Topik: Primitif-Primitif Keluaran Grafika Raster

Pada sistem grafika random-scan primitif-primitif grafika dibuat langsung oleh display driver pada peranti peraga. Bila berupa plotter, maka parameter garis, misalnya kedua koordinat titik ujungnya, diterjemahkan menjadi kecepatan dan arah gerakan mekanis pena plotter horisontal dan vertikal. Begitu juga bila berupa random-scan CRT, kecuali gerakan tersebut berupa perubahan pada deflektor. Pada sistem raster-scan perlu ada satu tahap lagi untuk memetakan primitif tersebut pada suatu "matriks" pixel. Tahap ini dikenal sebagai proses scan-conversion. Dalam pembahasan pembuatan primitif keluaran disini selanjutnya secara default algoritma-algoritma adalah untuk sistem peragaan raster-scan.

Jadi pertanyaan kita sekarang adalah bagaimana memberi harga elemen-elemen matriks tersebut sehingga menampakkannya membentuk primitif-primitif yang kita harapkan.

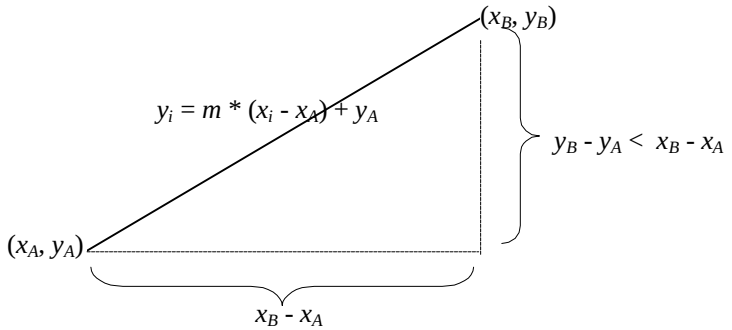
Pengertian primitif Grafika
Pada berbagai sistem grafika tingkat primitivitas grafis berbeda-beda. Beberapa sistem lanjut penggambaran shaded poligon telah dilakukan oleh perangkat kerasnya sendiri sebagai satu perintah tingkat mesin. Untuk kepentingan pemahaman grafika komputer sepenuhnya dalam kuliah ini perlu dipahami bagaimana kompleksitas dari konversi-konversi raster-scan tingkat ini sehingga terbentuk obyek primitif titik, garis, lingkaran, dsb. pada "matriks peragaan" sistem grafika kita.

Kriteria Algoritma
Suatu penggambaran grafika komputer bisa jadi terdiri dari ribuan bahkan jutaan primitif-primitif grafika ini. Jadi dengan sendirinya adanya peningkatan efisiensi dalam pembuatan setiap primitif tersebut maka secara proporsional akan mereduksi waktu komputasi keseluruhan penggambaran. Efisiensi ini pada umumnya dilakukan dengan sedapat mungkin

- mengurangi penggunaan operasi aritmetik perkalian/penjumlahan,
- Mengurangi penggunaan komputasi floating-point,
- memanfaatkan koherensi komputasi sebelumnya secara inkremental/dekremental,
- dan pemodelan aljabar yang lebih langsung.

Primitif Garis

Untuk menyederhanakan pembahasan maka kita batasi dahulu pada penggambaran diruang sub-kuadran (0, π/4). Dalam ruang ini maka $x_A < x_B$, dan gradien $m = (y_B - y_A)/(x_B - x_A)$ berharga $0 < m < 1$. Untuk kasus-kasus lain bisa secara umum diterangkan dengan cara yang mirip. Dalam sub-kuadran ini suatu garis adalah kumpulan titik antara koordinat (x_A, y_A) dan (x_B, y_B) yaitu (x_i, y_i), yang memenuhi $y_i = m * (x_i - x_A) + y_A$ untuk semua $x_A \le x_i \le x_B$. Dari sini bisa diketahui bahwa x_i berinkremen satu piksel demi satu piksel sementara yi berinkremen bilangan floating point antara 0.0 hingga 1.0.



Bagaimana halnya pada kasus raster yang mana x_i dan y_i harus berharga integer? Dalam hal ini y_i akan digantikan suatu harga integer yang palig mendekati y_i.

