Algoritmi de rasterizare a liniilor

Victor Moraru

victor.moraru@calc.utm.md

Planul cursului

Problema rasterizării liniilor

Considerații

Ecuațiile liniilor

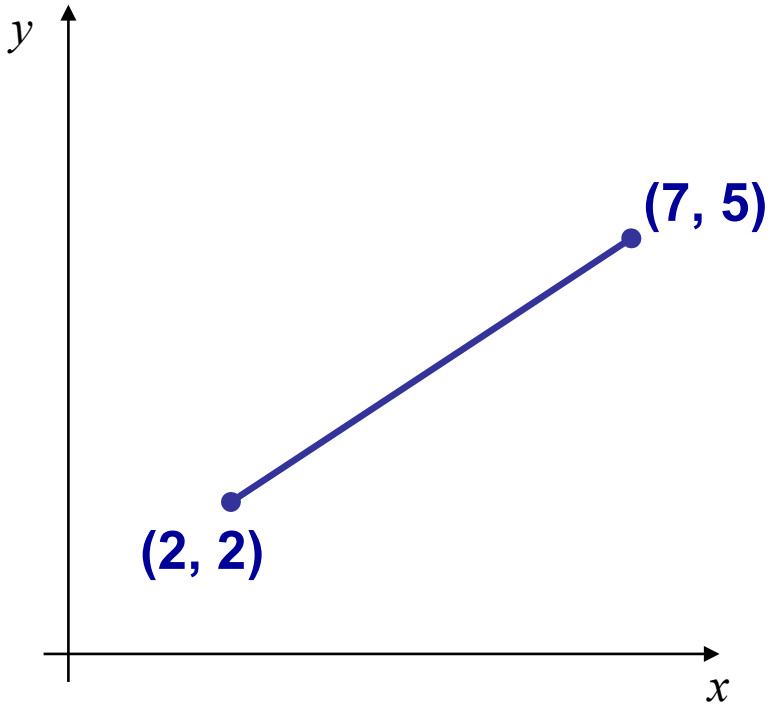
Algoritmi de rasterizare

- Soluție naiva
- Algoritmul DDA
- Algoritmul Bresenham

Concluzii

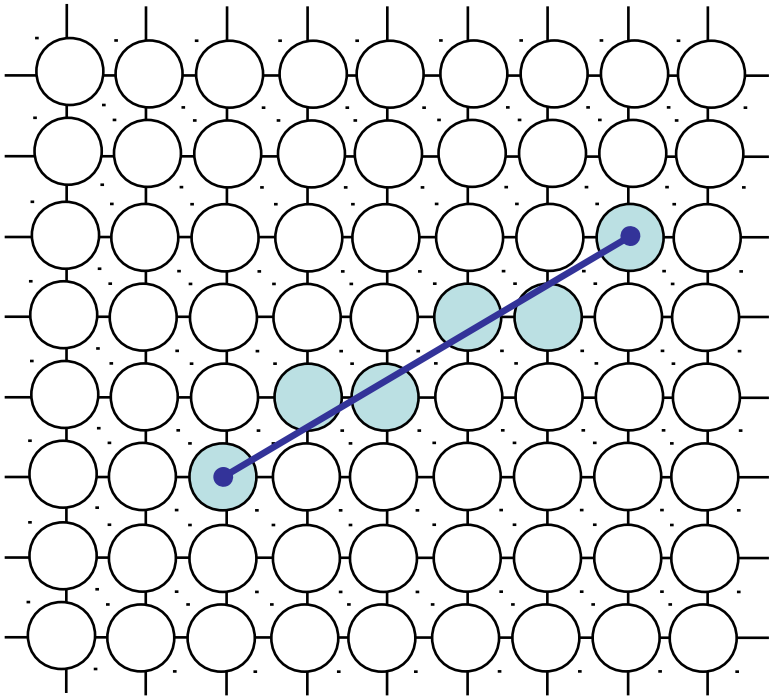
Problema rasterizarii liniilor

Un segment de linie este definit prin coordonatele punctelor de început și de sfârșit



Problema rasterizarii liniilor

Ce se întâmplă atunci când desenam acest segment pe un ecran bazat pe pixeli?



Cum alegem pixelii care for fi colorati?

Câteva considerații

Considerații:

- Linia trebuie sa aibă un aspect corect
  - Evitam treptele
- Procesul trebuie sa fie "ușor" și rapid !
  - Cate linii utilizam pentru a desena o scena?

    In linii generale, foarte multe...

Câteva considerații

Considerații:

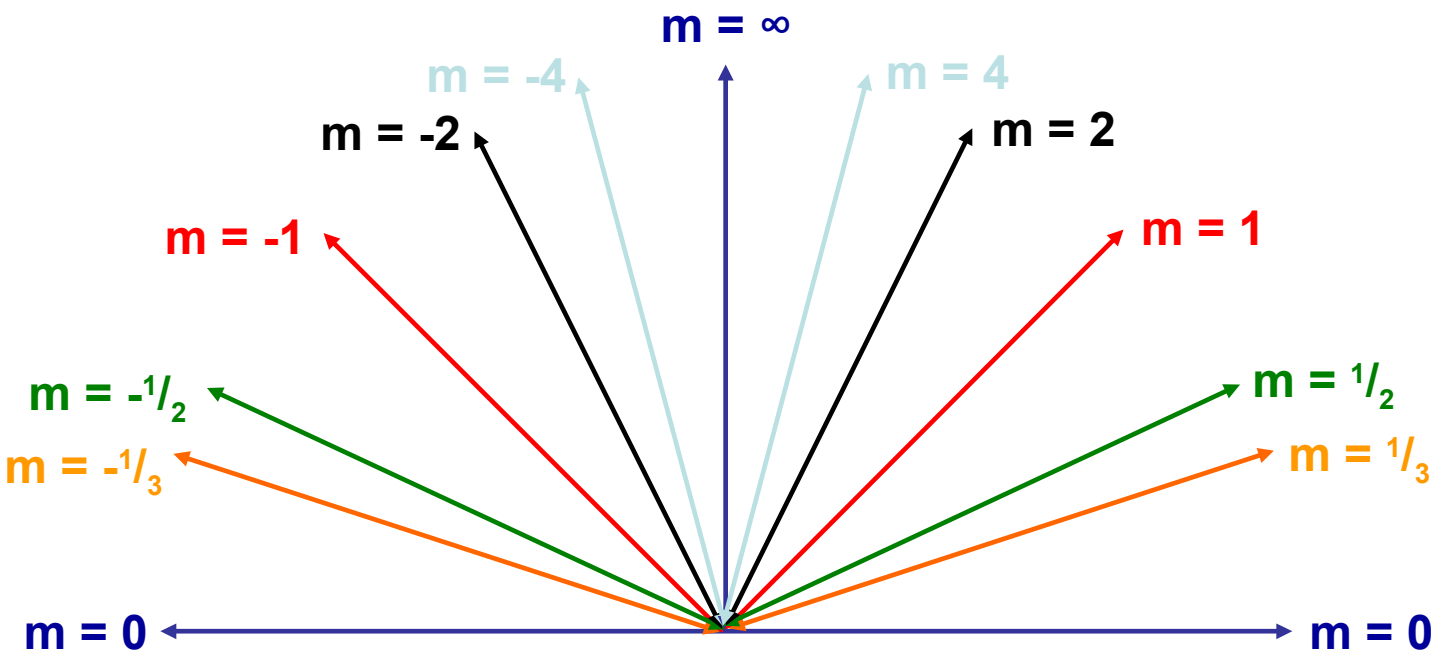
- Linia trebuie sa aibă un aspect corect
  - Evitam treptele
- Procesul trebuie sa fie "ușor" și rapid !
  - Cate linii utilizam pentru a desena o scena?

