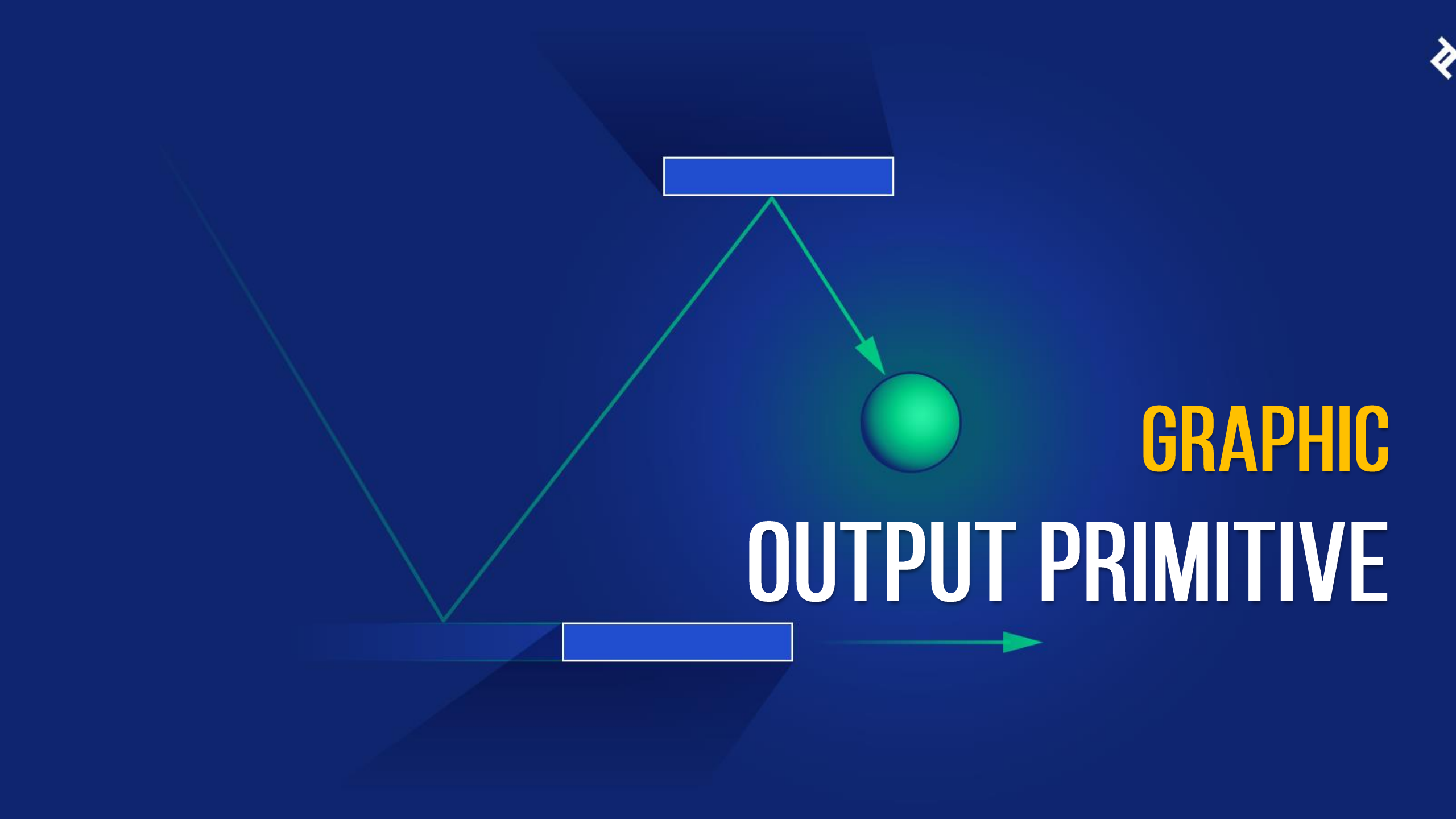


GRAPHIC
OUTPUT PRIMITIVE

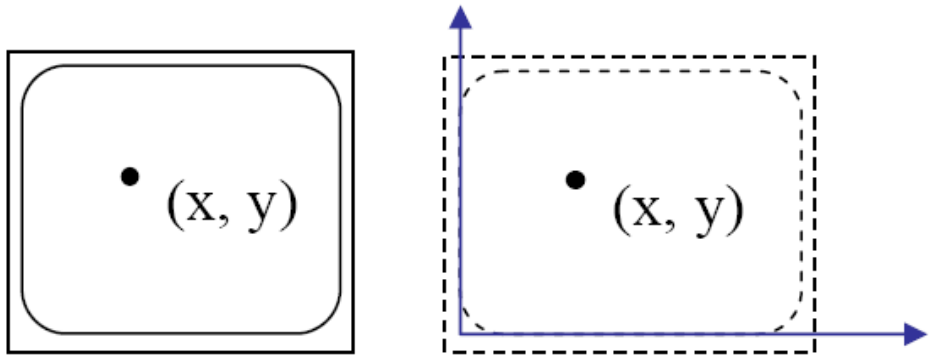
Materi

- Menggambar garis
- Algoritma DDA
- Algoritma Bressenham
- Menggambar Lingkaran
- Open Graphic Library (Open GL)

