**BAB 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.

Jadi, output primitif adalah paket pemrograman grafika dilengkapi dengan fungsi untuk menyatakan scene dalam bentuk struktur dasar geometri.

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. Kemudian peralatan output membuat garis sesuai posisi titik-titik tersebut. Untuk peralatan analog seperti plotter dan random-scan display garis lurus dapat dihasilkan dengan halus. Pada peralatan digital garis lurus dihasilkan dengan menetapkan titik diskrit antara titik awal dan akhir. Posisi titik diskrit sepanjang garis lurus data diperhitungkan dari persamaan garis tersebut. Untuk menentukan nilai suatu titik, dapat digunakan prosedur dasar dimana x sebagai nilai kolom pixel dan y sebagai nilai scan line sebagai berikut :

**setPixel(x,y)**

Bila nilai x dan y sudah tersimpan pada frame buffer untuk dapat menampilkannya pada layer menggunakan fungsi dasar

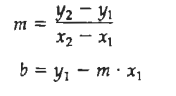
**getPixel(x,y)**

1. **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 (x1, y1) dan (x2, y2). Kita dapat menentukan nilai kemiringan m dan y memotong b dengan perhitungan berikut:

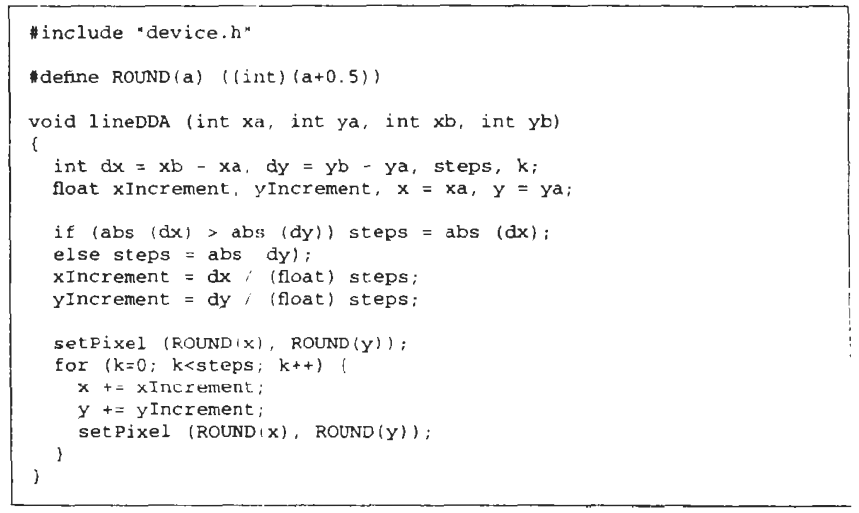


Untuk penambahan x sepanjang garis yaitu dx akan mendapatkan penambahan y sebesar :

**y = m.∆x**

**Algoritma DDA**

Algoritma Digital Differential Analyzer (DDA) adalah algoritma pembentukan garis berdasarkan perhitungan dx maupun dy dengan menggunakan rumus dy = m.dx. 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 (x0 , y0 ) dan titik akhir (x1 , y1).
3. Hitung x = x1 - x0 dan y = y1 - y0 .
4. Tentukan step, yaitu jarak maksimum jumlah penambahan nilai x maupun nilai y dengan cara :

* bila nilai | y| > | x| maka step = nilai | y|.
* bila tidak maka step = | x|.

1. Hitung penambahan koordinat pixel yaitu x\_increment = x / step dan y\_increment = y / step.
2. Koordinat selanjutnya (x+x\_incerement, y+y\_increment).
3. Posisi pixel pada layer ditentukan dengan pembulatan nilai koordinasi tersebut.
4. Ulangi step 6 dan 7 untuk menentukan posisi pixel selanjutnya, sampai x = x1 dan y = y1 .

**Algoritma Bresenham**

Algoritma Bresenham merupakan algoritma penggambaran garis yang akurat dan efisien dengan menggunakan penambahan perhitungan yang dapat disesuaikan untuk menampilkan lingkaran dan kurva lainnya.

Langkah-langkah untuk membentuk garis menurut algoritma ini adalah :

1. Tentukan dua titik yang akan dihubungkan dalam pembentukan garis.
2. Tentukan salah satu titik di sebelah kiri sebagai titik awal (x0, y0 ) dan titik lainnya sebagai titik akhir (x1 , y1 ).
3. Hitung x, y, 2x, dan 2y – 2x.
4. Hitung parameter p0 = 2y – x.
5. Untuk setiap xk sepanjang jalur garis, dimulai dengan k = 0

* bila pk<0 maka titik selanjutnya (xk+1, yk ) dan pk+1 = pk+2y
* bila tidak maka titik selanjutnya adalah (xk+1, yk+1) dan

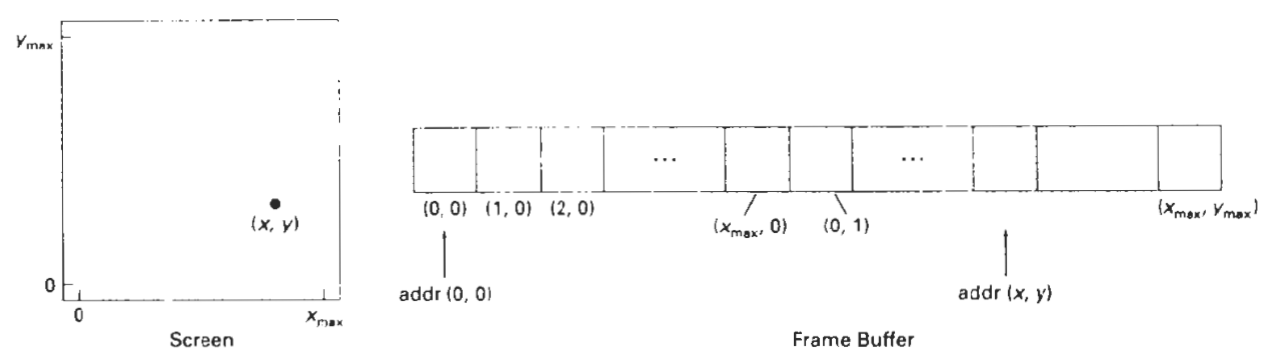
pk+1 = pk+2y–2x.

