III.

Trasarea primitivelor grafice

Primitive grafice

Sunt elementele de bază pe care programatorul le poate folosi pentru a realiza desenele necesare unei anumite aplicaţii.

Ex.: puncte, segmente de dreaptă, caractere, dreptunghiuri, linii poligonale, poligoane, conice, curbe cubice, cuadrice, suprafeţe bicubice, cub, paralelipiped, etc.

Primitive grafice 2D

Simple Graphics Primitives

- Lines:

```
void LineCoord(int xmin, int ymin, int xmax, int ymax);
void Line(point pt1, point pt2);
```

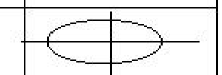
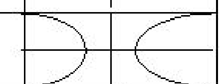
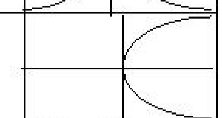
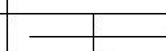
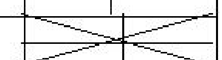
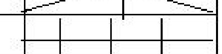
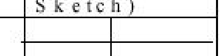
- Polygons:

```
void polyLine(int count, point* vertices);
void polygon(int count, point* vertices);
```

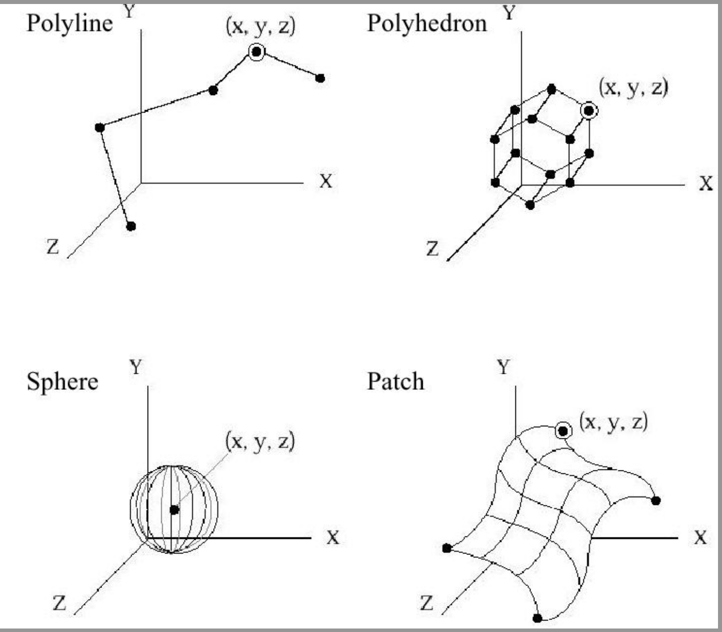
- Circles and ellipses:

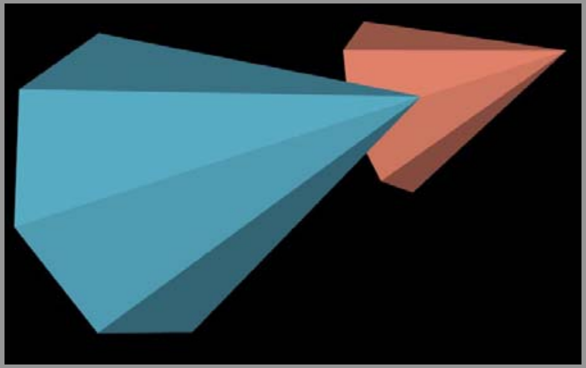
```
void circle(point center, int radius);
void ellipse(point center, int radiusX, int radiusY);
void ellipseArc(point center, int radiusX, int radiusY,
                float startAngle, float endAngle);
```

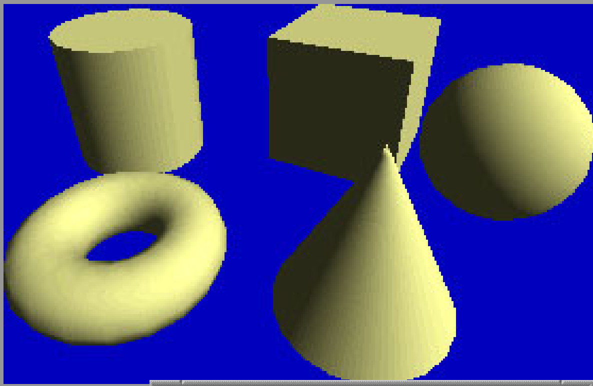
Summary of Conic Sections

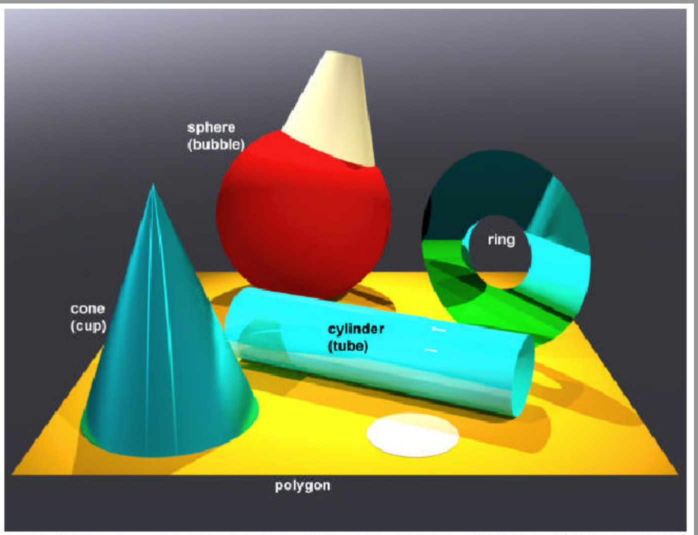
Name	Equation	Conditions	Type	Sketch
Ellipse	$\alpha x^2 + \beta y^2 = k$	$\beta < 0 < k, \alpha$	Central	
Hyperbola	$\alpha x^2 + \beta y^2 = k$	$\beta < 0 < k, \alpha$	Central	
Parabola	$\alpha y^2 + \beta x = 0$ $\beta x^2 + \alpha y = 0$		Noncentral	
Empty set	$\alpha x^2 + \beta y^2 = k$	$\alpha, \beta < 0 < k$	(Central)	(No Sketch)
Point	$\alpha x^2 + \beta y^2 = 0$	$\alpha , \beta > 0$	Central	
Pair of lines	$\alpha x^2 + \beta y^2 = 0$	$\beta < 0 < \alpha$	Central	
Parallel lines	$\alpha x^2 = k$	$\alpha , k > 0$	Central	
Empty set	$\alpha x^2 = k$	$\alpha < 0 < k$	(Central)	(No Sketch)
'Repeated' line	$\alpha x^2 = 0$		Central	

Primitive grafice 3D

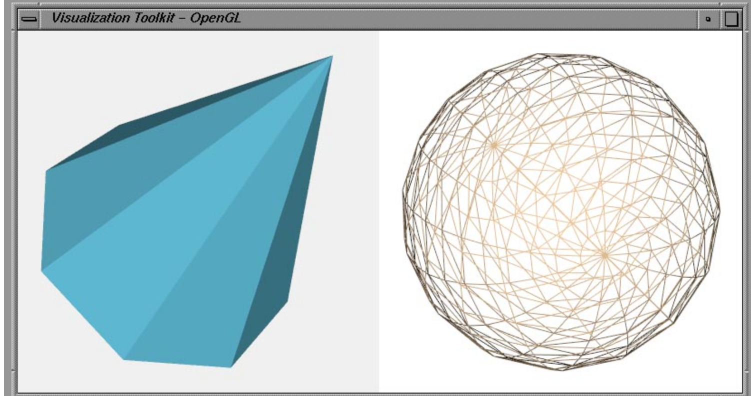


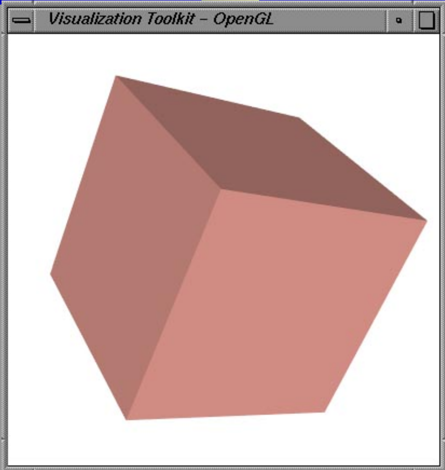






The suite of geometry primitives which comprise all visible Radiance surfaces. The geometry primitives in parentheses ( ) have inward pointing normals.

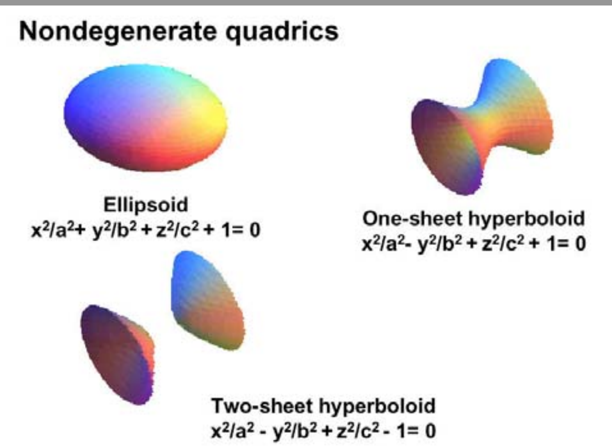


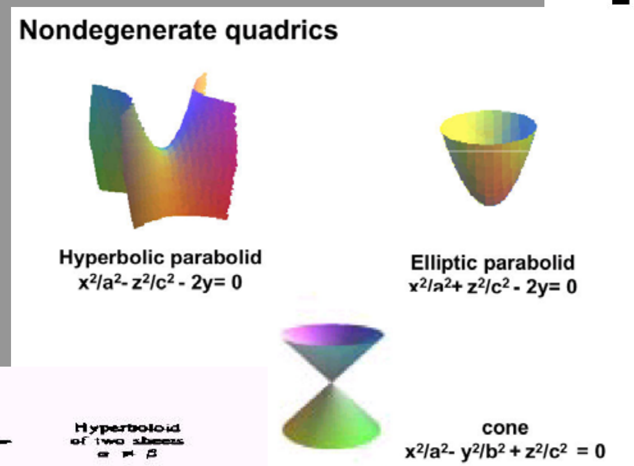


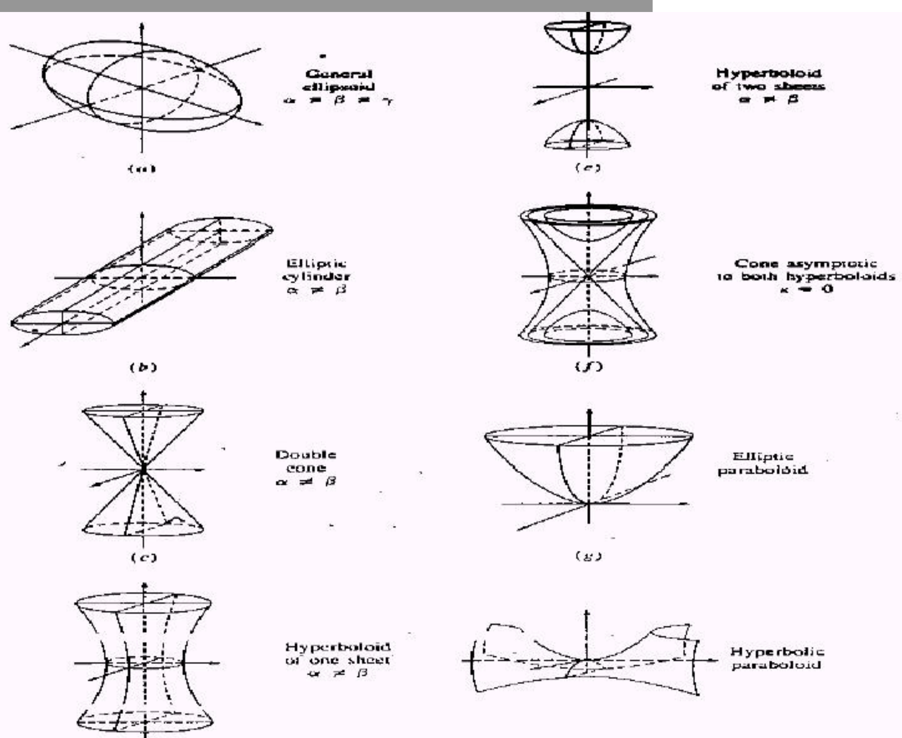
Cuadrice

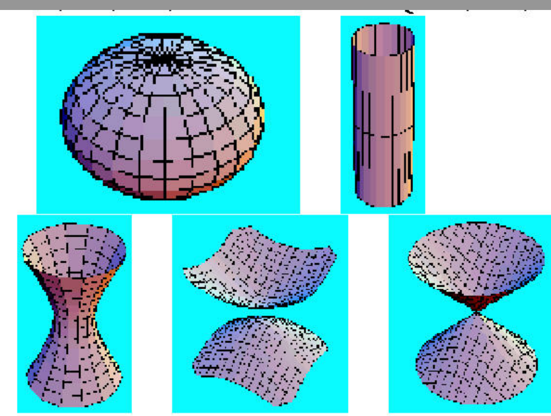
Degenerate quadrics

- planes (no quadratic terms),
- pairs of parallel planes (e.g. $x^2 - 1 = 0$)
- pairs of intersecting planes (e.g. $x^2-1 = 0$)
- elliptic cylinders (e.g. $x^2+z^2-1=0$)
- hyperbolic cylinders (e.g. $x^2-z^2-1=0$)
- parabolic cylinders (e.g. $x^2 -z = 0$)