Menggambar Garis

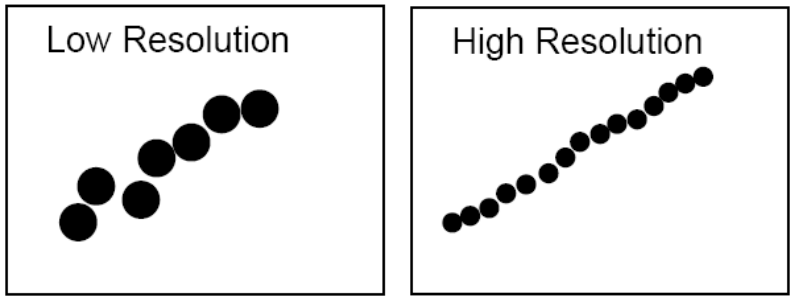
- Garis adalah kumpulan titik-titik yang tersusun sedemikian rupa sehingga memiliki pangkal dan ujung.
- Suatu titik pada layar terletak pada posisi (x,y), untuk menggambarkannya plot suatu pixel dengan posisi yang berkesesuaian.
- Contoh Program :

  Setpixel (x,y)

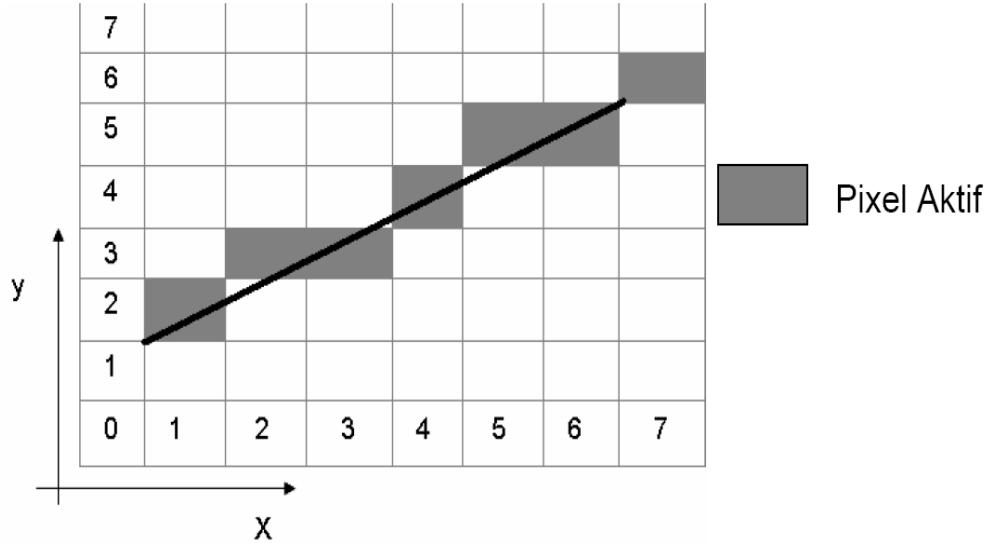


Menggambar Garis

- Penampilan garis pada layar komputer dibedakan berdasarkan Resolusi-nya.
  - Resolusi : keadaan pixel yang terdapat pada suatu area tertentu
  - Contoh : Resolusi 640x480, berarti pada layar kompuer terdapat 640 pixel per-kolom dan 480 pixel per-baris.
  - Resolusi dapat pula dibedakan menjadi kasar, medium dan halus.



Menggambar Garis



Pixel	X	Y
1	1	2
2	2	3
3	3	3
4	4	4
5	5	5
6	6	5
7	7	6

- Untuk menggambarkan garis seperti gambar di atas, diperlukan pixel aktif.
- Parameter pixel address yang membentuk garis pada layar adalah :

Menggambar Garis

- Untuk menampilkan atau menggambarkan garis pada layar dibutuhkan minimal 2 titik (endpoint), yaitu titik awal dan akhir.
  - Awal garis dimulai dengan titik atau pixel pertama, P1 diikuti titik kedua, P2.
  - Untuk mendapatkan titik-titik selanjutnya sampai ke Pn perlu dilakukan inkrementasi atas nilai koordinat sumbu X dan Y pada titik sebelumnya.
  - Perhitungan inkrementasi untuk masing-masing sumbu adalah berbeda :

