berikut :

- Rotasi terhadap sumbu z

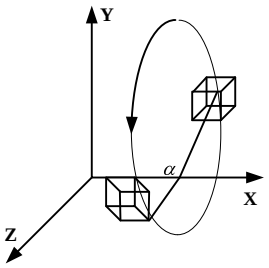
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



Gambar 10-4 Rotasi pada sumbu z

- Rotasi terhadap sumbu x

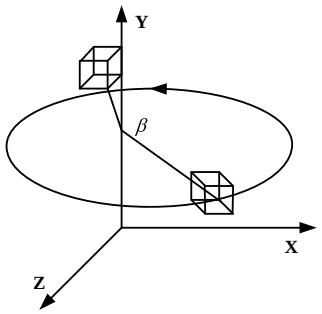
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



Gambar 10-5 Rotasi pada sumbu x

- Rotasi terhadap sumbu y

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



Gambar 10-6 Rotasi pada sumbu y

11 Konsep Dasar OpenGL

OpenGL adalah suatu *library* untuk pemodelan dan grafik tiga dimensi yang mempunyai kelebihan dalam kecepatan dan dapat digunakan pada *flatform* yang luas. OpenGL bukanlah bahasa pemrograman tetapi merupakan suatu *Application Programming Interface* (*API*).

11.1 Sintaks Perintah OpenGL

Sintaks perintah OpenGL mengikuti aturan penulisan dari *library* dimana fungsi tersebut berasal, format penulisan fungsi OpenGL adalah:

```
<awalan library><perintah><optional jumlah argumen><optional tipe argumen>
```

Semua perintah OpenGL menggunakan awalan *gl* diikuti dengan huruf kapital pada setiap kata membentuk nama perintah (sebagai contoh *glClearColor*). Untuk mendefinisikan konstanta diawali dengan *GL_*, dengan menggunakan huruf kapital dan garis bawah untuk memisahkan kata (seperti *GL_POLY_STIPPLE*). Terkadang beberapa huruf dan angka ditambahkan pada akhir perintah (seperti *3f* pada *glVertex3f*). Dalam hal ini angka 3 menunjukkan berapa banyak argumen yang harus ada pada perintah tersebut dan akhiran huruf f menunjukkan jenis datanya yaitu *floating*. Sebagai contoh pada dua perintah berikut ini :

```
glVertex3i(1,0,-2);
```

```
glVertex3f(1.0, 0.0, -2.0);
```

adalah sama yaitu meletakkan titik di layar pada koordinat x = 1, y = 0 dan z = -2, perbedaannya yaitu pada perintah pertama menspesifikasikan titik dengan tipe data *integer 32-bit*, sedangkan yang kedua dengan tipe data *single precision floating point*.

Beberapa perintah OpenGL menambahkan perintah huruf akhir *v* yang menunjukkan bahwa perintah tersebut menggunakan pointer ke *array/vektor*. Di bawah ini contoh perbedaannya.

```
float color_array[]={1.0,0.0,0.0}
```

```
glColor3f (1.0,0.0,0.0);
glColor3fv(color_array);
```

11.2 Library yang Berhubungan dengan OpenGL

OpenGL menyediakan set perintah untuk menggambar dan semua penggambaran yang lebih tinggi tingkatnya harus dilakukan dengan mengambil fungsi dasar dari perintah ini. Maka dari itu dapat dibuat *library* itu sendiri di atas program OpenGL yang mempermudah pemrograman lebih lanjut. Fungsi asli dari OpenGL sendiri selalu diawali dengan *gl* yang terdapat pada library *opengl32.dll* dan file *header gl.h*. Sedangkan beberapa *library* yang telah ditulis untuk menyediakan fungsi-fungsi tambahan pada OpenGL adalah :

