Scan-conversion

Marian Ioan MUNTEANU

Al.I.Cuza University of Iasi, Romania
webpage: http://www.math.uaic.ro/∼munteanu

15 Octombrie 2012

Cuprins

Scan-conversion pentru segmente de dreaptă

Algoritmul de rasterizare pentru un segment de dreaptă trebuie să calculeze coordonatele pixelilor care se află pe linia ideală sau care sunt cât mai aproape posibil de aceasta:

- secvența de pixeli aproximează cel mai bine linia
- să fie în același timp cât mai "dreaptă" posibil

Vom considera drepte având **panta cuprinsă între** -1 **și** 1
$\Longrightarrow$ aproximarea va trebui să conțină exact un pixel pe fiecare coloană

- dreapta este "mai orizontală decât verticală"
- creșterea pe axa Ox este mai mare decât creșterea pe axa Oy

Axa Ox mai poartă numele de *driving axis (DA)* fiind axa de control pentru algoritm, în timp ce axa Oy este numită *passive axis (PA)* și coordonata corespunzătoare este incrementată atât cât e nevoie.

Scan-conversion pentru segmente de dreaptă

Algoritmul de rasterizare pentru un segment de dreaptă trebuie să calculeze coordonatele pixelilor care se află pe linia ideală sau care sunt cât mai aproape posibil de aceasta:

- secvența de pixeli aproximează cel mai bine linia
- să fie în același timp cât mai "dreaptă" posibil

Vom considera drepte având **panta cuprinsă între** -1 **și** 1
$\Longrightarrow$ aproximarea va trebui să conțină exact un pixel pe fiecare coloană

- dreapta este "mai orizontală decât verticală"
- creșterea pe axa Ox este mai mare decât creșterea pe axa Oy

Axa Ox mai poartă numele de *driving axis (DA)* fiind axa de control pentru algoritm, în timp ce axa Oy este numită *passive axis (PA)* și coordonata corespunzătoare este incrementată atât cât e nevoie.

Scan-conversion pentru segmente de dreaptă

Algoritmul de rasterizare pentru un segment de dreaptă trebuie să calculeze coordonatele pixelilor care se află pe linia ideală sau care sunt cât mai aproape posibil de aceasta:

- secvența de pixeli aproximează cel mai bine linia
- să fie în același timp cât mai "dreaptă" posibil

Vom considera drepte având **panta cuprinsă între** -1 **și** 1
$\Longrightarrow$ aproximarea va trebui să conțină exact un pixel pe fiecare coloană

- dreapta este "mai orizontală decât verticală"
- creșterea pe axa Ox este mai mare decât creșterea pe axa Oy

Axa Ox mai poartă numele de *driving axis (DA)* fiind axa de control pentru algoritm, în timp ce axa Oy este numită *passive axis (PA)* și coordonata corespunzătoare este incrementată atât cât e nevoie.

Scan-conversion pentru segmente de dreaptă

Algoritmul de rasterizare pentru un segment de dreaptă trebuie să calculeze coordonatele pixelilor care se află pe linia ideală sau care sunt cât mai aproape posibil de aceasta:

- secvența de pixeli aproximează cel mai bine linia
- să fie în același timp cât mai "dreaptă" posibil

Vom considera drepte având **panta cuprinsă între -1 și 1**

$\Longrightarrow$ aproximarea va trebui să conțină exact un pixel pe fiecare coloană

- dreapta este "mai orizontală decât verticală"
- creșterea pe axa Ox este mai mare decât creșterea pe axa Oy

Axa Ox mai poartă numele de *driving axis (DA)* fiind axa de control pentru algoritm, în timp ce axa Oy este numită *passive axis (PA)* și coordonata corespunzătoare este incrementată atât cât e nevoie.