1. Ulangi langkah nomor 5 untuk menentukan posisi pixel selanjutnya, sampai x=x1 dan y=yk.

**Algoritma Garis Paralel**

1. **Loading The Frame Buffer**

Ketika segmen garis lurus dan objek lainnya memindai dikonversi untuk tampilan dengan sistem raster, posisi frame-buffer harus dihitung. Kami telah mengasumsikan bahwa ini dicapai dengan prosedur setpixel, yang menyimpan nilai di-sitas untuk piksel pada alamat yang sesuai dalam array frame-buffer. Algoritma pemindaian-konversi menghasilkan posisi piksel pada interval unit berturut-turut. Hal ini memungkinkan kita untuk menggunakan metode inkremental untuk menghitung alamat frame-buffer

*Gambar 3.2*

Sebagai contoh spesifik, misalkan array frame-buffer ditujukan dalam urutan utama dan posisi pixel itu bervariasi dari (0.0) di pojok kiri bawah layar ke (x,,) y,,, di pojok kanan atas (Gbr. 3-2). Untuk sistem bilevel (1 bit per piksel), alamat bit frame-buffer untuk posisi piksel (x, y) dihitung sebagai

addr(*x, y*) = addr(0, 0) + *y*(*xmax*+ 1) + *x* (3-1)

Bergerak melintasi garis scan, kita dapat menghitung alamat frame-buffer untuk piksel di (*X* + 1, *y*) sebagai offset berikut dari alamat untuk posisi (*x*, *y*):

addr(*x +* 1,*y*) = addr(*x*, *y)* + 1 (3-2)

Melangkah secara diagonal hingga garis scan berikutnya dari (x, y), kita sampai ke alamat frame-buffer (x + 1, y + 1) dengan perhitungan

addr(*x* + 1, *y* + 1) = addr(*x*, *y*) + *xmax*+ 2 (3-3)

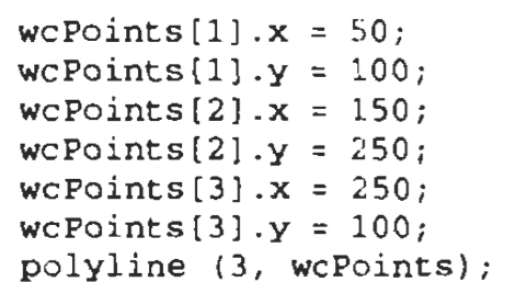
Dimana konstan xmax  + 2 yang sudah di komputasi terlebih dahulu untuk semua segmen garis. Perhitungan inkremental serupa dapat diperoleh dari rumus (3-1) untuk langkah unit dalam arah layar negatif x dan y. Setiap penghitungan alamat ini hanya melibatkan satu tambahan bilangan bulat.

Metode-metode untuk mengimplementasikan prosedur setPixel untuk menyimpan nilai intensitas pixel tergantung dengan kapabilitas dari sistem tertentu dan dengan persyaratan desain software package.

1. **Line Function**

Sebuah prosedur untuk menspesifikasikan segmen garis lurus dapat diatur dalam beberapa bentuk yang berbeda. Dalam PHIGS, GKS dan beberapa package lainnya, fungsi garis dua dimensional adalah

dimana parameter n diberikan nilai integer sama dengan jumlah posisi koordinat yang akan dimasukkan, dan wcPoints adalah array inputan dari nilai world coordinate untuk titik akhir dari segmen garis. Fungsi ini digunakan untuk menentukan satu set segmen garis lurus yang terhubung n-1. Karena serangkaian segmen garis tersambung terjadi lebih sering daripada segmen garis terisolasi dalam pengaplikasian grafis, polyline menyediakan fungsi baris yang lebih umum. Untuk menampilkan satu segmen straight-line, Donald Hearn dan M. Pauline Baker menetapkan n = 2 dan mendaftar nilai *x* dan *y* dari dua koordinat titik akhir di wcPoints.

Sebagai contoh penggunaan polyline, pernyataan berikut menghasilkan dua segmen garis terhubung, dengan titik akhir di (50, 103, (150, 250), dan (250, 100):

Referensi koordinat dalam fungsi polyline dinyatakan sebagai nilai koordinat absolut. Ini berarti nilai yang ditentukan adalah posisi titik aktual dalam sistem koordinat yang digunakan.

Pengimplementasian prosedur polyline dicapai dengan pertama kali melakukan rangkaian transformasi koordinat, kemudian membuat urutan panggilan ke device-level line-drawing.

Dalam PHIGS, input garis akhir sebenarnya ditentukan dalam pemodelan koordinat, yang kemudian dikonversi ke koordinat dunia (world coordinate). Selanjutnya, koordinat dunia dikonversi menjadi koordinat normalisasi, kemudian ke koordinat perangkat.

1. **Algoritma Pembentuk Lingkaran**

Lingkaran merupakan objek grafik yang paling sering digunakan pada grafik sederhana. Lingkaran dapat didefinisikan sebagai kumpulan titik yang memiliki jarak r dari posisi pusat (xc ,yc). Persamaan lingkaran dengan titik pusat (xc ,yc) dan radius r dapat dispesifikasikan menggunakan koordinat rectangular berikut

**+=**

Lingkaran juga dapat didefinisikan menggunakan koordinat polar. Lingkaran yang sama dapat didefinisikan sebagai berikut :

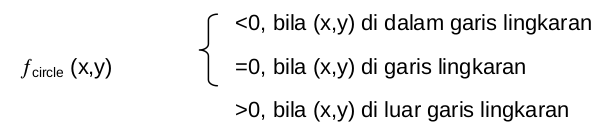
x = r cos θ +

y = r sin θ +

dimana 0 ≤ θ ≤ 2π

**Algoritma Lingkaran Midpoint**

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 dari suatu lingkaran, yaitu oktan kedua dari x = 0 ke x = R/ 2, dan menggunakan prosedur circle point untuk menampilkan titik dari seluruh lingkaran.



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 (xc ,yc) kemudian diperoleh (xc,yc) (= 0,r).
2. Hitung nilai dari parameter P0 = 5/4 – r1-r
3. Tentukan nilai awal k = 0, untuk setiap posisi x k berlaku sbb :

* Bila pk<0, maka titik selanjutnya adalah (xk +1,yk )

Pk+1 = Pk+1

* Bila pk>0, maka titik selanjutnya adalah (xk +1,yk -1)

Pk+1 = pk+2xk+1+1-2y k+1

Dimana 2xk+1 = 2xk+2 dan 2y k+1 = 2y k – 2

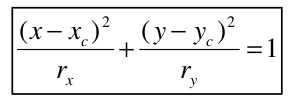
1. Tentukan titik simetris pada ketujuh oktan yang lain.
2. Gerakkan setiap posisi pixel (x,y) pada garis melingkar dari lingkaran dengan titik pusat dan tentukan nilai koordinat : x= x + xc dan y = y + yc
3. Ulangi langkah ke 3 -5, sampai dengan x>=y.
4. **Algoritma Pembentuk Elips**

Elips adalah lingkaran memanjang. Karena itu, kurva elips bisa dihasilkan dengan memodifikasi prosedur menggambar lingkaran untuk memperhitungkan dimensi elips yang berbeda di sepanjang sumbu mayor dan minor.

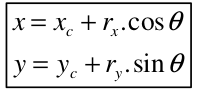
Elips didefinisikan sebagai himpunan titik sehingga jumlah jarak dari dua posisi yang ditentukan (fokus) adalah sama untuk semua titik. Jika jarak ke dua fokus dari titik P = (x, y) pada elips diberi label dan , maka persamaan umum dari sebuah elips dapat dinyatakan sebagai

**+ = constant**

Ellips merupakan salah satu objek grafis dengan persamaan koordinat rectangular sebagai berikut :



dan persamaan polar :



Teknik yang digunakan untuk menggambarkan garis dan lingkaran yang telah dibicarakan sebelumnya dapat diimplementasikan untuk menggambarkan ellips. Elips merupakan objek yang memiliki empat bagian yang simetris dari karakteristik ini, dapat disusun suatu algoritma yang memplot pixel di kuadran pertama dan menentukan titik di tiga kuadran lainnya.

