

MODUL PRAKTIKUM GRAFIKA KOMPUTER 2D



Disusun oleh:
Muhammad Fuad DR
03/165203/PA/9277

LABORATORIUM KOMPUTASI

**JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS GADJAH MADA
YOGYAKARTA
2006**

BAB I

Software Untuk Grafika Komputer

Software untuk menggambar grafik pada komputer ada dua jenis yaitu software yang berbentuk library atau pustaka pada suatu bahasa pemrograman (paket pemrograman grafika) dan software yang berbentuk aplikasi khusus. Pada software yang berbentuk library suatu bahasa pemrograman akan dilengkapi fungsi fungsi grafik yang berasal dari paket software grafik tersebut yang termasuk contoh dari jenis ini adalah Open Gl yang dibuat Silicon Graphics. Sedang pada paket aplikasi khusus gambar grafik dibuat tanpa mengetahui bagaimanahal itu dapat terjadi, contoh dari jenis ini adalah Blender ataupun Qcad. Pada Modul ini yang digunakan adalah software jenis pertama dengan menggunakan fungsi grafik pada Bahasa Pemrograman Pascal.

Note: Semua kode pada modul ini menggunakan prosedur prosedur di bawah. tambahkan pada setiap awal kode program!.

```
procedure init;
  var  gd, gm : integer;
  begin
    gm:=detect;  gd:=0;
    InitGraph(gd, gm, '');
    if GraphResult <> grOk then
      begin
        Writeln('Graph driver ',gd,' graph mode ',gm,' not
supported');
        Halt(1);
      end;
    end;
  procedure destroy;
  begin
    closegraph;
  end;
```

BAB II

Output Primitif

1. TITIK

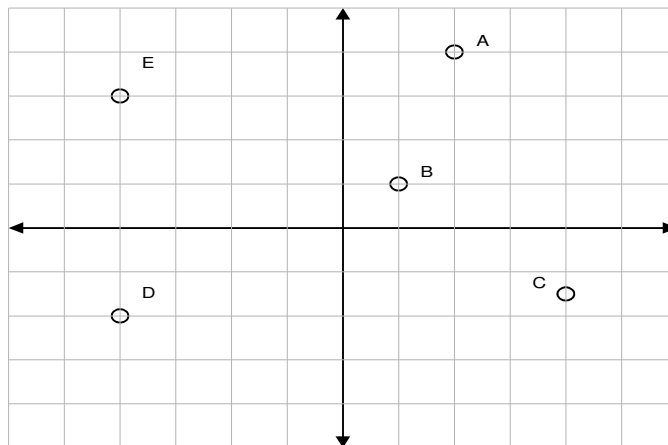
Titik dalam Grafika Komputer bisa didefinisikan sebagai suatu posisi tertentu dalam suatu sistem koordinat. Sistem koordinat yang dipakai bisa *Polar Coordinates* atau *Cartesian Coordinates*. Biasanya dalam pemrograman grafis, yang paling umum digunakan adalah *Cartesian Coordinates*.

Dalam *Cartesian Coordinates*, titik didefinisikan sebagai kombinasi dua bilangan yang menentukan posisi tersebut dalam koordinat x dan y (2D)

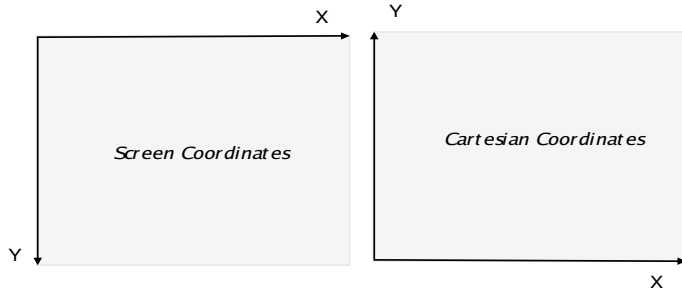
Contoh Penerapan

Jika kita ingin menempatkan titik-titik A(2,4), B(1,1), C(4,-1.5), D(-4,-2), E(-4,3)

Kita bisa menggambar sebagai berikut:



GAMBAR 1 : TITIK DALAM CARTESIAN COORDINATES Ada 2 definisi koordinat dalam komputer terutama dalam Sistem Operasi Windows, yaitu Screen Coordinate, dan Cartesian Coordinate, keduanya sering membingungkan. Untuk lebih jelasnya mari kita lihat gambar berikut:

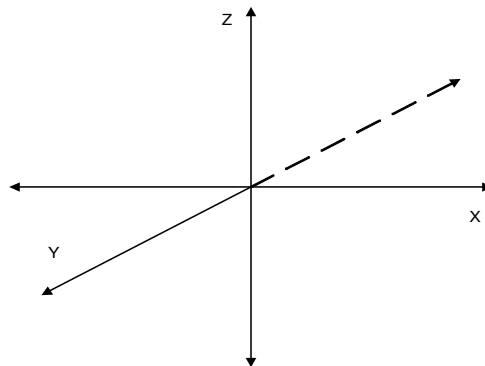


GAMBAR 2 : PERBEDAAN SCREEN DAN CARTESIAN COORDINATES

Prinsipnya, karena monitor didesain untuk menggambar dari atas ke bawah, maka sumbu y pada *Screen Coordinates* dan *Cartesian Coordinates* berbeda arah, untuk *Screen Coordinates*, sumbu Y arahnya ke bawah, sedangkan pada *Cartesian Coordinates*, sumbu Y arahnya ke atas. Biasanya dalam *rendering pipeline*, hal yang terakhir dilakukan adalah mengkonversi *Cartesian Coordinates* ke *Screen Coordinates*.

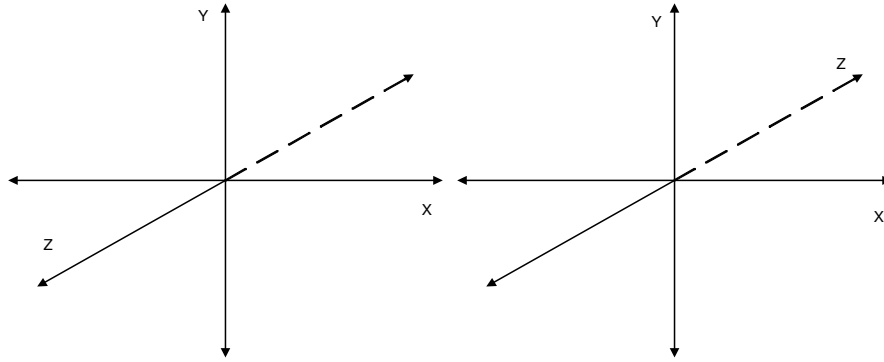
Dalam Sistem Operasi Linux, koordinat yang dipakai antara *Cartesian dan Screen sama*, yaitu Y positif ke atas.