Scan-conversion pentru segmente de dreaptă

Algoritmul de rasterizare pentru un segment de dreaptă trebuie să calculeze coordonatele pixelilor care se află pe linia ideală sau care sunt cât mai aproape posibil de aceasta:

- secvența de pixeli aproximează cel mai bine linia
- să fie în același timp cât mai "dreaptă" posibil

Vom considera drepte având **panta cuprinsă între -1 și 1**
$\Longrightarrow$ aproximarea va trebui să conțină exact un pixel pe fiecare coloană

- dreapta este "mai orizontală decât verticală"
- creșterea pe axa Ox este mai mare decât creșterea pe axa Oy

Axa Ox mai poartă numele de *driving axis (DA)* fiind axa de control pentru algoritm, în timp ce axa Oy este numită *passive axis (PA)* și coordonata corespunzătoare este incrementată atât cât e nevoie.

Scan-conversion pentru segmente de dreaptă

Algoritmul de rasterizare pentru un segment de dreaptă trebuie să calculeze coordonatele pixelilor care se află pe linia ideală sau care sunt cât mai aproape posibil de aceasta:

- secvența de pixeli aproximează cel mai bine linia
- să fie în același timp cât mai "dreaptă" posibil

Vom considera drepte având **panta cuprinsă între -1 și 1**
$\Longrightarrow$ aproximarea va trebui să conțină exact un pixel pe fiecare coloană

- dreapta este "mai orizontală decât verticală"
- creșterea pe axa Ox este mai mare decât creșterea pe axa Oy

Axa Ox mai poartă numele de *driving axis (DA)* fiind axa de control pentru algoritm, în timp ce axa Oy este numită *passive axis (PA)* și coordonata corespunzătoare este incrementată atât cât e nevoie.

Scan-conversion pentru segmente de dreaptă

Algoritmul de rasterizare pentru un segment de dreaptă trebuie să calculeze coordonatele pixelilor care se află pe linia ideală sau care sunt cât mai aproape posibil de aceasta:

- secvența de pixeli aproximează cel mai bine linia
- să fie în același timp cât mai "dreaptă" posibil

Vom considera drepte având **panta cuprinsă între -1 și 1**
$\Longrightarrow$ aproximarea va trebui să conțină exact un pixel pe fiecare coloană

- dreapta este "mai orizontală decât verticală"
- creșterea pe axa Ox este mai mare decât creșterea pe axa Oy

Axa Ox mai poartă numele de *driving axis (DA)* fiind axa de control pentru algoritm, în timp ce axa Oy este numită *passive axis (PA)* și coordonata corespunzătoare este incrementată atât cât e nevoie.

Scan-conversion pentru segmente de dreaptă

Algoritmul de rasterizare pentru un segment de dreaptă trebuie să calculeze coordonatele pixelilor care se află pe linia ideală sau care sunt cât mai aproape posibil de aceasta:

- secvența de pixeli aproximează cel mai bine linia
- să fie în același timp cât mai "dreaptă" posibil

Vom considera drepte având **panta cuprinsă între -1 și 1**
$\Longrightarrow$ aproximarea va trebui să conțină exact un pixel pe fiecare coloană

- dreapta este "mai orizontală decât verticală"
- creșterea pe axa Ox este mai mare decât creșterea pe axa Oy

Axa Ox mai poartă numele de *driving axis (DA)* fiind axa de control pentru algoritm, în timp ce axa Oy este numită *passive axis (PA)* și coordonata corespunzătoare este incrementată atât cât e nevoie.

Scan-conversion pentru segmente de dreaptă

Dorim să rasterizăm segmentul $P_0(x_0,y_0)P_1(x_1,y_1)$.

Strategia cea mai simplă de abordare este următoarea:

- pornim cu coordonata $x_{\text{minimă}} = x_0$;
- mărim x cu pasul 1;
- pentru fiecare x_i calculăm $y_i = m \cdot x_i + n$;
- rotunjim y_i.

În această manieră se selecționează întotdeauna pixelul cel mai apropiat de linia ideală. Pentru identificarea fiecărui pixel trebuie să se efectueze 3 operații: o adunare, o înmulțire și o rotunjire.

Scan-conversion pentru segmente de dreaptă

Dorim să rasterizăm segmentul $P_0(x_0,y_0)P_1(x_1,y_1)$.
Strategia cea mai simplă de abordare este următoarea:

- pornim cu coordonata $x_{\text{minimă}} = x_0$;
- mărim x cu pasul 1;
- pentru fiecare x_i calculăm $y_i = m \cdot x_i + n$;
- rotunjim y_i.