Menggambar Garis

Jenis	Sumbu-X	Sumbu-Y
Horisontal	Gerak (X=X+1)	Konstan
Vertikal	Konstan	Gerak (Y=Y+1)
Diagonal	Gerak (X=X+1)	Gerak (Y=Y+1)
Bebas	Gerak (X=X+n)	Gerak (Y=Y+n)

n dan m adalah nilai inkrementasi

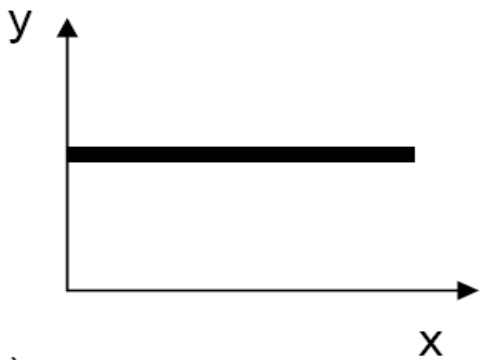
- Persamaan Umum Garis :

$$y = mx + c$$

Menggambar Garis

Garis Horisontal

- Garis yang membentang secara paralel dengan sumbu X dengan asumsi titik P1 pada koordinat X1 lebih kecil daripada X2 dari P2, sedangkan Y1 dan Y2 konstant



- Algoritma:
  1. Menentukan titik awal (P1) dan titik akhir (P2)
  2. Periksa posisi sumbu (koordinat)Jika titik ahir < titik awal,Lakukan inkrementasi sumbu X dari titik awal sampai titik akhir, Jika tidak, maka lakukan dekrementasi sumbu X dari titik awal sampai titik akhir
  3. Tampilkan garis menggunakan parameter koordinat yang telah dihitung.

Menggambar Garis

Garis Vertikal

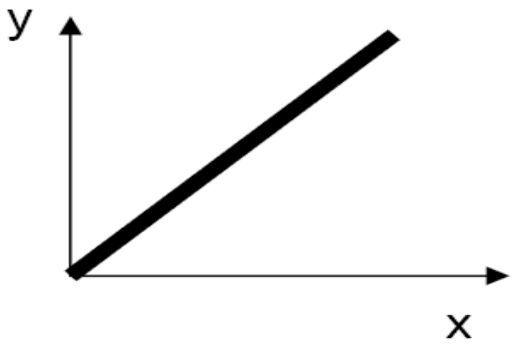
- Garis yang membentang secara paralel dengan sumbu Y dengan asumsi titik P1 pada koordinat Y1 lebih kecil daripada Y2 dari P2, sedangkan X1 dan X2 konstant

- Algoritma:
  1. Menentukan titik awal (P1) dan titik akhir (P2)
  2. Periksa posisi sumbu (koordinat)Jika titik ahir < titik awal,Lakukan inkrementasi sumbu Y dari titik awal sampai titik akhirJika tidak, maka lakukan dekrementasi sumbu Y dari titik awal sampai titik akhir
  3. Tampilkan garis menggunakan parameter koordinat yang telah dihitung.

Menggambar Garis (6)

Garis Diagonal

- Garis yang membentang secara paralel 45 derajat dari sumbu X atau sumbu Y dengan asumsi titik awal P1 dengan koordinat X1 dan Y1 lebih kecil daripada X2 dan Y2 atau sebaliknya.



- Algoritma:
  1. Menentukan titik awal (P1) dan titik akhir (P2)
  2. Periksa posisi sumbu (koordinat)Jika titik ahir < titik awal,Lakukan inkrementasi sumbu X dan sumbu Y dari titik awal sampai titik akhirJika tidak, maka lakukan dekrementasi sumbu X dan sumbu Y dari titik awal sampai titik akhir
  3. Tampilkan garis menggunakan parameter koordinat yang telah dihitung.

Algoritma DDA

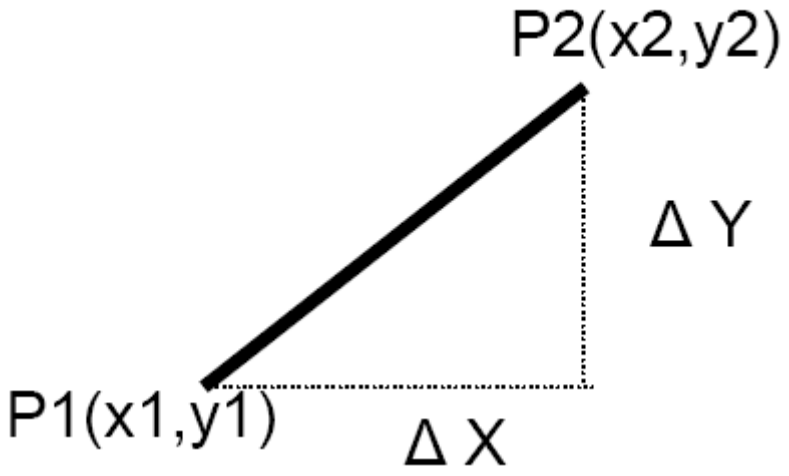
- Garis yang membentang antara 2 titik, P1 dan P2, selalu membentuk sudut yang besarnya sangat bervariasi.
- Sudut yang terbentuk menentukan kemiringan suatu garis atau disebut gradient/ slop atau disimbolkan dengan parameter m.
- Jika titik-titik yang membetuk garis adalah : (x1,y1)dan (x2,y2)