Untuk koordinate 3D, sama dengan 2D, hanya saja ditambah 1 sumbu yaitu sumbu z (axis-z). Ada beberapa cara untuk menggambarkan sumbu X, Y dan Z, ini. Pertama dengan sumbu z mengarah ke atas(gambar 3).



GAMBAR 3 : KOORDINAT DENGAN Z MENGARAH KE ATAS

dan koordinat dengan koordinat y mengarah ke atas.



GAMBAR 4 : KOORDINAT DENGAN Y MENGARAH KE ATAS

2. Garis

Umumnya persamaan garis lurus pada koordinat kartesius diwujudkan dalam persamaan garis : $y=m.x+b$ jika dimisalkan pada dua titik(x_0,y_0 dan x_1,y_1) akan dibuat sebuah garis lurus, kita dapat menentukan nilai “m’ dan “b” dengan persamaan berikut:

$$m = \frac{y_1 - y_0}{x_1 - x_0}$$

$$b = y_1 - m \cdot x_1$$

algoritma untuk menggambar garis pada komputer didasarkan pada dua persamaan di atas. dimana m adalah gradien atau kemiringan garis tersebut.

1. Algoritma *digital differential analyzer*(DDA),

Prinsip algoritma ini adalah mengambil nilai integer terdekat dengan jalur garis berdasarkan atas sebuah titik yang telah ditentukan sebelumnya(titik awal garis).

Algoritma pembentukan garis DDA:

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

4. Tentukan langkah, yaitu dengan cara jarak maksimum jumlah penambahan nilai x maupun nilai y, dengan cara:
 - Bila nilai absolut dari dx lebih besar dari absolut dy, maka langkah= absolut dari dx.
 - Bila tidak maka langkah= absolut dari dy
5. Hitung penambahan koordinat pixel yaitu $x_increment=dx/langkah$, dan $y_increment=dy/langkah$
6. Koordinat selanjutnya $(x+x_increment, y+y_increment)$
7. Posisi pixel pada layar ditentukan dengan pembulatan nilai koordinat tersebut.
8. Ulangi nomor 6 dan 7 untuk menentukan posisi pixel selanjutnya, sampai $x=x1$ dan $y=y1$.

Contoh Prosedur DDA dalam pascal:

```
uses graph,crt;
{tambahkan pada bagian ini prosedur penginisialisasian device,
lihat pada bab 1}
procedure drawLine(xstart,ystart,xend,yend:integer);
var
  step,k:integer;
  dx,dy:real;
  x_inc,y_inc,x,y:real;
begin
  dx:=xend-xstart;
  dy:=yend-ystart;
  x:=xstart;
  y:=ystart;
  if abs(dx) > abs(dy) then
    step:=round(abs(dx))
  else
    step:=round(abs(dy));
  x_inc:=dx/step;
  y_inc:=dy/step;
  putPixel(round(x),round(y),30);
  for k:=1 to step do
```

```

begin
    x:=x+x_inc;
    y:=y+y_inc;
    putPixel(round(x),round(y),30);
end;
end;

begin
    init;
    {menggambar garis dari titik 10,10 ke 500,10}
    drawLine(10,10,500,10);
    readkey;
    destroy;
end.

```

2. Algoritma garis Bresenham

Tidak seperti Algoritma DDA, Algoritma Bresenham tidak membulatkan nilai posisi pixel setiap waktu. Algoritma Bresenham hanya menggunakan penambahan nilai integer yang juga dapat diadaptasi untuk menggambar lingkaran.

Berikut ini langkah langkah untuk membentuk garis menurut algoritma Bresenham:

1. Tentukan dua titik yang akan dihubungkan
2. Tentukan salah satu titik di sebelah kiri sebagai titik awal yaitu(x_0, y_0) dan titik lainnya sebagai titik akhir(x_1, y_1).
3. Hitung $dx, dy, 2dx$ dan $2dy-2dx$.
4. Hitung parameter

$$p_0 = 2dy - dx$$
5. Untuk setiap x_k sepanjang jalur garis, dimulai dengan $k=0$,
 - Bila $p_k < 0$, maka titik selanjutnya adalah (x_{k+1}, y_k) , dan $p_{k+1} = p_k + 2dy$

- Bila tidak, maka titik selanjutnya adalah (x_{k+1}, y_{k+1}) , dan $p_{k+1} = p_k + 2dy - 2dx$.

6. Ulangi langkah nomor 5 untuk menentukan posisi pixel selanjutnya, samapai $x=x_1$ dan $y=y_1$.

Contoh Prosedur algoritma Bressenham untuk menggambar garis dari titik (10,10) ke (500,10).

```
uses graph,crt;
{tambahkan pada bagian ini prosedur penginisialisasian device,
lihat pada bab 1}
procedure DrawBressLine(xa,ya,xb,yb:integer);
var
  dx,p,dy,xEnd:integer;
  x,y:real;
begin
  dx:= abs(xb-xa);
  dy:= abs(yb-ya);
  p:=2*dy-dx;
  if xa > xb then
  begin
    x:=xb;
    y:=yb;
    xEnd:=xa;
  end
  else
  begin
    x:=xa;
    y:=ya;
    xEnd:=xb;
  end;
  putPixel(round(x),round(y),30);

  while x < xEnd do
  begin
    x:=x+1;
    if p < 0 then
```

```

        p:=p+(2*dy)
    else
    begin
        y:=y+1;
        p:=p+(2*(dy-dx));
    end;
    putPixel(round(x),round(y),30);
end;
end;

begin
    init;
    DrawBressLine(10,10,500,10);
    readkey;
    destroy;
end.

```

4. Algoritma Pembentuk Lingkaran

Secara umum prosedur pembentuk lingkaran dibuat dengan rumus dasar $x^2+y^2=R^2$. Terdapat beberapa cara untuk membentuk suatu lingkaran namun tidak efisien. Lingkaran dapat dibuat dengan menggambarkan seperempat lingkaran karena bagian lain dapat dibuat sebagai bagian yang simetris.

a. Algoritma Simetris delapan titik

Pada algoritma ini pembuatan lingkaran 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 sebenarnya hanya diperlukan untuk menghitung segmen 45' dalam menentukan lingkaran selengkapnya. Dengan titik pusat lingkaran tertentu, delapan titik simetris dapat ditampilkan dengan prosedur Circle Point Sebagai berikut:

```

procedure CirclePoints(x, y, value:integer);
begin
    putPixel(x,y,value);
    putPixel(-x,y,value);

```

```

        putPixel(x, -y, value);
        putPixel(-x, -y, value);
        putPixel(y, x, value);
        putPixel(-y, x, value);
        putPixel(y, -x, value);
        putPixel(-y, -x, value);
    end;

```

b. Algoritma Lingkaran Midpoint

Algoritma Lingkaran Midpoint juga disebut algoritma lingkaran Bresenham. Bresenham mengembangkan generator lingkaran yang cukup efisien. Algoritma yang digunakan membentuk semua titik berdasarkan titik pusat dengan penambahan semua jalur sekeliling lingkaran. Algoritma ini diturunkan dari algoritma Midpoint untuk pembentukan garis. Dalam hal ini hanya diperhatikan bagian 45° dari suatu lingkaran, yaitu oktan kedua dari $x=0$ ke $x=R/\sqrt{2}$, dan menggunakan CirclePoints untuk menampilkan titik dari seluruh lingkaran.