1. **Other Curve**

Umumnya kurva yang ditemui meliputi fungsi kerucut, trigonometri dan eksponensial, distribusi probabilitas, polinomial umum, dan fungsi spline. Menampilkan kurva ini dapat dihasilkan dengan metode yang mirip dengan yang dibahas untuk fungsi lingkaran dan elips. Kita dapat memperoleh posisi di sepanjang jalur kurva secara langsung dari representasi eksplisit y = f (x) atau dari bentuk parametrik atau dapat menerapkan metode titik tengah tambahan untuk plot kurva yang dijelaskan dengan fungsi implisit f(x, y) = 1).

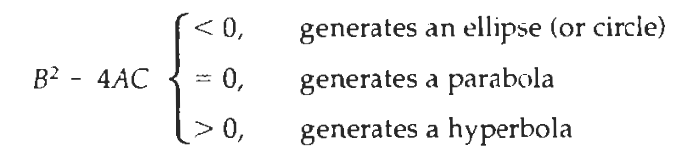
Metode sederhana untuk menampilkan fungsi kurva yang ditentukan adalah dengan proksi dengan segmen garis lurus. Representasi parametrik berguna dalam kasus ini untuk mendapatkan posisi titik akhir spaced line sama di sepanjang path kurva. Untuk dapat menghasilkan posisi dengan jarak yang sama dari representasi eksplisit dengan memilih variabel independen sesuai dengan kemiringan kurva.

Pendekatan garis lurus atau kurva digunakan untuk membuat grafik kumpulan data dari titik koordinat diskrit. Kita bisa menggabungkan titik-titik diskrit dengan segmen garis lurus, atau dengan menggunakan regresi linier (kuadrat terkecil) untuk memperkirakan data dengan satu garis lurus. Pendekatan nonlinier kuadrat digunakan untuk menampilkan kumpulan data dengan beberapa fungsi yang mendekati, biasanya polinomial. Banyak fungsi yang memiliki simetri yang bisa dieksploitasi untuk mengurangi perhitungan posisi koordinat sepanjang path kurva.

**Conic Section**

Secara umum, bagian kerucut (atau kerucut) dapat digambaran dengan persamaan: **A+ B+ Cxy + Dx + Ey + F = 0**

di mana nilai untuk parameter A, B, C, D, E, dan F menentukan jenis kurva yang untuk ditampilkan. Dengan set koefisien ini, kita dapat menentukan kerucut tertentu yang akan dihasilkan dengan mengevaluasi diskriminan:  **- 4AC**

****

Elips, hiperbola, dan parabola sangat berguna dalam aplikasi animasi tertentu. Kurva ini menggambarkan gerakan orbital dan lainnya untuk objek mengalami gaya gravitasi, elektromagnetik, atau nuklir. Persamaan eksplisit untuk lintasan parabola dari objek yang ditampilkan dapat ditulis sebagai



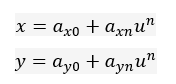
Metode yang digunakan dalam algoritma midpoint ellipse dapat langsung diterapkan dapatkan titik di sepanjang satu sisi sumbu simetri hiperbolik dan path parabola di dua wilayah: (1) di mana besarnya kemiringan kurva kurang dari

1, dan (2) di mana besarnya kemiringan lebih besar dari 1. Untuk melakukan ini, pertama-tama kita lakukan pilih bentuk Persamaan yang tepat yaitu 

dan kemudian gunakan fungsi yang dipilih untuk mengatur ekspresi untuk parameter keputusan di kedua wilayah.

**Polynomials and Spline Curves**

Merancang bentuk objek atau jalur gerak biasanya dilakukan dengan menentukan beberapa titik untuk menentukan kontur kurva umum, kemudian menyesuaikannya. Titik-titik yang dipilih dengan polinomial. Salah satu cara untuk mencapai kesesuaian kurva adalah dengan membuat bagian kurva polinomial kubik antara setiap pasangan titik yang ditentukan. Setiap bagian kurva kemudian digambarkan dalam bentuk parametrik sebagai:



di mana parameter u bervariasi selama interval 0 hingga 1. Nilai untuk koefisien u dalam persamaan parametrik ditentukan dari kondisi batas pada bagian kurva.

1. **Parallel Curve Algorithm**

Sebuah metode titik tengah paralel untuk menampilkan lingkaran adalah untuk membagi busur melingkar dari 900 ke 450 menjadi subarcs sama dan menetapkan prosesor yang terpisah untuk masing-masing subarc. Kita kemudian perlu menyiapkan computations untuk menentukan nilai awal y dan nilai parameter keputusan P*k* untuk setiap prosesor. Posisi pixel kemudian dihitung melalui setiap subarc dan posisi di oktan lingkaran lainnya kemudian diperoleh secara simetri.

Untuk tampilan elips atau kurva lainnya, kita cukup menggunakan metode partisi pemindaian-garis. Setiap prosesor menggunakan persamaan kurva untuk menemukan posisi persimpangan di sepanjang garis pemindaian yang ditetapkan. Dengan prosesor yang ditetapkan ke masing-masing piksel, setiap prosesor akan menghitung jarak (atau jarak yang dikuadratkan) dari kurva ke piksel yang ditetapkan. Jika jarak yang dihitung kurang dari nilai yang ditentukan, piksel diplot.

1. **Curve Function**

Gambar Primitif yang digeneralisasi kemudian akan merujuk algoritma yang sesuai, seperti metode titik tengah, untuk menghasilkan lingkaran. Dengan input interaktif, sebuah lingkaran dapat didefinisikan dengan dua titik koordinat: posisi tengah dan satu titik pada keliling.

Fungsi untuk menghasilkan lingkaran dan elips sering mencakup kemampuan menggambar bagian kurva dengan menentukan parameter untuk titik akhir garis. Metode lain untuk menunjuk busur lingkaran atau elips adalah dengan memasukkan posisi awal dan akhir koordinat lengkung.

1. **Pixel Addressing and object Geometric**

Deskripsi objek diberikan dalam kerangka referensi dunia, dipilih sesuai dengan aplikasi tertentu, dan koordinat dunia input pada akhirnya dikonversi ke posisi tampilan layar.

Koordinat piksel, namun, merujuk area layar terbatas. Jika kita ingin mempertahankan geometri objek dunia yang ditentukan, kita perlu mengkompensasi pemetaan titik input matematis ke area piksel yang terhingga.

Salah satu cara untuk melakukan ini adalah hanya menyesuaikan dimensi objek yang ditampilkan untuk memperhitungkan jumlah tumpang tindih area piksel dengan batas objek.

**Screen Grid Coordinates**

Alternatif untuk mengatasi posisi tampilan dalam hal pusat pixel adalah dengan referensi koordinat layar sehubungan dengan grid garis batas pixel horizontal dan vertikal yang berjarak satu unit terpisah.

Posisi koordinat layar kemudian adalah sepasang nilai integer yang mengidentifikasi posisi persimpangan kotak antara dua piksel.