Algoritma DDA

maka

$$m = \frac{\Delta y}{\Delta x}$$

$$m = \frac{y2 - y1}{x2 - x1}$$



Algoritma DDA

- Algoritma DDA bekerja bekerja atas dasar penambahan nilai x dan nilai y.
- Pada garis lurus, turunan pertama dari x dan y adalah konstanta.
- Sehingga untuk memperoleh suatu tampilan dengan ketelitian tinggi, suatu garis dapat dibangkitkan dengan menambah nilai x dan y masing-masing sebesar $\varepsilon\Delta x$ dan $\varepsilon\Delta y$, dengan ε besaran dengan nilai yang sangat kecil.

Algoritma DDA

- Kondisi ideal ini sukar dicapai, karenanya pendekatan yang mungkin dilakukan adalah berdasarkan piksel-piksel yang bisa dialamati/dicapai atau melalui penambahan atau pengurangan nilai x dan y dengan suatu besaran dan membulatkannya ke nilai integer terdekat.

Algoritma DDA

untuk |m| < 1:

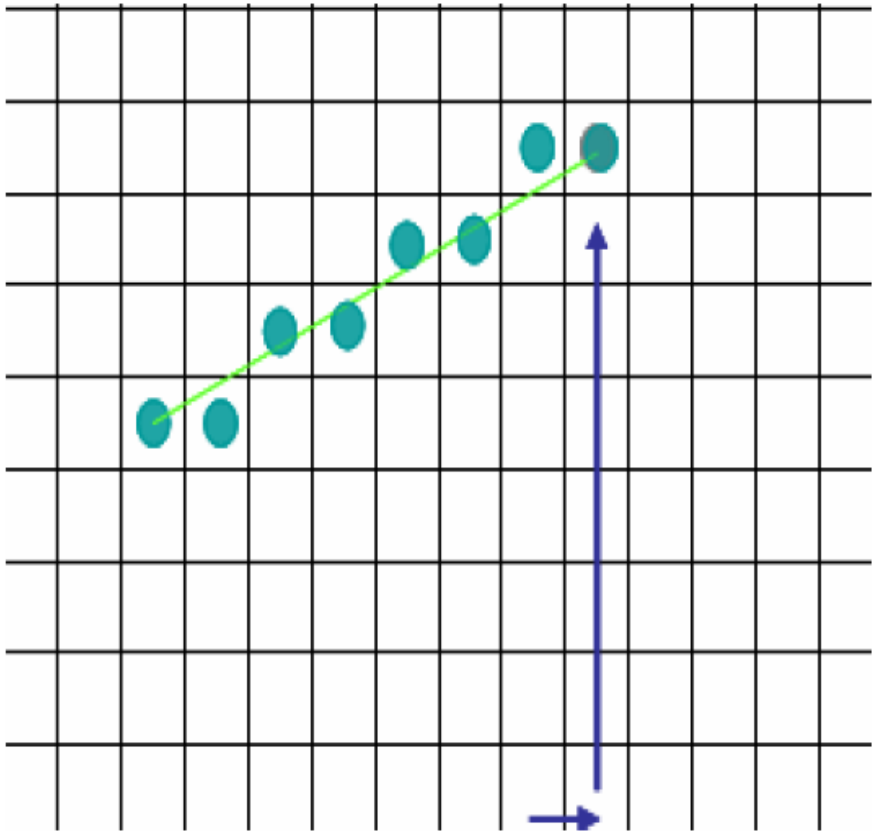
$$x_{k+1} = x_k + \partial x$$

$$= x_k + 1$$

$$y_{k+1} = y_k + \partial y$$

$$= y_k + m.\partial x$$

$$= y_k + m.(1)$$



Algoritma DDA

untuk |m| >1:

$$x_{k+1} = x_k + \partial x$$

$$= x_k + \frac{1}{m}$$

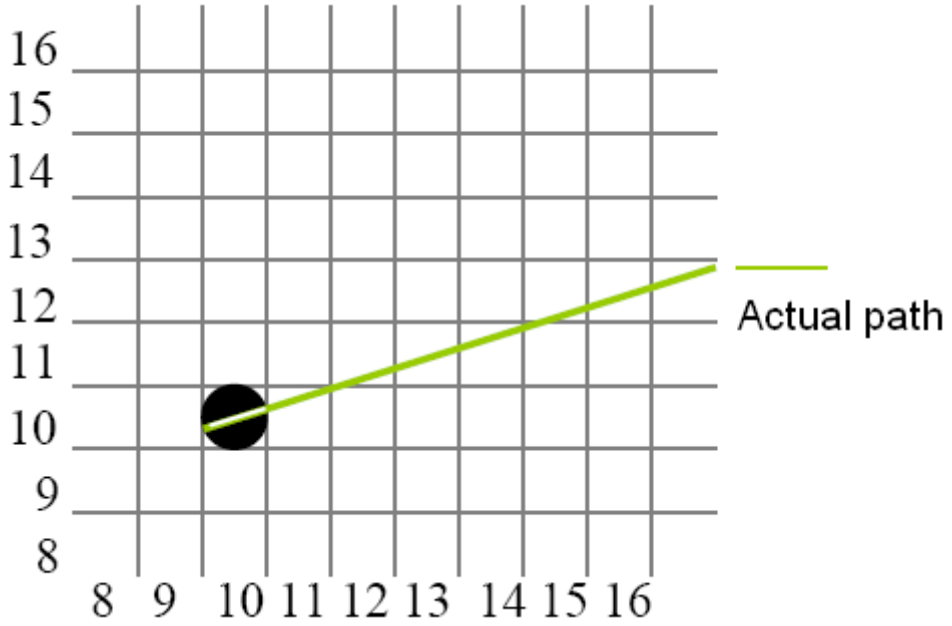
$$y_{k+1} = y_k + \partial y$$

$$= y_k + 1$$

Algoritma DDA

```
#define ROUND(a) ((int)(a+0.5))
void lineDDA (int xa, int xb, int ya, int yb)
{
     int dx = xb - xa, dy = yb - ya, steps, k;
     float xincrement, yincrement, x  = xa, y=ya;
     if (abs (dx)> abs (dy))
        steps = abs (dx);
     else
        steps = abs (dy);
     xincrement = dx / (float) steps;
     yincrement = dy / (float) steps;
     setPixel (ROUND(x), ROUND(y);
     for (k=0; k<steps; k++){
         x +=xincrement;
         y =+yincrement;
         setPixel (ROUND(x), ROUND(y));
     }
}
```

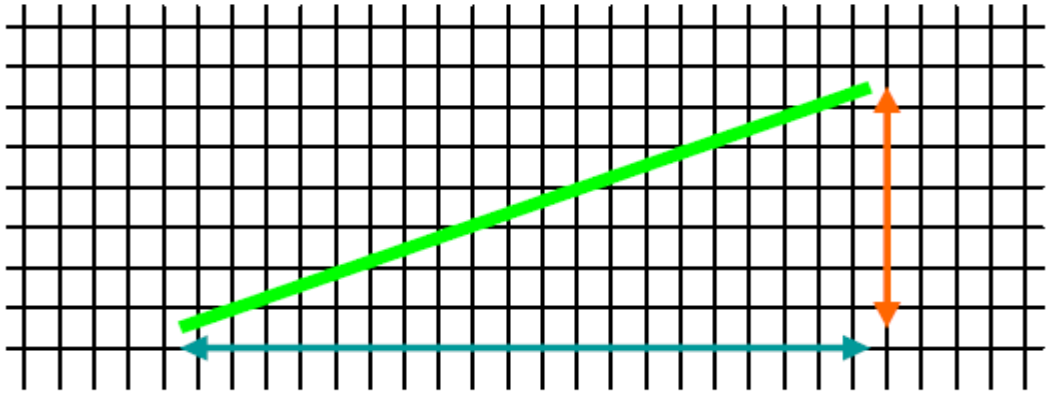
Algoritma Bressenham



Pixel selanjutnya ?

Algoritma Bressenham

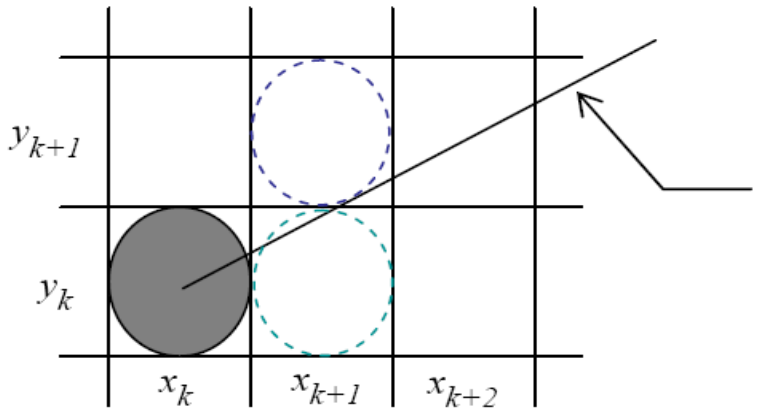
- Algoritma Bresenhma memilih titik terdekat dari actual path.
- Setiap sampling akan diincrement menjadi 1 atau 0