Langkah langkah untuk membentuk lingkaran algoritma Circle Midpoint:

1. Tentukan radius r dengan titik pusat lingkaran (x_c, y_c) kemudian diperoleh

$$(x_0, y_0) = (0, r)$$

2. Hitung nilai dari parameter

$$P_0 = 5/4 - r$$

3. Tentukan nilai awal $k=0$, untuk setiap posisi x_k berlaku sebagai berikut:

- o Bila $P_k < 0$, maka titik selanjutnya adalah (x_{k+1}, y_k) dan

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

Bila tidak, maka selanjutnya adalah (x_{k+1}, y_{k+1}) , dan $P_{k+1} = P_k + 2x_{k+1} + 1 -$

$$2y_{k+1}$$

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

4. Tentukan titik simetris pada ketujuh oktan yang lain

- Gerakkan setiap posisi pixel(x,y) pada garis melingkar dari lingkaran dengan titik pusat (xc,yc) dan tentukan nilai koordinat:

$$x=x+x_c, y=y+y_c$$

- Ulangi langkah ke-3 sampai 5, sehingga $x \geq y$

Contoh algoritma lingkaran midpoint

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

$$P_0 = 1 - r = 1 - 10 = -9$$

Koordinat titik awal adalah $(x,r)=(0,8)$.

K	P_k	(X_{k+1}, Y_{k+1}) oktan-1
		(0,8)
0	-7	(1,8)
1	-4	(2,8)
2	1	(3,7)
3	-6	(4,7)
4	3	(5,6)
5	2	(6,5)

Prosedur algoritma lingkaran midpoint

Input yang digunakan pada prosedur ini adalah koordinat titik pusat dan radius lingkaran. Posisi pixel ditentukan dengan rutin setPixel.

```
uses graph, crt;
{tambahkan pada bagian ini prosedur penginisialisasian device,
lihat pada bab 1}
procedure circlePlotPoints(xCenter, yCenter, x, y: integer);
begin
    putPixel(xCenter+x, yCenter+y, 30);
    putPixel(xCenter-x, yCenter+y, 30);
    putPixel(xCenter+x, yCenter-y, 30);
```

```

    putPixel(xCenter-x, yCenter-y,30);
    putPixel(xCenter+y, yCenter+x,30);
    putPixel(xCenter-y, yCenter+x,30);
    putPixel(xCenter+y, yCenter-x,30);
    putPixel(xCenter-y, yCenter-x,30);
end;
procedure circleMidPoint (xCenter,yCenter,radius:integer);
var
    x,y,p:integer;
begin
    x:=0;
    y:=radius;
    p:=1-radius;
    circlePlotpoints(xCenter,yCenter,x,y);
    while x<y do
        begin
            x:=x+1;;
            if p<0 then
                p:=p+(2*x+1)
            else
                begin
                    y:=y-1;
                    p:=p+(2*(x-y)+1);
                end;
            end;
        circlePlotPoints(xCenter,yCenter,x,y);
    end;
end;
begin
    init;
    circleMidPoint(100,100,90);
    readkey;
    destroy;
end.

```

5. Algoritma Pembentukan Elips

Elips merupakan modifikasi dari bentuk lingkaran, dengan memasukkan mayor dan minor axis pada prosedur lingkaran. Elips ditentukan oleh satu set titik dengan

memperhitungkan jumlah jarak dari kedua posisi(foci). Bila jarak ke kedua foci dari sembarang titik $p(x,y)$ pada elips diberi label d_1 dan d_2 , maka persamaan elips menjadi

$$d_1+d_2=\text{konstan}$$

Untuk menggambarkan jarak d_1 dan d_2 dengan ketentuan koordinat masing masing $F_1(x_1,y_1)$ dan $F_2(x_2,y_2)$

$$\sqrt{(x-x_1)^2+(y-y_1)^2}+\sqrt{(x-x_2)^2+(y-y_2)^2}=\text{konstan}$$

Dimana mayor dan minor axis elips dalam posisi paralel dengan sumbu x dan sumbu y pada contoh ini, parameter r_x disebut semi major axis dan r_y disebut semi minor axis, sehingga persamaan elips dengan parameter r_x dan r_y menjadi

$$\left(\frac{x-x_c}{r_x}\right)^2 + \left(\frac{y-y_c}{r_y}\right)^2=1$$

Algoritma elips Midpoint

Untuk algoritma pembentukan elips, pendekatan yang dilakukan sama dengan penggunaan pada saat menampilkan lingkaran. Dengan parameter untuk elips pada posisi standar, yaitu r_x , r_y , dan (x_c,y_c) . Bila elips ditampilkan pada posisi standar, maka dapat dilakukan dengan memutar elips tersebut menurut koordinat titik pusat, dan mendapatkan kembali mayor dan minor axis.

Metode midpoint untuk elips dijalankan pada kuadran pertama dalam dua bagian. Bagian pertama menurut kemiringan elips $r_x < r_y$. Penambahan dengan unit step pada arah sumbu x dilakukan bila slope lebih kecil dari 1, dan dengan unit step menurut sumbu y bila kemiringan lebih besar dari 1.

Bagian 1 dan 2 dapat digunakan untuk bermacam macam cara. Pertama dimulai dari posisi $(0,r_y)$ dan step searah jarum jam sepanjang jalur elips pada kuadran pertama. Pergeseran dengan unit step dalam x pada saat slope lebih besar dari 1.