    In linii generale, foarte multe...

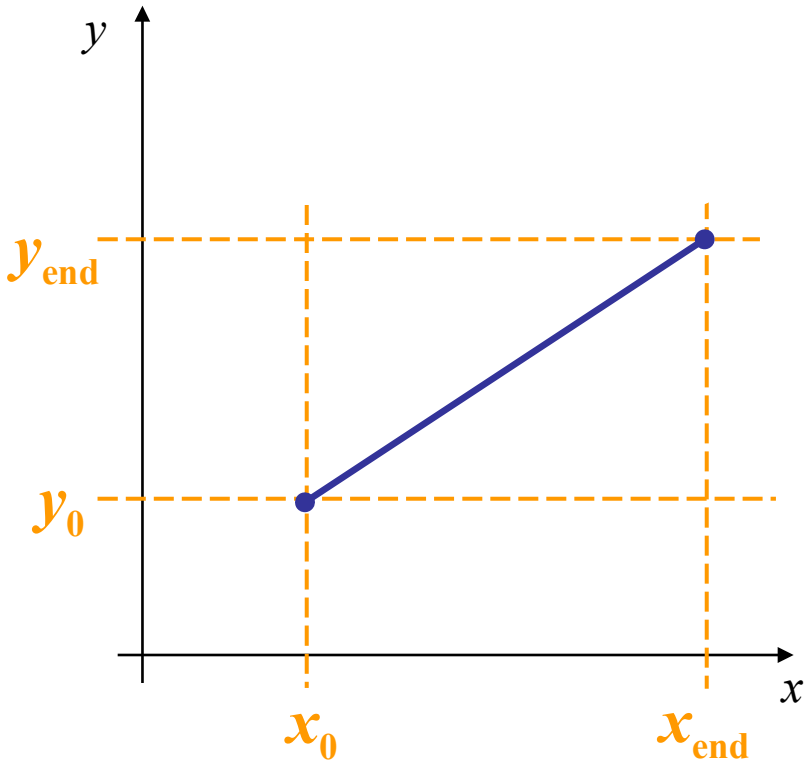
Linii și pante

Panta liniei (m) e definită prin coordonatele punctelor prin care o descriem

Câteva exemple de pante



Ecuațiile liniei



$$y = m \cdot x + b$$

unde :

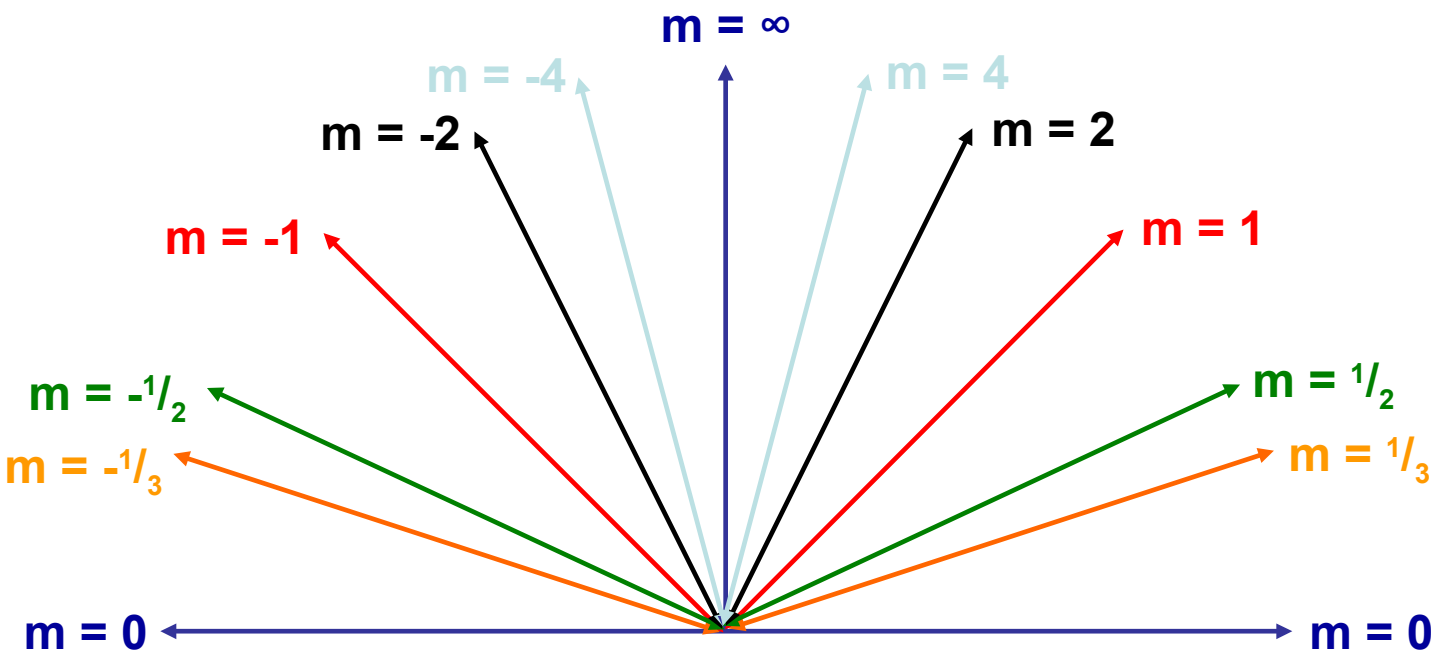
$$m = \frac{y_{end} - y_0}{x_{end} - x_0}$$