- *OpenGL Utility Library (GLU)* yang didalamnya terdapat sejumlah rutin yang menggunakan level bawah dari perintah OpenGL. Rutin-rutin ini mempunyai awalan *glu*. *Library* ini digunakan sebagai bagian dari implementasi OpenGL.
- *OpenGL Extension* untuk *X-Windows* yang menyediakan fungsi untuk menciptakan *OpenGL context* dan mengasosiasikannya dengan mesin yang menggunakan *X-Windows*. Rutin-rutin ini mempunyai awalan *glx.*

Auxiliary atau *aux library* terdapat pada *library glaux.lib* dan *file header glaux.h.* Perintah yang akan digunakan selalu menggunakan awalan *aux*

- *OpenGL Utility Toolkit (GLUT)* adalah *toolkit* untuk sistem *windows* yang ditulis oleh Mark Kilgard untuk menyembunyikan perintah *API* sistem *windows* yang kompleks.

11.3 Menggambar Objek Geometri

Pada OpenGL ada dua dasar operasi gambar yaitu membersihkan *windows* dan menggambar objek geometri termasuk titik, garis dan Polygon.

11.3.1 Membersihkan Windows

Menggambar pada layar komputer berbeda dengan menggambar pada kertas putih yang dari pabriknya sudah berwarna putih. Pada komputer, *memory* untuk menampilkan gambar biasanya diisi dengan gambar yang berasal dari perintah gambar paling akhir, jadi perlu dibersihkan dengan warna latar belakang sebelum digambar lagi. Warna latar belakang yang dipilih tergantung dari aplikasi yang akan dibuat. Sintaks *glClearColor(Glclamp red,Glclamp green, Glclamp blue, Glclamp alpha)* digunakan untuk memilih warna, yang akan digunakan untuk membersihkan latar belakang dalam *mode RGBA*. Selanjutnya perintah *glClear(Glbitfield mask)* digunakan untuk membersihkan *buffer* yang dispesifikasikan dengan warna yang telah ditentukan. Contoh berikut ini perintah yang digunakan untuk membersihkan layar latar belakang dengan warna hitam dan *buffer* apa yang akan dibersihkan. Dalam hal ini, *buffer* warna yang akan dibersihkan karena *buffer* warna merupakan tempat gambar disimpan.

```
glClearColor(0.0,0.0,0.0);
glClear(GL_COLOR_BUFFER_BIT);
```

11.3.2 Spesifikasi Warna

Pada OpenGL mendeskripsikan objek dengan warna objek adalah proses yang berjalan sendiri-sendiri. Karena pada umumnya seorang *programmer* akan mengatur warna terlebih dahulu lalu menggambar objek. Sebelum warna diubah maka semua objek yang digambar sesudah perintah tersebut akan menggunakan warna terakhir yang terdapat pada *coloring scheme.*

Untuk warna digunakan perintah *glColor3f()*, jika lebih dari tiga maka argumen keempat adalah *alpha* yang akan dijelaskan pada bagian *blending* sebagai salah satu efek yang dipunyai OpenGL. Contoh berikut menunjukkan urutan langkah dalam proses spesifikasi warna sebelum objek digambar.

```
glColor3f(0.0,1.0,0.0);        //setting warna
draw_object(A);          //gambar objek A
```

11.3.3 Memaksa Proses Menggambar Sampai Selesai

Kebanyakan sistem grafik modern sudah menggunakan sistem *graphics pipeline*. Dimana CPU utama memberikan *issue* perintah menggambar dan hardware lain yang melakukan *transformasi, clipping*, *shading*, *texturing* dan lain-lain. Pada arsitektur yang demikian, proses tidak dilakukan pada satu computer karena setiap komputer mempunyai tugas sendiri. CPU utama tidak harus menunggu proses pada masing-masing komputer tadi selesai, tapi bisa dengan memberikan *issue* perintah gambar yang berikutnya. Untuk inilah OpenGL menyediakan perintah *glFlush()* yang memaksa *client* untuk segera mengirim paket *network* walaupun belum penuh. Program sebaiknya ditulis menggunakan perintah ini karena *glFlush()* tidak memaksa proses gambar untuk selesai tetapi memaksa proses gambar untuk segera dieksekusi, sehingga dijamin semua perintah gambar yang sebelumnya akan segera dieksekusi dalam suatu waktu tertentu.