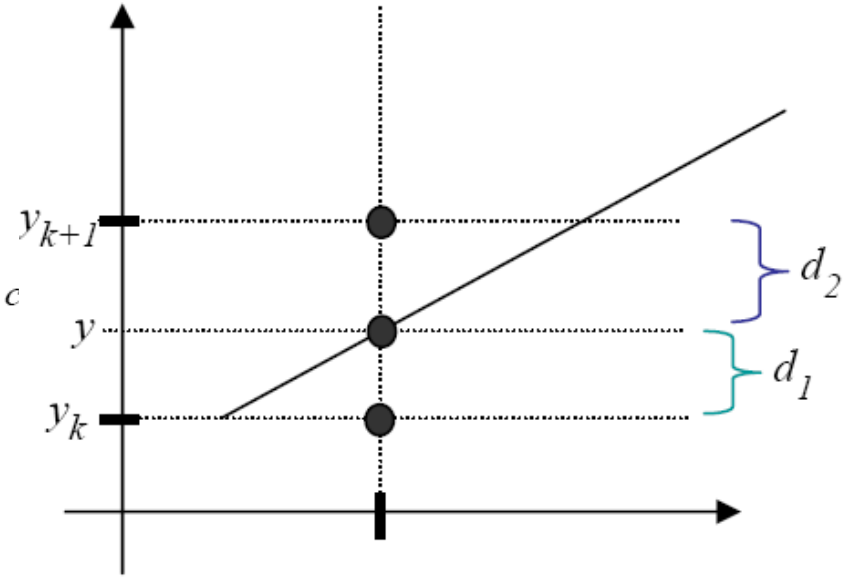
  - Kondisi awal :Jika m < 1, maka m bernilai positif
  - Bresenham melakukan inkremen 1 untuk x dan 0 atau

Algoritma Bressenham

- Jika current pixel (xk,yk)



$y = mx + c$



- Dimanakah pixel berikutnya akan di-plot, apakah di (xk+1, yk+1)= (xk+1, yk), atau (xk+1, yk+1)?

Algoritma Bressenham

- Tentukan nilai parameter keputusan, pk:

$$p_k = \Delta x(d_1 - d_2)$$

p_k	*negatif*	$d_1 < d_2$; pixel pada scanline y_k adalah di dekat actual path • plot (x_{k+1}, y_k)
p_k	*positif*	$d_1 > d_2$; pixel pada scanline y_{k+1} adalah di dekat actual path • plot (x_{k+1}, y_{k+1})

Algoritma Bressenham

Algoritma Bresenham untuk |m| < 1:

1. Input 2 endpoints, simpan endpoints kiri sebagai (x0, y0).
2. Panggil frame buffer (plot titik pertama)
3. Hitung konstanta Δx, Δy, 2Δy, 2Δy–2Δx dan nilai awal parameter keputusan p0= 2Δy –Δx
4. Pada setiap xk di garis, dimulai dari k=0, ujilah :

   jika pk< 0, plot(xk+1, yk) dan pk+1= pk+ $2\Delta y$

   jika tidak, maka plot (xk+1, yk+1)dan pk+1= pk+ $2\Delta y - 2\Delta x$
5. Ulangi tahap 4 Δx kali

Algoritma Bressenham

Algoritma Bresenham untuk |m| >1:

1. Input 2 endpoints, simpan endpoints kiri sebagai (x0, y0).
2. Panggil frame buffer (plot titik pertama)
3. Hitung konstanta Δx, Δy, 2Δx, 2Δx–2Δy dan nilai awal parameter keputusan p0= 2Δx –Δy
4. Pada setiap yk di garis, dimulai dari k=0, ujilah :

   jika pk< 0, plot(xk, yk +1) dan pk+1= pk+ $2\Delta x$

   jika tidak, maka plot (xk+1, yk+1)dan pk+1= pk+ $2\Delta x - 2\Delta y$
5. Ulangi tahap 4 Δy kali

Algoritma Bressenham

```
void lineBres (int xa, int xb, int ya, int yb)
{
     int dx = abs (xa-xb), dy = abs (ya-yb);
     int p = 2 * dy - dx;
     int twoDy = 2 * dy, twoDx = 2 * (dy-dx);
     int x, y , xEnd;
     if (xa>xb){
         x = xb;
         y = yb;
         xEnd = xa;
     }
     else {
          x = xa;
          y = ya;
          xEnd = xb;
     }
     setPixel(x,y);
```

Algoritma Bressenham

```
        while (x < xEnd){
              x++;
              if(p<0)
                 p += twoDy;
              else
              {
                  y++;
                  p += twoDx;
              }
        setPixel (x,y);
        }
}
```

Algoritma Bressenham

- Latihan : Hitunglah posisi piksel hingga membentuk sebuah garis yang menghubungkan titik (12,10) dan (17,14) !
- Jawab :
  1. (x0, y0) =(12, 10)
  2. $\Delta x = 5,\ \Delta y = 4,\ 2\Delta y = 8,\ 2\Delta y - 2\Delta x = -2$
  3. $p0 = 2\Delta y - \Delta x = 3$