$$b = y_0 - m \cdot x_0$$

Linii și pante

Panta liniei (m) e definită prin coordonatele punctelor prin care o descriem

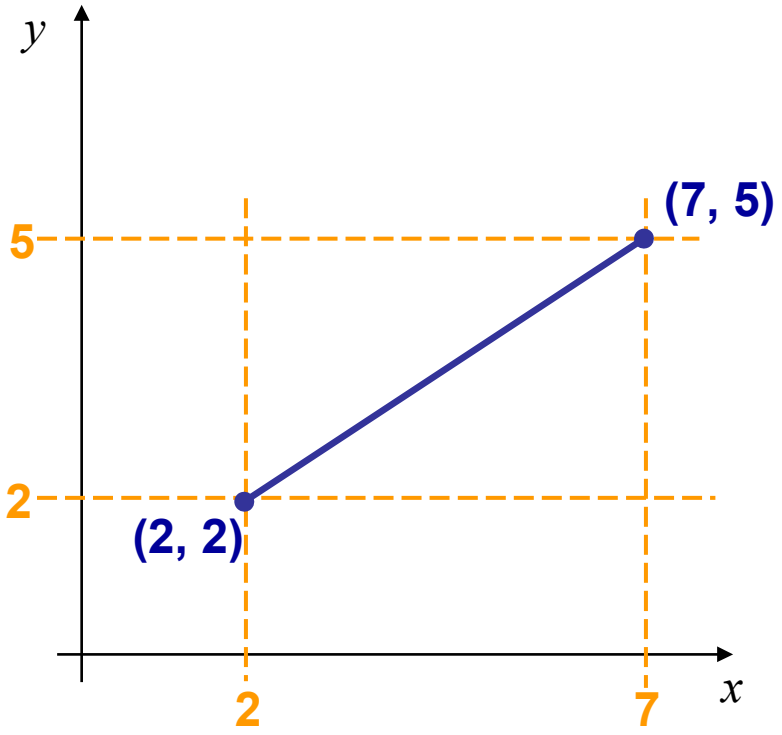
Câteva exemple de pante



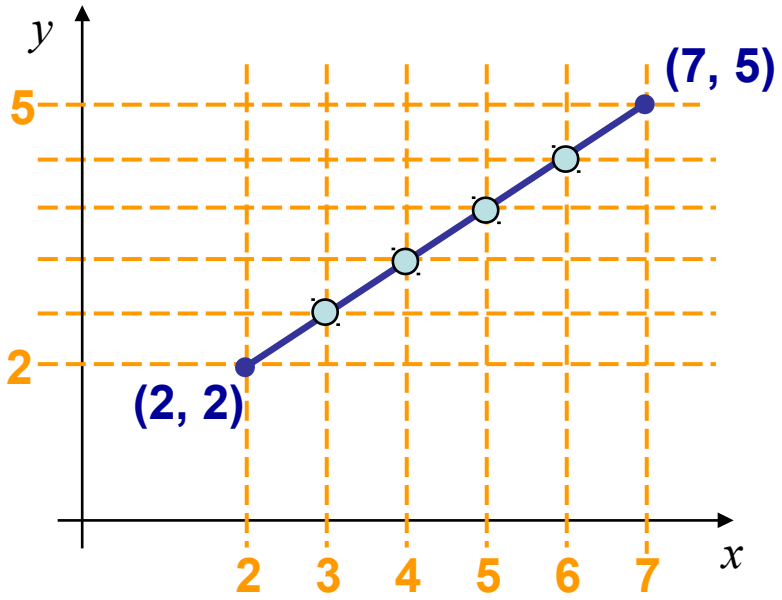
O soluție simplă

Am putea pur și simplu calcula y pentru fiecare valoare a coordonatei x

Să considerăm următorul exemplu:



O soluție simplă



Calculăm m si b:

$$m = \frac{5-2}{7-2} = \frac{3}{5}$$

$$b = 2 - \frac{3}{5} * 2 = \frac{4}{5}$$

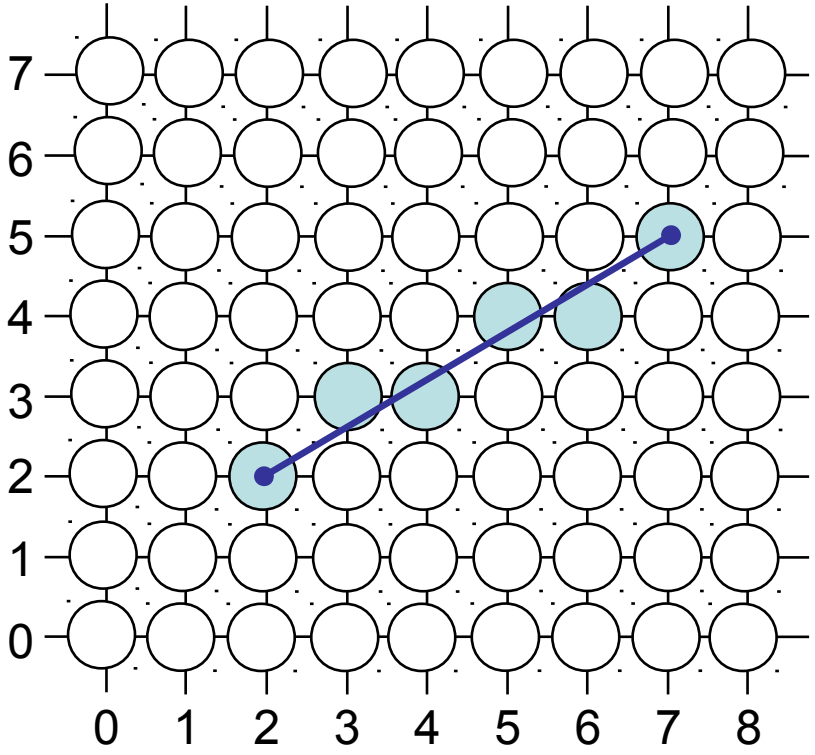
Pentru fiecare x calculăm y :

$$y(3) = \frac{3}{5} \cdot 3 + \frac{4}{5} = 2\frac{3}{5} \qquad y(4) = \frac{3}{5} \cdot 4 + \frac{4}{5} = 3\frac{1}{5}$$

$$y(5) = \frac{3}{5} \cdot 5 + \frac{4}{5} = 3\frac{4}{5} \qquad y(6) = \frac{3}{5} \cdot 6 + \frac{4}{5} = 4\frac{2}{5}$$

O soluție simplă

Rotunjim rezultatele şi colorăm pixelii



$$y(3) = 2\frac{3}{5} \approx 3$$

$$y(4) = 3\frac{1}{5} \approx 3$$

$$y(5) = 3\frac{4}{5} \approx 4$$

$$y(6) = 4\frac{2}{5} \approx 4$$

O soluție simplă

Această soluție este relativ lenta deoarece:

- Ecuația $y = mx + b$ necesită înmulțirea m cu x
- Rotunjirea rezultatelor pentru a obține coordonata y e și ea lenta