Alternatif lain, dimulai dari $(r_x,0)$ dan seleksi titik dalam arah berlawanan dengan arah jarum jam. Pergeseran unit step y ke unit step x pada saat kemiringan lebih besar dari -1. dengan prosesor paralel, posisi pixel dapat dihitung dalam dua bagian sekaligus

Pembentukan elips menurut algoritma Circle midpoint sebagai berikut:

1. Tentukan r_x, r_y dan pusat elips (x_c, y_c) kemudian diperoleh

$$(x_0, y_0) = (0, r_y)$$

2. Hitung nilai parameter

$$P1_0 = r_y^2 - r_x^2 r_y + 1/4 r_x^2$$

3. Tentukan nilai awal $k=0$, untuk setiap posisi x_k berlaku sebagai berikut :

- Bila $p1_k < 0$ maka titik selanjutnya adalah (x_{k+1}, y_k)

$$p1_{k+1} = p1_k + 2r_y^2 x_{k+1} + r_y^2$$

- Bila tidak, maka titik selanjutnya adalah (x_{k+1}, y_{k-1}) dan

$$p1_{k+1} = p1_k + 2r_y^2 x_{k+1} - 2r_x y_{k+1} + r_y^2$$

dengan

$$2r_y^2 x_{k+1} = 2r_y^2 x_k + 2r_y^2$$

Dan

$$2r_x y_{k+1} = 2r_x^2 y_k + 2r_x^2$$

Teruskan sampai $2r_y^2 x \geq 2r_x^2 y$

4. Tentukan nilai parameter pada bagian kedua menggunakan titik terakhir (x_0, y_0) yang telah dihitung pada bagian pertama, sebagai berikut

$$P^2_{k+1} = 2r_y^2 (x_0 + 1/2)^2 + 2r_x^2 (y_0 - 1)^2 - r_x^2 r_y^2$$

5. Setiap posisi y_k pada bagian kedua, dimulai dengan $k=0$

- Bila $p2_k > 0$ maka titik selanjutnya adalah (x_k, y_{k-1})

$$p2_{k+1} = p2_k + 2r_x^2 y_{k+1} + r_x^2$$

- Bila tidak, maka titik selanjutnya adalah (x_{k+1}, y_{k-1}) dan

$$p2_{k+1} = p2_k + 2r_y^2 x_{k+1} - 2r_x y_{k+1} + r_y^2$$

6. Tentukan titik simetris pada ketiga kuadran lainnya

7. Gerakkan setiap posisi(x,y) pada garis melingkar dari elisp dengan titik pusat(xc,yc) dan tentukan nilai koordinat

$$x=x+xc \quad y=y+yc$$

8. Ulangi langkah untuk bagian pertama di atas, sehingga $2r_y^2x \geq 2r_x^2y$

Contoh algoritma elips Midpoint

Untuk menggambarkan algoritma midpoint dalam pembentukan elips dengan titik pusat(0,0) dan mayor axis rx=6, serta minor axis ry=8, perhitungan berdasarkan pada kuadran pertama sebagai berikut:, nilai parameter dapat ditentukan

$$2r_y^2x=0$$

$$2r_y^2x=2r_x^2r_y$$

$$p1_0 = r_y^2 + r_x^2 r_y - 1/4 r_x^2$$

$$= -215$$

Koordinat titik awal (x,r) =(0,8)

K	P _k	(X _{k+1} , Y _{k+1}) oktan-1
		(0,8)
0	-215	(1,8)
1	-23	(2,8)
2	297	(3,7)
3	241	(4,6)
4	-108	(5,5)
5	236	(5,4)
6	-16	(6,3)
7	600	(6,2)
8	492	(6,1)
9	456	(6,0)

Prosedur algoritma elips Midpoint

Prosedur berikut menampilkan posisi pixel pada monitor dengan algoritma

Midpoint. Input yang digunakan adalah koordinat titik pusat mayor axis, dan minor axis.

Posisi pixel ditentukan dengan rutin setPixel.

```
uses graph,crt;
    {tambahkan pada bagian ini prosedur penginisialisasian device,
    lihat pada bab 1}
procedure elipsPlotPoints(xCenter,yCenter,x,y:integer);
begin
    putPixel(xCenter+x, yCenter+y,30);
    putPixel(xCenter-x, yCenter+y,30);
    putPixel(xCenter+x, yCenter-y,30);
    putPixel(xCenter-x, yCenter-y,30);
end;

procedure elipsMidPoint(xCenter,yCenter,Rx, Ry:integer);
var
    Rx2,Ry2,x,y,twoRx2,twoRy2,py,px,p:integer;
begin
    Rx2:=Rx*Rx;
    Ry2:=Ry*Ry;
    x:=0;
    y:=Ry;
    twoRx2:=2*Rx2;
    twoRy2:=2*Ry2;
    px:=0;
    py:=twoRx2*y;
    elipsPlotPoints(xCenter,yCenter,x,y);
    //bagian1
    p:=round(Ry2-(Rx2*Ry)+(0.25*Rx2));
    while px<py do
    begin
        x:=x+1;
        px:=px+twoRy2;
        if p<0 then
            p:= p+(Ry2+px)
        else
            begin
```

```

        y:=y-1;
        py:=py-twoRx2;
        p:=p+(Ry2+px-py);
    end;
    elipsPlotPoints(xCenter,yCenter,x,y);
end;
//bagian 2
p:=round(Ry2*(x+0.5) *(x+0.5)+Rx2*(y-1) *(y-1)-Rx2*Ry2);
while y>0 do
begin
    y:=y-1;
    py:=py-twoRx2;
    if p>0 then
        p:=p+(Rx2-py)
    else
        begin
            x:=x+1;
            px:=px+twoRy2;
            p:=p+Ry2+px-py;
        end;
    elipsPlotPoints(xCenter,yCenter,x,y);
end;
end;
begin
    init;
    elipsMidPoint(130,120,120,190);
    readkey;
    destroy;
end.

```

6. Fill area primitive

Terdapat dua dasar pendekatan untuk mengisi area pada raster system. Pertama, menentukan overlap internal untuk scan line yang melintasi area. Metode lain yaitu dengan memulai dari titik tertentu pada posisi di dalam polygon dan menggambar dengan arah menyebar ke pinggir, sampai batas polygon.

