UNIVERSITATEA TEHNICĂ A MOLDOVEI
Facultatea Calculatoare, Informatică şi Microelectronică
Departamentul Informatică şi Ingineria Sistemelor

GRAFICA PE CALCULATOR

TEMA 9. ALGORITMI FUNDAMENTALI DE SINTEZĂ A IMAGINILOR

l. u., dr. NASTAS Andrei

CONȚINUT

9.1. Algoritmi de generare a vectorilor în spațiul discret

Algoritmi performanți utilizați în implementarea funcțiilor de afișare ale sistemelor grafice.

După cum se știe echipamentele care produc imagini folosesc metoda raster și primesc imaginea codificată numeric. Imaginea este păstrată în memoria rastru.

Suprafața de afișare este tratată ca o matrice de celule discrete, numite **puncte** sau **pixeli**.

Fiecărui pixel îi corespunde o adresă distinctă (*xp*, *yp*), pe suprafața de afișare, căreia îi este atașat un sistem de coordonate carteziene 2D.

Între adresele pixelilor și celulele memoriei rastru în care sunt păstrate intensitățile (culorile) de afișare ale pixelilor, există o corespondență biunivocă.

9.1. Algoritmi de generare a vectorilor în spațiul discret

Algoritmii de generare a primitivelor grafice în spațiul discret determină adresele pixelilor care aproximează cel mai bine o primitivă.

Pentru vizualizarea primitivei este necesar să se apeleze o rutină dedicată echipamentului folosit, care înscrie în celula corespunzătoare unui pixel de adresă dată valoarea intensității (culorii) cu care se dorește afișarea pixelului respectiv.

O astfel de rutină este ***putpixel()***, din pachetul de funcții ale limbajului C++.

Orice bibliotecă grafică conține o subrutină care poate fi apelată în programele de aplicație în scopul trasării unui segment de dreaptă; de exemplu, subrutinele ***line()*** și ***lineto()***, care se utilizează în limbajul C++.

În cadrul subrutinei de afișare a unui segment de dreaptă *trebuie să se determine acei pixeli care aproximează cel mai bine segmentul teoretic*, adică acel segment determinat matematic de coordonatele capetelor sale.

9.1. Algoritmi de generare a vectorilor în spațiul discret

Cu cât rezoluția suprafeței de afișare este mai bună, cu atât liniile sunt mai bine aproximate. Cu toate acestea, pentru anumite pante aproximarea discretă este vizibilă chiar și la o rezoluție bună. Majoritatea algoritmilor de aproximare a vectorilor în spațiul discret se bazează pe metode incrementale.

Astfel, plecând de la ecuația dreptei determinate de punctele (x_1, y_1) și (x_2, y_2),

$$y = m \cdot x + b, \tag{9.1}$$

unde $m = \frac{y_2 - y_1}{x_2 - x_1}$

și $b = y_1 - m \cdot x_1$.

Dacă $x_1 < x_2$, se poate determina o succesiune de puncte de pe dreaptă prin incrementarea lui *x* de la x_1 la x_2 și calcularea lui *y* din ecuația dreptei.

În acest caz, afișarea segmentului (x_1, y_1) – (x_2, y_2) poate fi descrisă astfel: se afișează pixelul cu adresa (*xp* = *x*, *yp* = întregul cel mai apropriat de *y*), în culoarea specificată.