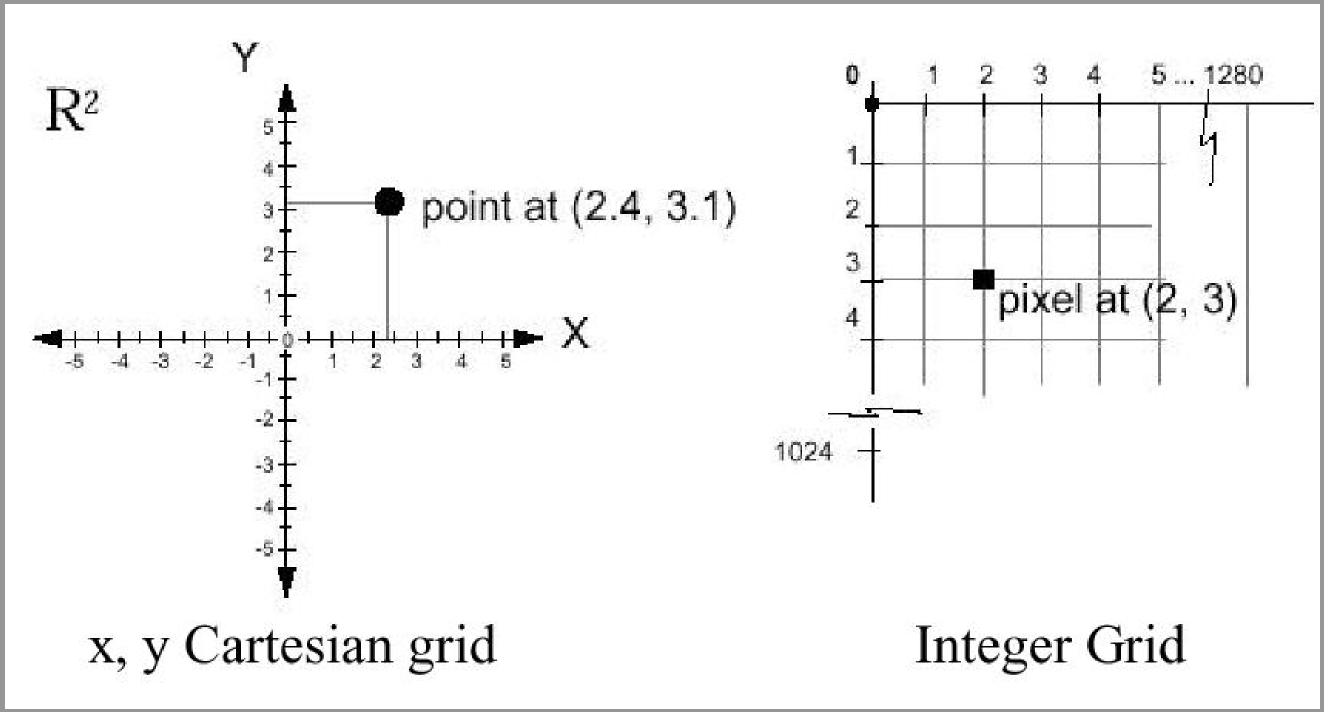






http://amath.colorado.edu/appm/staff/fast/java/qg.html

Coordonate dispozitiv



Optimizarea alg.de trasare a primitivelor

? maxim viteza de generare a unei primitive. Prin:

1. $\searrow$ nr. de operaţii în virgulă mobilă, în favoarea celor cu nr.întregi

2. min.nr.de * şi /

Metoda de trasare incrementală (1)

– bazată pe scheme cu diferenţe între mărimi asociate unor pct. succesive ale primitivei (exprimate prin relaţii liniare de transformare iterativă a unor variabile, implementate prin op. + şi - în nr. întregi)

– eficienţă în trasarea curbelor de grad unui şi doi

Metoda de trasare incrementală (2)

– construiește cea mai bună reprezentare discretă a curbei

(a) var. de adresare fizică (oper.simple de increm./decrem. – deplasări pe –, |, $\times$ între 2 pct. vecine de pe curbă)

(b) var. de stare (starea curentă a procesului iterativ, ușor actualizabile pt. trecerea la următorul pas)

Metoda de trasare incrementală (3)

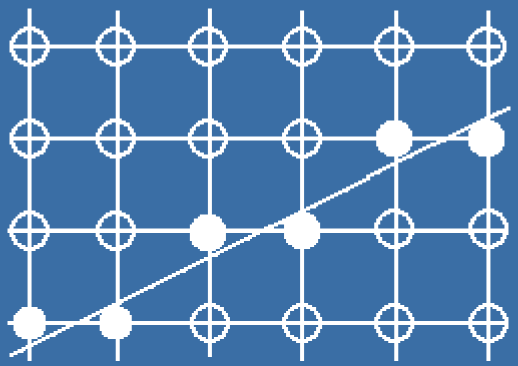
Reguli de conexiune discretă:

(a) $\forall P_0$ pct. generat la capăt de curbă, $\exists! P_1$ pct. vecin pe $-$, $|$ sau $\times$ generat pe curbă

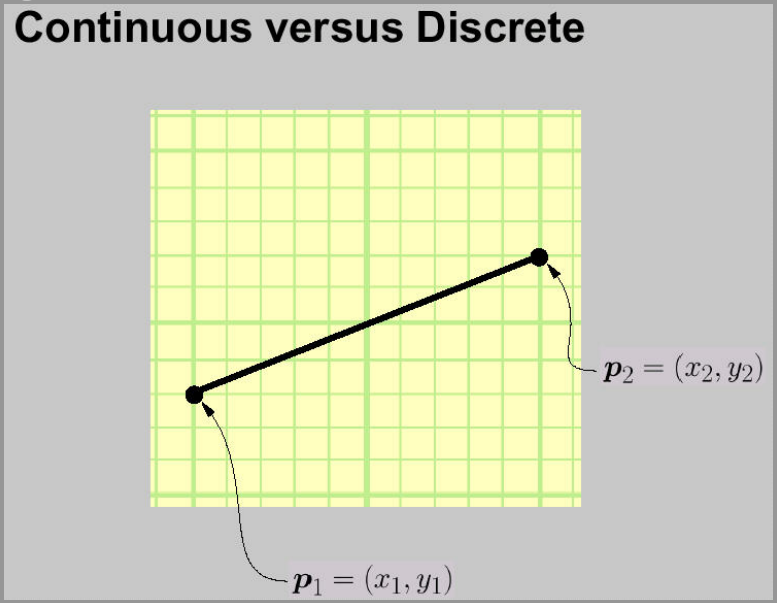
(b) $\forall P_0$ pct. în „Int”(curbă), $\exists! P_1, P_2$ 2 pct. vecine pe $-$, $|$ sau $\times$ generate pe curbă a.î. $P_1\widehat{P_0}P_2 = 135^\circ \ \vee \ 180^\circ$.

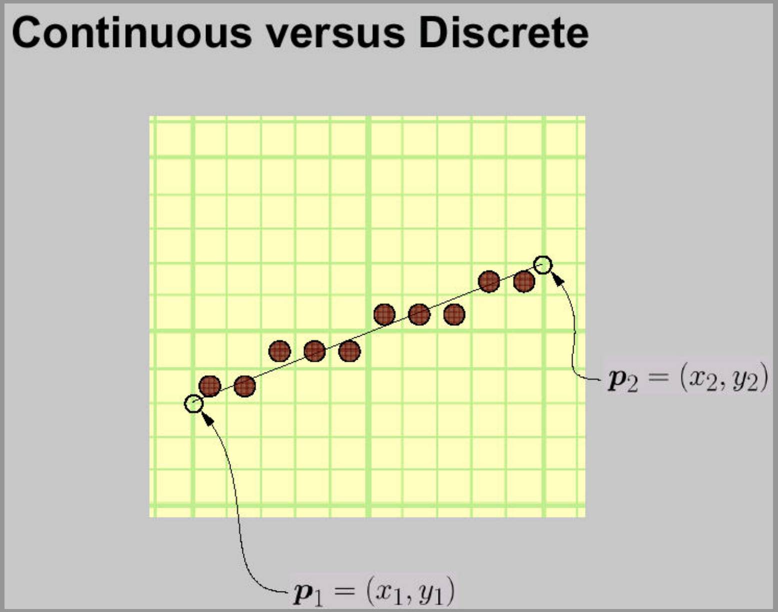
Segmente de linii

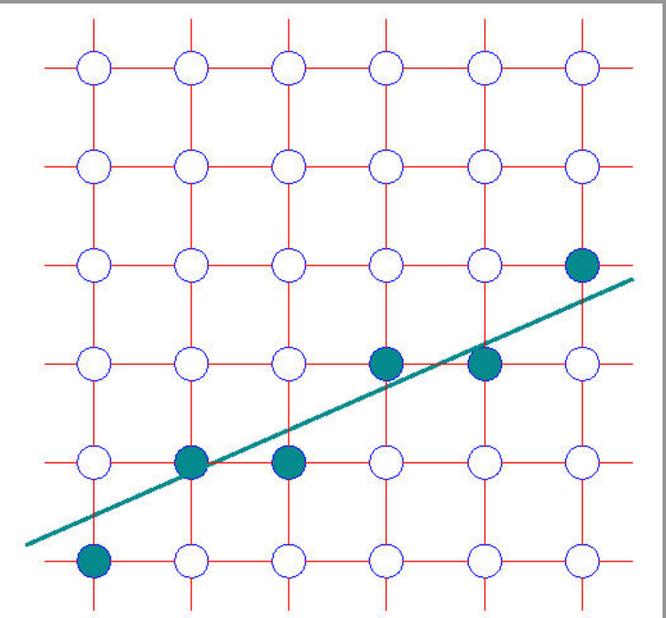
Trasare=conversie prin baleiaj=conversie scan= ? pixeli cei mai apropiaţi se imaginea ideală a segmentului

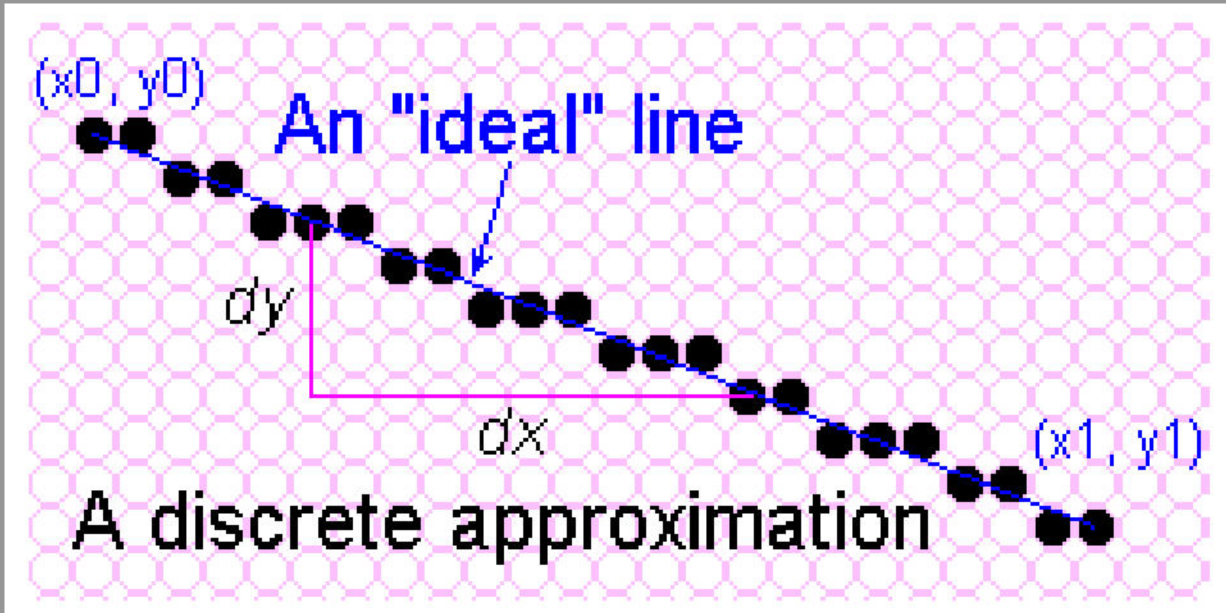


Segmente de linii









Algoritmul incremental de bază: algoritmul DDA (Digital Differential Analyzer)

? AB discret, $A(x_A, y_A)$, $B(x_B, y_B)$,

$x_A, y_A, x_B, y_B \in \mathbb{N} \cap$ spațiul vizibil

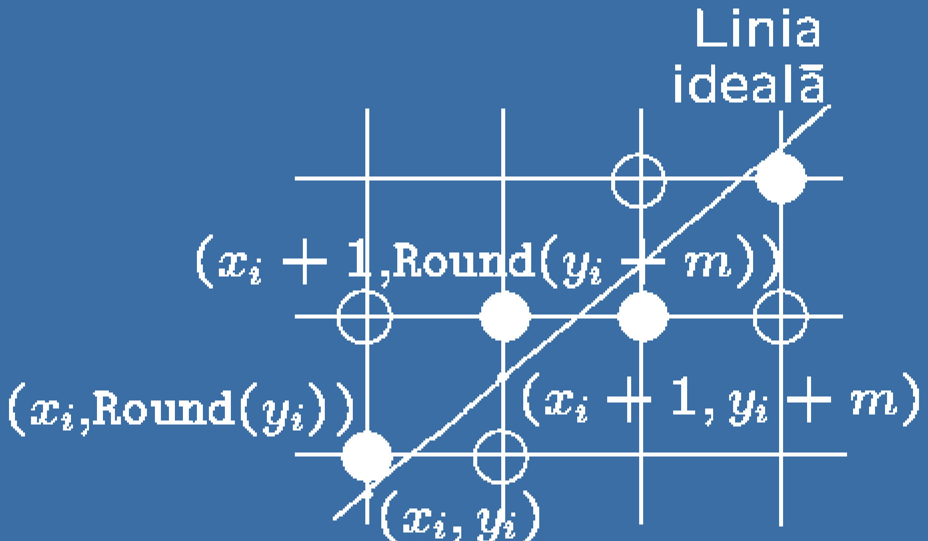
Ec.dr.AB: $y = y_A + m(x - x_A), m = \frac{y_B - y_A}{x_B - x_A}$,

Stategie simplă: Pp. $x_B > x_A$. Calculez m.

