

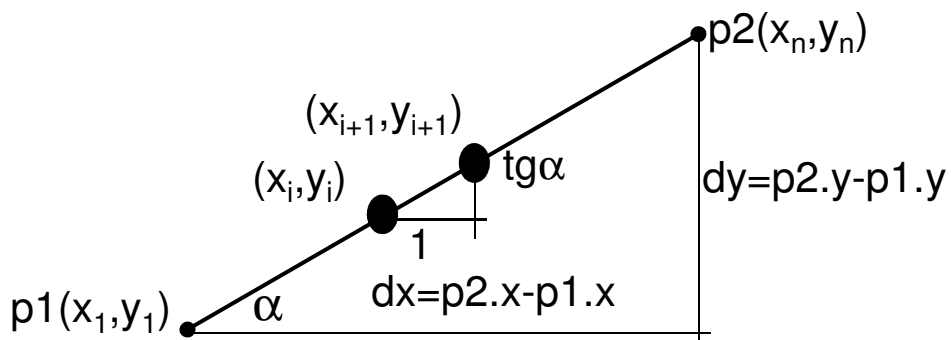
Računarska grafika

Rasterizacija linije



Osnovni inkrementalni algoritam

- Drugi naziv u literaturi *digitalni diferencijalni analizator* (DDA)
- Pretpostavke (privremena ograničenja koja mogu da se otklone jednostavnim uopštavanjem algoritma):
 - 1. nagib (*slope*) linije u granicama: $0 < \text{nagib} < 1$; (nagib = $\text{tg } \alpha$)
 - 2. linija zadata pomoću krajnjih tačaka p1 i p2 za koje važi: $p1.x < p2.x$ (p1 je levo u odnosu na p2)



Algoritam

- Nagib prave je racionalan broj: $\text{nagib} = dy/dx = (p2.y-p1.y)/(p2.x-p1.x)$
- U svakom koraku algoritma, inkrementira se x: $x_{i+1}=x_i+1$
- Važi: $y_{i+1}-y_i = \text{nagib}*(x_{i+1}-x_i) = \text{nagib}*1$
- Sledi: $y_{i+1} = y_i + \text{nagib}$
- Da bi se dobio i-ti piksel potrebno je da se izvrši zaokruživanje dobijene vrednosti y_{i+1}

```
Procedure LineDraw(p1,p2: Point; v: Value);
  Var
    x: Integer; y, nagib: Real;
  Begin
    nagib:= (p2.y - p1.y) / (p2.x - p1.x);
    y:=p1.y;
    For x:=p1.x To p2.x Do Begin
      SetPixel (x, Round(y), v);
      y:=y+nagib
    End
  End;
```

Problemi i rešenja

- Jedan potencijalan problem – kumulativna greška kod dodavanja nagib na y
- Rešenje:

```
Procedure LineDraw(p1,p2: Point; v: Value);
  Var
    x,y: Integer; nagib: Real;
  Begin
    nagib:= (p2.y - p1.y) / (p2.x - p1.x);
    y:=p1.y;
    For x:=p1.x To p2.x Do Begin
      SetPixel(x,y,v);
      y:=p1.y+Round((x-p1.x+1)*nagib)
    End
  End;
```

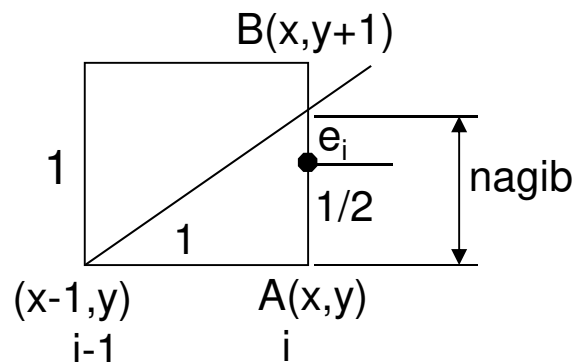
- Veći problem – aritmetika realnih brojeva je spora u odnosu na celobrojnu