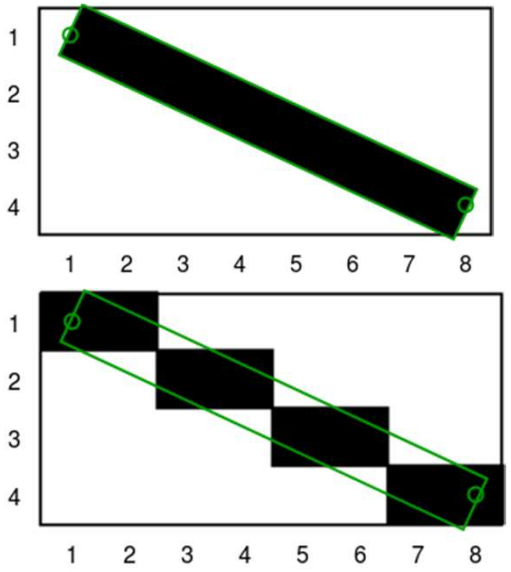


Fig. 9.1. Descompunerea în rastru a dreptei

9.1. Algoritmi de generare a vectorilor în spațiul discret

Această descriere are două deficiențe majore:

1) Nu se ține cont de panta dreptei; astfel, cu cât panta este mai mare, cu atât numărul de puncte calculate și afișate va fi mai mic, iar punctele obținute vor fi din ce în ce mai distanțate, deoarece variația lui *y* între două valori succesive ale lui *x* crește.

2) Calculul fiecărui punct de pe segment presupune efectuarea unei înmulțiri și a unei adunări între două numere reale.

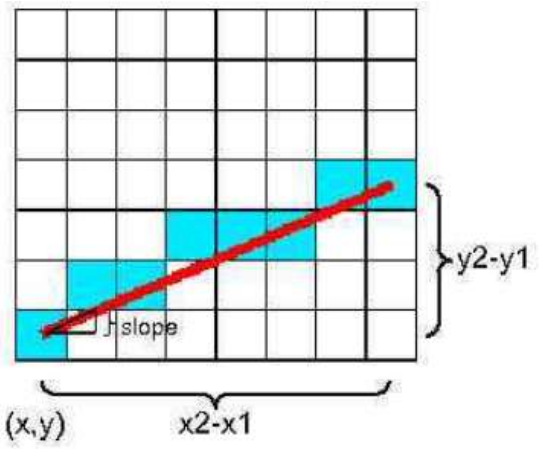
La mai multe procesoare execuția unei operații de înmulțire sau de împărțire între două numere întregi durează de cel puțin 10 ori mai mult decât execuția unei operații de adunare sau scădere.

De asemenea, nu toate procesoarele folosite pentru execuția subrutinelor de bază ale unui sistem grafic operează cu numere reale (operațiile sunt simulate).

Chiar și în cazul existenței unui coprocesor care execută operațiile cu numere reale, acestea sunt mai lente decât operațiile cu numere întregi.

9.1.1. Algoritmul DDA (Digital Differential Analyser)

DDA este folosit pentru desenarea liniei drepte pentru a forma o linie, triunghi sau poligon în grafica computerizată. DDA analizează probele de-a lungul liniei la intervale regulate de o coordonată ca întreg și pentru cealaltă coordonată se rotunjește între întregul cel mai apropiat de linie. Prin urmare, pe măsură ce linia avansează, scanarea primei coordonate întregi și rotunjirea celui de-al doilea la cel mai apropiat număr întreg. Prin urmare, o linie trasată folosind DDA pentru coordonarea *x* va fi x_0 la x_1 dar pentru *y* coordonate va fi $y = mx + b$ și pentru a desena funcție va fi F_n (*x*, *y* rotunjit).



9.1.1. Algoritmul DDA (Digital Differential Analyser)

Prima deficiență poate fi eliminată astfel: Dacă $0 \leq \text{abs}(m) \leq 1$, atunci succesiunea de puncte se va obține prin incrementarea lui *x*, altfel prin incrementarea lui *y*.

Algoritmii de generare a vectorilor în spațiul discret care vor fi prezentați în continuare, nu conțin calcule de înmulțire sau împărțire pentru determinarea punctelor de valori de pe vectori, iar unii dintre ei operează cu numere întregi.

Fie segmentul dat, $(x_1, y_1) – (x_2, y_2)$ și punctele (x', y'), (x'', y'') două puncte consecutive de pe segment.

Atunci: $m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{y" - y'}{x" - x'}$.