În această manieră se selecționează întotdeauna pixelul cel mai apropiat de linia ideală. Pentru identificarea fiecărui pixel trebuie să se efectueze 3 operații: o adunare, o înmulțire și o rotunjire.

Scan-conversion pentru segmente de dreaptă

Dorim să rasterizăm segmentul $P_0(x_0,y_0)P_1(x_1,y_1)$.
Strategia cea mai simplă de abordare este următoarea:

- pornim cu coordonata $x_{\text{minimă}} = x_0$;
- mărim x cu pasul 1;
- pentru fiecare x_i calculăm $y_i = m \cdot x_i + n$;
- rotunjim y_i.

În această manieră se selecționează întotdeauna pixelul cel mai apropiat de linia ideală. Pentru identificarea fiecărui pixel trebuie să se efectueze 3 operații: o adunare, o înmulțire și o rotunjire.

Scan-conversion pentru segmente de dreaptă

Dorim să rasterizăm segmentul $P_0(x_0,y_0)P_1(x_1,y_1)$.
Strategia cea mai simplă de abordare este următoarea:

- pornim cu coordonata $x_{\text{minimă}} = x_0$;
- mărim x cu pasul 1;
- pentru fiecare x_i calculăm $y_i = m \cdot x_i + n$;
- rotunjim y_i.

În această manieră se selecționează întotdeauna pixelul cel mai apropiat de linia ideală. Pentru identificarea fiecărui pixel trebuie să se efectueze 3 operații: o adunare, o înmulțire și o rotunjire.

Scan-conversion pentru segmente de dreaptă

Dorim să rasterizăm segmentul $P_0(x_0,y_0)P_1(x_1,y_1)$.
Strategia cea mai simplă de abordare este următoarea:

- pornim cu coordonata $x_{\text{minimă}} = x_0$;
- mărim x cu pasul 1;
- pentru fiecare x_i calculăm $y_i = m \cdot x_i + n$;
- rotunjim y_i.

În această manieră se selecționează întotdeauna pixelul cel mai apropiat de linia ideală. Pentru identificarea fiecărui pixel trebuie să se efectueze 3 operații: o adunare, o înmulțire și o rotunjire.

Scan-conversion pentru segmente de dreaptă

Dorim să rasterizăm segmentul $P_0(x_0,y_0)P_1(x_1,y_1)$.
Strategia cea mai simplă de abordare este următoarea:

- pornim cu coordonata $x_{\text{minimă}} = x_0$;
- mărim x cu pasul 1;
- pentru fiecare x_i calculăm $y_i = m \cdot x_i + n$;
- rotunjim y_i.

În această manieră se selecționează întotdeauna pixelul cel mai apropiat de linia ideală. Pentru identificarea fiecărui pixel trebuie să se efectueze 3 operații: o adunare, o înmulțire și o rotunjire.

Scan-conversion pentru segmente de dreaptă

Dorim să rasterizăm segmentul $P_0(x_0,y_0)P_1(x_1,y_1)$.
Strategia cea mai simplă de abordare este următoarea:

- pornim cu coordonata $x_{\text{minimă}} = x_0$;
- mărim x cu pasul 1;
- pentru fiecare x_i calculăm $y_i = m \cdot x_i + n$;
- rotunjim y_i.

În această manieră se selecționează întotdeauna pixelul cel mai apropiat de linia ideală. Pentru identificarea fiecărui pixel trebuie să se efectueze 3 operații: o adunare, o înmulțire și o rotunjire.

Scan-conversion pentru segmente de dreaptă

Dacă însă scriem

$$y_{i+1} = m \cdot x_{i+1} + n = m \cdot (x_i + 1) + n = m \cdot x_i + m + n = y_i + m$$

se observă că putem defini valorile coordonatelor x si y ale pixelilor prin incrementare, calculând valorile variabilelor la un anumit pas în funcție de valorile calculate la pasul precedent.

Am evitat astfel operația de înmulțire, dar, la fiecare pas, rămâne încă o operație de rotunjire, iar variabilele utilizate sunt reale. Operațiile cu numere reale sunt mai lente iar utilizarea aritmeticii reale înseamnă a introduce erori la rotunjire.