Bresenham-ov algoritam

- 1962. J. Bresenham je studirao problem generisanja prave linije na digitalnom ploteru i razvio algoritam koji se koristi i danas
 - algoritam objavljen na konferenciji 1963. i u časopisu 1965.
- Tehnika se zasniva na odlučivanju:
 - da li za inkrementiranu vrednost koordinate X treba da se inkrementira (eventualno dekrementira) vrednost koordinate Y
- Pretpostavke za izvođenje su iste kao kod DDA algoritma:
 - 1. nagib linije u granicama: $0 < \text{nagib} < 1$; (nagib = $\text{tg } \alpha$)
 - 2. linija zadata pomoću krajnjih tačaka p1 i p2 za koje važi: $p1.x < p2.x$ (p1 je levo u odnosu na p2)
- Osnovna pozitivna karakteristika Bresenham-ovog algoritma je da se *Real* aritmetika zamenjuje *Integer* aritmetikom

Pojam greške

- Posmatra se prava koja prolazi kroz tačku (piksel) $(x-1, y)$ i uočavaju se dve sledeće kandidat-tačke: $A(x, y)$ i $B(x, y+1)$

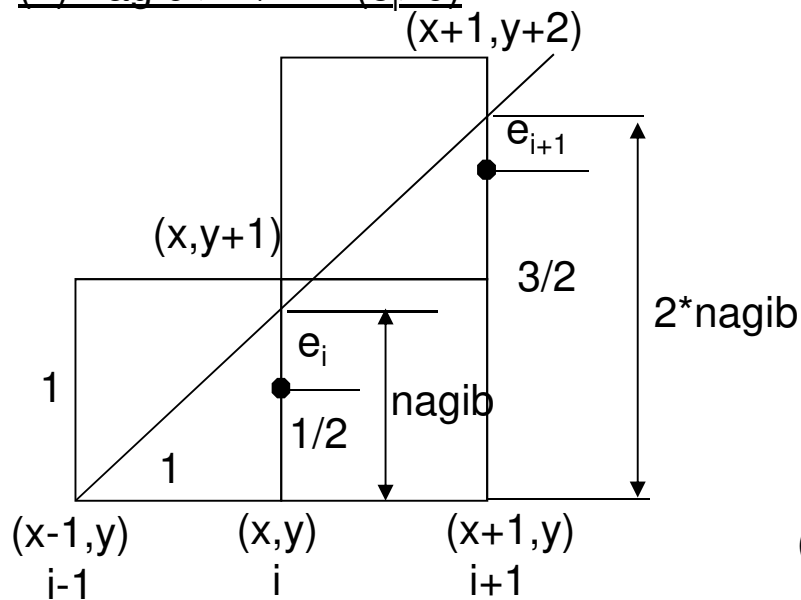


- Uvodi se pojam greške (odstupanja) e stvarne prave od sredine vertikalne duži koja spaja dva piksela ($e = \text{nagib} - 1/2$):
 - $e > 0$, ako prava prolazi kroz "gornju" polovinu duži $\Rightarrow B$
 - $e = 0$, ako prava prolazi kroz polovinu duži $\Rightarrow A$ ili B
 - $e < 0$, ako prava prolazi kroz "donju" polovinu duži $\Rightarrow A$

Slučajevi greške

- Dva osnovna slučaja su:

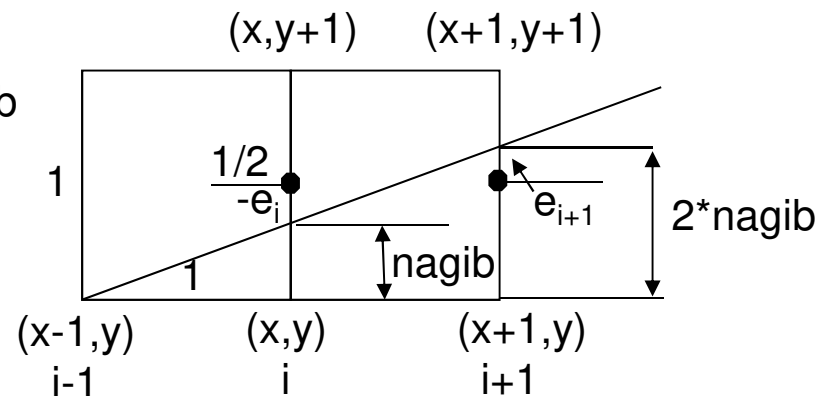
(1) nagib $> 1/2$ ($e_i > 0$)