1) Pentru $\text{abs}(m) \leq 1$ și $x_1 < x_2$ se generează segmentul incrementând pe *x*, deci:

$$x'' = x' + 1,\ x'' - x' = 1,\ y'' = y' + m. \tag{9.2}$$

2) Pentru $\text{abs}(m) > 1$ și $y_1 < y_2$ segmentul este generat incrementântu-l pe *y*, deci:

$$y'' = y' + 1,\ y'' - y' = 1,\ x'' = x' + 1/m. \tag{9.3}$$

Cazurile: $\text{abs}(m) \leq 1$ și $x_1 > x_2$ și $\text{abs}(m) > 1$ și $y_1 > y_2$ se reduc la cazurile la cazurile (1) și (2) prin interschimbarea valorilor variabilelor x_1 și x_2, respectiv y_1 și y_2.

Calculul punctelor de pe segmentul de dreapta conține numai operații de adunare și scădere, dar între numere reale.

9.1.2. Algoritmul Bresenham

Algoritmul Bresenham a fost dezvoltat de J. E. Bresenham în 1962 și este mult mai precis și mult mai eficient decât DDA. Acesta scanează coordonatele dar, în loc să le rotunjească, ia valoarea incrementală în considerare prin adăugarea sau scăderea și prin urmare poate fi utilizată pentru desenarea cercurilor și curbelor. Prin urmare, dacă se va trasa o linie între două puncte x și y, atunci coordonatele următoare vor fi (*xa* + 1, *ya*) și (*xa* + 1, *ya* + 1) unde a este valoarea incrementală a următoarelor coordonate și diferența dintre cele două se va calcula prin scăderea sau adăugarea ecuațiilor formate de ele.

9.1.2. Algoritmul Bresenham

Algoritmul Bresenham este bazat tot pe metoda incrementală, dar conține numai operații cu numere întregi.

Algoritmul este definit pentru vectori cu panta cuprinsă între 0 și 1.

Pentru fiecare valoare a lui x se alege acel punct al spațiului discret care este mai aproape de punctul de pe vectorul teoretic.

Selecția se bazează pe distanțele de la cele două puncte candidat (punctul de deasupra vectorului și cel de sub vector) la punctul corespunzător de pe vector.

Fie $m = \frac{y_2 - y_1}{x_2 - x_1}$ panta vectorului și (x_i, y_i) ultimul punct al spațiului discret ales în procesul de generare a vectorului.

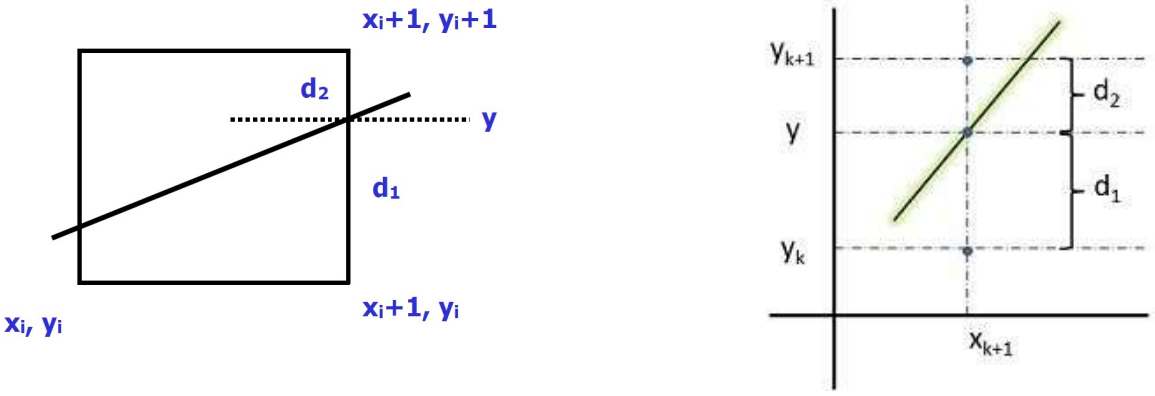


Fig. 9.2. Reprezentarea dreptei după Algoritmul Bresenham

9.1.2. Algoritmul Bresenham

Notăm cu: d_1 distanța de la vectorul teoretic la punctul $O(x_i + 1, y_i)$ și cu d_2 distanța de la vectorul teoretic la $D(x_i + 1, y_i + 1)$.