Secara naif kita dapat menghitung semua titik (x_i, y_i) berdasarkan persamaan garis di atas dengan mendekati yi mengguakan fungsi pembulatan round(yi).

```
m = (yB - yA)/(xB - xA);
```

```
for (xi = xA, xi <  xB; xi++) {
        yi = m * (xi-xA) + yA ;
        setpixel(xi, round(yi), v) ;
}
```

Fungsi round(xi) mengkomputasi secara floating point harga y_i dan memberikan hasil integer. Oleh sebab itu algoritma ini amatlah tidak efisien. Berikut ini kita modifikasialgoritma di atas dengan mengurangi perkalian dengan penjumlahan yang memanfaatkan koherensi inkremental.

Algoritma DDA

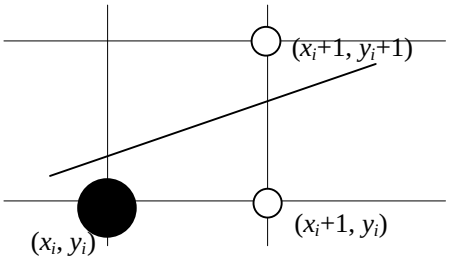
Mengganti operasi penghitungan y_i dalam bentuk fungsi dengan fungsi inkremental penjumlahan y_i dengan gradien *m*.

```
m = (yB - yA)/(xB - xA);
yi = yA ;
for (xi = xA; xi <= xB; xi++) {
        setpixel(xi, round(yi), v)
        xi++ ;
        yi += m ;
}
```

Algoritma DDA ini lebih baik dari algoritma naif di atas namun adanya penggunaan variabel floating point serta fungsi round(yi) menyebabkan ketidak efisienan algoritma ini.

Algoritma Midpoint Untuk Garis

Algoritma ini berusaha menghilangkan sama sekali operasi floating point dan fungsi round berdasarkan suatu fungsi keputusan (decision function) integer. Ide algoritma ini sebenarnya sudah muncul sebelumnya dalam algoritma klasik yaitu algoritma Bresenham untuk garis (1965). Namun, disini hanya dibahas metoda Midpoint karena konsepnya lebih umum untuk primitif-primitif grafika lainnya sementara algoritma Bresenham hanya untuk primitif-primitif tertentu saja. Sebelum membahas algoritmanya maka berikut ini akan dibahas penjabaran untuk menemukan fungsi keputusan tsb guna dapat kita kembangkan sendiri ke primitif-primitif grafika lainnya. Fungsi keputusan ini digunakan untuk menentukan “pemilihan” piksel berikutnya setelah piksel (x_i, y_i) di antara piksel $(x_i + 1, y_i)$ atau $(x_i + 1, y_i + 1)$. Untuk memudahkan selanjutnya masing-masing kita sebut piksel E dan piksel NE.



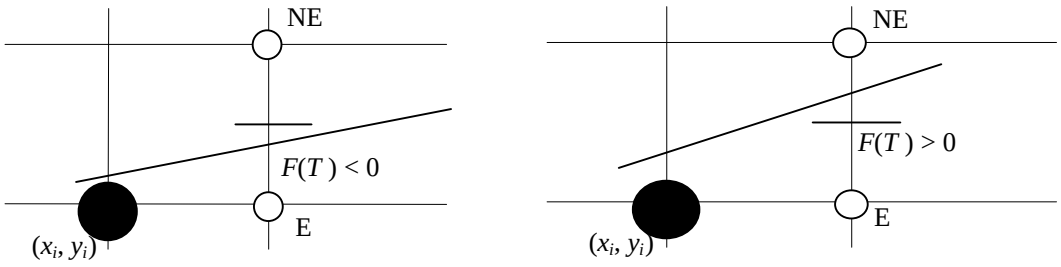
Masalah komputasi pencarian harga y_i direduksi menjadi masalah pemeriksaan tanda sebagai berikut untuk memilih titik mana yang akan "dipilih". Misalkan garis tsb. memiliki fungsi $y = dy/dx \,.\, x + C$, atau dapat ditulis kembali sebagai persamaan $dy \,.\, x - dx \,.\, y + C \,.\, dx = 0$, untuk semua titik (x, y) pada garis. Lalu kita definisikan fungsi keputusan $F(x, y)$ sbb.