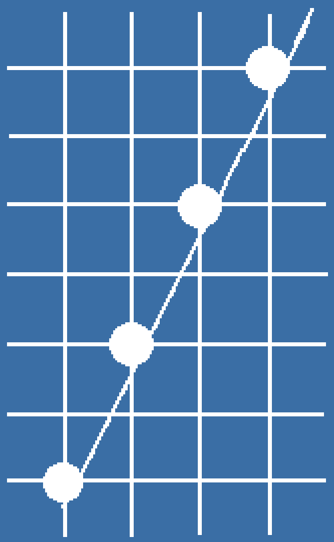
Fie $x_0 = x_A < x_1 = x_0 + 1 < \cdots < x_i < \cdots < x_n = x_B$

Calculez $x_i \to y_i = mx_i + y_A - mx_A$.

Algoritmul DDA (2)



(a)



(b)

Algoritmul DDA (3)

Trasez (x_i,Round(y_i)).

Eficienţă: la fecare pas $-$ 1 $*$ în virg.mob., 1 $+$, 1 fcţ. Round

? putem elimina $*$. Da! Cum: y_i++!

Pp. aprins (x_i, y_i). Următorul: (x_{i+1}, y_{i+1}) cu $x_{i+1} = x_i + 1$ şi $y_{i+1} = \mathrm{Round}(y_A + m(x_{i+1} - x_A)) = \mathrm{Round}(y_A + m(x_i + 1 - x_A)) = \mathrm{Round}(y_i + m)$.

Algoritmul DDA (4)

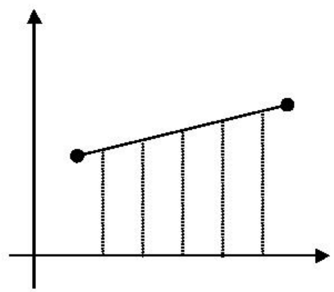
Caz $x_B < x_A$? increment pt x, $s = \text{sgn}(x_B - x_A)$
Caz $|m| > 1$? calcul $x = f(y)$, incrementare după y cu $s = \text{sgn}(y_B - y_A)$.

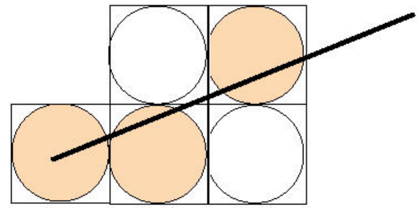
```
Proc Line(       {Presupune că −1 ≤ m ≤ 1, x0 < x1}
   x0, y0,       {Punct stânga}
   x1, y1,       {Punct dreapta}
   value : int)  {Valoarea dată pixelilor liniei}
var
```

```
    x:  int;          {x variază între x0 şi x1}
    dy, dx, x, m : real;
begin
    dy := y1 - y0;
    dx := x1 - x0;
    y := y0;  m := dy/dx;
    for x := x0 to x1 do
        begin
            {Setează val.pixel}
            WritePixel(x, Round(y), value);
            y := y + m
        end
end; {Line}
```

Algoritmul DDA

DDA(Digital Differential Analyzer) algorithm



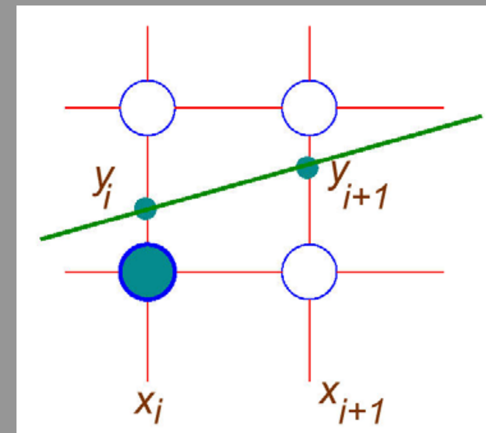


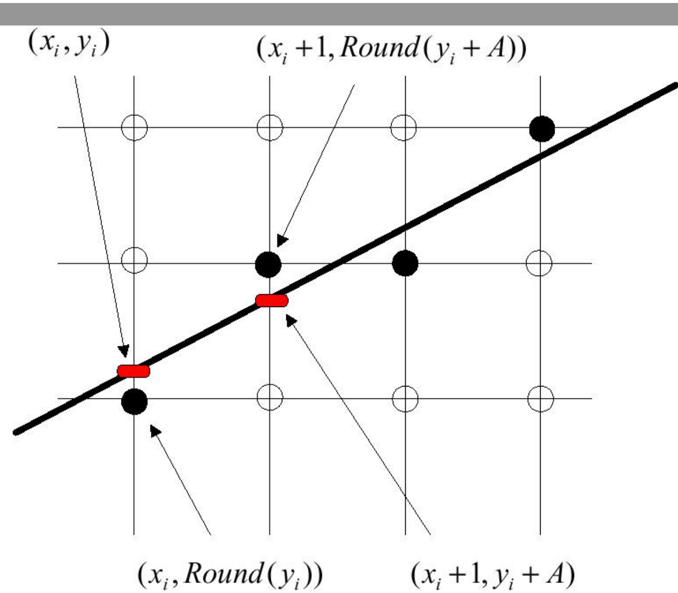
for positive slope,

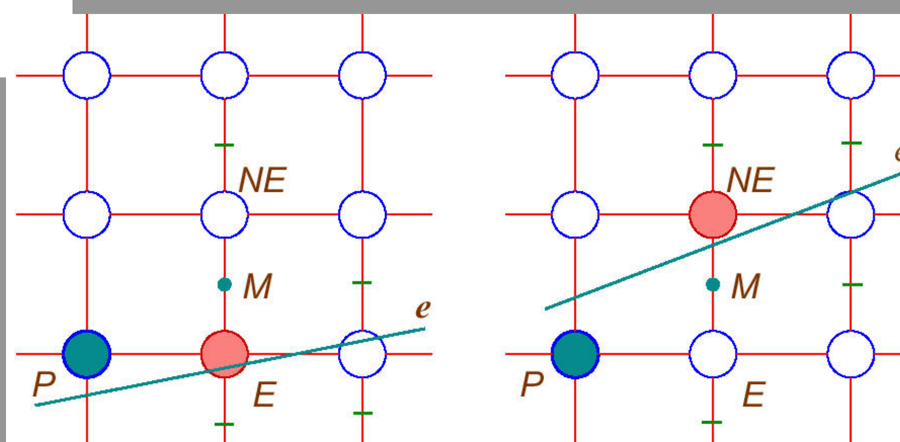
if $m \le 1,\ \Delta x = 1,\ \ y_{k+1} = y_k + m$

if $m \ge 1,\ \Delta y = 1,\ \ x_{k+1} = x_k + \dfrac{1}{m}$

- round-off error
- floating-point operation







M *above* $e \Longrightarrow$ Choose $E = (x_P + 1, y_P)$.

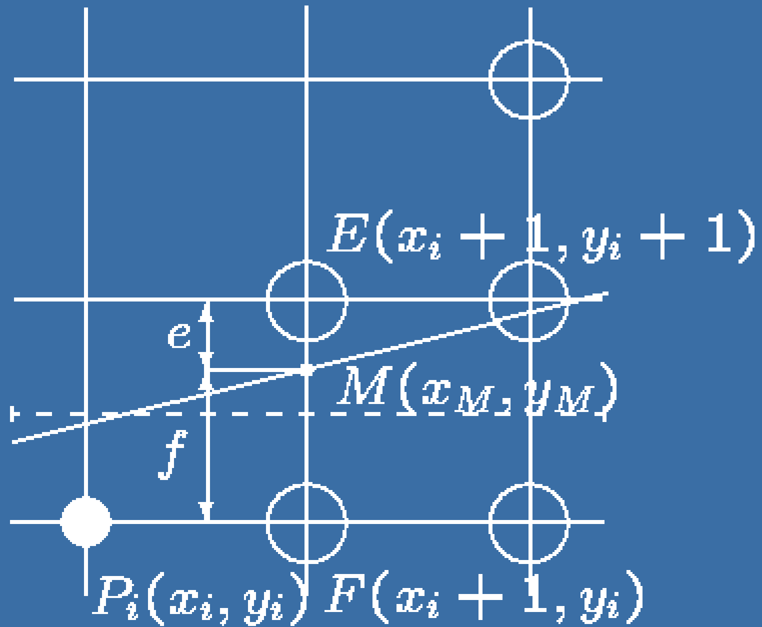
M *below* $e \Longrightarrow$ Choose $NE = (x_P + 1, y_P + 1)$

Alg.Bresenham (=alg.pct.de mijloc) (1)

DDA: Round calcule în virg.mob.!

Bresenham: operaţii numai în nr. întregi

Caz: $m \in (0, 1)$. Pp. pas i aprins $P(x_i, y_i)$.



Alg.Bresenham (2)

Pas $i+1$: $\xrightarrow{\text{reg.conexiune discretă}}$

$$E(x_i+1, y_i+1) \text{ sau } F(x_i+1, y_i).$$

Criteriul de decizie: apropierea de seg. ideal, adică $\begin{cases} F, & \text{dacă } f < e \\ E & \text{altfel} \end{cases}$

<u>Pas 0:</u> $x_A ? x_B \xrightarrow{\text{[interschimbare]}} x_A < x_B$

$T(-x_A, -y_A)$ a.î. ec. dreptei $y = \frac{dy}{dx}x$,

$dx := x_B - x_A, dy := y_B - y_A$; $x_0 = y_0 = 0$.

Alg.Bresenham – pasul i (3)

$M = \mathrm{Dr}(x = x_i + 1) \cap AB$:

$x_M = x_i + 1, y_M = \frac{dy}{dx} x_M$.

? $f = y_M - y_i, \; e = y_i + 1 - y_M$.

$$dx(f - e) = dx(2y_M - 2y_i - 1) =$$
$$= dx\left[2\frac{dy}{dx}(x_i + 1) - 2y_i - 1\right] =$$
$$= 2dy(x_i + 1) - 2y_i dx - dx.$$

$dx > 0$ (pas 0) $\to$ sgn$(f - e) =$sgn$(dx(f - e))$.

Factor de decizie $d_{i+1} : \overset{\text{not}}{=} dx(f - e)$.

Alg.Bresenham – pasul i (4)

? Incremental

$$\begin{cases} d_{i+1} = 2(x_i dy - y_i dx) + 2dy - dx, \\ d_i = 2(x_{i-1} dy - y_{i-1} dx) + 2dy - dx \end{cases} \Rightarrow$$

$$d_{i+1} - d_i = 2[dy(x_i - x_{i-1}) - dx(y_i - y_{i-1})]$$

$$\overset{x_i - x_{i-1} = 1}{\Rightarrow} \quad d_{i+1} = d_i + 2dy - 2dx(y_i - y_{i-1})$$

$$d_1 = 2(x_0 dy - y_0 dx) + 2dy - dx = 2dy - dx.$$

Variabilele de stare sunt (x_i, y_i, d_i).