$$e_i = \text{nagib} - 1/2$$

$$e_{i+1} = 2\text{nagib} - 3/2 = e_i + \text{nagib} - 1$$

(2) nagib $\leq 1/2$ ($e_i \leq 0$)



$$-e_i = 1/2 - \text{nagib} \Rightarrow e_i = \text{nagib} - 1/2$$

$$e_{i+1} = 2\text{nagib} - 1/2 = e_i + \text{nagib}$$

Izvođenje

- I dalje je nagib realan broj, a cilj je da algoritam bude celobrojni
 - (1) nagib > 1/2 ($e_i > 0$)
 - $e_i = \text{nagib} - 1/2$
 - $e_{i+1} = e_i + \text{nagib} - 1$
 - $e_i = dy/dx - 1/2 \quad /*2dx$
 - $e_{i+1} = e_i + dy/dx - 1 \quad /*2dx$
 - $2dx e_i = 2dy - dx$
 - $2dx e_{i+1} = 2dx e_i + 2dy - 2dx$
 - (2) nagib ≤ 1/2 ($e_i ≤ 0$)
 - $e_i = \text{nagib} - 1/2$
 - $e_{i+1} = e_i + \text{nagib}$
 - $e_i = dy/dx - 1/2 \quad /*2dx$
 - $e_{i+1} = e_i + dy/dx \quad /*2dx$
 - $2dx e_i = 2dy - dx$
 - $2dx e_{i+1} = 2dx e_i + 2dy$
- Za odlučivanje je dovoljan znak greške, apsolutna vrednost nije bitna:
 - $e'_i = 2dy - dx$
 - $e'_{i+1} = e'_i + 2(dy - dx)$
 - $e'_i = 2dy - dx$
 - $e'_{i+1} = e'_i + 2dy$

Algoritam sa ograničenjima

- Pseudokod:

```
x:=p1.x; y:=p1.y; dy:= (p2.y - p1.y); dx:= (p2.x - p1.x);
e:= 2*dy - dx;      {Integer!}
for i:=1 to dx do
  begin
    SetPixel(x,y,v);      // postavlja piksel (x,y) na vrednost v
    if e > 0 then begin y:= y+1; e:= e+2*(dy-dx) end
                else e:= e + 2*dy;
    x:= x+1;
  end;
  SetPixel(x,y,v);
```

- Ograničenje (1)

- može da se ukloni da algoritam važi za: $0 \leq \text{nagib} \leq +\infty$,
tako što se za strme prave ($1 \leq \text{nagib} \leq +\infty$) zamene vrednosti za x i y prilikom crtanja tačke

- Ograničenje (2)

- uklanja se tako što crtanje uvek počinje od tačke sa manjom x koordinatom

Kompletan algoritam (1)

- Kompletan Bresenham-ov algoritam za prave sa ne-negativnim nagibom (nagib>0):

```
Procedure BresenhamLineDraw(p1,p2: Point; v: Value);
  Var
    dx,dy,x,y,incr1,incr2,xend,e,t: Integer;  strma: Boolean;
  Begin
    dx:= Abs(p2.x - p1.x);          dy:= Abs(p2.y - p1.y);
    strma:= dy > dx;
    If strma Then
      Begin
        t:= dx;  dx:= dy;          dy:= t;
        t:= p1.x; p1.x:= p1.y; p1.y:= t;
        t:= p2.x; p2.x:= p2.y; p2.y:= t;
      End;
    incr1:=2*(dy-dx);  incr2:= 2*dy;  e:=2*dy - dx;
```

Kompletan algoritam (2)

```
{nastavak}
If p1.x > p2.x Then
  Begin x:= p2.x; y:= p2.y; xend:= p1.x End
Else
  Begin x:= p1.x; y:= p1.y; xend:= p2.x End;
While x < xend Do
  Begin
    If strma Then SetPixel(y,x,v) Else SetPixel(x,y,v);
    If e > 0 Then Begin y:= y + 1; e:= e + incr1 End
    Else e:= e + incr2;
    x:= x + 1
  End;
  If strma Then SetPixel(y,x,v) Else SetPixel(x,y,v)
End;
```

- Slučaj za $\text{nagib} < 0$ rešiti kroz samostalno vežbanje.
- Rasterizacija kružnice – na vežbama