$$F(x, y) = dy \,.\, x - dx \,.\, y + C \,.\, dx$$

Untuk (x, y) di atas garis maka fungsi ini berharga negatif dan sebaliknya untuk (x, y) di bawah garis berharga positif. Untuk memilih E atau NE maka digunakan koordinat tengah dari E dan NE yaitu $T = (x_i + 1, y_i + ½)$.

$$F(x_i + 1, y_i + ½) = dy \,.\, (x_i + 1) - dx \,.\, (y_i + ½) + C \,.\, dx$$

Jika $F(T) > 0$ yang berarti juga T berada di bawah garis, maka NE yang dipilih (garis lebih dekat ke NE daripada ke E) dan sebaliknya jika $F(T) < 0$ maka E yang dipilih. Bagaimana jika $F(T) = 0$? Kita boleh menentukan NE atau E asal konsisten (untuk pembahasan berikut kita pilih E).



Karena perhitungan F(T) tsb masih memerlukan operasi floating-point maka kita akan mencoba menemukan keherensinya dengan memeriksa kemungkinan-kemungkinan yang akan menjadi titik berikutnya. Namun tentu saja pemilihan ini bergantung pada hasil di atas.

Jika telah dipilih E maka kita memilih di antara $(x_i + 2, y_i)$ atau $(x_i + 2, y_i + 1)$ dengan titik pemeriksaan di T' = $(x_i + 2, y_i + ½)$. Dengan memasukkan dalam fungsi $F(\ )$ maka di peroleh

$$F(x_I + 2, y_i + ½) = dy \,.\, (x_i + 2) - dx \,.\, (y_i + ½) + C \,.\, dx$$

Selisih harga dua harga fungsi, $\Delta_E = F(T') - F(T)$ adalah

$$\begin{aligned}\Delta_E &= F(x_i + 2, y_i + ½) - F(x_i + 1, y_i + ½)\\ &= (dy \,.\, (x_i + 2) - dx \,.\, (y_i + ½) + B \,.\, dx) - (dy \,.\, (x_i + 1) - dx \,.\, (y_i + ½) + B \,.\, dx)\\ &= dy\end{aligned}$$

Sementara jika telah dipilih NE yang dipilih maka maka kita memilih di antara $(x_i + 2, y_I + 1)$ atau $(x_i + 2, y_i + 2)$ dengan titik pemeriksaan di T'' = $(x_i + 2, y_i + 1½)$. Kita akan dapatkan

$$F(x_i + 2, y_i + 1½) = dy \,.\, (x_i + 2) - dx \,.\, (y_i + 1½) + B \,.\, dx$$

Selisih harga dua fungsi, $\Delta_{NE} = F(T'') - F(T)$ adalah

$$\begin{aligned}\Delta_{NE} &= F(x_i + 2, y_i + 1½) - F(x_i + 1, y_i + ½)\\ &= (dy \,.\, (x_i + 2) - dx \,.\, (y_i + 1½) + B \,.\, dx) - (dy \,.\, (x_i + 1) - dx \,.\, (y_i + ½) + B \,.\, dx)\\ &= dy - dx\end{aligned}$$

Dengan Δ_E atau Δ_{NE} maka fungsi keputusan $F(x_i, y)$ dapat dihitung secara inkremental dari harga sebelumnya tanpa menggunakan fungsi asalnya yang dapat berisi faktor floating-point. Lalu berapa harga $F(x_i, y)$ yang pertama kalinya? Titik pertama adalah (x_A, y_A). Maka kita mulai hitung $F(\ )$ mulai pada saat memilih (x_A+1, y_A) atau (x_A+1, y_A+1) dengan titik tengah $(x_A+1, y_A + ½)$.

$$\begin{aligned}F(x_A+1, y_A + ½) &= dy \,.\, (x_A + 1) - dx \,.\, (y_A+½) + C \,.\, dx\\ &= dy \,.\, x_A - dx \,.\, y_A + C \,.\, dx + dy - dx/2\\ &= F(x_A, y_A) + dy - dx/2 = dy - dx/2\end{aligned}$$