Alg.Bresenham – pasul i (5)

Criteriul decizional este:
– dacă $d_{i+1} \geq 0$ se alege $E(x_i+1, y_i+1)$ şi $d_{i+2} = d_{i+1} + 2(dy - dx)$
– dacă $d_{i+1} < 0$, atunci se alege $F(x_i + 1, y_i)$ şi $d_{i+2} = d_{i+1} + 2dy$

```
Procedure MidpointLine (x0, y0, x1, y1, value: integer)
var
   dx, dy, incrE, incrF, d, x, y : integer;
begin
   dx := x1 - x0; dy := y1 - y0;
   d := 2 * dy - dx;          {Val.iniţială variabilă decizie}
   incrE := 2 * (dy - dx){Increment pentru mutarea la E}
   incrF := 2 * dy;           {Increment pentru mutarea la F}
   x := x0;
```

```
y := y0;
WritePixel(x, y, value){Pixel de start}
while x < x1 do
        begin
        if d < 0 then
            begin            {Alege F}
                  d := d + incrF;
                  x := x + 1
            end
        else
            begin            {Alege E}
                  d := d + incrE;
                  x := x + 1;
                  y := y + 1;
            end;
        WritePixel (x, y, value) {Cel mai apropiat pixel}
        end {while}
end; {MidpointLine}
```

Alg.Bresenham – de ce alg.pct.mijloc

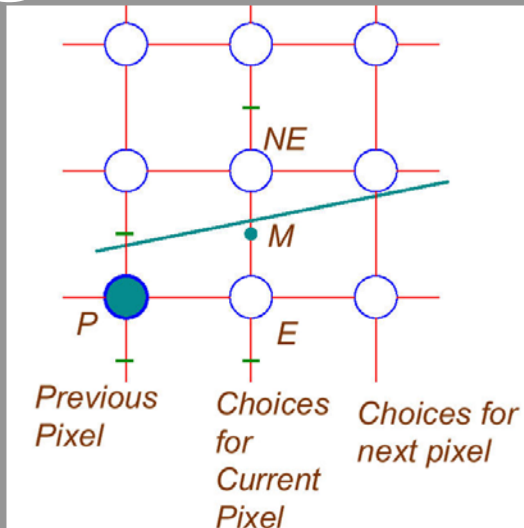
Alg. pct de mijloc pt. seg. dreaptă pe $D(x, y)$ $= ax + by + c$, $a > 0$ are $\underline{d_{i+1}} = 2D(x_i + 1, y_i + \frac{1}{2})$.

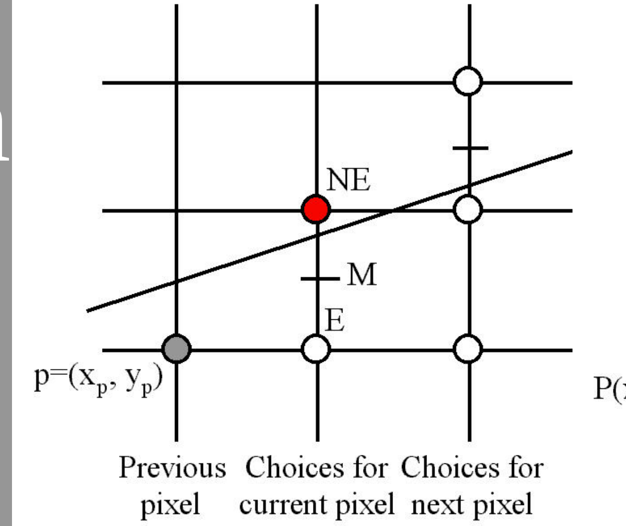
– $\underline{d_{i+1} < 0}$ $\Rightarrow$ alege E şi $d_{i+2} = 2D(x_i + 2, y_i + 1/2) = 2[a(x_i+2)+b(y_i+1/2)+c] = d_{i+1}+2dy$

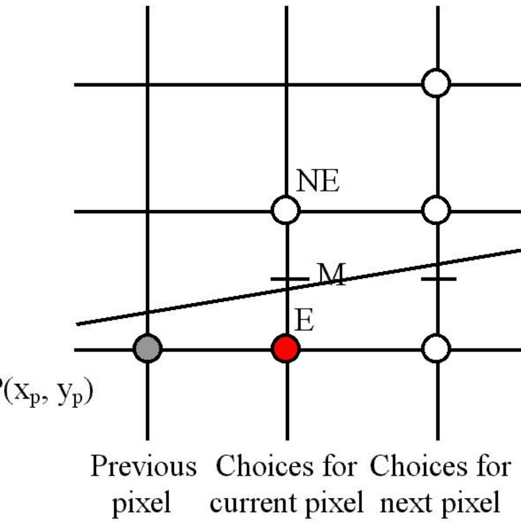
– $\underline{d_{i+1} \geq 0}$ $\Rightarrow$ alege F şi $d_{i+2} = 2D(x_i + 2, y_i + 3/2) = d_{i+1} + 2(a + b) = d_{i+1} + 2(dy - dx)$.

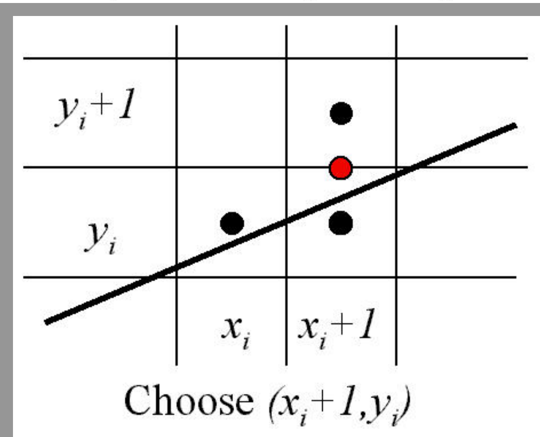
$d_1 = 2D(x_A + 1, y_A + 1/2) = 2D(x_A, y_A) + 2a + b = 2a + b = 2dy - dx$,

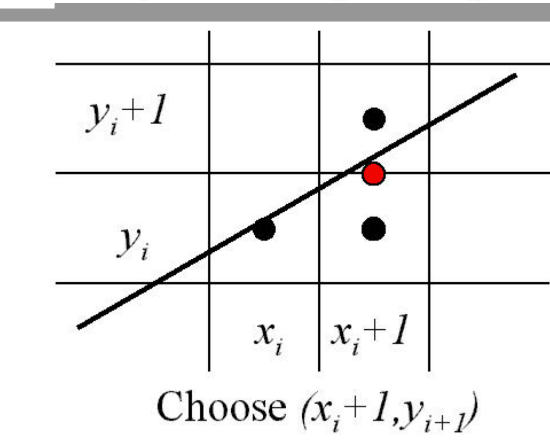
Alg.Bresenham

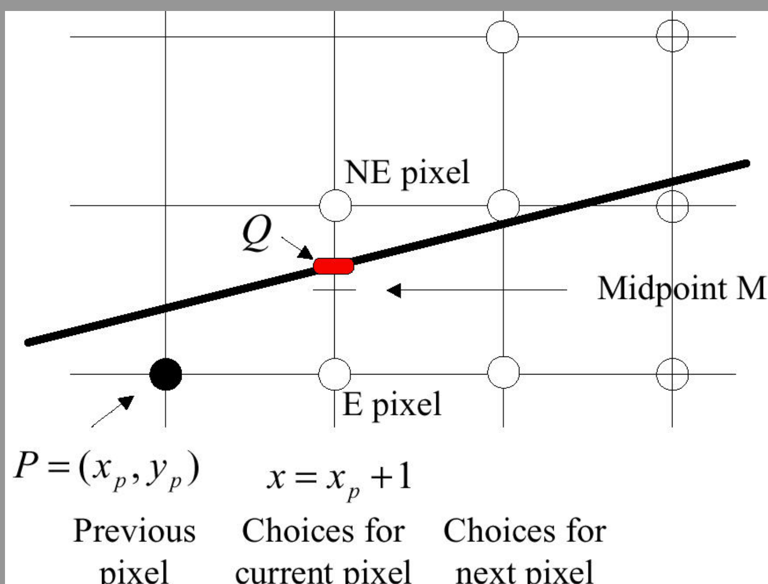






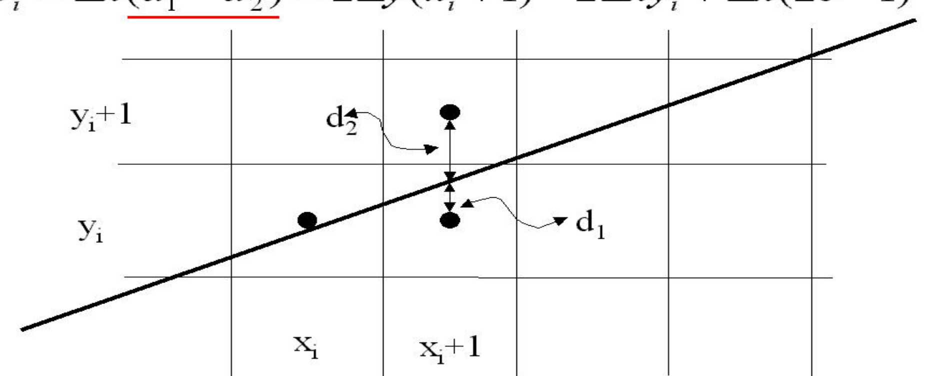




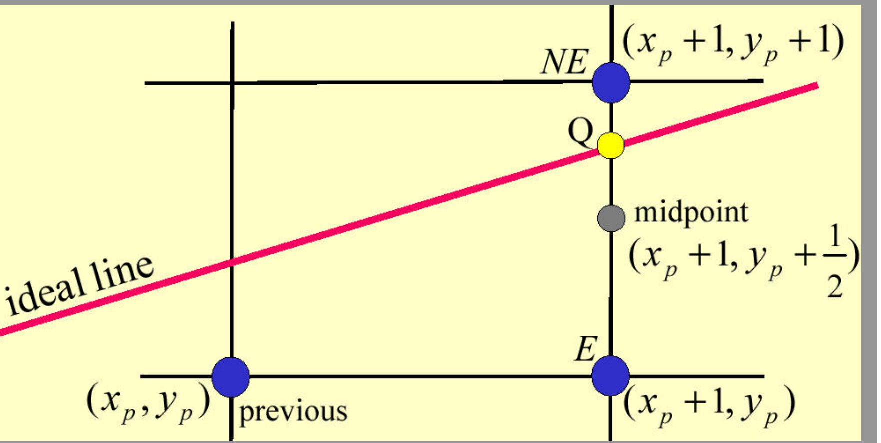


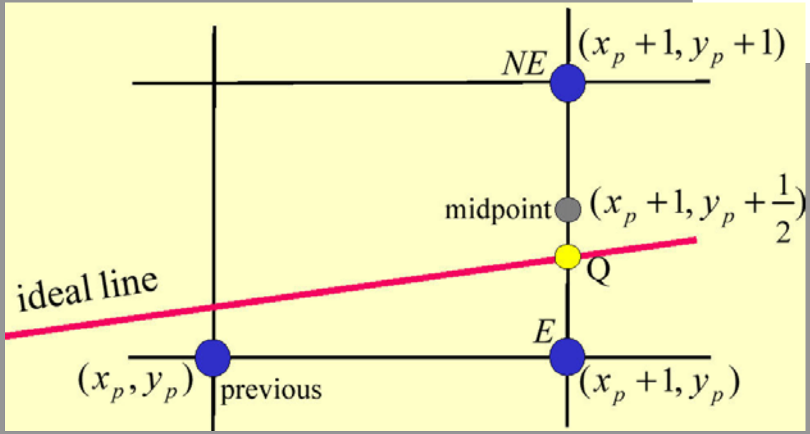
- Decision variable is:

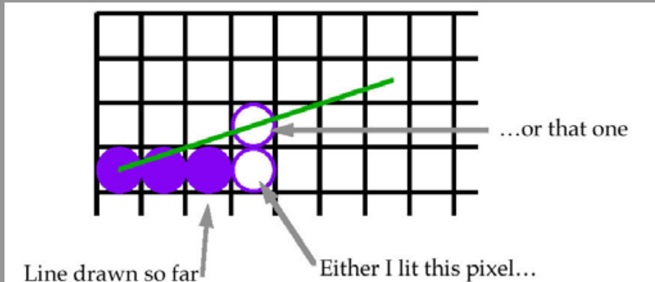
$$p_i = \Delta x(d_1 - d_2) = 2\Delta y(x_i + 1) - 2\Delta x y_i + \Delta x(2c - 1)$$



Alg.Bresenham







Bresenham s Line Algorithm (continue)

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

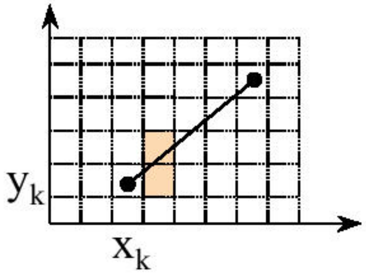
$$p_{k+1} = p_k + 2\Delta y - 2\Delta x(y_{k+1} - y_k)$$

if $p_k < 0$, then $y_{k+1} = y_k$

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

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

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

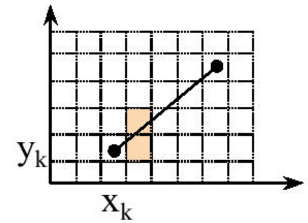


$$\text{from DDA, } y_{k+1} = y_k + m = y_k + \frac{\Delta y}{\Delta x}$$

$$\frac{\Delta y}{\Delta x} ? 0.5 \xrightarrow{<} 2\Delta y - \Delta x < 0 \quad (y_{k+1} = y_k) \Rightarrow 2\frac{\Delta y}{\Delta x} ? 0.5$$

$$\xrightarrow{\geq} 2\Delta y - \Delta x \geq 0 \quad (y_{k+1} = y_k + 1) \Rightarrow 2\frac{\Delta y}{\Delta x} - 1 ? 0.5$$