Scan-conversion pentru segmente de dreaptă

Dacă însă scriem

$$y_{i+1} = m \cdot x_{i+1} + n = m \cdot (x_i + 1) + n = m \cdot x_i + m + n = y_i + m$$

se observă că putem defini valorile coordonatelor x si y ale pixelilor prin incrementare, calculând valorile variabilelor la un anumit pas în funcție de valorile calculate la pasul precedent.

Am evitat astfel operația de înmulțire, dar, la fiecare pas, rămâne încă o operație de rotunjire, iar variabilele utilizate sunt reale. Operațiile cu numere reale sunt mai lente iar utilizarea aritmeticii reale înseamnă a introduce erori la rotunjire.

Algoritmul lui Bresenham

Considerațiile care stau la baza acestui algoritm, numit și *algoritmul punctului de mijloc*, sunt acelea de a încerca să se folosească doar operații din aritmetica întregilor.

Presupunem că ultimul pixel ales este $P(x_P, y_P)$. Următorul pixel al rasterizării va fi E, ori NE.

Fie Q punctul în care linia reală intersectează coloana $x = x_P + 1$. Alegem ca următor pixel, dintre E și NE, pe cel care minimizează distanța față de Q.

Astfel, dacă M este mijlocul segmentului (E, NE), va trebui să alegem punctul de aceeași parte cu Q față de M. Trebuie să calculăm deci de ce parte se află Q față de M.

Algoritmul lui Bresenham

Considerațiile care stau la baza acestui algoritm, numit și *algoritmul punctului de mijloc*, sunt acelea de a încerca să se folosească doar operații din aritmetica întregilor.

Presupunem că ultimul pixel ales este $P(x_P, y_P)$. Următorul pixel al rasterizării va fi E, ori NE.

Fie Q punctul în care linia reală intersectează coloana $x = x_P + 1$. Alegem ca următor pixel, dintre E și NE, pe cel care minimizează distanța față de Q.

Astfel, dacă M este mijlocul segmentului (E, NE), va trebui să alegem punctul de aceeași parte cu Q față de M. Trebuie să calculăm deci de ce parte se află Q față de M.

Algoritmul lui Bresenham

Considerațiile care stau la baza acestui algoritm, numit și *algoritmul punctului de mijloc*, sunt acelea de a încerca să se folosească doar operații din aritmetica întregilor.

Presupunem că ultimul pixel ales este $P(x_P, y_P)$. Următorul pixel al rasterizării va fi E, ori NE.

Fie Q punctul în care linia reală intersectează coloana $x = x_P + 1$.

Alegem ca următor pixel, dintre E și NE, pe cel care minimizează distanța față de Q.

Astfel, dacă M este mijlocul segmentului (E, NE), va trebui să alegem punctul de aceeași parte cu Q față de M. Trebuie să calculăm deci de ce parte se află Q față de M.

Algoritmul lui Bresenham

Considerațiile care stau la baza acestui algoritm, numit și *algoritmul punctului de mijloc*, sunt acelea de a încerca să se folosească doar operații din aritmetica întregilor.

Presupunem că ultimul pixel ales este $P(x_P, y_P)$. Următorul pixel al rasterizării va fi E, ori NE.

Fie Q punctul în care linia reală intersectează coloana $x = x_P + 1$. Alegem ca următor pixel, dintre E și NE, pe cel care minimizează distanța față de Q.

Astfel, dacă M este mijlocul segmentului (E, NE), va trebui să alegem punctul de aceeași parte cu Q față de M. Trebuie să calculăm deci de ce parte se află Q față de M.

Algoritmul lui Bresenham

Considerațiile care stau la baza acestui algoritm, numit și *algoritmul punctului de mijloc*, sunt acelea de a încerca să se folosească doar operații din aritmetica întregilor.

Presupunem că ultimul pixel ales este $P(x_P, y_P)$. Următorul pixel al rasterizării va fi E, ori NE.

Fie Q punctul în care linia reală intersectează coloana $x = x_P + 1$. Alegem ca următor pixel, dintre E și NE, pe cel care minimizează distanța față de Q.