1. Algoritma scan line

Titik potong diurutkan dari kiri ke kanan. Posisi yang berhubungan pada frame buffer antara sepasang titik potong diberi warna tertentu. Posisi empat pixel sebagai titik potong antara garis batas polygon ditentukan oleh dua buah pixel pada koordinat dari $x=8$ ke $x=13$ dan dari $x=23$ ke $x=34$

2. Algoritma boundary fill.

Metode ini bermanfaat untuk paket aplikasi grafik interaktif dimana titik dalam dapat dengan mudah ditentukan. Prosedur boundary fill menerima inout koordinat suatu titik (x,y) , warna isi dan garis batas. Dimulai dari titik (x,y) , prosedur memeriksa posisi titik tetangga, yaitu apakah merupakan warna batas. Bila tidak, maka titik tersebut digambar dengan warna isi. Proses ini dilanjutkan sampai semua titik pada batas diperiksa. Prosedur berikut menampilkan metode rekursif mengisi 4 bidang dengan intensitas pada parameter fill.

```
procedure boundaryFill( x,y,fill,boundary:integer);
var
    current:integer;
begin
    current:= getPixel(x,y);
    if (current <> boundary) and (current <> fill) then
        begin
            putPixel(x,y,fill);
            boundaryFill(x+1,y,fill,boundary);
            boundaryFill(x-1,y,fill,boundary);
            boundaryFill(x,y+1,fill,boundary);
            boundaryFill(x,y-1,fill,boundary);
        end;
    end;
```

3. Algoritma flood fill

Pendekatan lain untuk mengisi suatu bidang polygon adalah 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 memiliki beberapa warna. Pertama tama yang dibuata adalah membuat nilai pixel baru, sehingga smua pixel memiliki warna yang sama. Prosedur berikut menggambarkan metode flood-fill untuk mengisi warna suatu polygon.

```
procedure floodFill( x,y,fillColor,oldColor:integer);
begin
    if getPixel(x,y) = oldcolor then
    begin
        putPixel(x,y,fillcolor);
        floodFill(x+1,y , fillColor, oldColor);
        floodFill(x-1,y , fillColor, oldColor);
        floodFill(x,y+1 , fillColor, oldColor);
        floodFill(x,y-1 , fillColor, oldColor);
    end;
end;
```

BAB III

Transformasi Dua Dimensi

1 Transformasi Dasar

a. Translasi

Translasi dilakukan dengan penambahan translasi pada suatu titik koordinat dengan translation vector atau shift vector, yaitu (tx, ty) . Koordinat baru titik yang ditranslasi dapat diperoleh dengan menggunakan rumus

$$x' = x + tx$$

$$y' = y + ty$$

translasi adalah transformasi dengan bentuk yang tetap memindahkan objek apa adanya. Titik yang akan ditranslasi akan dipindahkan ke lokasi lain menurut garis lurus.

Contoh Translasi

Untuk menggambarkan translasi suatu objek yang berupa segitiga dengan koordinat $A(10,10)$, $B(30,10)$, dan $C(10,30)$ dengan translation vector $(10,20)$, pertama tama dihitung koordinat hasil translasi

Titik A

$$x'A = xA + tx = 10+10 = 20$$

$$y'A = yA + ty = 10+20 = 30$$

Hasil translasi titik A'(20,30)

Titik B

$$x'B = xB + tx = 30+10 = 40$$

$$y'B = yB + ty = 10+20 = 30$$

Hasil translasi titik B'(40,30)

Titik C

$$x'C = xC + tx = 10+10 = 20$$

$$y'C = yC + ty = 30+20 = 50$$

Hasil translasi titik C'(20,50)

Dengan demikian hasil translasi segitiga dengan koordinat A'(20,30) B'(40,30)

C'(20,50). Segitiga yang baru sama bentuknya dengan segitiga yang lama.

Contoh Program translasi garis (10,10,500,10) dengan vector translasi 20,10

```
uses graph,crt;
var
    x0,y0,x1,y1:integer;

{tambahkan pada bagian ini prosedur penginisialisasian device, lihat
pada bab 1}
procedure drawLine(xstart,ystart,xend,yend:integer);
var
    step,k:integer;
    dx,dy:real;
    x_inc,y_inc,x,y:real;
begin
    dx:=xend-xstart;
    dy:=yend-ystart;
    x:=xstart;
    y:=ystart;
    if abs(dx) > abs(dy) then
        step:=round(abs(dx))
    else
        step:=round(abs(dy));
    x_inc:=dx/step;
    y_inc:=dy/step;
    putPixel(round(x),round(y),30);
    for k:=1 to step do
    begin
        x:=x+x_inc;
        y:=y+y_inc;
        putPixel(round(x),round(y),30);
    end;
end;
```

```

procedure transLine(xstart,ystart,xend,yend,xtrans,ytrans:integer; var
xstartout,ystartout,xendout,yendout:integer);
begin
    xstartout:=xstart+xtrans;
    ystartout:=ystart+ytrans;
    xendout:=xend+xtrans;
    yendout:=yend+ytrans;
end;
begin
    init;
    {menggambar garis dari titik 10,10 ke 500,10}
    drawLine(10,10,500,10);
    transLine(10,10,500,10,20,10,x0,y0,x1,y1);
    drawLine(x0,y0,x1,y1);
    readkey;
    destroy;
end.

```

Penskalaan

Transformasi skala adalah perubahan ukuran suatu objek. Koordinat baru diperoleh dengan melakukan perkalian koordinat dengan scaling factor, yaitu (sx, sy) dimana sx adalah scaling factor untuk sumbu x dan sy adalah scaling factor untuk sumbu y. Koordinat baru titik yang diskala dapat diperoleh dengan

$$x' = x \cdot sx$$

$$y' = y \cdot sy$$

scaling factor sx dan sy dapat diberikan sembarang nilai positif. Nilai lebih dari 1 menandakan bahwa sebuah objek diperbesar sedang nilai kurang dari 1 menunjukkan bahwa objek diperkecil.

Contoh penskalaan

Untuk menggambarkan skala suatu objek yang merupakan segiempat dengan koordinat A(10,10), B(30,10), C(30,20), D(10,20) diskala dengan scaling factor(3,2),

pertama tama dihitung koordinat hasil skala satu demi satu, pertama tama dihitung koordinat hasil skala satu demi satu

Titik A

$$x'A = xA.sx = 10 * 3 = 30$$

$$y'A = yA.sy = 10 * 2 = 20$$

Hasil skala titik A'(30,20)

Titik B

$$x'B = xB.sx = 30 * 3 = 90$$

$$y'B = yB.sy = 10 * 2 = 20$$

Hasil skala titik B'(90,20)

Titik C

$$x'C = xC.sx = 30 * 3 = 90$$

$$y'C = yC.sy = 20 * 2 = 40$$

Hasil skala titik C'(90,40)

Titik D

$$x'D = xD.sx = 10 * 3 = 30$$

$$y'D = yD.sy = 20 * 2 = 40$$

Hasil skala titik D'(30,40)