11.3.4 Menggambar di Bidang Tiga Dimensi

Untuk menggambar grafik jenis apapun pada komputer biasanya dimulai dengan *pixel*. *Pixel* adalah elemen terkecil dari layar monitor yang mempunyai atribut warna dan posisi. Sedangkan untuk membentuk garis, Polygon, objek dan lain-lain dapat dilakukan melalui urutan *pixel* yang berbeda. Menggambar dengan menggunakan OpenGL mempunyai perbedaan dengan bahasa lain, yaitu tidak perlu memikirkan koordinat layar secara fisik tetapi hanya perlu menspesifikasikan posisi koordinat dengan volume penglihatan. OpenGL memikirkan sendiri bagaimana caranya

menggambar titik, garis, dan lainnya yang berada dalam ruang tiga dimensi ke gambar dua dimensi pada layar komputer.

Area gambar yang dibatasi ini adalah ruang koordinat kartesian yang mempunyai *range* dari -100 hingga 100 untuk sumbu x, y dan z. Secara sederhana bidang ini dapat dianggap sebagai bidang gambar untuk perintah-perintah OpenGL.

Untuk menggambar titik digunakan suatu perintah OpenGL yaitu : *glVertex*. Fungsi ini dapat mempunyai 2 sampai 4 parameter dari berbagai macam tipe data. Sebagai contoh perintah *glVertex* di bawah ini akan menspesifikasikan sebuah titik pada posisi 4 sumbu x, 4 sumbu y dan 0 untuk sumbu z. *glVertex3f(4.0f,4.0f,0.0f);.*

Setelah diketahui cara untuk menspesifikasikan sebuah titik di ruang pada OpenGL. Selanjutnya yang harus ditambahkan adalah informasi tambahan mengenai titik tersebut, apakah titik tersebut akhir dari sebuah garis, atau merupakan sebuah titik sudut dari sebuah Polygon atau lainnya, karena definisi geometrik dari sebuah *vertex* sebenarnya bukanlah hanya sebuah titik pada layar tetapi lebih merupakan sebuah titik dimana terjadi interseksi antara dua buah garis atau kurva.

Primitif adalah interpretasi sejumlah set atau deretan titik pada sebuah bentuk yang digambar pada layar. Pada OpenGL terdapat sepuluh macam primitif dari mulai menggambar sebuah titik hingga Polygon. Untuk itu digunakan perintah *glBegin* sebagai cara memberitahu OpenGL untuk memulai menginterpretasi sederetan titik sebagai salah satu bentuk primitif. Dan untuk mengakhiri deretan titik ini digunakan perintah *glEnd.* Sebagai contoh ada dua perintah berikut ini :

Perintah 1 :

```
glBegin(GL_POINTS);            //spesifikasikan titik sebagai primitif
glVertex3f(0.0f,0.0f,0.0f);    //spesifikasikan posisi titik
glVertex3f(4.0f,4.0f,4.0f);    //spesifikasikan titik lain
glEnd();                 //mengakhiri perintah menggambar titik
Perintah 2 :
glBegin( GL_LINES);            //spesifikasikan garis sebagai primitif
glVertex3f(0.0f,0.0f,0.0f);    //spesifikasikan posisi titik awal garis
glVertex3f(4.0f,4.0f,4.0f);    //spesifikasikan titik akhir garis
glEnd();                 //mengakhiri perintah menggambar titik
```

Perintah 1 hasilnya berupa dua buah titik di layar pada posisi (0.0f,0.0f,0.0f) dan posisi (4.0f,4.0f,4.0f) sedangkan pada perintah 2 akan menghasilkan garis yang melalui titik (0.0f,0.0f,0.0f) dan (4.0f,4.0f,4.0f).