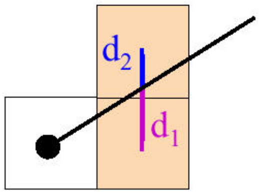
Bresenham s Line Algorithm in 1965



$(\mathrm{x_k}, \mathrm{y_k}) \longrightarrow (\mathrm{x_k}, \mathrm{y_k})$

or

$(\mathrm{x_k}, \mathrm{y_k}+1)$



$$y_k = mx_k + b$$

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

$$d_1 = y - y_k = m(x_k + 1) + b - y_k$$

$$d_2 = (y_k + 1) - y = y_k + 1 - m(x_k + 1) - b$$

$$d_1 - d_2 = 2m(x_k + 1) - 2y_k + 2b - 1$$

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

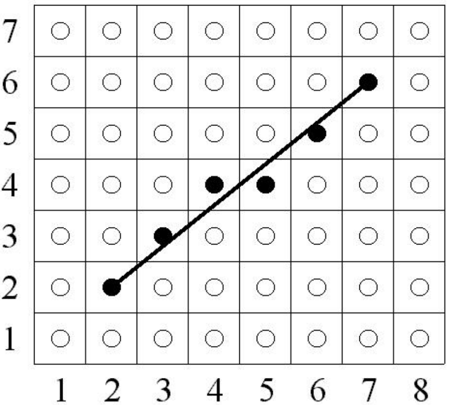
$$= 2\Delta y x_k - 2\Delta x y_k + C$$

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

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

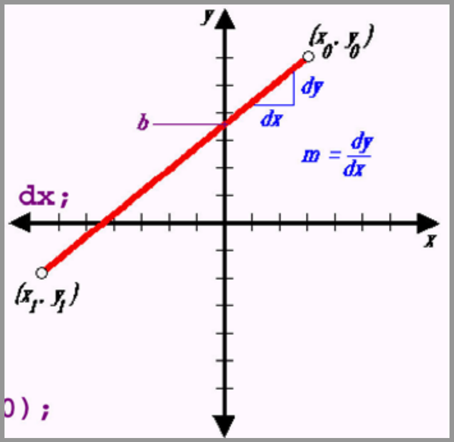
$$p_0 = 2\Delta y - \Delta x$$

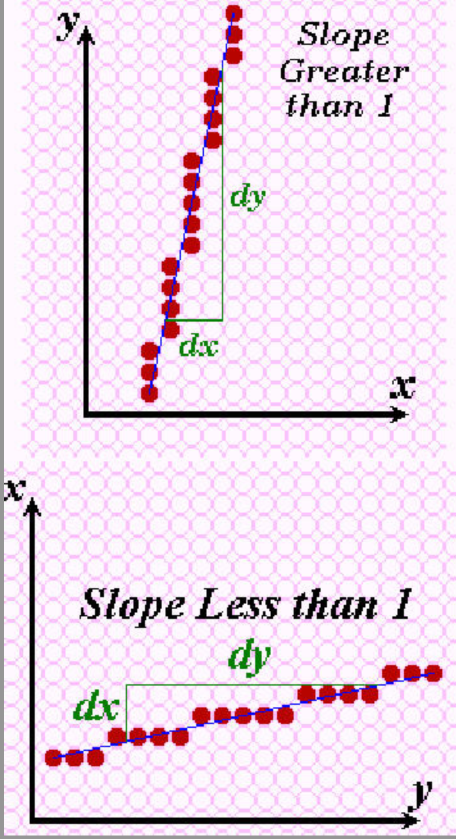
Alg.Bresenham

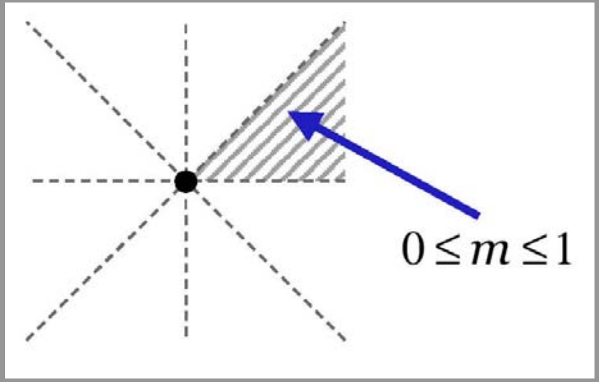


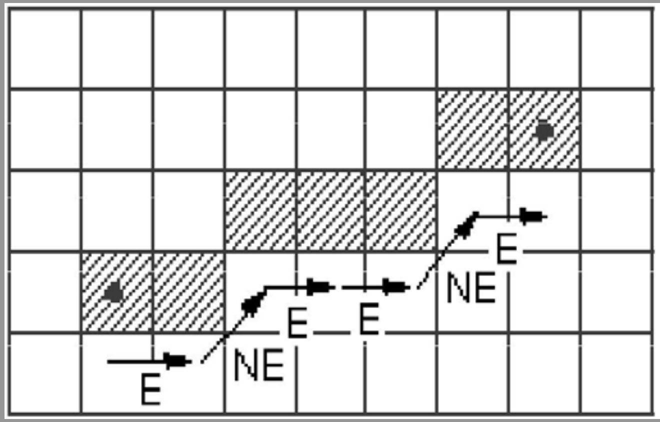
$\Delta x=5,\ \Delta y=4$

i	x	y	p
1	2	2	3
2	3	3	1
3	4	4	-1
4	5	4	7
5	6	5	5
6	7	6	3

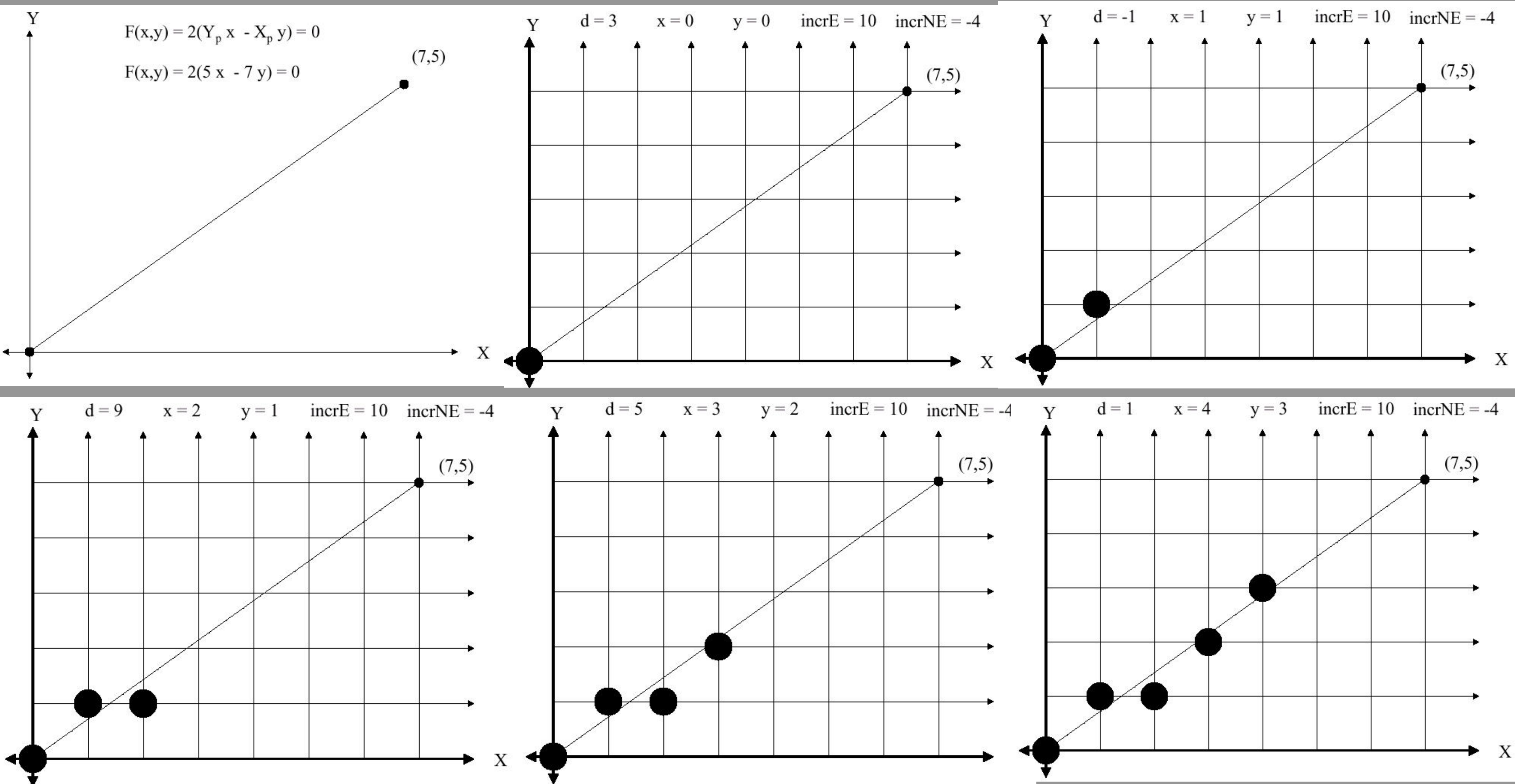




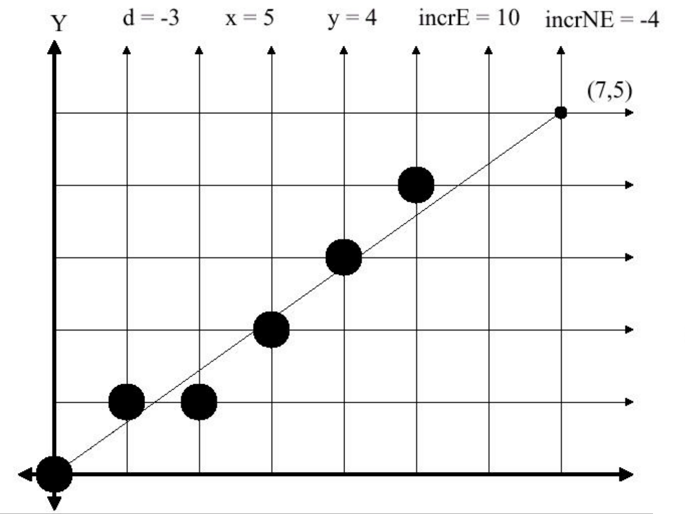


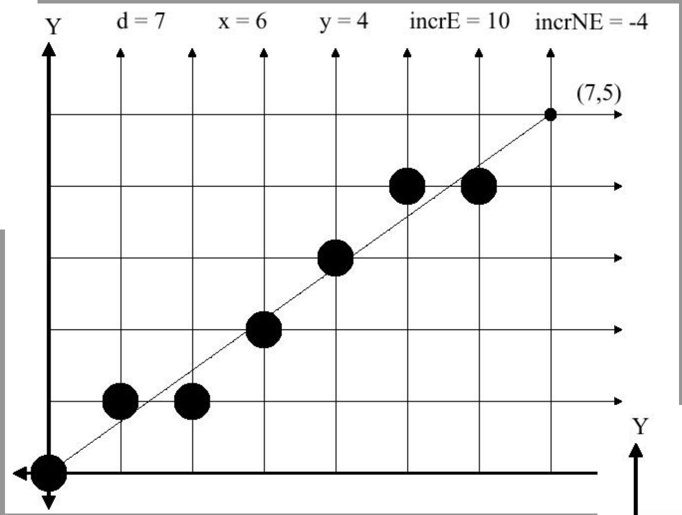


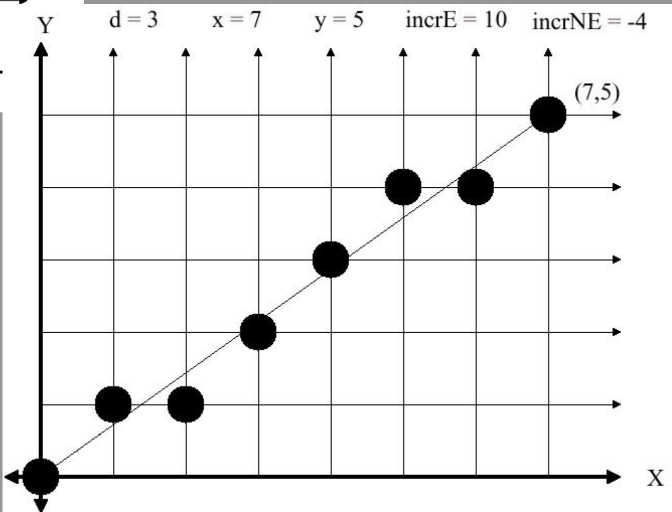
Alg.Bresenham



Alg.Bresenham







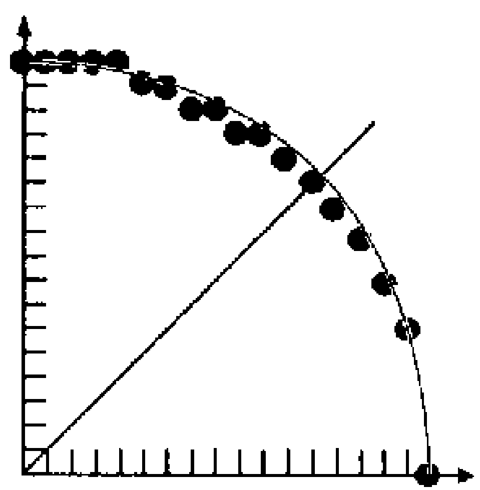
Trasare cercuri (1)