Vom căuta o soluție mai rapidă

Câteva cuvinte despre pantă

In exemplul precedent am calculat y pentru fiecare valoare a lui x

Care ar fi o alta soluție? Calculam x pentru fiecare y dat

Obținem: $$x = \frac{y - b}{m}$$

unde: $$m = \frac{y_{end} - y_0}{x_{end} - x_0}$$ și $$b = y_0 - m \cdot x_0$$

O soluție simplă

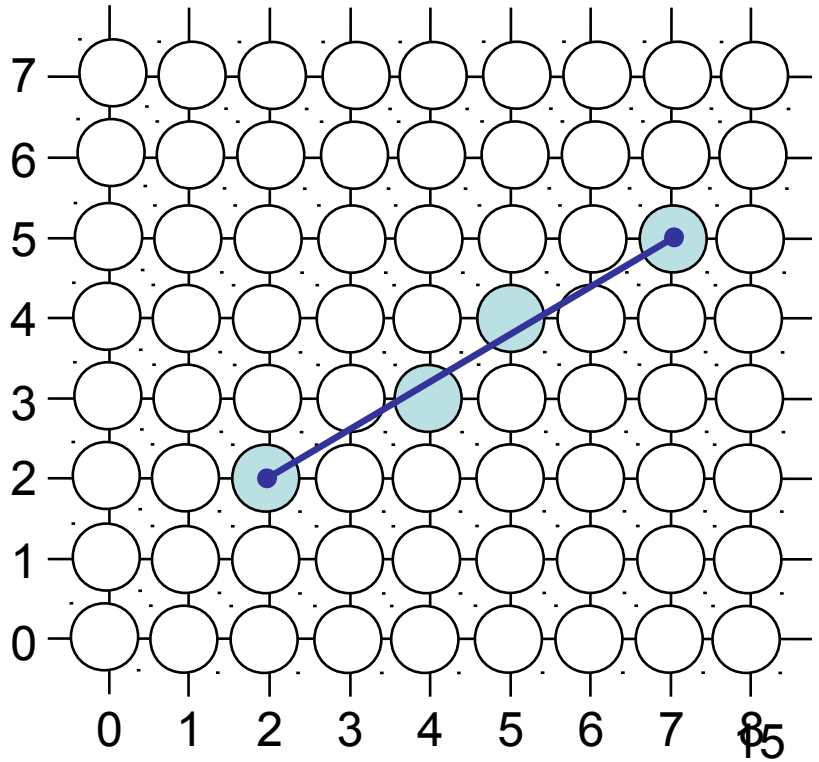
Am obține:

$$x(3)=3\frac{2}{3}\approx 4 \qquad x(4)=5\frac{1}{3}\approx 5$$

Ceea ce nu merge prea bine !

Cum sa alegem metoda cea mai buna?

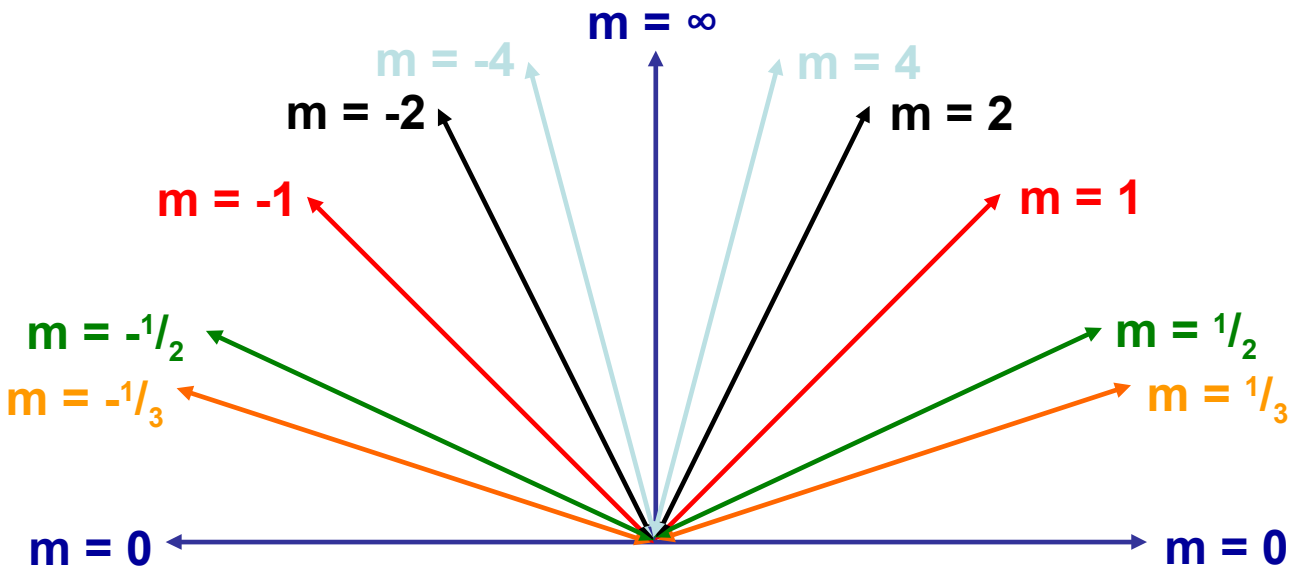
Panta liniei ar putea fi un indiciu...



Câteva cuvinte despre pantă

Dacă valoarea pantei e intre -1 și 1 vom calcula y bazându-ne pe coordonata x

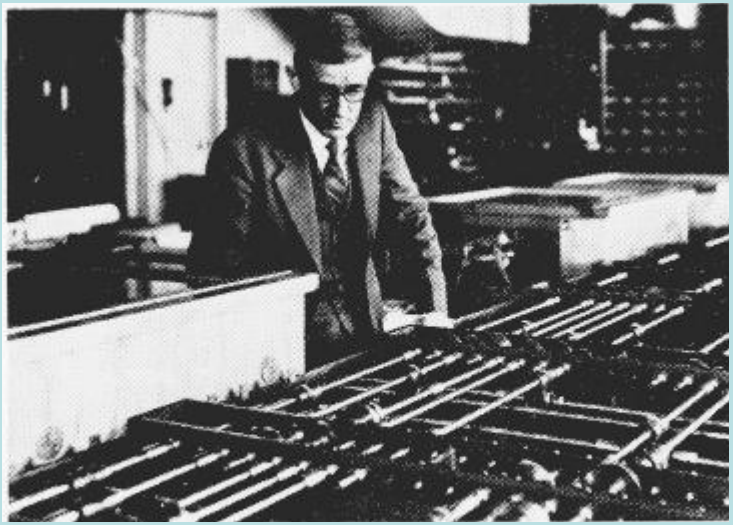
In caz contrar vom proceda invers



Algoritmul DDA

Algorimul DDA (*digital differential analyzer)* utilizeaza un algoritm incremental pentru a accelera conversia

y_{k+1} se calculează doar in baza y_k



Analizorul diferential original a fost o mașină fizică dezvoltata de Vannevar Bush la MIT in anii 1930, în scopul de a rezolva ecuații diferențiale ordinare.