Algoritma untuk menggambar garis dan membuat kurva yang dibahas pada bagian sebelumnya masih valid ketika diterapkan pada posisi input yang dinyatakan sebagai koordinat kisi layar. Parameter keputusan dalam algoritme ini sekarang hanyalah ukuran perbedaan pemisahan kisi layar, bukan perbedaan pemisahan dari pusat piksel.

**Maintaining Geometric: Properties of Displayed Objects**

Ketika kita mengonversi deskripsi geometris objek menjadi representasi piksel, kita mengubah titik dan garis matematika menjadi aras layar terbatas. Jika kita ingin mempertahankan pengukuran geometris asli yang ditentukan oleh koordinat input untuk suatu objek, kita perlu memperhitungkan ukuran piksel terbatas ketika kita mengubah definisi objek menjadi tampilan layar.

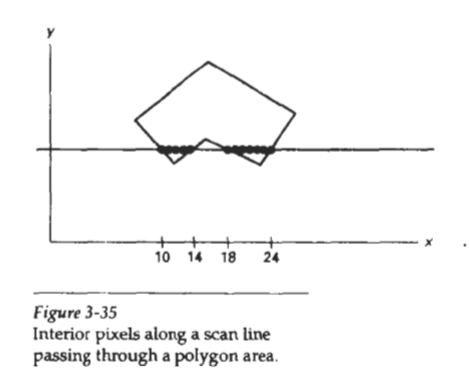
1. **Fill Area Primitif**

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.

**Scan Line Polygon Fill Algorithm**

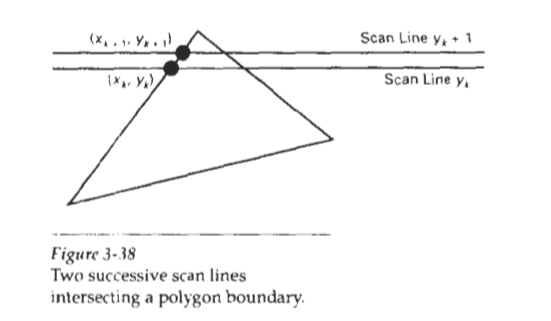
Pemberian warna pada polygon dilakukan dengan cara menscan secara horisontal dari kiri ke kanan. Hal ini dilakukan untuk mendapatkan titik potong dengan tepi polygon, kemudian mengurutkan nilai-nilai titik potong x dari kiri ke kanan dan memberi warna pada piksel-piksel diantara dua pasangan berurutan (x1-x2). Hal ini dilakukan dari garis scan yang paling bawah (nilai y terkecil) hingga garis scan yang paling atas. Metode ini bisa juga digunakan untuk pengisian warna pada obyek-obyek sederhana lainnya,misalnya lingkaran, ellip dan lain-lain.



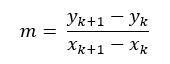
Gambar 3-35 mengilustrasikan prosedur pemindaian garis untuk Pengisian solid area poligon. Untuk setiap garis pindai yang melintasi poligon, algoritma area-fill menempatkan titik persimpangan garis pindaian dengan tepi poligon. Titik-titik persimpangan ini kemudian disortir dari kiri ke kanan, dan posisi frame-buffer yang sesuai antara setiap pasangan persimpangan diatur ke warna pengisian yang ditentukan. Dalam contoh Gambar 3-35, posisi persimpangan empat piksel dengan batas poligon menentukan dua rentangan piksel interior dari x = 10 ke x = 14 dan dari x = 18 ke x = 24.

Beberapa persimpangan garis pindai pada simpul poligon memerlukan penanganan khusus. Garis pindai yang melewati titik memotong dua sisi pada posisi itu, menambahkan dua titik ke daftar persimpangan untuk garis pindai.

Perhitungan yang dilakukan dalam pemindaian-konversi dan algoritma grafik lainnya biasanya mengambil keuntungan dari berbagai properti koherensi adegan yang akan ditampilkan. Yang kami maksud dengan koherensi adalah bahwa sifat-sifat satu bagian adegan terkait dalam beberapa hal dengan bagian adegan yang lain sehingga hubungan tersebut dapat digunakan untuk mengurangi pemrosesan. Metode koherensi sering melibatkan perhitungan inkremental yang diterapkan sepanjang garis pindai tunggal atau antara garis pindaian berturut-turut. Dalam menentukan persimpangan tepi, kita dapat mengatur perhitungan koordinat tambahan sepanjang tepi dengan mengeksploitasi fakta bahwa kemiringan tepi konstan dari satu garis pindai ke yang berikutnya.



Gambar 3-38 menunjukkan dua garis pindaian berturut-turut melintasi tepi kiri poligon. Kemiringan garis batas poligon ini dapat dinyatakan dalam istilah koordinat persimpangan garis pindai:



Karena perubahan koordinat y antara dua garis pemindaian sederhana:



**Inside-Outside Tests**

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

**Scan Line Fill of Curved Boundary Area**

Secara umum, Scan Line Fill of Curved Boundary Area membutuhkan lebih banyak pekerjaan daripada Scan Line Polygon Fill, karena perhitungan persimpangannya melibatkan batas-batas non linier. Untuk kurva sederhana seperti lingkaran atau elips, melakukan scan line fill nya adalah proses yang mudah. Kita hanya perlu menghitung dua scan line fill pada sisi yang berlawanan dari kurva. Simetri antara kuadran (dan antara oktan untuk lingkaran) digunakan untuk mengurangi perhitungan batas.

**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.

**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.

1. **Fill Area Function**

Unfuk menampilkan poligon terisi dalam PHlGS dan GKS dengan fungsi

**fillArea (n, wcvertices)**

Area poligon yang ditampilkan dibatasi oleh serangkaian n segmen garis lurus menghubungkan set posisi vertex yang ditentukan dalam wcvertices. Paket-paket ini tidak menyediakan fungsi isian untuk objek dengan batas melengkung.

Implementasi fungsi area Isi tergantung pada tipe yang dipilih isi interior. Kita dapat menampilkan batas poligon yang mengelilingi interior berlubang, atau kita dapat memilih warna solid atau isi pola tanpa batas untuk tampilan poligon. Untuk isian padat, fungsi Area isian diimplementasikan dengan garis pindai mengisi algoritma untuk menampilkan area warna tunggal. Berbagai opsi atribut untuk menampilkan area isian poligon di I'HlGS dibahas pada bab selanjutnya.

Primitif poligon lain yang tersedia di PHlGS adalah mengisi Area Set. Fungsi ini memungkinkan serangkaian poligon untuk ditampilkan dengan menentukan daftar, simpul untuk setiap poligon. Juga, dalam paket grafik lainnya, fungsi sering disediakan untuk menampilkan berbagai area isian yang umum digunakan selain poligon umum. Beberapa contoh adalah fillRectangle, fillCircle, fillCircleArc, fillEllipse, dan filLEllipseArc.

1. **Cell Array**

Array sel adalah primitif yang memungkinkan pengguna untuk menampilkan bentuk arbitrasi yang didefinisikan sebagai grid dua dimensi. Matriks nilai warna yang telah ditentukan sebelumnya adalah dipetakan oleh fungsi ini ke wilayah koordinat persegi panjang tertentu. Fungsi dari PHIGS ini adalah

**cellArray(wcPoint, n, m colorArray)**

dimana colorArray adalah matriks n by m dari nilai warna integer dan wcpoits daftar batas-batas wilayah koordinat persegi panjang: (, ) dan (, ) .