Fie $x^2 + y^2 = R^2$, $R \in \mathbb{N}$; general: $T(x_0, y_0)$

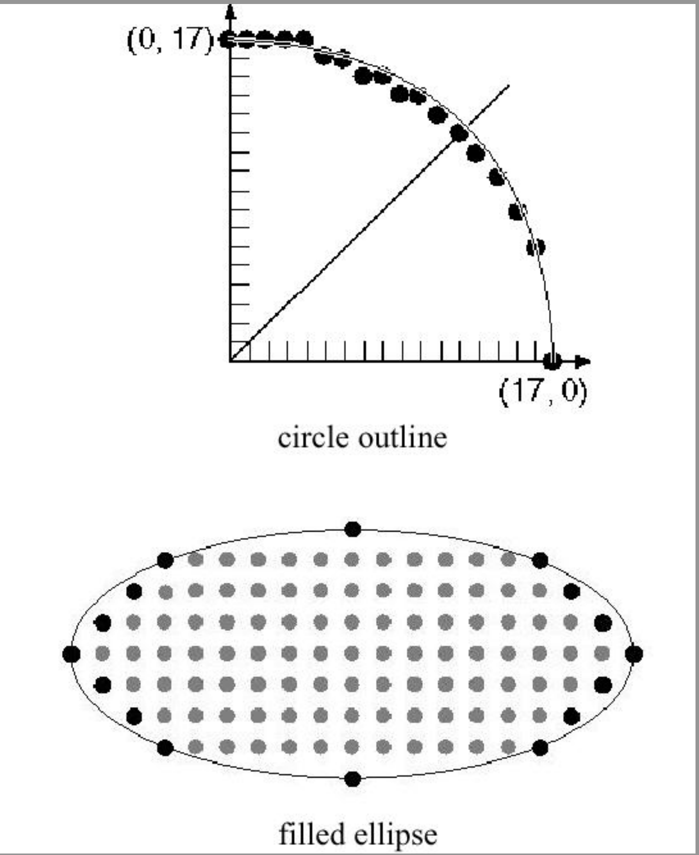
Metoda I: $y = \pm\sqrt{R^2 - x^2}$, trasare sfert cerc (+ simetrii), $x_0 = 0 < x_1 = 1 < \cdots < x_R = R$, $y_i = \sqrt{R^2 - x_i^2}$.

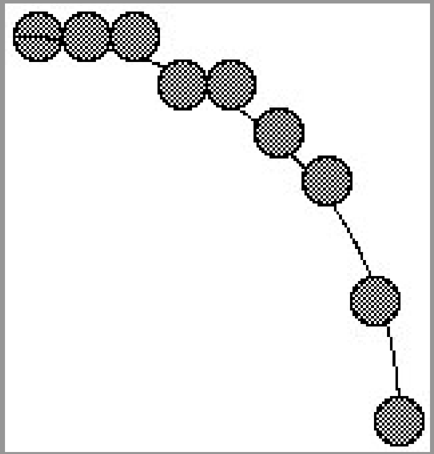
Eficiență ?: per pas: 1 $*$, 1 $\sqrt{\ }$, 1 Round.

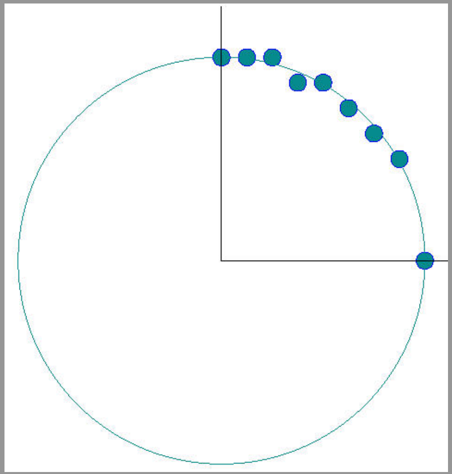
Corectitudine ? Găuri!

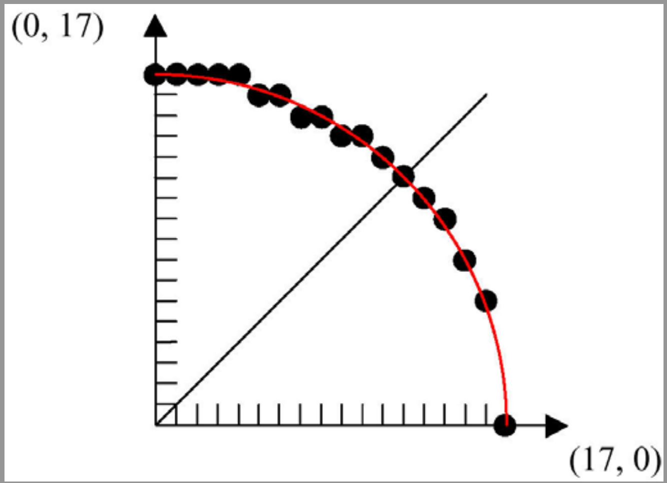


Trasare cercuri-probleme

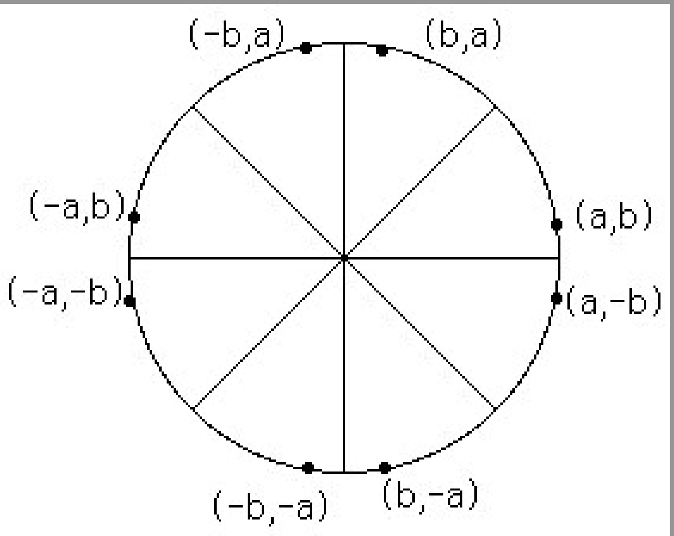


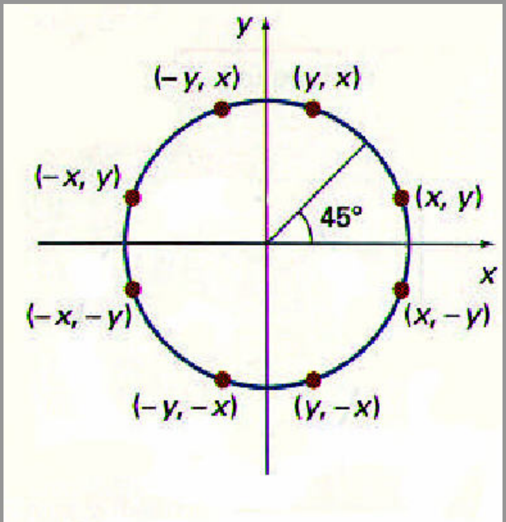




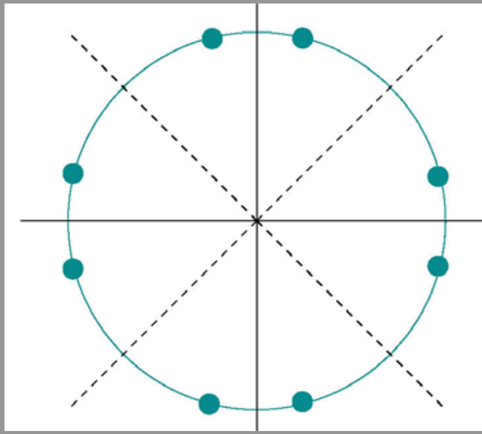


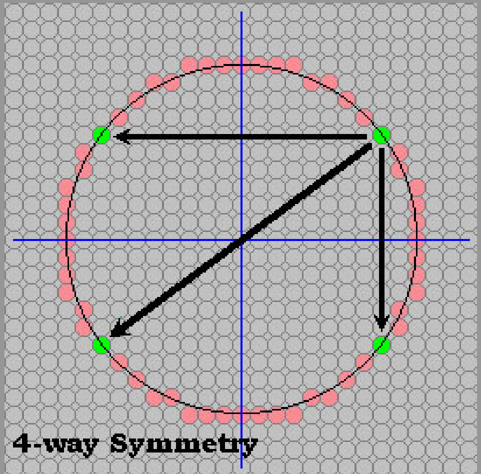
Trasare cercuri- simetrii

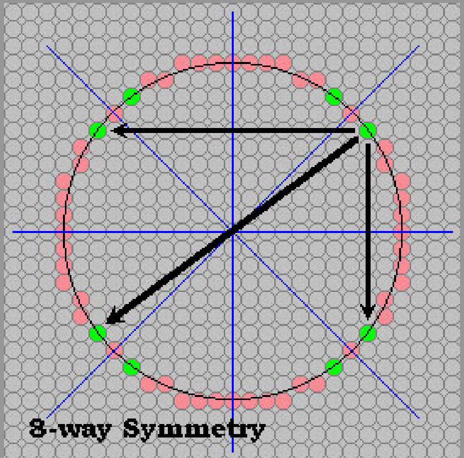


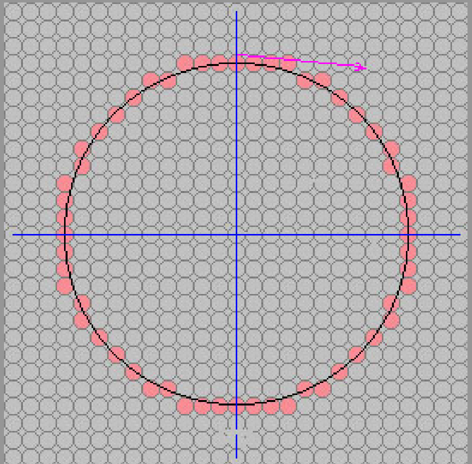


Symmetry of a circle. Calculation of a circle point (x, y) in one octant yields the circle points shown for the other seven octants.





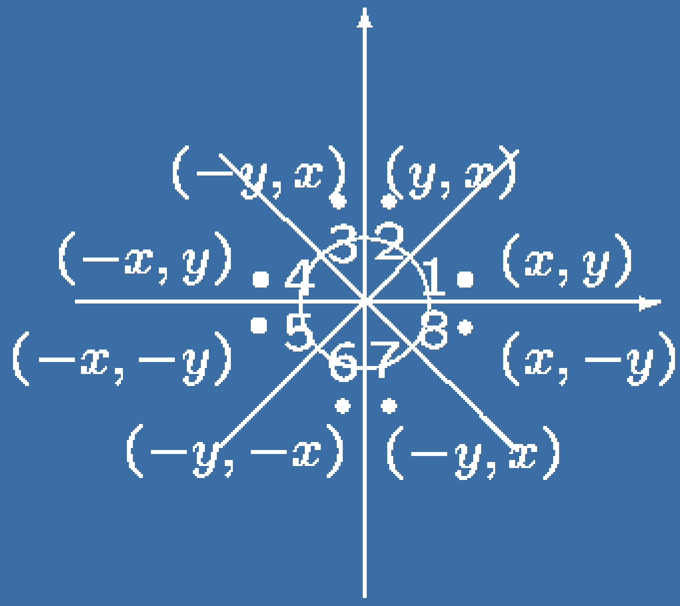




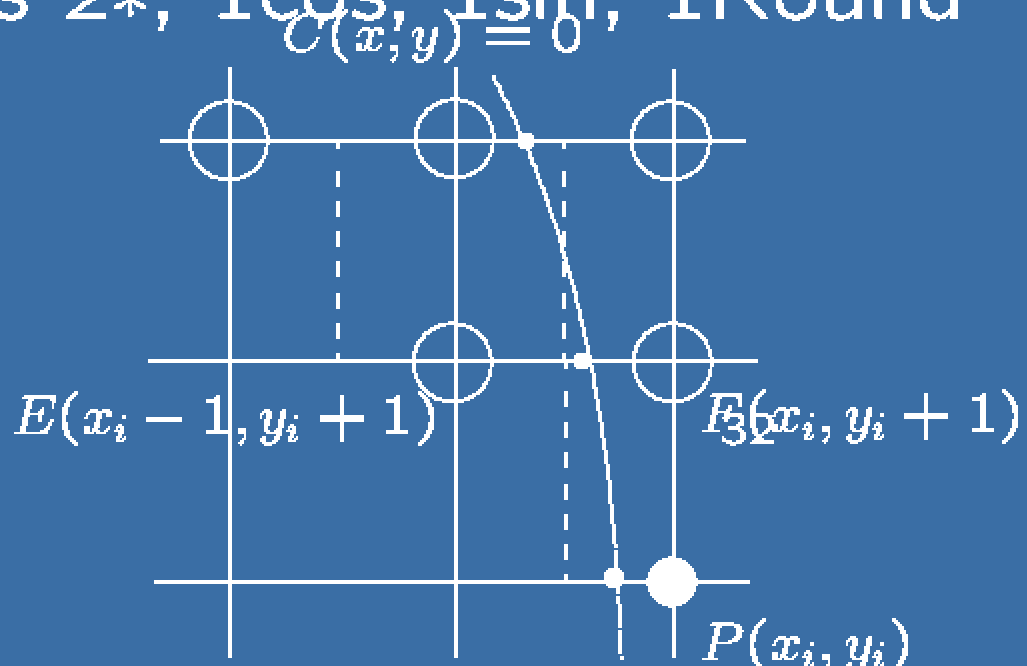
Metoda II: $\{(R\cos\theta_i, R\sin\theta_i)\}_i$, $\theta_0 = 0° < \cdots < \theta_i < \cdots < \theta_n = 45°$ + simetrii