Ketika menggambar titik tunggal, secara *default size* titik adalah satu. Untuk mengubah *size* titik ini digunakan perintah *glPointSize()* dengan parameter ukurannya. Selain

menggunakan pengubahan titik dapat juga dilakukan pengubahan letak dari garis ketika melakukan penggambaran.

11.3.5 Metode Hidden_Surface Removal

Jika ada dua objek digambar, gambar A kemudian gambar B maka pada suatu sudut pandang tertentu akan nampak objek B menutupi objek A, tapi jika melihat dari sudut pandang yang berlawanan maka objek A harus berada di depan objek B. Relasi saling menutupi ini harus dipertahankan dalam menggambar layar yang realistik. Oleh karena itu digunakan metode *hidden_surface removal* yaitu penglihatan dari objek solid yang terhalang oleh objek lain. Dalam OpenGL dikenal *buffer* yang bertugas menangani masalah ini yaitu *depth buffer* yang bekerja dengan cara menyimpan kedalaman dan jarak dari sudut pandang tiap *pixel* pada *windows.*

Perhitungan grafika dalam *hardware* ataupun *software* mengkonversikan semua permukaan objek yang digambar menjadi sekelompok *pixel* pada *windows* dimana permukaan akan tampak jika tidak dihalangi oleh sesuatu. Disini juga dilakukan perhitungan jarak dari mata ke objek. Dengan menggunakan *depth buffer*. Jika *pixel* baru lebih dekat ke mata daripada nilai pixel sebelumnya di dalam *depth buffer* maka nilai kedalaman dan warna *pixel* yang baru akan menggantikan *pixel* sebelumnya dan begitu juga sebaliknya. Sehingga metode *hidden_surface removal* ini akan semakin meningkatkan kinerja penggambaran karena informasi yang terbaru sudah dibandingkan dengan informasi lama dan menggambar salah satu saja. Untuk menggunakan *depth buffer* maka tinggal mengaktifkan perintah OpenGL, yaitu *glEnable(GL_DEPTH_TEST)* dan ini hanya dilakukan sekali.

11.3.6 Teknik Viewing pada OpenGL

Garis besar dalam ilmu grafika komputer adalah bagaimana menghasilkan gambar dua dimensi dari objek tiga dimensi karena objek tersebut digambar di layar komputer yang merupakan bidang dua dimensi. Beberapa operasi pada komputer yang mengkonversikan koordinat objek tiga dimensi ke posisi *pixel* pada layar komputer :

Transformasi, yang dipresentasikan dengan perkalian matriks, termasuk pemodelan objek (*modelling*), penglihatan (*viewing*), dan operasi proyeksi (*projection*). Termasuk juga di dalamnya rotasi, translasi, skala, proyeksi *ortographic* dan *perspective.*

- Karena layar adalah *windows* segiempat, maka objek yang terletak di luar *windows* harus dibuang (*clipped*) dari *clipping plane.*
- Langkah akhir menyelesaikan *mapping* dari koordinat yang ditransformasikan menjadi *pixel* di layar yang dinamakan transformasi *viewport.*

Proses transformasi untuk menghasilkan ruang pandang yang diinginkan hampir sama dengan cara orang untuk mengambil foto dengan menggunakan kamera. Langkah-langkahnya adalah sebagai berikut :

- Menyiapkan *tripot* sebagai penyanggah kamera dan mengarahkan kamera menuju sebuah ruang pandang sesuai yang diinginkan (pada OpenGL diistilahkan *Viewing Transformation*).
- Mengatur ruang pandang (objek) yang akan difoto dilihat dari kamera sesuai yang diinginkan (*Modelling Transformation*).
- Memilih lensa kamera, ingin memperbesar atau sebaliknya (*Projection Transformation*).
- Mengatur seberapa besar ukuran foto yang akan dihasilkan (*Viewport Transformation*).