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

Menggambar Lingkaran

- Persamaan lingkaran

$$(x - x_0)^2 + (y - y_0)^2 = r^2$$

$$dengan$$

$$y = y_0 \pm \sqrt{r^2 - (x - x_0)^2}$$

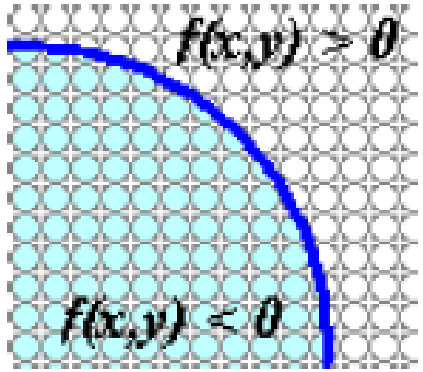
Menggambar Lingkaran

- Fungsi discriminator

$$(x-x_0)^2+(y-y_0)^2=r^2$$

- Dapat ditulis sebagai fungsi

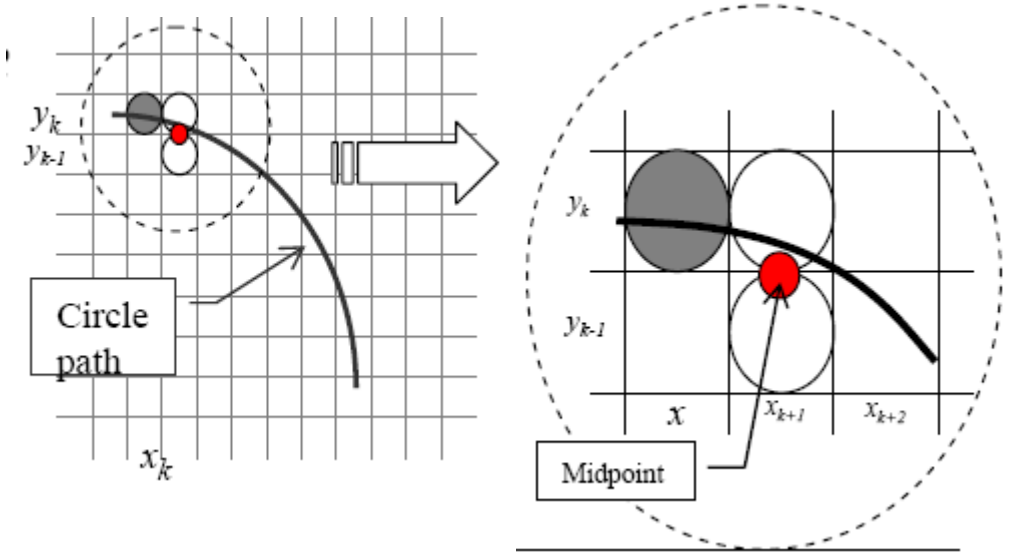
$$f(x,y)=(x)^2+(y)^2-r^2$$



- Fungsi discriminator
  - f(x,y)<0 untuk titik di dalam lingkaran
  - f(x,y)=0 untuk titik yang terletak pada lingkaran
  - f(x,y)>0 untuk titik di luar lingkaran

Midpoint Algorithm

- Bila diketahui suatu titik : (xk,yk), maka titik berikutnya apakah di(xk+1, yk), or (xk+1, yk-1) ?
- Misal titik tengahnya (midpoint) : (xk+1, yk) = 0.5
- Gunakan fungsi discriminator untuk mendapatkan :



Midpoint Algorithm

- Dengan menggunakan midpoint di antara 2 kandidat pixel, kita dapat mencari Parameter Keputusan, Pk, untuk mendapatkan plot pixel berikutnya :

$$P_k = f(x_k + 1, y_k - \tfrac{1}{2})$$

$$= (x_k + 1)^2 + (y_k - \tfrac{1}{2})^2 - r^2$$

Midpoint Algorithm

1. Input radius, r, and titik tengah lingkaran (xc, yc). Titik awal di-plot pada (0, r) –yang merupakan titik tengah lingkaran asli,
2. Hitung nilai awal Parameter Keputusan :
3. Pada xk, dimulai dengan k = 0, uji nilai Pk:

   Jika Pk< 0, maka titik selanjutnya (x_{k+1}, y_k)

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

   Jika Pk>= 0, maka titik selanjutnya (x_k+1, y_k-1)

   $P_{k+1} = P_k + 2X_k + 1 - 2\ y_k$

   dimana : $2x_k = 2x_k + 2$, $2y_k = 2y_k - 2$
4. Tentukan titik simetri pada 7 octant lainnya.
5. Ambil titik aktual untuk titik tengah lingkaran pada (xc, yc) dimana (x + xc, y + yc).
6. Ulangi langkah 3 sampai 5 hingga tercapai x ≥y.