Jadi secara inkremental kita hitung fungsi keputusan

- Setelah titik (x_A, y_A) maka $F = dy - dx/2$. Selanjutnya,
- Untuk $F > 0$ pilih NE (inkremen y_i) dan update $F = F + \Delta_{NE}$
- Untuk $F \leq 0$ pilih E (y_i tetap) dan update $F = \Delta_E$

Karena pada langkah pertama di atas F bisa berharga real karena adanya pembagi dengan 2 maka kita gunakan D = 2F sebagai fungsi keputusan yang bebas floating point namun dengan perubahan tanda yang sama yaitu

- Setelah titik (x_A, y_A) maka $D = 2dy - dx$. Selanjutnya,
- Untuk $D > 0$ pilih NE (inkremen y_i) dan update $D = D + 2\,\Delta_{NE}$
- Untuk $D \leq 0$ pilih E (y_i tetap) dan update $D = D + 2\,\Delta_E$

Jadi algoritmanya dalam bahasa C sbb.

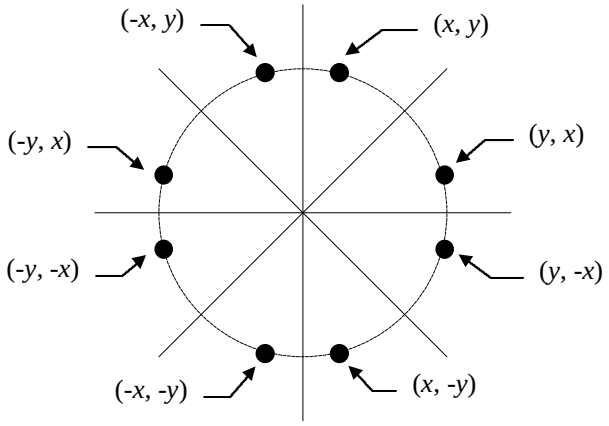
```
MidpointLine(int xA, int yA, int xB, int yB, v) {
     Int Dx, Dy, d, incE, incNE, xi, yi;
     Dx = xB - xA; Dy = yB - yA;
     d = 2 * Dy - Dx;
     incE = 2 * Dy; incNE = 2 * (Dy - Dx);
     xi = xA; yi = yA;
     writepixel(xi, yi, v);
     while (xi <= xB) {
          if (d <= 0) {
               d = d + incE;
          }
          else {
               d = d + incNE;
               yi++;
          }
          xi++;
          writepixel(xi, yi, v)
     }
}
```

Bagaimana dengan arah subkuadran lainnya? Untuk m = 0 maka penggambaran garis dipercepat dengan hanya menginterasi xi saja. Untuk ($\pi/4$, $\pi/2$) algoritma dimodifikasi dengan menukarkan x dan y.

Pengembangan lebih lanjut oleh Wu dan Rokne (1987) dengan look-ahead dua pixel berikutnya. Wyvill (90) menyarankan pemanfaatan kesimetrian garis sehingga algoritma dapat dimodifikasi untuk proses dari kedua arah. Algoritma ini bisa pula dimodifikasi untuk sistem paralel. Misalnya dengan 10 prosesor akan digambarkan suatu garis dengan Dx = 45. Pembagiannya adalah: titik 1, 11, 21, 31, dan 41 diproses oleh prosesor pertama, titik 2, 12, 22, 32, dan 42 oleh prosesor kedua, dst. Bentuk fungsi rekursif akan sedikit berbeda karena incremental titik bukan lagi 1 tapi 10 dan ada masing-masing prosesor memiliki variabel p sendiri.