Algoritmul DDA

Fie lista de puncte care determină linia din exemplul precedent:

(2, 2), (3, $2^{3}/_{5}$), (4, $3^{1}/_{5}$), (5, $3^{4}/_{5}$), (6, $4^{2}/_{5}$), (7, 5)

Remarcați ca coordonatele x sunt modificate regulat, una cate una, iar coordonatele y sunt modificate in funcție de panta liniei

Algoritmul DDA are la bază anume acest fenomen

Algoritmul DDA

Când panta liniei e intre -1 și 1 linia începe la primul punct urmând incrementarea coordonatei x cu 1, coordonata y se calculează după cum urmează:

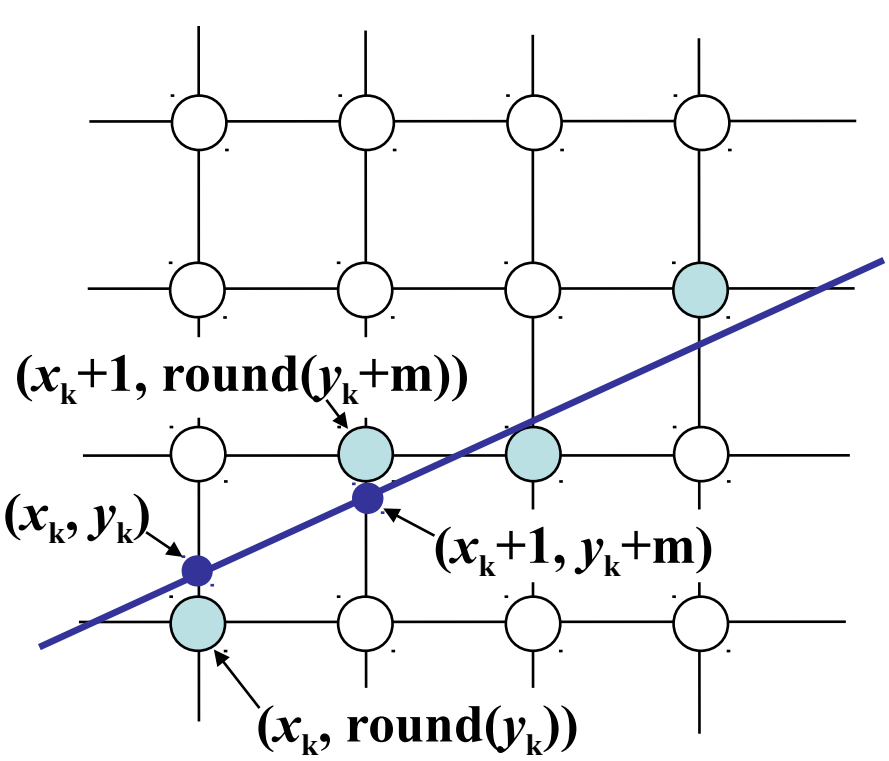
$$y_{k+1} = y_k + m$$

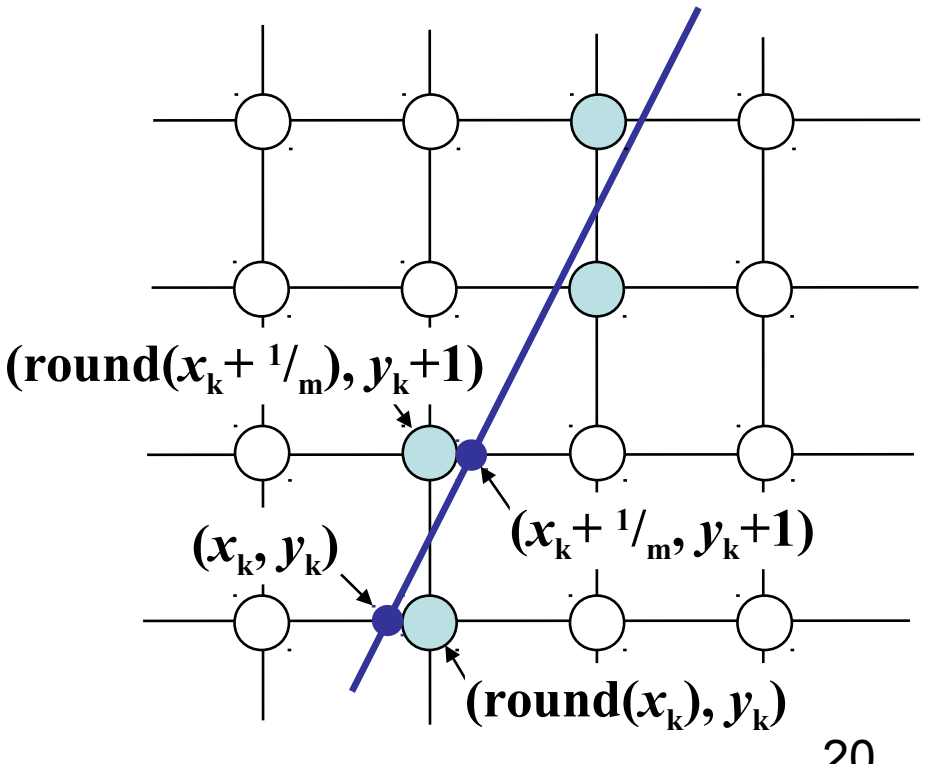
Când panta e in afara limitelor din primul caz, incrementăm y cu 1 și calculam x in felul următor:

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

Algoritmul DDA

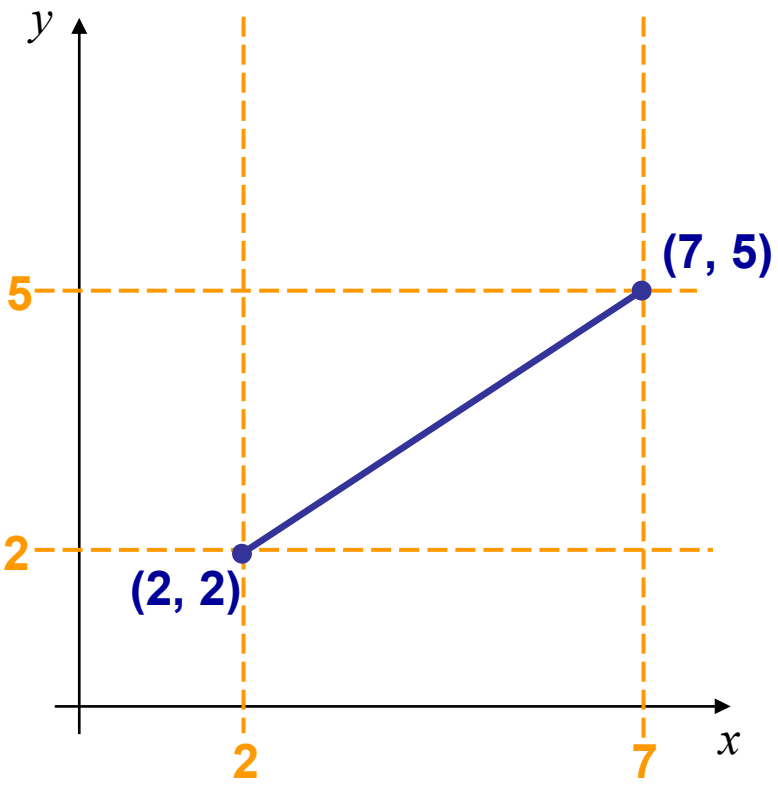
Valorile calculate vor trebui rotunjite ca sa corespundă pixelilor