Astfel, dacă M este mijlocul segmentului $(E,\ NE)$, va trebui să alegem punctul de aceeași parte cu Q față de M. Trebuie să calculăm deci de ce parte se află Q față de M.

Algoritmul lui Bresenham

$$F(x, y) = a \cdot x + b \cdot y + c = 0.$$

Notații:

$$dx = x_1 - x_0 \ ; \ dy = y_1 - y_0 \quad \longrightarrow \quad m = \frac{dy}{dx}$$

$$F(x, y) = dy \cdot x - dx \cdot y + n \cdot dx = 0, \quad a = dy, \ b = -dx, \ c = y_0 x_1 - x_0 y_1.$$

Ce informații ne poate furniza funcția F?

F ia valoarea 0 în toate punctele dreptei, valori pozitive în puncte din semiplanul inferior și valori negative în semiplanul superior.

Algoritmul lui Bresenham

$$F(x, y) = a \cdot x + b \cdot y + c = 0.$$

Notații:

$$dx = x_1 - x_0 \ ; \ dy = y_1 - y_0 \quad \longrightarrow \quad m = \frac{dy}{dx}$$

$$F(x, y) = dy \cdot x - dx \cdot y + n \cdot dx = 0, \quad a = dy, \ b = -dx, \ c = y_0 x_1 - x_0 y_1.$$

Ce informații ne poate furniza funcția F?

F ia valoarea 0 în toate punctele dreptei, valori pozitive în puncte din semiplanul inferior și valori negative în semiplanul superior.

Algoritmul lui Bresenham

$$F(x,y) = a \cdot x + b \cdot y + c = 0.$$

Notații:

$dx = x_1 - x_0$; $dy = y_1 - y_0 \quad \longrightarrow \quad m = \frac{dy}{dx}$

$F(x,y) = dy \cdot x - dx \cdot y + n \cdot dx = 0, \quad a = dy,\ b = -dx,\ c = y_0 x_1 - x_0 y_1.$

Ce informații ne poate furniza funcția F?

F ia valoarea 0 în toate punctele dreptei, valori pozitive în puncte din semiplanul inferior și valori negative în semiplanul superior.

Algoritmul lui Bresenham

$$F(x, y) = a \cdot x + b \cdot y + c = 0.$$

Notații:

$$dx = x_1 - x_0 \ ; \ dy = y_1 - y_0 \quad \longrightarrow \quad m = \frac{dy}{dx}$$

$$F(x, y) = dy \cdot x - dx \cdot y + n \cdot dx = 0, \quad a = dy, \ b = -dx, \ c = y_0 x_1 - x_0 y_1.$$

Ce informații ne poate furniza funcția F?

F ia valoarea 0 în toate punctele dreptei, valori pozitive în puncte din semiplanul inferior și valori negative în semiplanul superior.

Algoritmul lui Bresenham

$$F(x, y) = a \cdot x + b \cdot y + c = 0.$$

Notații:

$$dx = x_1 - x_0 \ ; \ dy = y_1 - y_0 \quad \longrightarrow \quad m = \frac{dy}{dx}$$

$$F(x, y) = dy \cdot x - dx \cdot y + n \cdot dx = 0, \quad a = dy, \ b = -dx, \ c = y_0 x_1 - x_0 y_1.$$

Ce informații ne poate furniza funcția F?

F ia valoarea 0 în toate punctele dreptei, valori pozitive în puncte din semiplanul inferior și valori negative în semiplanul superior.

Algoritmul lui Bresenham

semnul $F(M) = F(x_P + 1, y_P + \frac{1}{2})$?

variabila de decizie

$$d = a \cdot (x_P + 1) + b \cdot \left(y_P + \frac{1}{2}\right) + c.$$

- dacă $d < 0$ atunci M este deasupra dreptei $\longrightarrow$ alegem E;
- dacă $d > 0$ atunci M este sub dreaptă $\longrightarrow$ alegem NE;
- dacă $d = 0$ atunci se pot alege oricare dintre cei doi pixeli $\longrightarrow$ alegem E (convenție).