Următorul punct ales va fi O dacă $d_1 < d_2$, sau punctul D în caz contrar.

Dacă $d_1 = d_2$ se poate alege oricare dintre cele două puncte.

Exprimăm diferența $d_1 - d_2$:

$$y = m \cdot (x_i + 1) + b, \tag{9.4}$$

este ordonata punctului de pe vectorul teoretic

$$d_1 = y - y_i = m \cdot (x_i + 1) + b - y_i, \tag{9.5}$$

$$d_2 = y_i + 1 - y = y_i + 1 - m \cdot (x_i + 1) - b, \tag{9.6}$$

$$d_1 - d_2 = 2 \cdot m \cdot (x_i + 1) - 2 \cdot y_i + 2 \cdot b - 1. \tag{9.7}$$

Se înlocuiește m cu d_y/d_x, apoi se înmulțește în ambele părți cu d_x.

Rezultă :

$$t_i = (d_1 - d_2) \cdot d_x = 2 \cdot d_y \cdot (x_i + 1) - 2 \cdot d_x \cdot y_i + 2 \cdot b \cdot d_x - d_x, \tag{9.8}$$

t_i reprezintă *eroarea de aproximare în pasul i*.

9.1.2. Algoritmul Bresenham

Valoarea $c = 2 \cdot b \cdot d_x - d_x + 2 \cdot d_y$ este aceeași pentru orice pas, deci:

$$t_i = 2 \cdot d_y \cdot x_i - 2 \cdot d_x \cdot y_i + c, \tag{9.9}$$

Notăm cu $(x_i + 1, y_i + 1)$ punctul care se alege în pasul curent.

Atunci, expresia erorii de aproximare pentru pasul următor este:

$$t_{i+1} = 2 \cdot d_y \cdot x_{i+1} - 2 \cdot d_x \cdot y_{i+1} + c. \tag{9.10}$$

1) Dacă $t_i \leq 0$ se alege punctul O, deci $x_{i+1} = x_i + 1$ și $y_{i+1} = y_i$.

Rezultă:
$$t_{i+1} = 2 \cdot d_y \cdot (x_i + 1) - 2 \cdot d_x \cdot y_i + c, \tag{9.11}$$

sau
$$t_{i+1} = t_i + 2 \cdot d_y.$$

Dacă $t_i > 0$ se alege punctul D, deci $x_{i+1} = x_i + 1$ și $y_{i+1} = y_i + 1$.

Rezultă:
$$t_{i+1} = 2 \cdot d_y \cdot (x_i + 1) - 2 \cdot d_x \cdot (y_i + 1) + c, \tag{9.12}$$

sau
$$t_{i+1} = t_{i+2} \cdot d_y - 2 \cdot d_x.$$

Se calculează eroarea de aproximare pentru primul pas înlocuind în expresia (9.9) pe x_i cu x_1 și y_i cu y_1:

$$t_1 = 2 \cdot d_y \cdot x_1 - 2 \cdot d_x \cdot y_1 + 2 \cdot d_y - d_x + 2 \cdot d_x (y_1 - (d_y / d_x) \cdot x_1), \tag{9.13}$$

$$t_1 = 2 \cdot d_y - d_x.$$

9.1.3. Generalizarea algoritmului Bresenham

Vectorii definiți în spațiul 2D, care nu sunt orizontali sau verticali, pot fi clasificați în opt clase geometrice, numite octanți.

Fie un vector $(x_1, y_1) – (x_2, y_2)$.