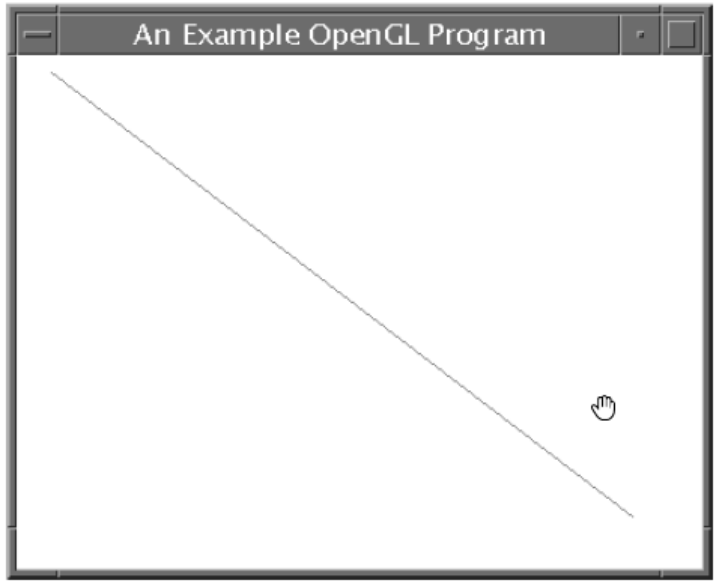
$$p_0 = \frac{5}{4} - r$$

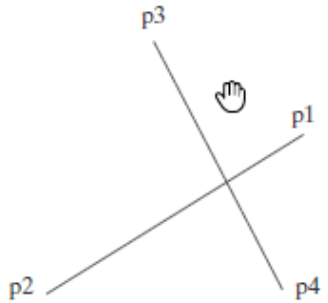
Tugas

- Buat program untuk menggambarkan garis dan lingkaran dengan menggunakan algoritma Bressenham dan algoritma midpoint
- Tugas dikerjakan secara berkelompok
- Tugas dikerjakan dalam waktu 1 minggu
- File dikompresi, format : zip, rar.

OPEN GRAPHIC LIBRARY (OPENGL)

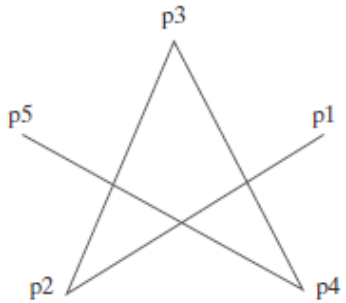
```
#include <GL/glut.h>      // (or others, depending on the system in use)

void init (void)
{
    glClearColor (1.0, 1.0, 1.0, 0.0);  // Set display-window color to white.

    glMatrixMode (GL_PROJECTION);       // Set projection parameters.
    gluOrtho2D (0.0, 200.0, 0.0, 150.0);
}

void lineSegment (void)
{
    glClear (GL_COLOR_BUFFER_BIT);  // Clear display window.

    glColor3f (0.0, 0.4, 0.2);      // Set line segment color to green.
    glBegin (GL_LINES);
        glVertex2i (180, 15);       // Specify line-segment geometry.
        glVertex2i (10, 145);
    glEnd ( );

    glFlush ( );     // Process all OpenGL routines as quickly as possible.
}

void main (int argc, char** argv)
{
    glutInit (&argc, argv);                          // Initialize GLUT.
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);    // Set display mode.
    glutInitWindowPosition (50, 100);   // Set top-left display-window position.
    glutInitWindowSize (400, 300);      // Set display-window width and height.
    glutCreateWindow ("An Example OpenGL Program"); // Create display window.

    init ( );                           // Execute initialization procedure.
    glutDisplayFunc (lineSegment);      // Send graphics to display window.
    glutMainLoop ( );                   // Display everything and wait.
}
```





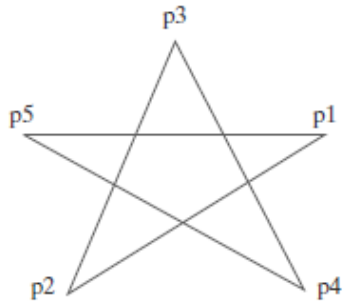
(a)

```
glBegin (GL_LINES);
   glVertex2iv (p1);
   glVertex2iv (p2);
   glVertex2iv (p3);
   glVertex2iv (p4);
   glVertex2iv (p5);
glEnd ( );
```



(b)

```
glBegin (GL_LINE_STRIP)
   glVertex2iv (p1);
   glVertex2iv (p2);
   glVertex2iv (p3);
   glVertex2iv (p4);
   glVertex2iv (p5);
glEnd ( );
```



(c)

```
glBegin (GL_LINE_LOOP);
   glVertex2iv (p1);
   glVertex2iv (p2);
   glVertex2iv (p3);
   glVertex2iv (p4);
   glVertex2iv (p5);
glEnd ( );
```