Primitif Lingkaran

Untuk sementara kita asumsikan penggambaran lingkaran yang berpusat di (0, 0). Karena sifat simetri 8 arah, penghitungannya cukup dilakukan pada 1/8 lingkaran lalu diaplikasikan pada ketujuh titik lainnya. Untuk selanjutnya kita bergerak searah jarum jam (clock-wise) dari $\pi/2$ ke $\pi/4$.




```
CirclePoints(int x, int y, int v) {
     WritePixel(x,y,v); WritePixel (y,x,v);
     WritePixel(-x,y,v); WritePixel (-y,x,v);
     WritePixel (x,-y,v); WritePixel (y,-x,v);
     WritePixel (-x,-y,v); WritePixel (-y,-x,v);
}
```

Secara naif tentu kita mungkin menghitung (x, y) dari persamaan $y^2 = r^2 - x^2$. Mulai $\pi/2$ ke $\pi/4$ dari arah ini maka x menaik satu demi satu sementara y berubah antara 0 dan –1. Secara naif kita mungkin menghitung y dari fungsi $y = (r^2 - x^2)^{½}$ namun tidak perlu kita bahas karena jelas tidak efisien dengan penjelasan yang sama seperti pada pembuatan garis.

Algoritma Midpoint Untuk Lingkaran

Sama halnya pada penggambaran garis, fungsi keputusan $F( )$ digunakan dengan memanfaatkan inkrementasi pada suatu posisi dengan posisi sebelumnya. Dalam hal ini

$F(x_i, y_i) = x_i^2 + y_i^2 – r^2$.

Seperti halnya pada garis maka titik tengah $T = (x_i +1, y_i – ½)$ digunakan untuk pemeriksaan fungsi keputusan. Jika sebelumnya telah dipilih titik (x_i, y_i), fungsi keputusan tersebut digunakan untuk memilih $(x_i +1, y_i)$ jika $F(T) > 0$ atau $(x_i +1, y_i -1)$ jika $F(T) \leq 0$. Untuk selanjutnya kita menyebut kedua pilihan itu masing-masing E dan SE.

Penurunan selanjutnya kita akan dapatkan harga fungsi keputusan di awal iterasi adalah $F = 5/4 – r$ serta F akan berubah dengan $\Delta_E = 2x_i + 3$ dan $\Delta_{SE} = 2x_i – 2y_i + 3$.

Karena kedua Δ itu mengandung variabel xi dan yi yang akan berlainan untuk setiap iterasi (pada garis kedua Δ berharga konstan) maka kedua Δ itu perlu diinkremen pada setiap iterasi sbb. Pada setiap pilihan baik Δ_E maupun Δ_{SE} mengalami perubahan. Jika memilih E maka kemudian

Δ_E berubah dengan $2(x_i+1)+ 3 - 2x_i + 3 + 2 = 2$

Δ_{SE} berubah dengan $2(x_i+1) – 2y_i + 3 - 2x_i – 2y_i + 3 + 2 = 2$

Jika memilih SE maka kemudian

Δ_E berubah dengan $2(x_i+1)+ 3 - 2x_i + 3 + 2 = 2$

Δ_{SE} berubah dengan $2(x_i+1) – 2(y_i -1) + 3 - 2x_i – 2y_i + 3 + 2 = 4$

Inisialisasi kedua Δ dihitung dengan Δ_E dan Δ_{SE} pada titik $(0, r)$ sebagai titik pertama lingkaran, sbb.
$\Delta\Delta_E = 2 . 0 + 3 = 3$, dan $\Delta_{SE} = 2 . 0 – 2 . r + 3 = 5 – r$;

Di atas disebutkan bahwa sebelum iterasi $F = 5/4 – r$ yang berarti fungsi keputusan merupakan floating point. Mengingat setiap harga Δ perupakan bilangan integer maka kita gunakan fungsi keputusan lain $d = F – ¼$ yang memiliki karakteristik yang sama dengan F pada pemeriksaan tanda SE atau E.

Jadi algoritmanya dalam C adalah

```
MidpointCircle(int r, int v) {
     Int Dx, Dy, d, incE, incNE, xi, yi;
     xi = 0; yi = r;
     d = 1 - r;
     IncE = 3;
     IncSE = -2 * r + 5;
     CirclePoints(xi, yi, v);
     while (y > x) {
          if (d <= 0) {
               d = d + incE;
               incE += 2;
               incSE += 2;
          }
          else {
               d = d + incSE;
               incE += 2;
               incSE += 4;
               yi--;
          }
          xi++;
          CirclePoints(xi, yi, v);
     }
}
```

Bagaimana jika lingkaran berpusat di suatu titik bukan (0, 0) misalnya (x_0, y_0)? Ini dapat dilakukan dengan memodifikasi algoritma CirclePoints dan pemanggilnya.