Octantul din care face parte se stabilește în funcție de $d_x = x_2 – x_1$ și $d_y = y_2 – y_1$, astfel:

Octantul 1: $d_x > 0$, $d_y > 0$ și $d_x \geq d_y$;

Octantul 2: $d_x > 0$, $d_y > 0$ și $d_x < d_y$;

Octantul 3: $d_x < 0$, $d_y > 0$ și $abs(d_x) < d_y$;

Octantul 4: $d_x > 0$, $d_y > 0$ și $abs(d_x) \geq d_y$;

Octantul 5: $d_x < 0$, $d_y < 0$ și $abs(d_x) \geq abs(d_y)$;

Octantul 6: $d_x < 0$, $d_y < 0$ și $abs(d_x) < abs(d_y)$;

Octantul 7: $d_x > 0$, $d_y < 0$ și $d_x < abs(d_y)$;

Octantul 8: $d_x > 0$, $d_y < 0$ și $d_x \geq abs(d_y)$.

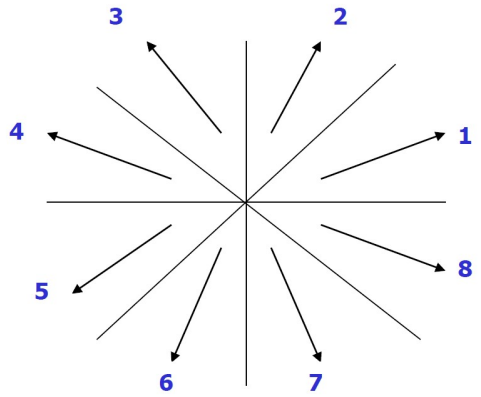


Fig. 9.3. Direcțiile vectorilor în octanți

9.1.3. Generalizarea algoritmului Bresenham

Diferența dintre algoritmul DDA și Bresenham

- DDA utilizează puncte plutitoare pe când algoritmul Bresenham utilizează puncte fixe.
- DDA rotunjește coordonatele la cel mai apropiat număr întreg, dar algoritmul Bresenham nu.
- Algoritmul Bresenham este mult mai precis și mai eficient decât DDA.
- Algoritmul Bresenham poate desena cercuri și curbe cu mult mai multă precizie decât DDA.
- DDA folosește multiplicarea și împărțirea ecuației, dar algoritmul Bresenham utilizează numai scăderea și adăugarea.

9.2. Algoritmi de generare a cercurilor

Un cerc este determinat prin centrul său și raza, sau un punct periferic.
Poate fi definit analitic în:

– coordonate carteziene
– sau polare (prin ecuații parametrice).

9.2.1. Calculul punctelor de pe cerc folosind coordonatele carteziene ale cercului

Calculul incremental al punctelor de pe cerc se poate baza pe ecuația în coordonate carteziene:

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

unde: (*xc*, *yc*) este centrul cercului, iar *r* este raza.

Presupunem că îl incrementăm pe *x* cu valori de la *xc* – *r* la *xc* + *r* cu pasul 1 și îl determinăm pe *y* utilizând expresia:

$$y = yc \pm \sqrt{r^2 - (x - xc)^2}, \quad (9.14)$$

sau îl incrementăm pe *x* din ecuația cercului.

Metoda are două deficiențe:

– Expresia aritmetică prin care se obține *y* (sau *x*) conține operații consumatoare de timp (ridicare la putere, extragere rădăcină pătrată);

– Punctele obținute nu sunt egal distanțate; astfel, dacă se incrementează *x*, atunci punctele din vecinătatea capetelor intervalului [*xc* – *r*, *xc* + *r*] vor fi foarte rare (punctele pentru care la o modificare cu 1 a lui *x* corespunde o modificare foarte mare a lui *y*).