Jika semua tahap di atas telah dilaksanakan dengan benar maka gambar foto akan dihasilkan dengan baik atau dalam istilah OpenGL dianggap ruang pandang sudah siap dipakai atau harus dipersiapkan terlebih dahulu sebelum objek digambarkan. Untuk menghasilkan koordinat yang dipakai di *windows* maka harus dispesifikasikan sebuah matriks 4 x 4 yang kemudian dispesifikasikan dengan koordinat dari tiap *vertex* yang mendefinisikan objek, baik itu untuk transformasi *viewing, modelling* atau *projection*.

Vertex dalam OpenGL selalu terdiri dari 4 koordinat x, y, z, w walaupun pada kebanyakan kasus *w* selalu 1 sedangkan pada 2 dimensi *w* = 0.

Transformasi *viewing* dan *modelling* telah dispesifikasikan lalu dikombinasikan sehingga membentuk matriks *Modelview*, yang akan dikalikan dengan koordinat objek untuk menghasilkan koordinat mata (*eye coordinates*). Selanjutnya OpenGL menggunakan matriks *projection* untuk menghasilkan koordinat *clip*. Pada tahap ini didefinisikan volume ruang pandang atau dengan kata lain objek di luar volume tersebut tidak akan ditampilkan pada layar. Setelah itu baru dilanjutkan dengan *perspective division* yaitu membagi koordinat dengan *w* untuk menghasilkan *Normalize Device Coordinate*. Dan pada akhirnya koordinat hasil transformasi tersebut diubah menjadi koordinat pada *windows* dengan menggunakan transformasi *viewport*. Koordinat akhir inilah yang dibutuhkan guna memanipulasi objek pada layar.

Berbagai macam transformasi yang sudah dijelaskan di atas masing-masing memiliki matriks sendiri, yaitu untuk transformasi *viewing* dan *modelling* digunakan matriks *modelview*, sedangkan untuk transformasi proyeksi digunakan matriks proyeksi. Untuk menspesifikasikan matriks *modelview*, *projection* atau *texture* yang akan dimodifikasi digunakan perintah *glMatrixMode(Glenum mode)* dengan argumen untuk mode yaitu : *GL_MODEL_VIEW, GL_PROJECTION* dan *GL_TEXTURE*. Perintah transformasi selanjutnya akan mempengaruhi matriks yang telah dispesifikasikan oleh *glMatrixMode().* Sedangkan untuk membersihkan masing-masing matriks tersebut di atas bisa digunakan *glLoadIdentity().* Contoh berikut menunjukkan bahwa sebelum melakukan transformasi *viewing* atau *modelling* maka *stack* matriks yang terakhir terlebih dahulu harus dibersihkan dengan matriks identitas.

```
glMatrixMode(GL_MODEL_VIEW);

glLoadIdentity();
```

Langkah ini sangat perlu karena setiap perintah transformasi selalu mengalikan *stack* matriks yang paling akhir dengan matriks baru dispesifikasikan dan hasilnya disimpan di *stack* matriks yang paling akhir tersebut. Jika tidak mengalikan *stack* matriks yang terakhir tadi dengan matriks identitas maka dianggap transformasi dilanjutkan dari matriks transformasi yang terakhir digunakan.

11.3.7 Transformasi Modelling

Transformasi *modelling* adalah transformasi untuk menentukan posisi dan orientasi dari sebuah model. Dalam hal ini kita dapat melakukan rotasi, transisi, penskalaan atau gabungan dari ketiganya.

Tiga rutin OpenGL untuk transformasi *modelling* adalah *glTranslatef(), glRotatef(),* dan *glScalef()*. Rutin-rutin tersebut mentransformasikan sebuah objek dengan jalan menggeser, memutar, membesarkan atau mengecilkan objek tersebut. Perintah-perintah tersebut sama dengan jika kita membuat sebuah matriks *translation*, rotasi, maupun matriks *scaling* dan kemudian memanggil perintah *glMultMatrix()* dengan martiks tersebut sebagai argumen.