Problemă: ? $\theta_{i+1} - \theta_i$ a.î. generare Pct. vecine şi Nu suprapunere

Eficienţă: per pas 2$*$, 1cos, 1sin, 1Round



(a)



(b)

Trasare cercuri (3)

Metoda III: studiu pe octanţi, metodă incrementală, calcule în numere întregi! Pp. sens trigonometric de parcurs al cercului.

Octant 1:

deplasări între pct. vecine $\uparrow$ sau $\nwarrow$

Octant 2:

deplasări între pct. vecine $\leftarrow$ sau $\nwarrow$

Algoritmul Bresenham pentru cercuri (1)

Fie $C(x,y) = x^2 + y^2 - R^2$ și octant 1.

Obs.: $C(x,y) \begin{cases} < 0 & (x,y) \in \text{Int(cerc)} \\ > 0 & (x,y) \in \text{Ext(cerc)} \\ = 0 & \text{cerc} \end{cases}$

Pas i: $P_i(x_i, y_i)$; $i+1$: $E(x_i - 1, y_i + 1) \vee F(x_i, y_i + 1)$ fcț. de $\min\{|C(E)|, |C(F)|\}$.

Factor de decizie:

$$\Delta_i \overset{\text{not}}{:=} C(x_i, y_i + 1) + C(x_i - 1, y_i + 1) = x_i^2 + (x_i - 1)^2 + 2(y_i + 1)^2 - 2R^2.$$

Algoritmul Bresenham pentru cercuri (2)

$$P_{i+1} = \begin{cases} F & \Delta_i < 0 \ (|C(F)| < |C(E)|) \\ E & \text{altfel} \end{cases},$$

$$\Delta_{i+1} = \begin{cases} \Delta_i + 4y_{i+1} + 2, & \Delta_i < 0 \\ \Delta_i - 4x_{i+1} + 4y_{i+1} + 2 & \text{altfel} \end{cases}$$

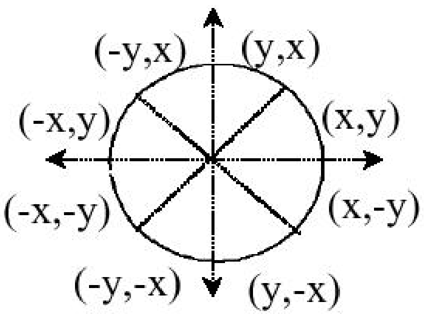
Variabile de stare:

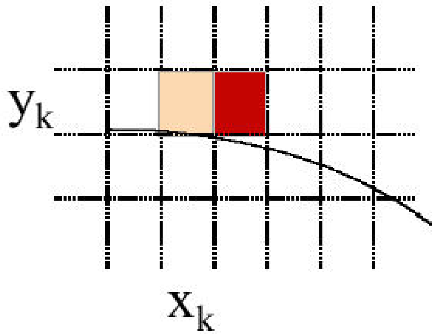
(x_i, y_i, Δ_i).

Valori inițiale: $x_0 = R$, $y_0 = 0$, $\Delta_0 = 3 - 2R$.

Alg.Bresenham pt. Cercuri

- Bresenham s Circle Algorithm(Midpoint Algorithm) in 1977





$$f_{circle}(x,y) = x^2 + y^2 - r^2 \begin{cases} < 0 & inside \\ = 0 & on \\ > 0 & \text{outside} \end{cases}$$

$$p_k = f_{circle}(x_k + 1, y_k - \frac{1}{2}) = (x_k + 1)^2 + (y_k - \frac{1}{2})^2 - r^2$$

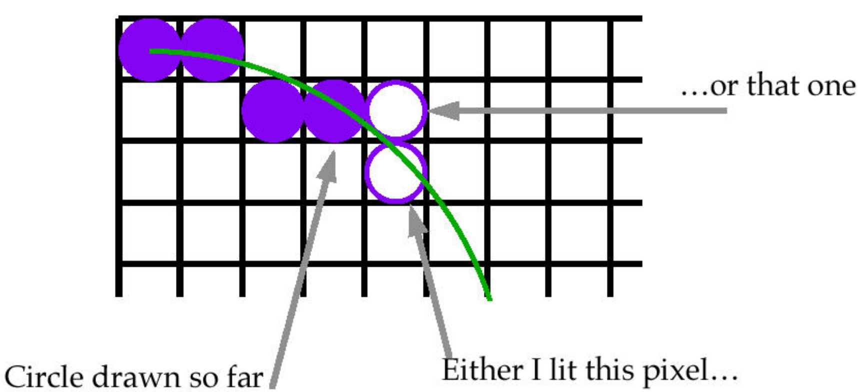
$$p_{k+1} = f_{circle}(x_{k+1} + 1, y_k - \frac{1}{2}) = \{(x_k + 1) + 1\}^2 + (y_{k+1} - \frac{1}{2})^2 - r^2$$

$$p_{k+1} = p_k + 2(x_k + 1) + (y_{k+1}^2 - y_k^2) - (y_{k+1} - y_k) + 1$$

$$p_{k+1} = \begin{cases} p_k + 2x_{k+1} + 1 & \text{if } p_k < 0 \\ p_k + 2x_{k+1} + 1 - 2y_{k+1} & \text{if } p_k \ge 0 \end{cases}$$

$$2x_{k+1} = 2x_k + 2$$

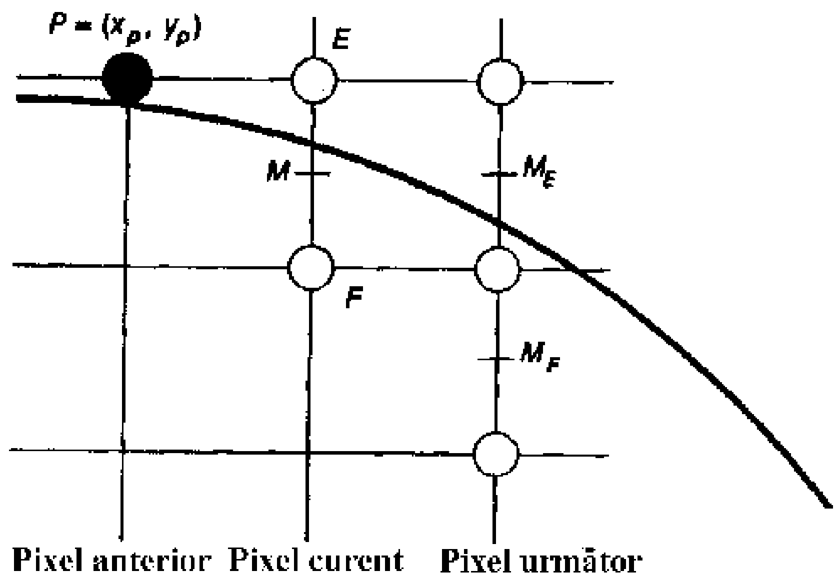
$$2y_{k+1} = 2y_k - 2 \quad \text{or} \quad 2y_{k+1} = 2y_k$$



Algoritmi incrementali pt. octantul 2

Ag.Bresenham, octant 2: ? $(x_i - 1, y_i + 1)$ sau $(x_i - 1, y_i)$, $\Delta_i = y_i^2 + (y_i + 1)^2 + 2(x_i - 1)^2 - 2R^2$.

Algoritmul punctului de mijloc. Pp. sens de parcurs în sens orar şi octant 2.



Alg.punctului de mijloc pt. cercuri (1)

Factor de decizie: $\Delta_{i+1} = C(x_i + 1, y_i - \frac{1}{2})$

$$\begin{cases} E \text{ și } \Delta_{i+2} = C(x_i + 2, y_i - \frac{1}{2}) = \Delta_{i+1} + \Delta^E_{i+1}, \\ \qquad \Delta^E_{i+1} = 2x_i + 3, \text{ dacă } \Delta_{i+1} < 0 \\ F \text{ și } \Delta_{i+2} = C(x_i + 2, y_i - \frac{3}{2}) = \Delta_{i+1} + \Delta^F_{i+1}, \\ \qquad \Delta^F_{i+1} = 2x_i - 2y_i + 5 \text{ altfel} \end{cases}$$

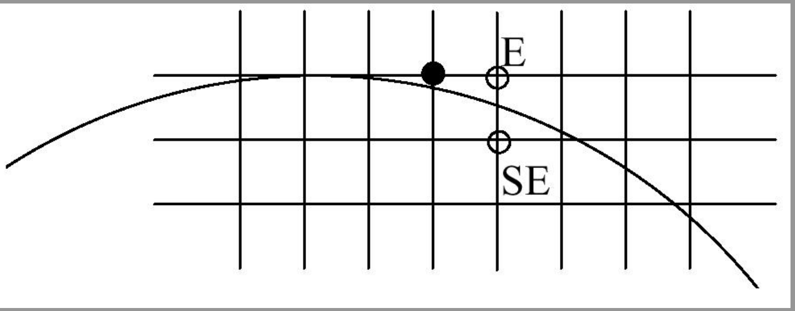
$(x_0, y_0) = (0, R)$, $C(1, R - \frac{1}{2}) = \frac{5}{4} - R$.

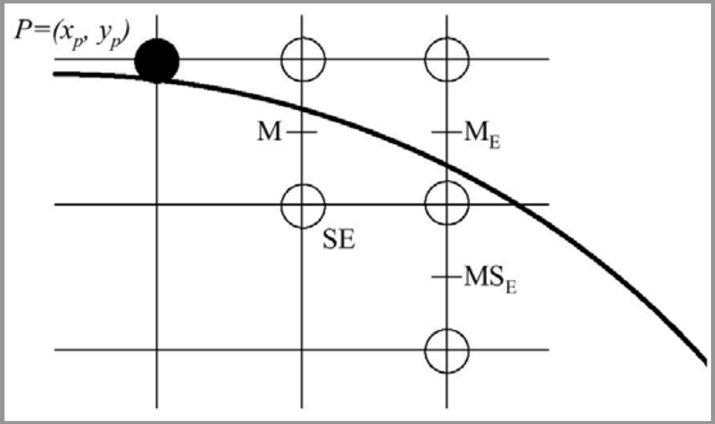
Operații în nr. raționale ?!

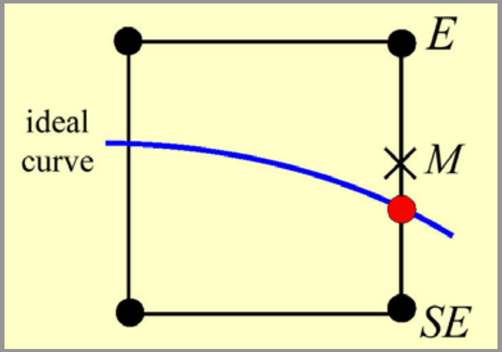
Var.decizională: $d_i = \Delta_i - \frac{1}{4} \Rightarrow d_1 = 1 - R$

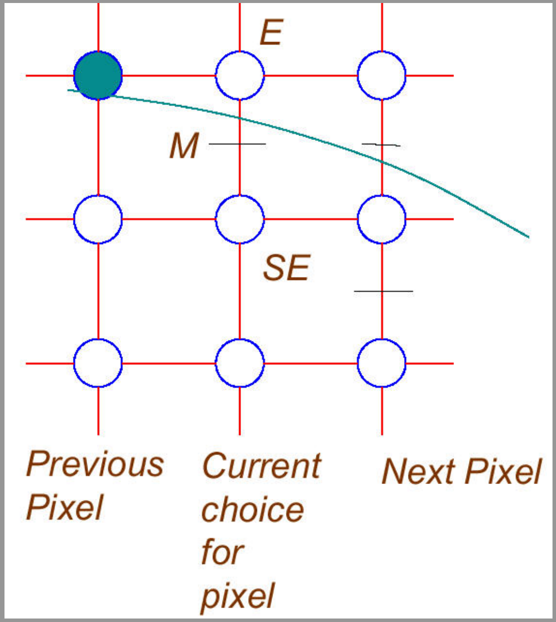
și $\Delta < 0 \;\Rightarrow\; d < -1/4 \overset{d \in Z}{\Rightarrow} d < 0$.