9.2.2. Calculul punctelor de pe cerc folosind ecuațiile parametrice ale cercului

Ecuațiile parametrice ale cercului:

$$\begin{cases} x = xc + r \cdot \cos(t), \\ y = yc + r \cdot \sin(t), \end{cases} \quad \text{unde } 0 \le t \le 6{,}28\ (2 \cdot \pi), \tag{9.15}$$

permit obținerea de puncte echidistante de pe circumferința modificându-l pe t de la 0 la 6,28 cu un pas constant.

Circumferința poate fi aproximată unind punctele obținute prin segmente de dreaptă (figura 9.4).

Dacă se dorește aproximarea circumferinței prin puncte atunci se alege pasul unghiular egal cu $1/r$.

Se poate evita evaluarea funcțiilor sinus și cosinus pentru fiecare punct calculat, știind că între fiecare două puncte consecutive există aceeași distanță unghiulară.

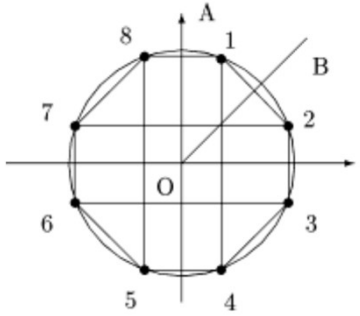


Fig. 9.4. Aproximarea circumferinței

9.2.2. Calculul punctelor de pe cerc folosind ecuațiile parametrice ale cercului

Fie (x_i, y_i) ultimul punct calculat și (x_{i+1}, y_{i+1}) punctul următor:

$$x_i = xc + \mathrm{dx}_i,\ dx_i = r \cdot \cos(t),$$
$$y_i = yc + dy_i,\ dy_i = r \cdot \sin(t), \tag{9.16}$$

$$x_{i+1} = xc + dx_{i+1},$$
$$dx_{i+1} = r \cdot \cos(t + \mathrm{pas}), \tag{9.17}$$

$$y_{i+1} = yc + dy_{i+1},$$
$$dy_{i+1} = r \cdot \sin(t + \mathrm{pas}). \tag{9.18}$$

Dar, știm că:

$$\cos(t + \mathrm{pas}) = \cos(t)\cdot\cos(\mathrm{pas}) - \sin(t)\cdot\sin(\mathrm{pas}),$$
$$\sin(t + \mathrm{pas}) = \cos(t)\cdot\sin(\mathrm{pas}) + \sin(t)\cdot\cos(\mathrm{pas}). \tag{9.19}$$

Fie $c = \cos(\mathrm{pas})$ și $s = \sin(\mathrm{pas})$. Aceste valori se calculează o singură dată la generarea unui cerc.

Rezultă următoarele relații de recurență pentru calculul punctelor de pe cerc:

$$dx_{i+1} = dx_i \cdot c - dy_i \cdot s,$$
$$dy_{i+1} = \mathrm{dx}_i \cdot \mathrm{s} - dy_i \cdot c. \tag{9.20}$$

9.2.2. Calculul punctelor de pe cerc folosind ecuațiile parametrice ale cercului

Timpul necesar calculului punctelor de pe cerc poate fi redus substanțial ținând cont de simetria punctelor de pe cerc.

Astfel, este suficient să se calculeze prin metoda descrisă mai înainte coordonatele punctelor dintr-un octant.

Fiecărui punct calculat îi corespunde alte 7 simetrice.

– De exemplu, în funcția cerc_sim(), se calculează numai punctele din primul octant;

– pentru fiecare punct calculat se apelează funcția punct_simetric(), care afișează nu numai punctul calculat, ci și simetricele sale (în raport cu unghiul *u*).