Skala dengan fixed point

Lokasi skala suatu objek dapat dikontrol dengan menentukan titik tertentu yang disebut fixed point(xf,yf). Koordinat fixed point dapat dapat pada sembarang posisi. Poligon kemudian diskala relative terhadap fixed point dengan melakukan skala jarak dari tiap titik terhadap fixed point.Untuk titik dengan koordinat (x,y) diperoleh(x',y') sebagai skala

$$x' = xf + (x - xf)sx$$

$$y' = yf + (y - yf)sy$$

Contoh Program penskalaan garis (10,10,50,10) dengan skala 2

```
uses graph,crt;
var
    x0,y0,x1,y1:integer;
{tambahkan pada bagian ini prosedur penginisialisasian device,
lihat pada bab 1}
procedure drawLine(xstart,ystart,xend,yend:integer);
var
    step,k:integer;
    dx,dy:real;
    x_inc,y_inc,x,y:real;
begin
    dx:=xend-xstart;
    dy:=yend-ystart;
    x:=xstart;
    y:=ystart;
    if abs(dx) > abs(dy) then
        step:=round(abs(dx))
    else
        step:=round(abs(dy));
    x_inc:=dx/step;
    y_inc:=dy/step;
    putPixel(round(x),round(y),30);
    for k:=1 to step do
    begin
        x:=x+x_inc;
        y:=y+y_inc;
        putPixel(round(x),round(y),30);
    end;
end;
procedure scaleLine(xstart,ystart,xend,yend,scale:integer; var
xstartout,ystartout,xendout,yendout:integer);
begin
    xstartout:=xstart*scale;
    ystartout:=ystart*scale;
    xendout:=xend*scale;
    yendout:=yend*scale;
end;
```

```

begin
    init;
    drawLine(10,10,50,10);
    scaleLine(10,10,50,10,2,x0,y0,x1,y1);
    drawLine(x0,y0,x1,y1);
    readkey;
    destroy;
end.

```

Rotasi

Rotasi dua dimensi memindahkan sebuah objek menurut garis melingkar. Untuk melakukan rotasi diperlukan sudut rotasi α dan pivot point (x_p, y_p) . Nilai positif dari sudut rotasi menentukan arah rotasi berlawanan dengan arah jarum jam. Sedangkan sudut rotasi negative memutar objek searah dengan jarum jam.

Rumus transformasi untuk rotasi suatu titik (x, y) dengan sudut rotasi α sebagai berikut:

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

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

Contoh Rotasi

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

Titik A

$$\begin{aligned}
 x'A &= x_p + (x_A - x_p) \cos 30^\circ - (y_A - y_p) \sin 30^\circ \\
 &= 10 + (10 - 10) * 0.9 - (10 - 10) * 0.5 = 10
 \end{aligned}$$

$$\begin{aligned}
 y'A &= y_p + (x_A - x_p) \sin 30^\circ + (y_A - y_p) \cos 30^\circ \\
 &= 10 + (10 - 10) * 0.5 + (10 - 10) * 0.9 = 10
 \end{aligned}$$

Hasil rotasi titik A'(10,10)

Titik B

$$\begin{aligned}x'B &= xp+(xB-xp) \cos 30' - (yB-yp)\sin 30' \\ &=10+(30-10)* 0.9 -(10-10) *0.5=28\end{aligned}$$

$$\begin{aligned}y'B &= yp+(xB-xp) \sin 30' - (yB-yp)\cos 30' \\ &=10+(30-10)*0.5 +(10-10)*0.9=20\end{aligned}$$

Hasil rotasi titik A'(28,20)

Titik C

$$\begin{aligned}x'C &= xp+(xC-xp) \cos 30' - (yC-yp)\sin 30' \\ &=10+(10-10)* 0.9 -(30-10) *0.5=0\end{aligned}$$

$$\begin{aligned}y'C &= yp+(xC-xp) \sin 30' - (yC-yp)\cos 30' \\ &=10+(10-10)*0.5 +(30-10)*0.9=28\end{aligned}$$

Hasil rotasi titik A'(0,28)

contoh program merotasi garis dengan pusat rotasi 40,50 dan sudut rotasi 180'

```
uses graph,crt;
var
    x0,y0,x1,y1:integer;
    {tambahkan pada bagian ini prosedur penginisialisasian device, lihat
pada bab 1}
procedure drawLine(xstart,ystart,xend,yend:integer);
var
    step,k:integer;
    dx,dy:real;
    x_inc,y_inc,x,y:real;
begin
    dx:=xend-xstart;
    dy:=yend-ystart;
    x:=xstart;
    y:=ystart;
    if abs(dx) > abs(dy) then
        step:=round(abs(dx))
    else
        step:=round(abs(dy));
```

```

x_inc:=dx/step;
y_inc:=dy/step;
putPixel(round(x),round(y),30);
for k:=1 to step do
begin
    x:=x+x_inc;
    y:=y+y_inc;
    putPixel(round(x),round(y),30);
end;
end;
procedure
rotateLine(xstart,ystart,xend,yend,xcenter,ycenter:integer;angle:real;
var xstartout,ystartout,xendout,yendout:integer);
begin
    xstartout:=round((xcenter+((xstart-xcenter)*cos(angle)))-((ystart-
ycenter)*sin(angle)));
    ystartout:=round((ycenter+((xstart-xcenter)*sin(angle)))-((ystart-
ycenter)*cos(angle)));
    xendout:=round((xcenter+((xend-xcenter)*cos(angle)))-((yend-
ycenter)*sin(angle)));
    yendout:=round((ycenter+((xend-xcenter)*sin(angle)))-((yend-
ycenter)*cos(angle)));
end;
begin
    init;
    drawLine(10,10,50,10);
    rotateLine(10,10,50,10,40,50,180,x0,y0,x1,y1);
    drawLine(x0,y0,x1,y1);
    readkey;
    destroy;
end.

```

Refleksi

Refleksi adalah transformasi membuat mirror dari suatu objek. Objek hasil refleksi dibuat relatif terhadap sumbu dari refleksi dengan memutar 180' terhadap sumbu refleksi. Sumbu refleksi dapat dipilih pada bidang xy. Jenis jenis refleksi ada berbagai

macam tetapi algoritma yang digunakan semua sama tinggal mengubah matriks transformasinya saja.

Contoh refleksi terhadap sumbu $y=0$.

```
uses graph,crt;
var
    x0,y0,x1,y1:integer;
    result: array [1..1,1..3] of integer;
    matTrans:array [1..3,1..3] of integer;
    before: array [1..3,1..3] of integer;

{tambahkan pada bagian ini prosedur penginisialisasian device, lihat
pada bab 1}
procedure drawLine(xstart,ystart,xend,yend:integer);
var
    step,k:integer;
    dx,dy:real;
    x_inc,y_inc,x,y:real;
begin
    dx:=xend-xstart;
    dy:=yend-ystart;
    x:=xstart;
    y:=ystart;
    if abs(dx) > abs(dy) then
        step:=round(abs(dx))
    else
        step:=round(abs(dy));
    x_inc:=dx/step;
    y_inc:=dy/step;
    putPixel(round(x),round(y),30);
    for k:=1 to step do
    begin
        x:=x+x_inc;
        y:=y+y_inc;
        putPixel(round(x),round(y),30);
    end;
end;
procedure initMatrix;
```

```

var
    i,j:integer;
begin
    for i:=1 to 3 do
        for j:=1 to 3 do
            begin
                if i=j then
                    matTrans[i,j]:=1
                else
                    matTrans[i,j]:=0;
            end;

            matTrans[3,3]:=-1;
        end;
    end;
procedure calculateMatrix(xstart,ystart:integer; var
xstartout,ystartout:integer);
var
    i,j,k,jumlah:integer;
begin
    initMatrix;
    before[1,1]:=xstart;
    before[1,2]:=ystart;
    before[1,3]:=1;

    for i:=1 to 3 do
        for j:=1 to 3 do
            begin
                jumlah:=0;
                for k:=1 to 3 do
                    begin
                        result[i,j]:=jumlah+(matTrans[i,k]*before[k,j]);
                    end;
                end;
            end;
        end;
        xstartout:=result[1,1];
        ystartout:=result[1,2];
    end;
procedure reflectLine(xstart,ystart,xend,yend:integer; var
xstartout,ystartout,xendout,yendout:integer);