Nilai warna piksel ditetapkan berdasarkan posisi relatif koordinat pusat piksel Jika pusat piksel terletak di dalam salah satu dari n oleh m mengoordinasikan sel, piksel tersebut diberi warna elemen yang sesuai di matriks colorArray.

1. **Character Generation**

Huruf, angka, dan karakter lain dapat digeser dalam berbagai ukuran dan

gaya. Gaya desain keseluruhan untuk satu set (atau keluarga) karakter disebut font.

Tipografi (atau font) dapat dibagi menjadi dua kelompok besar: serif dan sans serif. Jenis serif memiliki garis kecil atau aksen di ujung karakter utama stroke, sedangkan tipe sans-serif tidak memiliki aksen. Misalnya, teks dalam ini

buku diatur dalam font serif (Palatino). Tetapi kalimat ini dicetak dalam font sans-serif (Optima). Jenis serif umumnya lebih mudah dibaca; yaitu, lebih mudah dibaca lebih lama blok teks. Di sisi lain, karakter individu dalam tipe sans-serif adalah lebih mudah untuk mengenali kembali. Karena alasan ini, tipe sans-serif dikatakan lebih mudah dibaca. Sejak karakter sans-serif dapat dengan cepat dikenali, jenis huruf ini baik untuk pelabelan dan judul pendek.

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.

**BAB 4**

**ATTRIBUT 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.

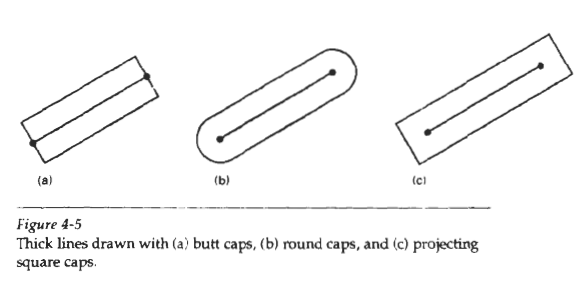
**4.1.1 Tipe Garis**

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.

**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.

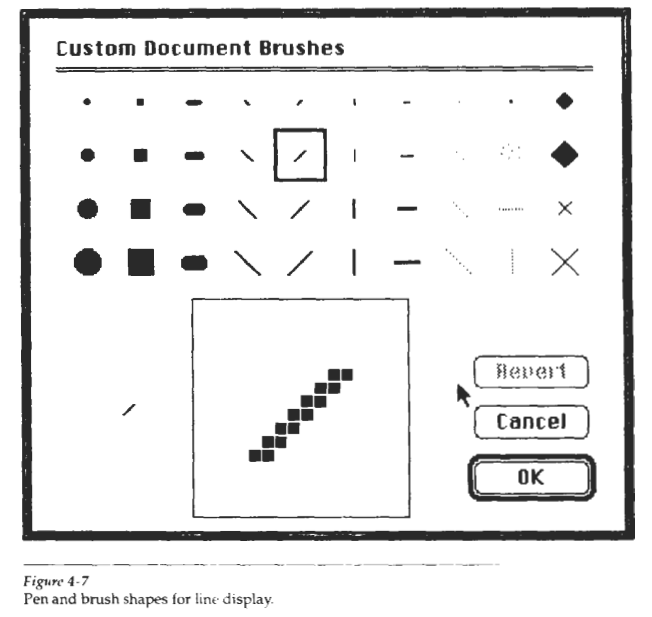
Pada implementasi raster, tebal garis standar diperoleh dengan menempatkan satu pixel pada tiap posisi, seperti algoritma Bressenham. Garis dengan ketebalan didapatkan dengan perkalian integer positif dari garis standar, dan menempatkan tambahan pixel pada posisi sejajar. Untuk garis dengan slope kurang 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.



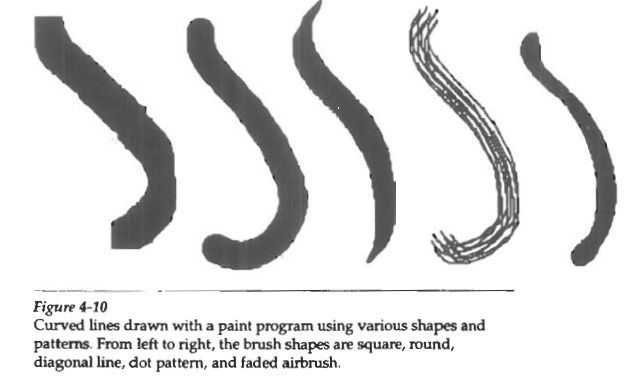
Membuat polyline tebal membutuhkan beberapa pertimbangan tambahan. Secara umum, metode yang kami pertimbangkan untuk menampilkan segmen garis tunggal tidak akan menghasilkan serangkaian segmen garis yang terhubung dengan lancar. Menampilkan garis tebal menggunakan bentang piksel horizontal dan vertikal, misalnya, meninggalkan celah piksel di batas antara garis lereng yang berbeda di mana ada pergeseran dari bentang horizontal ke bentang vertikal.

**4.1.3 Pilihan Pena dan Kuas**

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.

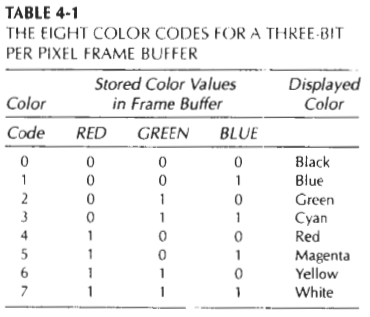
**4.2 Atribut Kurva**

Parameter untuk atribut kurva sama dengan atribut segmen garis. Kurva dapat ditampilkan dengan berbagai warna, tebal, dot-dash(Style garis) dan pilihan pen atau brush. Selain itu untuk pengisian suatu bidang tertentu termasuk memilih warna antara solid dan pattern tertentu dan memilih warna pattern yang ada.

**4.3.1 Tingkatan Warna dan Grayscale**

Dalam sebuah system raster warna, jumlah pilihan warna yang tersedia tergantung dengan jumlah tempat penyimpanan yang diberikan per piksel di frame buffer. Terdapat dua cara untuk menyimpan informasi warna di frame buffer :

1. Dengan menyimpan kode warna secara langsung di frame buffer
2. Dengan memasukkan kode warna di tabel yang berbeda dan menggunakan nilai piksel sebagai indeks untuk tabel tersebut

Untuk metode pertama, ketika sebuah kode warna ditentukan di dalam sebuah program, nilai biner yang sesuai ditempatkan dalam frame buffer untuk tiap komponen piksel dalam output primitif untuk ditampilkan dalam warna itu.