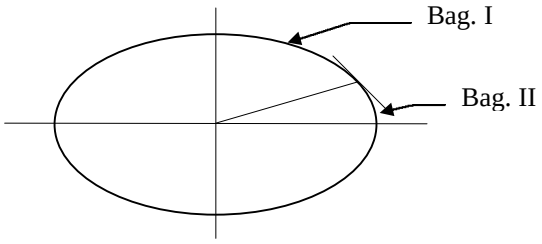
```
CirclePoints(int x, int y, int x0, int y0,int v)  {
     WritePixel(x0 + x, y0 + y, v); WritePixel(x0 + y, y0 + x, v);
     WritePixel(x0 - x, y0 + y, v); WritePixel(x0 - y, y0 + x, v);
     WritePixel(x0 + x, y0 - y, v); WritePixel(x0 + y, y0 - x, v);
     WritePixel(x0 - x, y0 - y, v); WritePixel(x0 - y, y0 - x, v);
}
```

Primitif Ellips

Suatu ellips dengan sumbu mayor 2a (arah sumbu x) dan sumbu minor 2b (arah sumbu y) adalah

$x^2 b^2 + y^2 a^2 = a^2 b^2$

Tidak seperti lingkaran, ellips hanya dapat dibagi kedalam empat ruang simetris saja. Selain itu dalam satu kuadran subkuadran tidak terbagi dengan sudut yang samayang harus dihitung sendiri-sendiri. Bagian I adalah ruang dengan gradien antara 0 hingga –1 dan bagian II adalah sisanya. Gradien dy/dx berharga -1 pada kuadran 1 adalah pada saat $2b^2 x + 2a^2 y\, dy/dx = 0$ atau $y = (b^2/a^2) x$.



Selanjutnya, secara umum penurunan formulasinya bisa meniru penurunan untuk lingkaran sbb. Pada bagian I, fungsi keputusan $F(x_i+1, y_i - ½) = (x_i+1)^2 b^2 + (y_i-½)^2 a^2 - a^2 b^2$ untuk memilih E:$(x_i + 1, y_i)$ atau SE:$(x_i + 1, y_i-1)$.

- Pada posisi awal F diinisialisasi $b^2 - a^2 b + a^2/4$.
- Kemudian pada setiap iterasi F akan berubah dengan Δ_E setelah memilih E atau Δ_{SE} setelah memilih SE dengan,
  $\Delta_E = b^2 (2x_i + 3)$
  $\Delta_{SE} = b^2 (2x_i + 3) + a^2 (-2 y_i + 2)$
- Kedua Δ ini merupakan fungsi dari x_i dan y_i maka secara inkremental Δ_E dan Δ_{SE} juga berubah sbb.
  Δ_E diinisialisasi $3 b^2$ kemudian dalam setiap iterasi berubah dengan $\Delta\Delta_E = 2b$
  Δ_{SE} diinisialisasi $3 b^2 - 2 b a^2 + 2 a^2$ dan jika SE dipilih akan berubah dengan $\Delta\Delta_{SE} = 2 b^2 - 2 a^2$

Pada bagian II fungsi keputusan $F(x_i+ ½, y_i - 1) = (x_i+ ½)^2 b^2 + (y_i-1)^2 a^2 - a^2 b^2$ untuk memilih S:$(x_i, y_i -1)$ atau SE:$(x_i + 1, y_i-1)$.