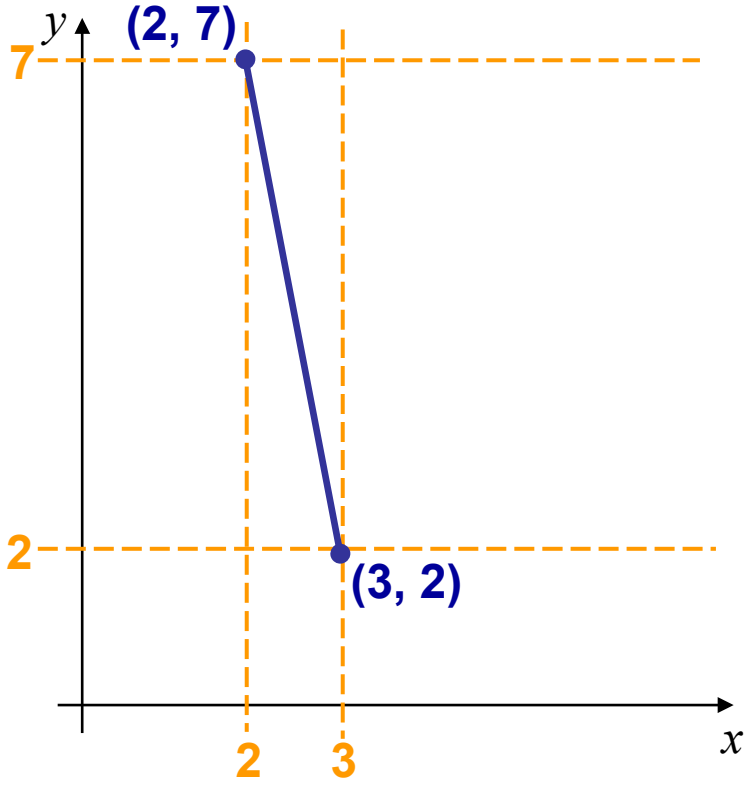




Algoritmul DDA

Încercați pentru exemplele următoare:





Algoritmul DDA: concluzii

Algoritmul DDA e mai rapid decât metoda precedentă

- Nu mai avem nici o înmulțire la calcularea coordonatelor

Din contra, exista două probleme majore:

- Acumularea erorilor de rotunjire poate duce pixelizarea departe de ceea ce dorim noi
- Operațiile de rotunjire implică aritmetica în virgulă flotantă ceea ce consumă multe resurse de calcul

Concluzii pentru a continua...

Operațiile de rasterizare a liniilor sunt consumatoare de timp ---> trebuie sa găsim metode bune pentru a le realiza

Algoritmul DDA este destul de bun, dar putem găsi soluții și mai bune: nu ne oprim aici...

Algoritmul Bresenham

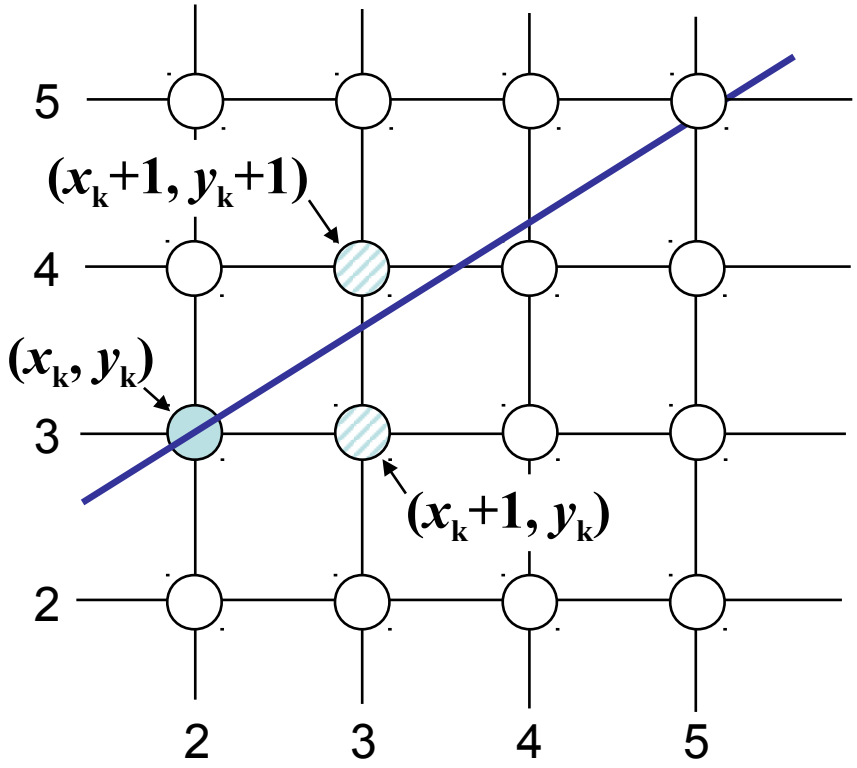
Algoritmul Bresenham este un alt algoritm incremental de rasterizare

Marele lui avantaj este că folosește doar calcule cu numere întregi

Jack Bresenham a lucrat timp de 27 de ani la IBM înainte de a intra mediul academic. Bresenham a dezvoltat algoritmi lui celebri de la IBM la începutul anilor 1960

Algoritmul Bresenham: idea

Ne deplasăm de-a lingul axei x cu pasi egali cu o unitate si de fiecare dată alegem dintre doua valori ale lui y

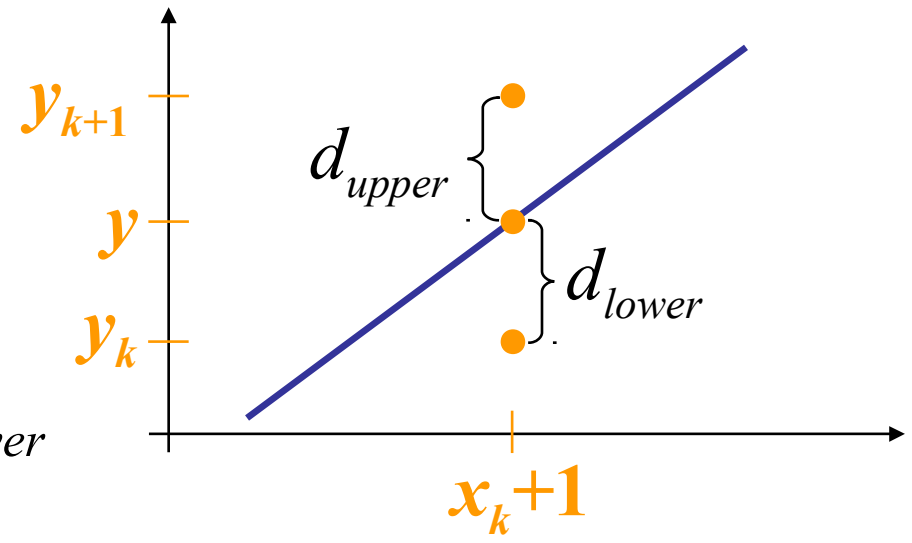


De exemplu, fiind in pozitia (2,3) vom avea de ales între pixelii (3,3) si (3,4)