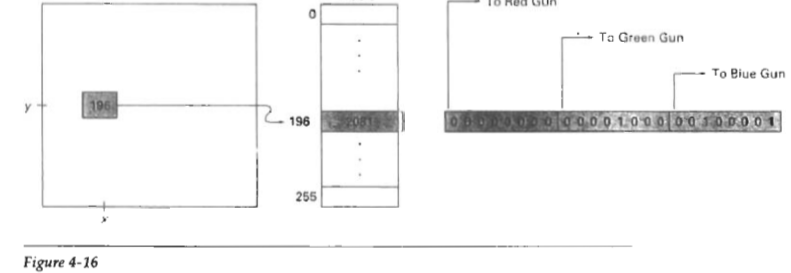
Seperti yang bisa dilihat di tabel 4-1, terdapat 3 bit untuk 3 warna, bit yang paling kiri untuk warna merah, bit yang tengah untuk warna hijau, dan bit kanan untuk warna biru. Menambahkan jumlah bit untuk tiap bit akan menghasilkan warna tertentu, dan tentunya menambah intensitas warna yang dihasilkan.

**4.3.2 Color Table**

Pada gambar 4-16, dapat dilihat skema untuk menyimpan nilai warna di dalam sebuah color lookup table (atau video lookup table), dimana nilai frame buffer sekarang digunakan sebagai indeks ke tabel warna. Sistem yang dapat membuat user memilih antara 256 warna untuk tampilan simultan dari palet hampir 17 juta warna, adalah sistem yang dapat mereferensikan tiap pikselnya ke 256 posisi tabel dan tiap masukan (entri) di tabel dapat menggunakan 24 bit untuk menentukan warna RGB.

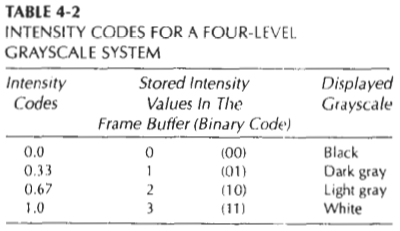
Pengguna dapat mengatur entri tabel warna dalam program aplikasi PHIGS dengan fungsi

**setColourRepresentation (ws, ci, colorptr)**

Parameter ws mengidentifikasi perangkat output workstation; parameter ci menentukan indeks warna, dan parameter colorptr yang merupakan nomor posisi tabel warna (0 hingga 255 untuk contoh pada Gambar 4-16); dan parameter colorptr menunjuk ke rona (hue) nilai warna RGB (r, g, b) yang masing-masing ditentukan dalam rentang dari 0 hingga 1.

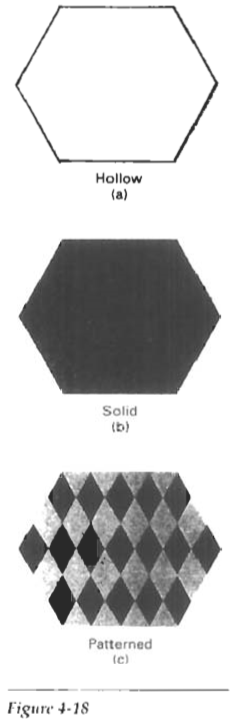
**4.3.3 Grayscale**

Jikalau menggunakan monitor tidak berwarna, fungsi warna tetap masih bisa digunakan di sebuah program aplikasi untuk me set warna abu-abu (grayscale). Nilai numerik di range 0-1 dapat digunakan untuk menentukan level/tingkatan grayscale, yang kemudian akan dikonversikan ke kode biner untuk disimpan di raster. Dapat dilihat di tabel 4-2 tentang intensitas yang dihasilkan dari dua kode biner.



**4. 4 Area-Fill Attributes**

**4.4.1 Fill Style**

Area ditampilkan dengan 3 fill styles dasar yaitu hollow(hampa/kosong) dengan warna di tepiannya, solid dan dengan pattern. Fill style dasar dapat dipilih di program PHIGS dengan fungsi:

**setInteriorStyle (fs)**

Area hollow ditampilkan menggunakan outline di sekitar objek dengan warna interior sama dengan warna background. Fill solid ditampilkan dalam satu warna hingga bagian tepi dari area, untuk memilih warna untuk solid interior atau tepian hollow dapat menggunakan

**setInteriorColourIndex (fc)**

Yang mana fc adalah kode warna.

**4.4.2 Pattern Fill**

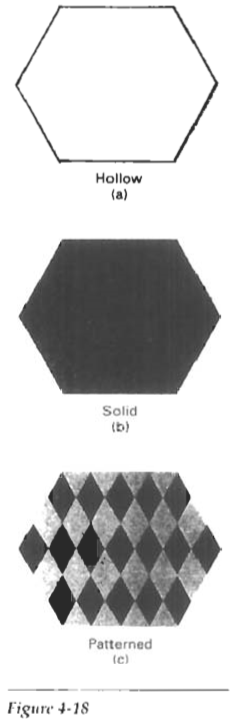
Untuk menggunakan Pattern dapat digunakan

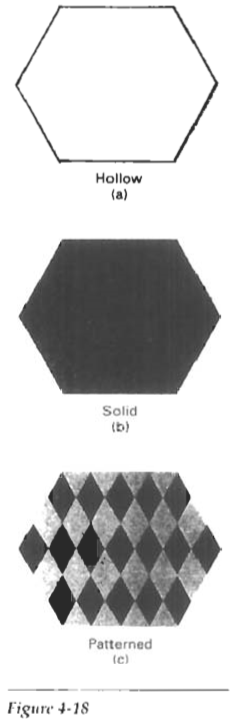
**setInteriorStyleIndex (pi)**

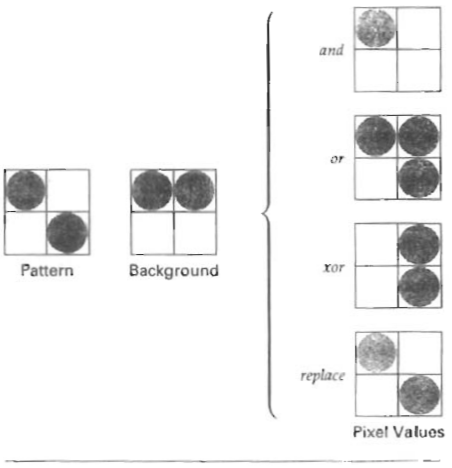
Parameter pi digunakan untuk menentukan posisi tabel. Jika tabel digunakan untuk pattern hatch, dan jika kita sudah memilih hatch fill untuk style interior di segmen program ini, maka nilai yang ditetapkan ke parameter pi adalah indeks untuk pola yang disimpan dalam tabel penetasan.

Untuk pattern style fill, entri tabel dapat dibuat pada perangkat output individual dengan set Pattern Representation

(**ws, p\_, nx, ny, cp)**

Fill pattern dapat dikombinasikan dengan pattern dari background menggunakan operasi Boolean and, or dan xor





**4.4.3 Soft Fill**

Soft fill adalah adalah sebuah prosedur yang mana boundary-fill dan flood-fill digunakan untuk mewarnai ulang area supaya warna fill terkombinasi dengan warna background. Salah satu kegunaan metode fill ini adalah untuk melembutkan warna fill pada tepian objek yang sudah menyaru ke antialias edge(tepi). Kegunaan lainnya untuk mewarnai ulang area warna yang yang tadinya diisi dengan brush semitransparent.

**4.5 Character Attributes**

Tampilan karakter yang ditampilkan dikendalikan oleh atribut seperti font, ukuran, warna, dan orientasi. Atribut dapat diatur untuk seluruh string karakter (teks) dan untuk karakter individu yang didefinisikan sebagai simbol penanda.