- Pada posisi awal F diinisialisasi $b^2 (x_i + ½)^2 - a^2(y_i - 1)^2 + a^2b^2$ berdasar (x_i, y_i) terakhir di bagian pertama.
- Kemudian pada setiap iterasi F akan berubah dengan Δ_S setelah memilih S atau Δ_{SE} setelah memilih SE dengan,
  $\Delta_S = a^2 (2y_i + 3)$
  $\Delta_{SE} = a^2 (2y_i + 2) + b^2 (-2 x_i + 3)$
- Kedua Δ ini merupakan fungsi dari x_i dan y_i maka secara inkremental Δ_E dan Δ_{SE} juga berubah sbb.
  Δ_E diinisialisasi $3 b^2$ kemudian dalam setiap iterasi berubah dengan $\Delta\Delta_E = 2b$
  Δ_{SE} diinisialisasi $3 b^2 - 2 b a^2 + 2 a^2$ dan jika SE dipilih akan berubah dengan $\Delta\Delta_{SE} = 2 b^2 - 2 a^2$

Primitif Kurva Lainnya

Kurva-kurva lain misalnya konus, trigonometris, eksponensial, distribusi probabilitas, polinomial umum, spline, dihitung secara naif dari fungsinya ($y = f(x)$) atau dari fungsi parametrisnya ($x = t(u)$ dan $y = s(u)$). Misalkan lingkaran dengan fungsinya: $y = - (r^2 – x^2)^{½}$, atau dengan fungsi parametrisnya: $x = r \cos(u)$ dan $y = r \sin(u)$.

Algoritma Midpoint

Metoda midpoint dapat diaplikasikan dengan menemukan fungsi keputusan dalam bentuk polinomial seperti pada primitif-primitif di atas kemudian menemukan sifat koherensi inkremen dari fungsi keputusan tsb. Jika garis memiliki inkremen yang konstan, lingkaran memiliki inkremen yang linear, maka fungsi polinomial berderajat *n* memiliki inkremen berupa fungsi polinomial berderajat *n*-1. Untuk selanjutnya fungsi inkremen yang polinomial dapat dicarikan inkremen dari inkremen tsb dengan derajat polinomial yang lebih rendah hingga konstanta.

Simetrisitas dapat mereduksi penghitungan total menjadi hanya bagian tertentu saja sementara bagian kosimetrisnya memanfaatkan hasil perhitungan ini. Contoh: pada lingkaran penghitungan cukup pada sudut 45 - 90 derajat.

Aproksimasi dengan Polyline

Polyline adalah sejumlah potongan garis lurus yang bersambungan. Selain dengan penghitungan presisi pixel demi pixel di atas. Dalam beberapa kasus aproksimasi suatu kurva sebagai sejumlah garis lurus bisa dilakukan, terutama jika bentuk fungsinya rumit. Dalam beberapa perangkat lunak seperti AutoCAD obyek-obyek gambar seperti circle, ellipse, arc, curve, dsb. digambarkan dengan cara ini untuk menghemat komputasi. Keterbatasannya adalah jika dilakukan “panning” hingga tingkat yang cukup tinggi maka diskontinyuitas kurva akan terlihat. Untuk menghindari hal tsb. pada struktur datanya disimpan juga persamaan fungsi asalnya sehingga jika zooming dilakukan maka akan digenerate polyline baru yang lebih "teliti". Ada beberapa cara pembuatan titik-titik ujung polyline ini:

- Dengan fungsi parametris: didapat titik-titik yang berjarak parametris tetap. Misalnya lingkaran dengan jarak sudut. $x = r\cos(u)$ dan $y = r\sin(u)$ iterasi $u = a, 2a, 3a, ...$ maka akan didapat (xi, yi) berjarak cukup konstan.
- Dengan fungsi eksplisit : titik-titik ujungnya dihitung dengan fungsi eksplisit $y = f(x)$ jika $|\text{gradien}| \leq 1$ atau $x = f(y)^{-1}$ jika $|\text{gradien}| > 1$

Algoritma Parallel

Pada sistem-sistem paralel algoritma-algoritma di atas dapat diadaptasikan dengan sejumlah cara:

- melakukan partisi posisi-posisi untuk masing-masing prosesor. Misalnya ada 10 prosesor unit, PU ke 1 memproses posisi ke 1, 11, 21, dst. dan prosesor 2 untuk posisi ke 2, 12, 22, dst. Modifikasi pada algoritma terletak pada increment dari fungsi diskriminan sendiri sesuai dengan increment x tsb.
- melakukan partisi frame buffer. Tiap PU memproses penggambaran dalam area yang dimilikinya.
- paralelisme total pada algoritma. Bagian-bagian algoritma dikerjakan secara pipelined.