Algoritmul lui Bresenham

semnul $F(M) = F(x_P + 1, y_P + \frac{1}{2})$?

variabila de decizie

$$d = a \cdot (x_P + 1) + b \cdot \left(y_P + \frac{1}{2}\right) + c.$$

- dacă $d < 0$ atunci M este deasupra dreptei $\longrightarrow$ alegem E;
- dacă $d > 0$ atunci M este sub dreaptă $\longrightarrow$ alegem NE;
- dacă $d = 0$ atunci se pot alege oricare dintre cei doi pixeli $\longrightarrow$ alegem E (convenție).

Algoritmul lui Bresenham

semnul $F(M) = F(x_P + 1, y_P + \frac{1}{2})$?

variabila de decizie

$$d = a \cdot (x_P + 1) + b \cdot \left(y_P + \frac{1}{2}\right) + c.$$

- dacă $d < 0$ atunci M este deasupra dreptei $\longrightarrow$ alegem E;
- dacă $d > 0$ atunci M este sub dreaptă $\longrightarrow$ alegem NE;
- dacă $d = 0$ atunci se pot alege oricare dintre cei doi pixeli $\longrightarrow$ alegem E (convenție).

Algoritmul lui Bresenham

semnul $F(M) = F(x_P + 1, y_P + \frac{1}{2})$?

variabila de decizie

$$d = a \cdot (x_P + 1) + b \cdot \left(y_P + \frac{1}{2}\right) + c.$$

- dacă $d < 0$ atunci M este deasupra dreptei $\longrightarrow$ alegem E;
- dacă $d > 0$ atunci M este sub dreaptă $\longrightarrow$ alegem NE;
- dacă $d = 0$ atunci se pot alege oricare dintre cei doi pixeli $\longrightarrow$ alegem E (convenție).

Algoritmul lui Bresenham

semnul $F(M) = F(x_P + 1, y_P + \frac{1}{2})$?

variabila de decizie

$$d = a \cdot (x_P + 1) + b \cdot \left(y_P + \frac{1}{2}\right) + c.$$

- dacă $d < 0$ atunci M este deasupra dreptei $\longrightarrow$ alegem E;
- dacă $d > 0$ atunci M este sub dreaptă $\longrightarrow$ alegem NE;
- dacă $d = 0$ atunci se pot alege oricare dintre cei doi pixeli $\longrightarrow$ alegem E (convenție).

Algoritmul lui Bresenham

Pasul următor:

– dacă a fost ales E :

$d_{\text{new}} = F(x_P + 2, y_P + \frac{1}{2}) = d + a.$

Notăm $deltaE = d_{new} - d = a$, astfel **d += deltaE**.

– dacă a fost ales NE :

$d_{new} = F(x_P + 2, y_P + \frac{3}{2}) = d + a + b.$

Notăm $deltaNE = d_{\text{new}} - d = a + b$, prin urmare **d += deltaNE**.

valoarea inițială a lui d:

$$F\left(x_0 + 1, y_0 + \frac{1}{2}\right) = dy - \frac{dx}{2}$$

Algoritmul lui Bresenham

Pasul următor:

– dacă a fost ales E :

$d_{\text{new}} = F(x_P + 2, y_P + \frac{1}{2}) = d + a.$

Notăm $deltaE = d_{new} - d = a$, astfel **d += deltaE**.

– dacă a fost ales NE :

$d_{new} = F(x_P + 2, y_P + \frac{3}{2}) = d + a + b.$

Notăm $deltaNE = d_{\text{new}} - d = a + b$, prin urmare **d += deltaNE**.

valoarea inițială a lui d:

$$F\left(x_0 + 1, y_0 + \frac{1}{2}\right) = dy - \frac{dx}{2}$$

Algoritmul lui Bresenham

Pasul următor:

– dacă a fost ales E :

$d_{\text{new}} = F(x_P + 2, y_P + \frac{1}{2}) = d + a.$

Notăm $deltaE = d_{new} - d = a$, astfel **d += deltaE**.