Vom dori sa alegem pixelul care e cel mai aproape de linia originală

Derivarea algoritmului lui Bresenham

Aflandu-ne in poziția x_k+1 vom nota diferențele dintre linia matematica şi puctele posibile prin d_{upper} si d_{lower}



Coordonata y a liniei matematice la pasul x_k+1 va fi:

$$y = m(x_k + 1) + b$$

Derivarea algoritmului lui Bresenham

Prin urmare, d_{upper} si d_{lower} se vor calcula după cum urmează:

$$d_{lower} = y - y_k$$
$$= m(x_k + 1) + b - y_k$$

si

$$d_{upper} = (y_k + 1) - y$$
$$= y_k + 1 - m(x_k + 1) - b$$

Putem utiliza aceasta pentru a lua decizia despre pixelul cela mai apropiat de linia matematica

Derivarea algoritmului lui Bresenham

Decizia se bazează pe diferența dintre pozițiile a doi pixeli:

$$d_{lower} - d_{upper} = 2m(x_k + 1) - 2y_k + 2b - 1$$

Înlocuim m cu $\Delta y/\Delta x$ unde Δx si Δy sunt diferențele dintre punctele la extremități:

$$\Delta x(d_{lower} - d_{upper}) = \Delta x(2\frac{\Delta y}{\Delta x}(x_k + 1) - 2y_k + 2b - 1)$$

$$= 2\Delta y \cdot x_k - 2\Delta x \cdot y_k + 2\Delta y + \Delta x(2b - 1)$$

$$= 2\Delta y \cdot x_k - 2\Delta x \cdot y_k + c$$

Derivarea algoritmului lui Bresenham

Prin urmare, parametrul de decizie p_k pentru pasul *k* de-a lungul liniei este calculat in felul următor:

$$p_k = \Delta x(d_{lower} - d_{upper})$$
$$= 2\Delta y \cdot x_k - 2\Delta x \cdot y_k + c$$

Semnul parametrului deczional p_k coincide cu cel al diferenței $d_{lower} - d_{upper}$

Daca p_k e negativ, alegem pixelul de jos, un caz contrar il vom selecta pe celalalt

Derivarea algoritmului lui Bresenham

Să ne amintim ca modificarea coordonatelor dea lungul axei x se efectueaza cu pasi unitari, deci toate calculele se efectueaza cu numere intregi.

La pasul k+1

$$p_{k+1} = 2\Delta y \cdot x_{k+1} - 2\Delta x \cdot y_{k+1} + c$$

Scazand p_k din aceasta, obtinem:

$$p_{k+1} - p_k = 2\Delta y(x_{k+1} - x_k) - 2\Delta x(y_{k+1} - y_k)$$

Derivarea algoritmului lui Bresenham

Dar $x_{k+1} = x_k + 1$, prin urmare:

$$p_{k+1} = p_k + 2\Delta y - 2\Delta x(y_{k+1} - y_k)$$

unde $y_{k+1} - y_k$ este sau 0 sau 1 in functie de semnul p_k

Primul parametru decizional p_0 este evaluat in (x_0, y_0) si este calcultat prin formula:

$$p_0 = 2\Delta y - \Delta x$$

Algoritmul Bresenham pentru trasarea liniilor

ALGORITMUL BRESENHAM PENTRU LINII
(pentru $|m| < 1.0$)

1. Introducem cele două puncte de la extremități, salvăm punctul din stânga în (x_0, y_0)
2. Desenăm punctul (x_0, y_0)
3. Calculăm constantele Δx, Δy, $2\Delta y$, si $(2\Delta y - 2\Delta x)$ şi primim prima valoare a pametrului de decizie :
$$p_0 = 2\Delta y - \Delta x$$
4. Pentru fiecare x_k de-a lungul liniei, incepând cu $k = 0$, realizam testul urmator. Daca $p_k < 0$, puctul următor care va fi desenat este (x_k+1, y_k) şi :
$$p_{k+1} = p_k + 2\Delta y$$

Algoritmul Bresenham pentru trasarea liniilor

In caz contrar, următorul punct care va fi desenat este (x_k+1, y_k+1) si:

$$p_{k+1} = p_k + 2\Delta y - 2\Delta x$$

5. Repetăm pasul 4 (Δx – 1) ori

Atentie! Derivarea algoritmului de mai sus a presupus ca panta este mai mica sau egală cu 1. Pentru alte pante va fi necesar sa ajustam usor algoritmul

Exemplu Bresenham

Sa desenam linia dintre punctele (20,10) si (30,18)

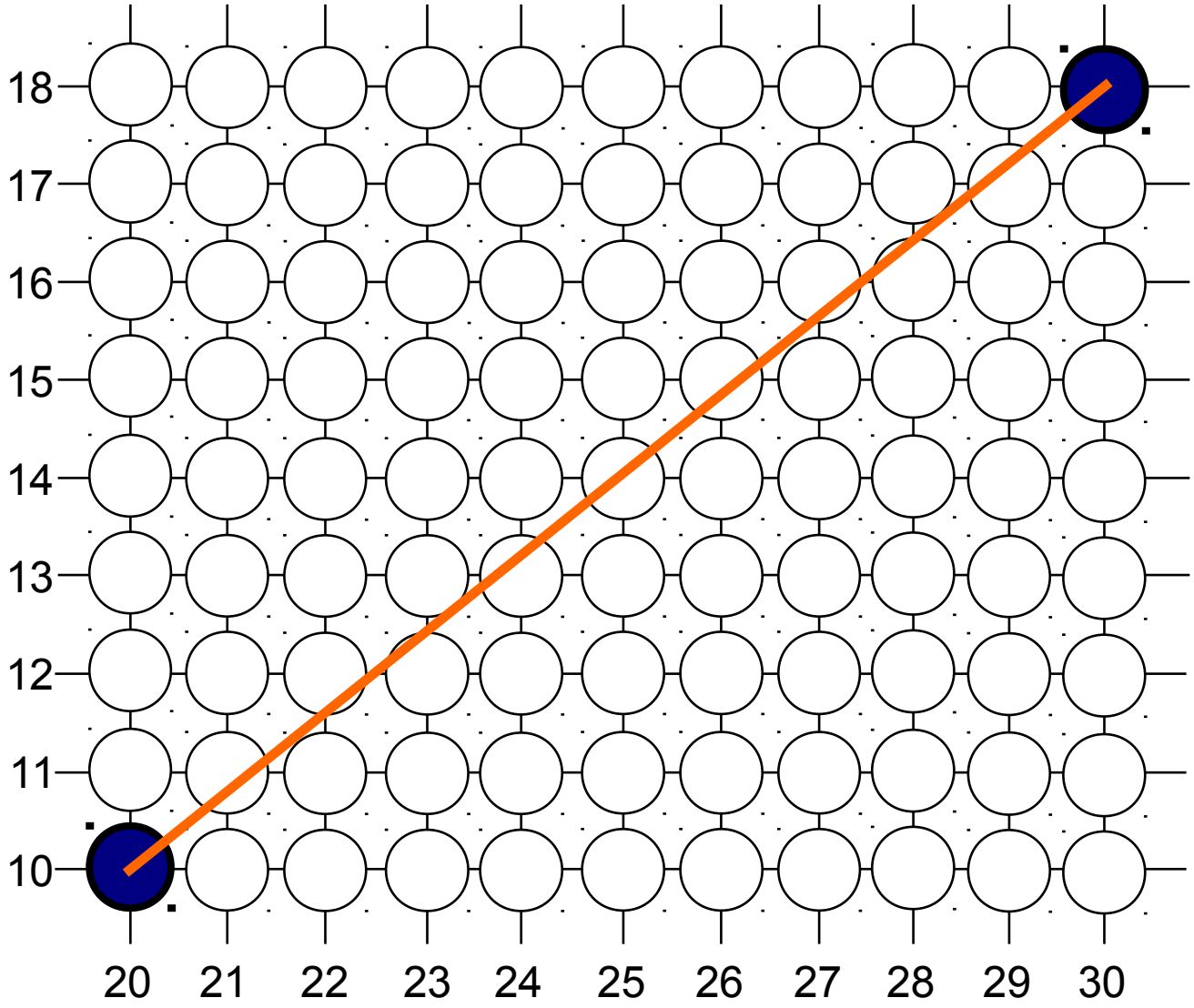
In primul rand calculam toate constantele:

- Δx: 10
- Δy: 8
- $2\Delta y$: 16
- $2\Delta y$ - $2\Delta x$: -4

Calculam valoarea initiala a parametrului p_0:

- $p0 = 2\Delta y - \Delta x$ = 6

Exemplu Bresenham

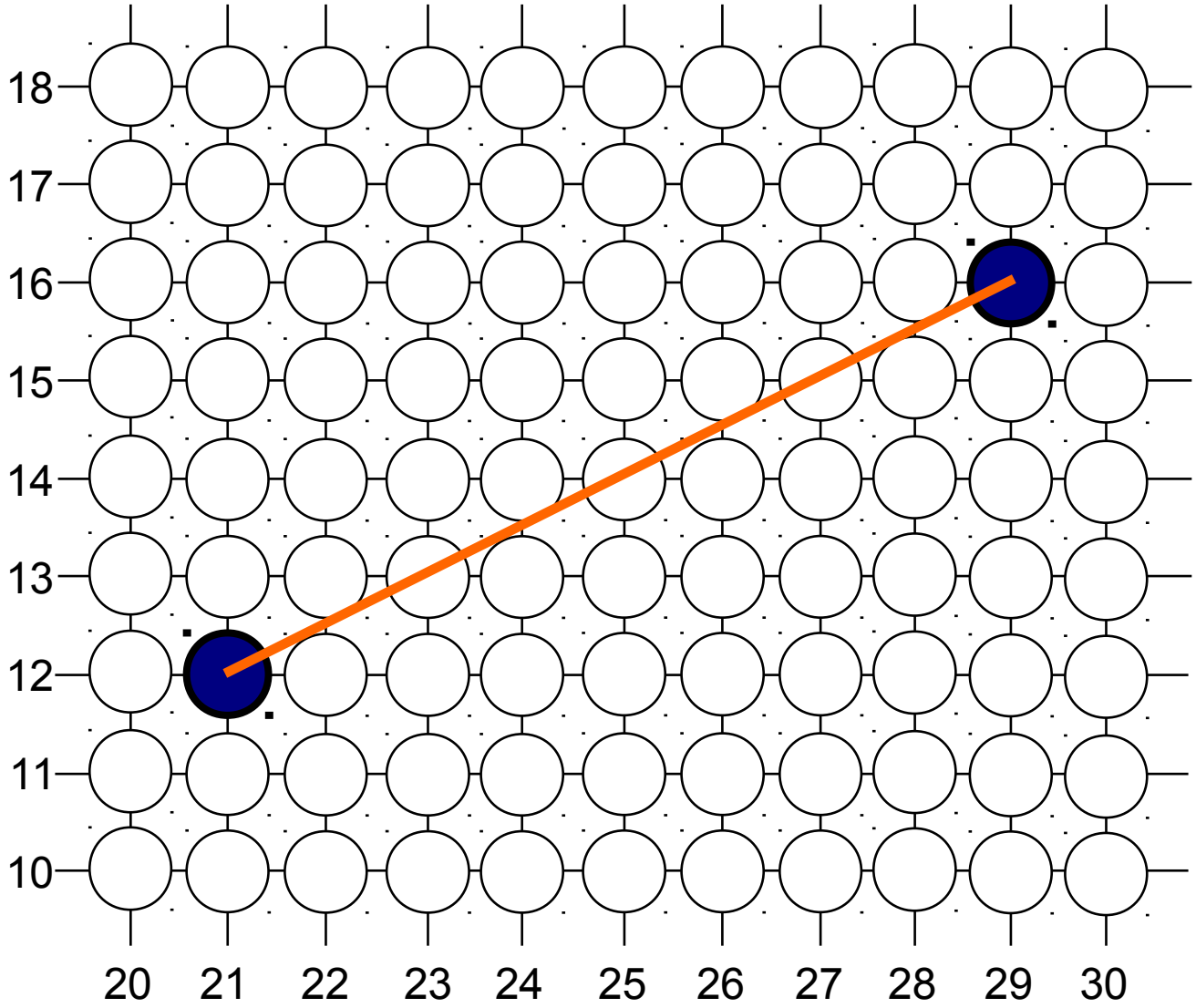


k	p_k	(x_{k+1},y_{k+1})
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		35

Exemplu Bresenham

Utilizati algoritmul Bresenham pentru a desena linia ce pleacă din punctul (21,12) si ajunge în punctul (29,16)

Exemplu Bresenham



k	p_k	(x_{k+1},y_{k+1})
0		
1		
2		
3		
4		
5		
6		
7		
8		
		37

Concluzii

Algoritmul Bresenham pentru trasarea liniilor are urmatoarele avantaje:

- Algoritm incremental rapid
- Foloseste doar operatii cu numere intregi

Comparand-l cu algoritmul DDA, cel din urma are urmatoarel probleme:

- Acumularea erorilor de rotunjire poate duce procesul de pixelizarea departe de ceea ce am dori noi
- Operațiile de rotunjire implica aritmetica in virgula flotantă ceea ce consuma multe resurse de calcul

Intrebări ?