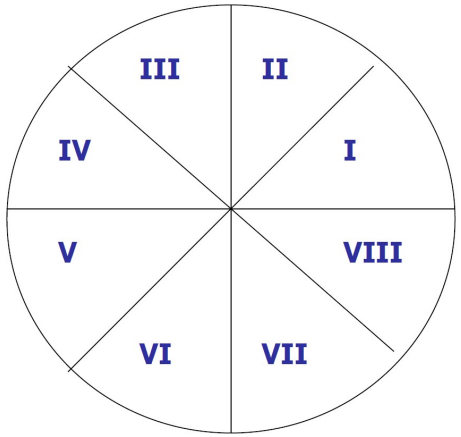


Fig. 9.5. Simetria punctelor pe cerc

9.2.3. Generarea cercurilor în spațiul discret. Algoritmul Bresenham

Algoritmul Bresenham, folosește metoda incrementală și eroarea axială ca măsură a aproprierii punctului ales de cercul teoretic.

Considerăm cercul cu centrul în originea sistemului de coordonate și raza *r*.

În cadrul algoritmului se calculează numai punctele din octantul al doilea, celelalte obținându-se prin simetrie.

Fie (x_i, y_i) ultimul punct al spațiului discret, ales pentru aproximarea cercului.

Următorul punct va fi unul dintre (x_{i+1}, y_i) și (x_{i+1}, y_{i-1}).

Ordonata punctului de pe cercul teoretic se obține din ecuația cercului în coordonate carteziene:

$$y^2 = r^2 - (x_i + 1)^2 \,. \qquad (9.21)$$

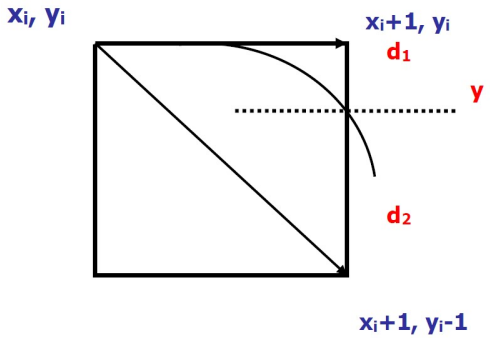


Fig. 9.6. Reprezentarea cercului după algoritmul Bresenham

9.2.3. Generarea cercurilor în spațiul discret. Algoritmul Bresenham

Se calculează distanțele dintre punctele de pe cerc și cele două puncte candidat:

$$d1 = y_i^2 - y^2 = y_i^2 - r^2 + (x_i + 1)^2, \qquad (9.22)$$
$$d2 = y^2 - (y_i - 1)^2 = r^2 - (x_i + 1)^2 - (y_i - 1)^2.$$

Notăm cu t_i eroarea de aproximare în pasul curent:

$$t_i = d1 - d2 = y_i^2 + 2(x_i + 1)^2 + (y_i - 1)^2 - 2r^2. \qquad (9.23)$$

În continuare se obține o relație de recurență pentru calculul erorii de aproximare în pasul următor:

$$t_{i+1} = y_{i+1}^2 + 2(x_{i+1} + 1)^2 + (y_{i+1} - 1)^2 - 2r^2. \qquad (9.24)$$

1) Dacă $t_i < 0$, atunci $x_{i+1} = x_i + 1$ și $y_{i+1} = y_i$.

Deci, $$t_{i+1} = y_i^2 + 2\big((x_i + 1) + 1\big)^2 + (y_i - 1)^2 - 2r^2,$$

sau $$t_{i+1} = t_i + 4 \cdot x_i + 6.$$

2) Dacă $t_i \geq 0$, atunci $x_{i+1} = x_i + 1$ și $y_{i+1} = y_i - 1$.

Deci, $$t_{i+1} = (y_i - 1)^2 + 2\big((x_i + 1) + 1\big)^2 + \big((y_i - 1) - 1\big)^2 - 2r^2,$$

sau $$t_{i+1} = t_i + 4 \cdot (x_i - y_i) + 10.$$

Valoarea erorii în primul pas se obține înlocuind în expresia lui t_i pe x_i cu $x_1 = 0$ și pe y_i cu $y_1 = r$.

Rezultă: $$t_1 = 3 - 2 \cdot r.$$

ÎNTREBĂRI