– dacă a fost ales NE :

$d_{new} = F(x_P + 2, y_P + \frac{3}{2}) = d + a + b.$

Notăm $deltaNE = d_{\text{new}} - d = a + b$, prin urmare **d += deltaNE**.

valoarea inițială a lui d:

$$F\left(x_0 + 1, y_0 + \frac{1}{2}\right) = dy - \frac{dx}{2}$$

Algoritmul lui Bresenham

Pasul următor:

– dacă a fost ales E :

$d_{\text{new}} = F(x_P + 2, y_P + \frac{1}{2}) = d + a.$

Notăm $deltaE = d_{new} - d = a$, astfel **d += deltaE**.

– dacă a fost ales NE :

$d_{new} = F(x_P + 2, y_P + \frac{3}{2}) = d + a + b.$

Notăm $deltaNE = d_{\text{new}} - d = a + b$, prin urmare **d += deltaNE**.

valoarea inițială a lui d:

$$F\left(x_0 + 1, y_0 + \frac{1}{2}\right) = dy - \frac{dx}{2}$$

Algoritmul lui Bresenham

Pasul următor:

– dacă a fost ales E :

$d_{\text{new}} = F(x_P + 2, y_P + \frac{1}{2}) = d + a.$

Notăm $deltaE = d_{new} - d = a$, astfel **d += deltaE**.

– dacă a fost ales NE :

$d_{new} = F(x_P + 2, y_P + \frac{3}{2}) = d + a + b.$

Notăm $deltaNE = d_{\text{new}} - d = a + b$, prin urmare **d += deltaNE**.

valoarea inițială a lui d:

$$F\left(x_0 + 1, y_0 + \frac{1}{2}\right) = dy - \frac{dx}{2}$$

Algoritmul lui Bresenham

Pasul următor:

– dacă a fost ales E :

$d_{\text{new}} = F(x_P + 2, y_P + \frac{1}{2}) = d + a.$

Notăm $deltaE = d_{new} - d = a$, astfel **d += deltaE**.

– dacă a fost ales NE :

$d_{new} = F(x_P + 2, y_P + \frac{3}{2}) = d + a + b.$

Notăm $deltaNE = d_{\text{new}} - d = a + b$, prin urmare **d += deltaNE**.

valoarea inițială a lui d:

$$F\left(x_0 + 1, y_0 + \frac{1}{2}\right) = dy - \frac{dx}{2}$$

Algoritmul lui Bresenham

Pasul următor:

– dacă a fost ales E :

$d_{\text{new}} = F(x_P + 2, y_P + \frac{1}{2}) = d + a.$

Notăm $deltaE = d_{new} - d = a$, astfel **d += deltaE**.

– dacă a fost ales NE :

$d_{new} = F(x_P + 2, y_P + \frac{3}{2}) = d + a + b.$

Notăm $deltaNE = d_{\text{new}} - d = a + b$, prin urmare **d += deltaNE**.

valoarea inițială a lui d:

$$F\left(x_0 + 1, y_0 + \frac{1}{2}\right) = dy - \frac{dx}{2}$$

Algoritmul lui Bresenham

Pasul următor:

– dacă a fost ales E :

$d_{\text{new}} = F(x_P + 2, y_P + \frac{1}{2}) = d + a.$

Notăm $deltaE = d_{new} - d = a$, astfel **d += deltaE**.

– dacă a fost ales NE :

$d_{new} = F(x_P + 2, y_P + \frac{3}{2}) = d + a + b.$

Notăm $deltaNE = d_{\text{new}} - d = a + b$, prin urmare **d += deltaNE**.

valoarea inițială a lui d:

$$F\left(x_0 + 1, y_0 + \frac{1}{2}\right) = dy - \frac{dx}{2}$$

Algoritmul lui Bresenham

Algoritmul:

```
int dy = y1 - y0;
int dx = x1 - x0;
int d = 2 * dy - dx;
int deltaE = 2 * dy;
int deltaNE = 2 * (dy-dx);
int x = x0;
int y = y0;
putpixel(x0, y0, LIGHTBLUE);
```

Algoritmul lui Bresenham

```
while (x < x1) {
    if (d <= 0) {                /* se alege E */
        d+= deltaE;
        x++;
      } else {                   /* se alege NE */
        d+= deltaNE;
        x++;
        y++;
      }
    putpixel(x, y, LIGHTGREEN);
  }                              /* end while */
```