Sintak perintah *glTranslate{f,d}(TYPEx,TYPEy,TYPEz)* digunakan untuk mengalikan matriks yang sedang aktif dengan sebuah matriks yang mentranslasikan objek berdasarkan nilai argumen x, y, z yang diberikan. Sedangkan sintaks perintah *glRotate{f,d}(TYPEx,TYPEy,TYPEz)* digunakan untuk mengalikan matriks yang sedang aktif dengan matriks yang memutar objek dengan arah yang berlawanan dengan arah jarum jam, sebesar sudut yang diberikan argumen *angle* dan berdasarkan sumbu yang diberikan argumen x, y dan z. Dan perintah yang terakhir yaitu *glScale{f,d}(TYPEx,TYPEy,TYPEz)* digunakan untuk mengalikan martiks yang sedang aktif dengan matriks yang memperbesar, memperkecil atau merefleksikan objek. Masing-masing koordinat x, y, z dari setiap titik pada objek dikalikan dengan argumen x, y, z. Perintah *glScale* merupakan satu-satunya perintah transformasi *modelling* yang mengubah ukuran objek. Jika nilai yang diberikan lebih besar dari 1.0 maka objek akan diperbesar, jika nilai yang diberikan kurang dari 1.0 maka objek akan diperkecil, dan jika nilai yang diberikan negatif, maka objek akan direfleksikan (dicerminkan).

11.3.8 Transformasi Viewing

Memanggil transformasi *viewing* dapat dianalogikan dengan mendefinisikan dan meletakkan kamera pada posisinya. Sebelum mendefinisikan transformasi *viewing* perlu untuk menggeser matriks yang sedang aktif dengan perintah *glLoadIdentity().* Penyelesaian transformasi *viewing* dapat dilakukan dengan beberapa jalan, seperti yang dijelaskan di bawah ini. Kita dapat juga untuk memilih untuk menggunakan letak dan orientasi *default* dari titik pandang, yang terletak di pusat koordinat dan memandang ke sumbu z negatif. Adapun cara yang digunakan untuk menyelesaikan transformasi *viewing* adalah :

- Menggunakan salah satu perintah dari transformasi *modelling* yaitu *glRotate*()* atau *glTranslate*().*

- Menggunakan rutin *utility library gluLookAt()* untuk mendefinisikan garis pandang. Rutin ini mengandung perintah untuk rotasi dan translasi.
- Membuat rutin sendiri yang mengandung perintah rotasi dan translasi. Beberapa aplikasi mungkin membutuhkan rutin sendiri yang memungkinkan aplikasi tersebut menentukan *viewing* yang sesuai.

Seringkali seorang *programmer* mengatur layar pada sekitar pusat sistem koordinat atau di lokasi lain yang sesuai, kemudian ingin melihatnya dari posisi pandang yang berubah-ubah untuk mendapatkan hasil yang sesuai. Rutin *gluLookAt()* pada OpenGL dirancang untuk tujuan tersebut. Rutin tersebut membutuhkan tiga set argumen untuk menentukan letak titik pandang, mendefinisikan titik referensi terhadap arah dari kamera dan menunjukkan dimana arah atas. Perintah yang digunakan adalah :

```
gluLookAt(glDouble eyeX, glDouble eyeY, glDouble eyeZ, glDouble
centerX, glDouble centerY, glDouble centerZ, glDouble upX, glDouble
upY, glDouble upZ);
```

Perintah tersebut mendefinisikan matriks *viewing* dan mengalikan dengan matriks yang sedang aktif. Titik pandang yang diperlukan ditentukan oleh *eyeX, eyeY, eyeZ*. Argumen *centerX*, *centerY, centerZ* menyatakan sembarang titik sepanjang garis pandang tetapi biasanya ada beberapa titik di tengah garis pandang tersebut. Argumen *upX, upY, upZ* menunjukkan dimana arah atas.