**4.5.1 Text Attributes**

Teks memiliki beberapa atribut seperti font dan style, beberapa contoh font seperti New York, Courier, Helvetica, Times New Roman dan lainnya. Sedangkan untuk style font ada **Bold** dan *Italics*. Font tertentu dan gaya terkait dipilih dalam program PHlGS dengan mengatur kode integer untuk parameter font teks t f dalam fungsi

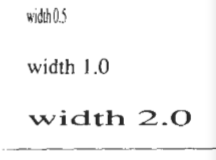
**setTextFont (t f)**

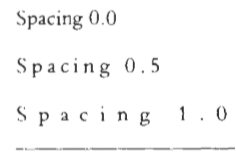
Untuk merubah warna teks dapat menggunakan

**setTextColourIndex (tc)**

Untuk merubah ukuran teks menggunakan

**setCharacterHeight (ch)**

Lalu kalau ingin melebarkan ukuran teks dapat menggunakan

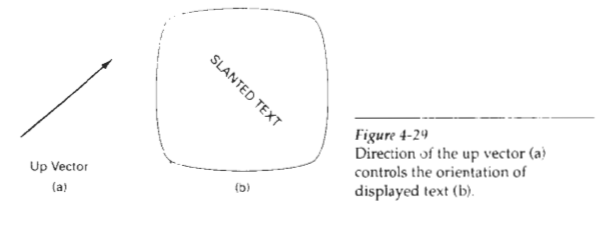
**setCharacterExpansionFactor (cw)**

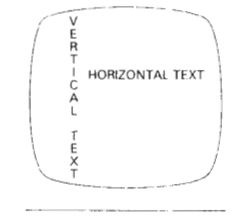
Untuk spacing antar huruf dapat diatur dengan

**setCharacterSpacing (cs)**

Orientasi karakter string yang ditampilkan dapat diatur sesuai dengan arah karakter up vector dengan

**setCharacterUpVector (upvect)**

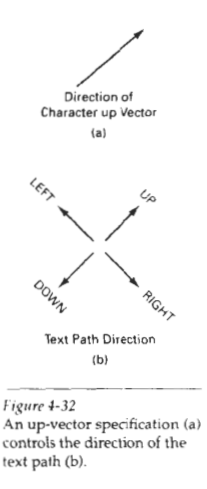
Lalu kita juga bisa merubah arah teks menjadi horizontal atau vertikal dengan

**setTextPath (tp)**

Atribut karakter lainnya adalah alignment, atribut ini menunjukkan bagaimana teks diposisikan menggunakan

**setTextAlignment (h, v)**

**4.5.2 Marker Attributes**

Simbol marker adalah karakter tunggal yang dapat ditampilkan dalam warna berbeda dan dalam berbagai ukuran. Atribut marker diterapkan oleh prosedur yang memuat karakter yang dipilih ke dalam raster pada posisi yang ditentukan dengan warna dan ukuran yang ditentukan.

Untuk memilih karakter tertentu sebagai marker symbol dapat menggunakan

**setMarkerType (mt)**

di mana parameter jenis penanda mt diatur ke kode integer. Kode umum untuk jenis marker adalah bilangan bulat 1 hingga 5, yang masing-masing menentukan titik (.), tanda silang vertikal (+), tanda bintang (\*), lingkaran (o) dan tanda silang diagonal (x). Jenis marker dipusatkan pada koordinat marker.

Untuk mengubah ukuran marker dapat menggunakan

**setmarkerSizeScaleFactor (ms)**

Dan untuk merubah warna marker menggunakan

**setPolyMarkerColourIndex (mc)**

**4.6 Bundled Attributes**

**4.6.1 Bundled Line Attributes**

Entri dalam tabel bundel untuk atribut garis di workstation diatur dengan fungsi

**setPolylineRepresentation (ws,li, It ,lw, l c)**

Parameter ws adalah pengidentifikasi workstation, dan parameter indeks garis mendefinisikan posisi tabel bundel. Parameter It, lw, dan lc kemudian di bundel dan dihubungkan nilai untuk mengatur jenis garis, lebar garis, dan spesifikasi warna garis, masing-masing untuk indeks tabel yang ditunjuk.

Sekelompok atribut line bundled dipilih untuk setiap workstation dengan menentukan nilai indeks tabel:

**setPolylineIndex(li)**

Perintah tersebut menentukan pada posisi tabel yang ditentukan oleh nilai parameter indeks baris li.

**4.6.2 Bundled Area-Fill Attributes**

Tabel entri untuk atribut area-fill diatur dengan

**setInteriorRepresentation (ws, fi, fs, pi, fc)**

yang mendefinisikan daftar atribut yang sesuai untuk mengisi indeks pada workstation ws. Parameter f s, pi, dan fc adalah nilai yang ditetapkan untuk fill-style, pattern index, dan fill color masing-masing pada workstation yg ditunjuk. Tabel bundle serupa juga dapat diatur untuk atribut tepi fill area poligon.

Bundle atribut tertentu kemudian dipilih dari tabel dengan fungsi

**setInteriorIndex (fi)**

**4.6.3 Bundled Text Attributes**

Fungsi

**setTextRepresentation (ws, ti, tf, tp ,te, ts, tc)**

Nilai bundle untuk font text, precision, faktor ekspansi, ukuran dan warna di posisi tabel untuk workstation ws yang ditentukan oleh nilai yang ditetapkan ke parameter indeks teks ti. Atribut lain seperti up vector character, text path, tinggi karakter dan text alignment diatur secara individu

**4.6.4 Bundled Marker Attributes**

Entri tabel untuk atribut marker yang dibundel diatur dengan

**setPolymarkerRepresentation (ws, mi, mt, ms, mc)**

Ini mendefinisikan jenis marker, faktor skala marker, dan warna marker untuk indeks mi pada workstation ws. Pilihan tabel bundel kemudian dibuat dengan fungsi

**setPolymarkerIndex (mi)**

**4.7 Inquiry Function**

Pengaturan saat ini untuk atribut dan parameter lainnya, seperti jenis dan status workstation, dalam daftar sistem dapat diambil dengan fungsi inquiry. Fungsi-fungsi ini memungkinkan nilai saat ini disalin ke dalam parameter yang ditentukan, yang kemudian dapat disimpan untuk digunakan kembali nanti atau digunakan untuk memeriksa kondisi sistem saat ini jika terjadi kesalahan.

Untuk memeriksa nilai atribut saat ini dengan menyebutkan nama atribut di fungsi penyelidikan. Misalnya saja fungsinya

**inquirePolyLineIndex (lastli)**

dan

**inquireInteriorColourIndex (lastfc)**

Salin nilai saat ini untuk indeks garis dan isi warna ke parameter lastli dan lastfc. Segmen program berikut akan menyarankan untuk menggunakan kembali nilai tipe baris saat ini setelah serangkaian garis digambar dengan tipe baris baru

**inquireLinetype (oldlt);**

**setLinetype (newlt);**

**setLinetype (odllt);**

**4.8 Antialiasing**

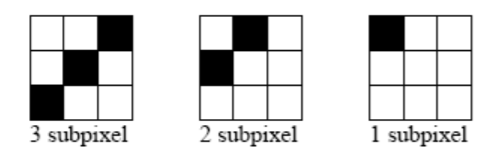
Antialiasing adalah teknik mengurangi artifak distorsi dalam merepresentasikan gambar resolusi tinggi pada resolusi yang lebih rendah. Artifak distorsi disebut aliasing. Antialiasing hanya berpengaruh pada garis di sekeliling objek. Biasanya memiliki pilihan pengaturan seperti MSAA (Multisample Anti-Aliasing) dan SSAA (Super Sample Anti-Aliasing, dikenal juga sebagai FSAA atau Full Screen Anti-Aliasing).