```

```
begin
    calculateMatrix(xstart, ystart, xstartout, ystartout);
    calculateMatrix(xend, yend, xendout, yendout);
end;
begin
    init;
    {menggambar garis dari titik 10,10 ke 500,10}
    drawLine(10,10,500,10);
    reflectLine(10,10,500,10,x0,y0,x1,y1);
    drawLine(x0,y0,x1,y1);
    readkey;
    destroy;
end.
```

BAB IV

Clipping Titik dan Garis

Clipping adalah pendefinisian gambar di dalam maupun di luar suatu bidang tertentu. Clipping dapat diaplikasikan pada world coordinate, sehingga hanya isi yang berada dalam window dipetakan ke device coordinate. Cara lain, world coordinate lengkap dapat dipetakan ke device coordinate lebih dahulu, kemudian di clip pada batas viewport.

1. Clipping titik

Pada clip window yang berbentuk segi empat, titik (x,y) akan ditampilkan bila

$$x_{wmin} \leq x \leq x_{wmax}$$

$$y_{wmin} \leq y \leq y_{wmax}$$

dimana batas clip window $(x_{wmin}, x_{wmax}, y_{wmin}, y_{wmax})$ dapat berada di dalam batas world coordinate atau viewport coordinate. Bila salah satu tidak terpenuhi maka titik tersebut tidak akan di clip.

Contoh Algoritma Clipping titik

```
uses graph, crt;
var
    x, y, xmin, ymin, xmax, ymax: integer;
    jawab: char;
    {tambahkan pada bagian ini prosedur penginisialisasian device,
    lihat pada bab 1}
procedure clipPoint(x, y, xmin, ymin, xmax, ymax: integer);
begin
    if (x >= xmin) and (x <= xmax) and (y >= ymin) and (y <= ymax) then
    begin
        init;
        putPixel(x, y, 30);
        readkey;
        destroy;
    end
end
```

```

        else
            writeln('Point is not visible');
end;
begin
    jawab:='n';
    while jawab = 'n' do
        begin
            writeln('enter the coordinates of clipping window ');
            writeln('enter x(min) y(min)');
            read(xmin);
            readln(ymin);
            writeln('enter x(max) y(max)');
            read(xmax);
            readln(ymax);
            writeln('enter the coordinates of point x y');
            read(x);
            readln(y);
            clipPoint(x,y,xmin,ymin,xmax,ymax);
            writeln('do yo want to exit (y/n)');
            readln(jawab);
        end
    end.
end.

```

2. Clipping Garis

Gambar di atas memprlihatkan hubungan antara posisi garis dengan viewport. Bagian bagian garis yang berada di luar viewport tidak akan ditampilkan. Berikut ini algoritma clipping garis yang palibg sering digunakan:

1. Periksa segmen garis untuk menentukan sepenuhnya berada di dalam clip window.
2. Bila tidak, maka dicoba untuk menentukan apakah sepenuhnya di luar window.
3. Bila tidak dapat mengidentifikasi sepenuhnya di dalam, ataupun di luar window, maka dilakukan perhitungan perpotongan dengan satu atau lebih batas clipping.

Clipping garis Cohen-Sutherland

Algoritma Cohen-Sutherland merupakan algoritma yang paling populer.

Biasanya metode ini mempercepat pemrosesan segmen garis dengan mengurangi jumlah perpotongan yang harus dihitung. Setiap endpoint dari garis dalam suatu gambar dinyatakan dalam 4 digit kode biner yang disebut region code, yang mengidentifikasi lokasi dari titik relatif terhadap batas clipping yang berbentuk segi empat.

Contoh Prosedur Clipping garis Cohen-Sutherland

```
uses graph,crt;
var
  pixels:array[0..1,0..3] of integer;
  x1,y1,x2,y2,xmin,ymin,xmax,ymax:integer;
  xn1,xn2,yn1,yn2,m:real;
  jawab:char;
{tambahkan pada bagian ini prosedur penginisialisasian device, lihat
pada bab 1}

procedure DrawBressLine(xa,ya,xb,yb:integer);
var
  dx,p,dy,xEnd:integer;
  x,y:real;
begin
  dx:= abs(xb-xa);
  dy:= abs(yb-ya);
  p:=2*dy-dx;
  if xa > xb then
  begin
    x:=xb;
    y:=yb;
    xEnd:=xa;
  end
  else
  begin
    x:=xa;
    y:=ya;
    xEnd:=xb;
  end;
end;
```

```

putPixel(round(x),round(y),30);
while x < xEnd do
begin
    x:=x+1;
    if p < 0 then
        p:=p+(2*dy)
    else
begin
        y:=y+1;
        p:=p+(2*(dy-dx));
    end;
    putPixel(round(x),round(y),30);
end;
end;
procedure su_co( x1,y1,x2,y2,xmin,ymin,xmax,ymax:integer);
var
    i,j,fl:integer;
begin
    for i:=0 to 1 do
        for j:=0 to 3 do
            pixels[i,j]:=0;
        if y1 > ymax then
pixels[0,0]:=1;
        if y1<ymin then
pixels[0,1]:=1;
        if x1>xmax then
pixels[0,2]:=1;
        if x1<xmin then
pixels[0,3]:=1;
        if y2>ymax then
pixels[1,0]:=1;
        if y2<ymin then
pixels[1,1]:=1;
        if x2>xmax then
pixels[1,2]:=1;
        if x2<xmin then
pixels[1,3]:=1;
        for j:=0 to 3 do