11.3.9 Transformasi Proyeksi

Transformasi proyeksi menentukan bagaimana suatu objek ditransformasikan ke layar. Ada dua macam proyeksi yang ada pada OpenGL yaitu *perspektif* dan *ortografik*. Proyeksi perspektif adalah proyeksi yang melihat objek seperti segala sesuatu yang ada pada kehidupan sehari-hari. Maksudnya adalah bahwa makin jauh suatu objek maka akan makin kecil penampakan objek tersebut. Jika ingin menampilkan objek sesuai dengan kenyataan maka dipilihlah proyeksi perspektif ini. Hal ini terjadi karena proyeksi perspektif menggunakan viewing volume seperti piramida terpotong bagian atasnya (frustum). Sintaks perintah gluPerspective(GLdouble fovy, GLdouble aspect, GLdouble zNear, GLdouble zFar) digunakan untuk menciptakan matriks pandangan perspektif dan mengalikan dengan matriks terakhir pada stack. Argumen fovy adalah sudut dari bidang pandang dari pada bidang x-z yang nilainya berkisar antara (0.0, 180.0). Argumen aspect adalah rasio sebagai hasil dari pembagian weight dan height pada frustum. Argumen zNear dan zFar adalah jarak antara titik pandang dan clipping plane sepanjang sumbu z negatif dan nilainya selalu positif.

Proyeksi orthographic adalah proyeksi yang menampilkan objek ke layar tanpa mengubah ukuran dari objek yang sesungguhnya yang akan ditampilkan. Proyeksi ini sering dipakai dalam desain arsitek dan CAD. Sintaks perintah glOrtho(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, GLdouble near, GLdouble far) digunakan untuk menghasilkan matriks orthographic viewing volume dan mengalikannya dengan matriks terakhir. Near clipping plane adalah segiempat dengan

sudut kiri bawah yang ditentukan oleh (left, bottom, -near) dan sudut kanan atas yang ditentukan oleh (right, top, -near).

`Transformasi Viewport`

Transformasi viewport menspesifikasikan daerah untuk gambar yang akan menempati layar. Viewport sendiri artinya bidang segiempat pada windows dimana image digambar. Transformasi perspektif dan viewport mengatur bagaimana ruang pandang terhadap objek dipetakan di layar komputer. Jadi cara kerja transformasi ini sama dengan proses akhir fotografi yaitu mendefinisikan ukuran dan lokasi objek pada gambar yang ditampilkan pada layar. Sintaks perintah glViewport(GLint x, GLint y, GLsizei width, GLsizei height) digunakan untuk mendefinisikan segiempat *pixel* dari *windows* dimana gambar terakhir telah dipetakan. Parameter (x,y) menspesifikasikan sudut kiri bawah dari *viewport* yang telah diinisialisasi (0,0), sedangkan *width* dan *height* adalah ukuran dari segiempat *viewport*, biasanya adalah lebar dan tinggi *windows*.

11.3.10 Pewarnaan pada OpenGL

OpenGL menspesifikasikan warna sebagai gabungan intensitas komponen merah, hijau, dan biru. Berdasarkan ini maka dapat dibentuk suatu ruang warna *RGB* yang merupakan kombinasi warna yang dapat digunakan.

11.3.11 Pencahayaan

Metode pencahayaan pada OpenGL terdiri dari tiga komponen cahaya yaitu *ambient, diffuse* dan *specular*. Cahaya *ambient* adalah cahaya yang datang ke permukaan objek dari sembarang arah sebagai akibat dari pantulan ruangan di sekeliling objek sehingga tidak mempunyai arah datang yang khusus. Cahaya *diffuse* adalah cahaya yang datang dari arah tertentu tetapi dipantulkan ke segala arah oleh permukaan objek. Jika cahaya diarahkan langsung tanpa melalui sudut tertentu maka objek pada sisi dimana sinar menyorot akan tampak lebih terang dari sisi yang lain. Cahaya *specular* adalah cahaya yang datang dari arah tertentu dan terpantul ke arah tertentu pula.