**4.8.1 Supersampling Straight Line Segments**

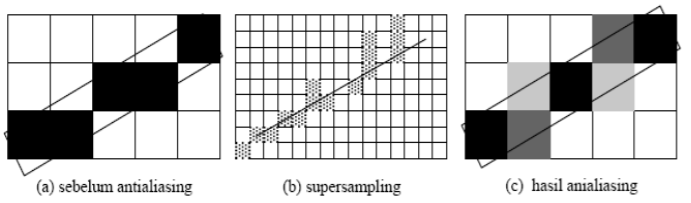
Secara lojik metoda ini "memperhalus" ukuran pixel ke dalam subpixel-subpixel dan "menggambarkan" garis pada grid subpixel tersebut. Lalu harga intensitas suatu pixel ditentukan sesuai dengan berapa banyak subpixelnya dikenai "garis" tersebut. Relasi: intensitas pixel ~ jumlah subpixel pada garis.

Ada dua cara penghitungan relasi tersebut :

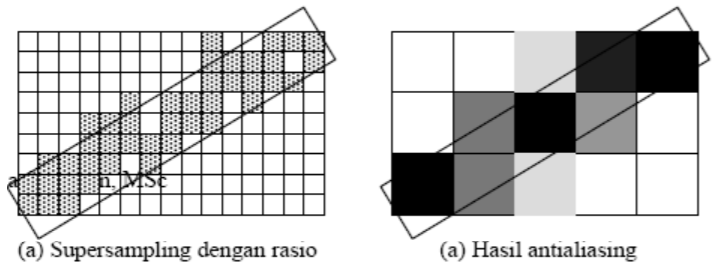
* Menganggap garis adalah garis dengan ketebalan infinitesimal 0 (hanya garis lojik). Untuk subsampling 3x3 ada 4 kemungkinan tingkatan: 3 subpixel, 2 subpixel, 1 subpixel, dan tidak ada. Pemberian intensitas sesuai dengan keempat tingkat tersebut.

****

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



**4.8.2 Pixel-Weighting Masks**

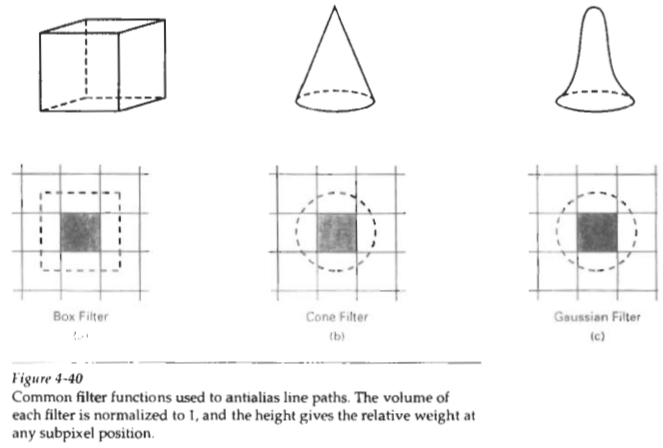
Alternatif menggunakan rasio secara langsung di atas, teknik fitering dalam pengolahan citra (bedanya: pengolahan citra pada pixel sedangkan di sini pada subpixel) dengan suatu mask (atau kernel) sesuai dengan subdivision pixel misalnya 3x3 subpixel digunakan untuk menghitung. Ada beberapa bentuk mask. Contohnya: - box mask (berefek averaging) - gaussian mask Kadang-kadang mask meliputi juga subpixel di pixel tetangganya untuk mendapatkan hasil yang lebih smooth.

**4.8.3 Area Sampling Straight Line Segments**

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. Metode 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.

**4.8.4 Filtering Techniques**

Metode yang lebih akurat untuk antialiasing garis adalah menggunakan filtering techniques(teknik filtering). Mirip dengan menerapkan weighted pixel mask, tapi sekarang kita bayangkan permukaan pembobotan terus menerus, (atau fungsi filter) yang menutupi pixel. Gambar 4-40 menunjukan contoh fungsi filter dari persegi, kerucut dan Gaussian



**4.8.5 Pixel Phasing**

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

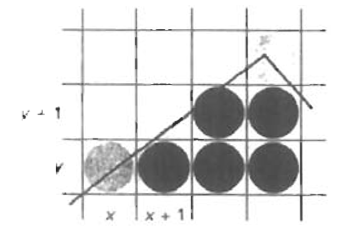
**4.8.6 Compensating for Line Intensity Differences**

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 horisontal/diagonal. Jadi secara visual efek ini dapat juga dikurangi dengan menaikkan intensitas garis yang mengarah diagonal sesuai dengan sudut dan mencapai maksimum pada 45 derajat dengan faktor akar 2 dari garis horisontal/vertikal.

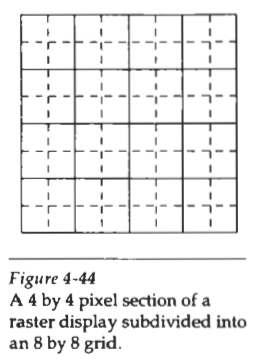
**4.8.7 Antialising Area Boundaries**

Antialiasing adalah Konsep untuk menghilangkan batas bergerigi yang ada pada suatu tampilan. Untuk melakukan Antialiasing, dapat melakukan Prosedur Algoritma Scan-Line untuk menghaluskan area yang d

Jika sebuah system mampu memposisikan ulang piksel, area boundaries dapat dihaluskan dengan menyesuaikan posisi batas pixel sehingga mereka sepanjang garis mendefinisikan area boundary. Di tabel berikut ini, piksel di posisi (*x, y*) memiliki ukuran setengah dari area yang ada di dalam polygon boundary. Oleh karena itu intensitas di posisi tersebut akan disesuaikan hingga setengah dari nilai yang ada



Pada posisi berikutnya(*x* + 1, *y* + 1) sepanjang boundary, intensitas disesuaikan hingga sekitar sepertiga dari nilai yang ditetapkan untuk titik itu.



Metode Supersampling dapat diterapkan dengan membagi area total dan menentukan jumlah sub piksel di dalam batas area. Partisi piksel menjadi empat subarea ditunjukkan pada Gambar 4-44. Kisi asli 4 x 4 piksel diubah menjadi kisi 8 x 8, dan memproses delapan garis pidai di kisi ini alih-alih empat.

Metode lain untuk menentukan persentase area piksel dalam batas, yang dikembangkan oleh Pitteway dan Watkinson, didasarkan pada algoritma garis tengah.