```

```

        begin
    if (pixels[0,j]=0) and (pixels[1,j]=0) then
        continue;
    if (pixels[0,j]=1) and (pixels[1,j]=1) then
    begin
        fl:=3;
        break;
    end;
    fl:=2;
        end;
        case fl of
    1:
    begin
        DrawBressLine(x1,y1,x2,y2);
    end;
    3:
        writeln('a" Line Is Not Visible...:-(');
    2:
    begin
        m:=(y2-y1)/(x2-x1);
        xn1:=x1;
        yn1:=y1;
        xn2:=x2;
        yn2:=y2;
        if pixels[0,0]=1 then
            begin
                xn1:=x1+(ymax-y1)/m;
                yn1:=ymax;
            end;
        if pixels[0,1]=1 then
            begin
                xn1:=x1+(ymin-y1)/m;
                yn1:=ymin;
            end;
        if pixels[0][2]=1 then
            begin
                yn1:=y1+(xmax-x1)*m;
                xn1:=xmax;

```

```

        end;
        if pixels[0][3]=1 then
        begin
            yn1:=y1+(xmin-x1)*m;
            xn1:=xmin;
        end;
    if pixels[1][0] = 1 then
        begin
            xn2:=x2+(ymax-y2)/m;
            yn2:=ymax;
        end;
        if pixels[1][1]=1 then
        begin
            xn2:=x2+(ymin-y2)/m;
            yn2:=ymin;
        end;
    if pixels[1][2]=1 then
        begin
            yn2:=y2+(xmax-x2)*m;
            xn2:=xmax;
        end;
    if pixels[1][3]=1 then
        begin
            yn2:=y2+(xmin-x2)*m;
            xn2:=xmin;
        end;
        DrawBressLine(round(xn1),round(yn1),round(xn2),round(yn2));
    end;
end;
begin
    jawab:='n';
    while jawab = 'n' do
    begin
        writeln('enter the coordinates of clipping window ');
        writeln('enter x(min) y(min)');
        read(xmin);
        readln(ymin);
    end;
end;

```

```

writeln('enter x(max) y(max)');
read(xmax);
readln(ymax);
    writeln('enter the coordinates of line');
writeln('enter X1 Y1');
    read(x1);
    readln(y1);
writeln('enter X2 Y2');
    read(x2);
    readln(y2);

init;
DrawBressLine(x1,y1,x2,y2);
readkey;
destroy;
    init;
    su_co(0,0,80,80,xmin,ymin,xmax,ymax);
    readkey;
    destroy;

writeln('do yo want to exit (y/n)');
readln(jawab);
end;
end.

```

Clipping garis Liang-Barsky

Clipping garis Liang-Barsky berdasarkan atas analisis persamaan parametrik dari segmen garis yang dapat ditulis dalam bentuk:

$$x=x1+u*dx$$

$$y=y1+u*dy$$

dimana $dx=x2-x1$ dan $dy=y2-y1$. Pada umumnya algoritma ini lebih efisien dibandingkan algoritma Cohen-Sutherland, karena perhitungan titik potong diabaikan. Setiap perubahan hanya memerlukan satu bagian dan hanya menghitung sekali untuk mendapatkan $u1$ dan $u2$. Sebaliknya algoritma Cohen-Sutherland berulang ualng menghitung titik potong sepanjang suatu garis.

Contoh Prosedur Clipping garis Liang-Barsky

```

uses graph,crt;
var
  x1,y1,x2,y2:real;
  jawab:char;
  xmin,ymin,xmax,ymax:integer;
  {tambahkan pada bagian ini prosedur penginisialisasian device,
  lihat pada bab 1}

procedure DrawBressLine(xa,ya,xb,yb:integer);
var
  dx,p,dy,xEnd:integer;
  x,y:real;
begin
  dx:= abs(xb-xa);
  dy:= abs(yb-ya);
  p:=2*dy-dx;
  if xa > xb then
  begin
    x:=xb;
    y:=yb;
    xEnd:=xa;
  end
  else
  begin
    x:=xa;
    y:=ya;
    xEnd:=xb;
  end;
  putPixel(round(x),round(y),30);
  while x < xEnd do
  begin
    x:=x+1;
    if p < 0 then
      p:=p+(2*dy)
    else
      begin
        y:=y+1;
        p:=p+(2*(dy-dx));
      end
  end
end

```

```

        end;
        putPixel(round(x),round(y),30);
    end;
end;
function clipt( d,q:real; var tL, tU:real ):boolean;
begin
    if d < 0 then
        begin
            if q < (tU * d ) then
                clipt:=false;
            if q < (tL * d ) then
                tL := q / d;
            end
            else if d > 0 then
                begin
                    if q < (tL * d ) then
                        clipt:= false;
                    if q < (tU * d ) then
                        tU := q / d;
                    end
                    else if q < 0 then
                        begin
                            clipt:= false;
                        end;
                    clipt:= true;
                end;
            end;
end;
procedure liang_barsky_line(x1,y1,x2,y2:real);
var
    dx,dy,tL,tU:real;
begin
    dx := x2 - x1;
    dy := y2 - y1;
    tL := 0;
    tU := 1;
    if clipt( -dx, x1 - xmin, tL, tU ) = true then
        begin
            if clipt( dx, xmax - x1, tL, tU ) = true then
                begin

```

```

    if clipt( -dy, y1 - ymin, tL, tU ) = true then
    begin
    if clipt( dy, 300 - ymax, tL, tU ) = true then
    begin
        if tU < 1 then
        begin
            x2 := x1 + (tU * dx);
            y2 := y1 + (tU * dy);
        end;
        if tL > 0 then
        begin
            x1 := x1+(tL * dx);
            y1 := y1+(tL * dy);
        end;
        DrawBressLine(round(x1),round(y1),round(x2),round(y2));
    end;
    end;
end;
end;
begin
    jawab:='n';
    while jawab = 'n' do
    begin
        writeln('enter the coordinates of clipping window ');
        writeln('enter x(min) y(min)');
        read(xmin);
        readln(ymin);
        writeln('enter x(max) y(max)');
        read(xmax);
        readln(ymax);
        writeln('enter the coordinates of line');
        writeln('enter X1 Y1');
        read(x1);
        readln(y1);
        writeln('enter X2 Y2');
        read(x2);
        readln(y2);
    end;
end;

```

```
init;
DrawBressLine(round(x1),round(y1),round(x2),round(y2));
readkey;
destroy;
    init;
    liang_barsky_line(x1,y1,x2,y2);
    readkey;
    destroy;
writeln('do yo want to exit (y/n)');
readln(jawab);
    end;
end.
```

Referensi:

- Didiet. Titik Dan Garis. [http://lynxluna.org/papers/math/Chap1.Titik dan Garis.pdf](http://lynxluna.org/papers/math/Chap1.Titik%20dan%20Garis.pdf). Diakses tanggal 19 oktober 2005
- Sutopo, Aries Hadi. *Pengantar grafika komputer*. Gava Media, Yogyakarta. 2002