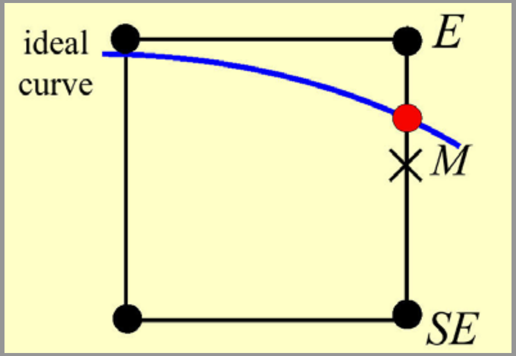
Alg.pct.de mijloc pt. cercuri











Alg.punctului de mijloc pt. cercuri (2)

Îmbunătăţire performanţe: incrementarea şi a diferenţei aplicate la variabila decizională:

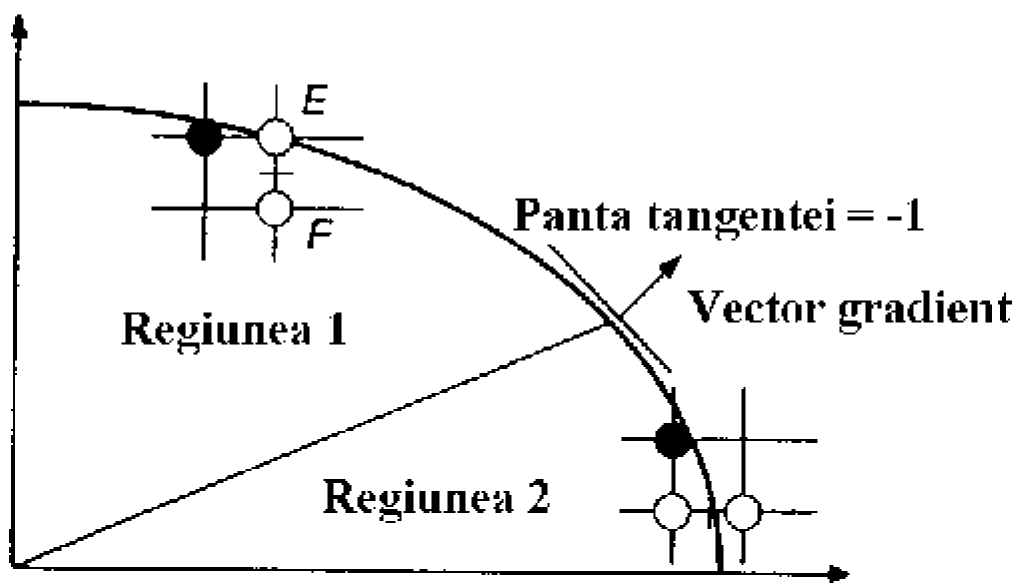
$$\begin{cases} \Delta^E_{i+2} = 2(x_i+1)+3 = \Delta^E_{i+1}+2, \\ \quad \Delta^F_{i+2} = 2(x_i+1)-2y_i+5 = \Delta^F_{i+1}+2, \text{ dacă } E \\ \Delta^E_{i+2} = 2(x_i+1)+3 = \Delta^E_{i+1}+2, \\ \quad \Delta^F_{i+2} = 2(x_i+1)-2(y_i-1)+5 = \Delta^F_{i+1}+4, \text{ alfte} \end{cases}$$

```
Procedure MidpointCircle(radius, value: integer);
{Trasează semicercul de la (0, R) la (R√2, R√2)}
{Se presupune că cercul este centrat în origine}
var
    x, y, d, deltaE, deltaF: integer;
begin
```

```
x := 0; y := radius; d := 1 − radius;
deltaE := 3; deltaF := −2 ∗ radius + 5;
WritePixel(x, y, value);
while y > x do
    begin
    if d < 0 then {Selectează E}
        begin
            d := d + deltaE; deltaE := deltaE + 2;
            deltaF := deltaF + 2; x := x + 1 end
    else
        begin
            d := d + deltaF;
            deltaE := deltaE + 2; deltaF := deltaF + 4;
            x := x + 1; y := y − 1 end;
    WritePixel(x, y, value);
    end {while}
end; {MidpointCircle}
```

Elipse drepte (1)

$$E(x,y) = b^2x^2 + a^2y^2 - a^2b^2 = 0$$



Elipse drepte (2)

Regiunea 1. Var.de decizie: $d_{i+1} = 4E(x_i + 1, y_i - \frac{1}{2})$.

$$d_{i+2} = \begin{cases} 4E(x_i + 2, y_i - \frac{1}{2}) = d_{i+1} + 4b^2(2x_i + 3), & E \\ 4E(x_i + 2, y_i - \frac{3}{2}) = d_{i+1} + 4a^2(-2x_i + 2), & F \end{cases}$$

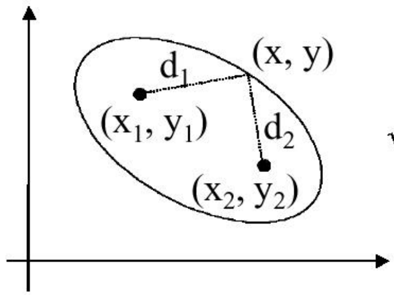
Trecerea în reg.2 când vector gradient

$\text{grad}\, E(x,y) = \frac{\partial E}{\partial x}\mathrm{i} + \frac{\partial E}{\partial y}\mathrm{j} = 2b^2x\mathrm{i} + 2a^2y\mathrm{j}$

are panta 1: dacă $a^2(y_i - \frac{1}{2}) \leq b^2(x_i + 1)$.

Elipse drepte

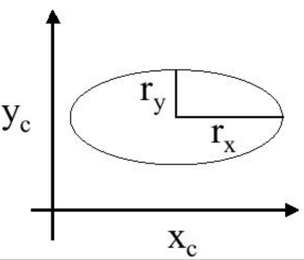
4. Ellipse-Generating Algorithms



$$d_1 + d_2 = \text{constant}$$

$$\sqrt{(x-x_1)^2+(y-y_1)^2}+\sqrt{(x-x_2)^2+(y-y_2)^2} = \text{constant}$$

$$Ax^2 + By^2 + Cxy + Dx + Ey + F = 0$$



$$\left(\frac{x-x_c}{r_x}\right)^2 + \left(\frac{y-x_c}{r_y}\right)^2 = 1$$

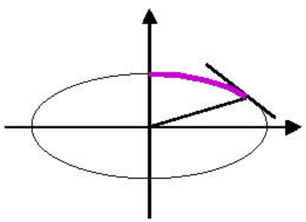
$$\Rightarrow \left(\frac{x}{r_x}\right)^2 + \left(\frac{y}{r_y}\right)^2 = 1$$

$$f_{ellipse}(x,y) = r_y^2x^2 + r_x^2y^2 - r_x^2r_y^2$$

Midpoint Ellipse Algorithm

$$f_{ellipse}(x,y) = r_y^2x^2 + r_x^2y^2 - r_x^2r_y^2$$

$$\begin{cases} <0, \text{ if } (x,y) \text{ is inside} \\ =0, \text{ if } (x,y) \text{ is on} \\ >0, \text{ if } (x,y) \text{ is outside} \end{cases}$$



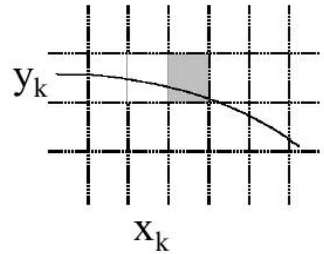
$$\frac{dy}{dx} = -\frac{2r_y^2x}{2r_x^2y} = -1$$

$$2r_y^2x \geq 2r_x^2y$$

$$2r_y^2x < 2r_x^2y$$

$$p_k = f_{ellipse}\left(x_k+1, y_k - \frac{1}{2}\right)$$

$$= r_y^2(x_k+1)^2 + r_x^2\left(y_k - \frac{1}{2}\right)^2 - r_x^2r_y^2$$



$$p_{k+1} = f_{ellipse}\left(x_{k+1}+1, y_{k+1} - \frac{1}{2}\right)$$

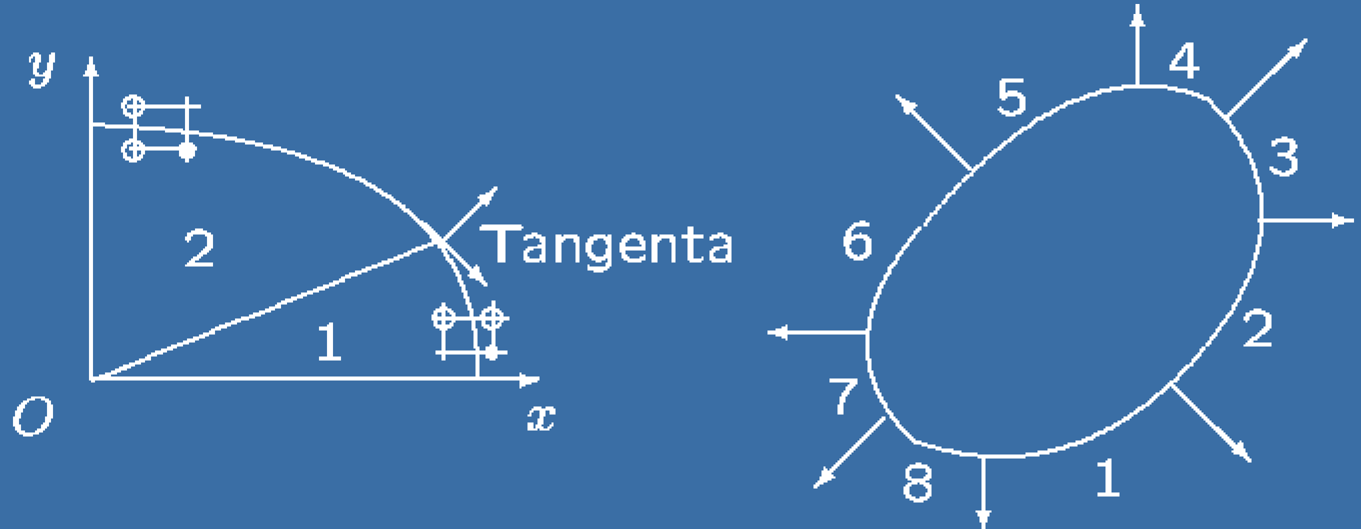
$$= r_y^2[(x_k+1)+1]^2 + r_x^2\left(y_{k+1} - \frac{1}{2}\right)^2 - r_x^2r_y^2$$

$$or = p_k + 2r_y^2(x_k+1) + r_x^2\left[\left(y_{k+1} - \frac{1}{2}\right)^2 - \left(y_k - \frac{1}{2}\right)^2\right]$$

$$increment = \begin{cases} 2r_y^2x_{k+1} + r_y^2, & \text{if } p_k < 0 \\ 2r_y^2x_{k+1} + r_y^2 - 2r_x^2y_{k+1}, & \text{if } p_k \geq 0 \end{cases}$$

Elipse oarecare (1)

$$G(x,y) := ax^2 + bxy + cy^2 + dx + ey + f = 0$$



(a) (b)

Octant	1	2	3	4	5	6	7	8
Δx	1	0,1	0,-1	-1	-1	0,-1	0,1	1
Δy	0,1	1	1	0,1	0,-1	-1	-1	0,-1

Elipse oarecare 2)

Octantul 1: Var.de decizie $d_i = G(x_i + 1, y_i + \frac{1}{2})$

$$d_{i+1} = \begin{cases} G(x_i + 2, y_i + \frac{1}{2}) = d_i + u_{i+1}, & \text{mişcare } \rightarrow \\ G(x_i + 2, y_i + \frac{3}{2}) = d_i + v_{i+1}, & \text{mişcare } \nearrow \end{cases}$$

$$\begin{cases} u_i = a(2x_i + 1) + b(y_i + 1/2) + d + 2a, & \text{mişcare } \rightarrow \\ v_i = (2a + b)x_i + (b + 2c)y_i + a + b/2 + & \\ \qquad +d + e + 2(a + b + c), & \text{mişcare } \nearrow \end{cases}$$

$$u_{i+1} = \begin{cases} u_i + k_1 & \text{mişcare } \rightarrow \\ u_i + k_2 & \text{mişcare } \nearrow \end{cases},$$

$$v_{i+1} = \begin{cases} v_i + k_2, & \text{mişcare } \rightarrow \\ v_i + k_3 & \text{mişcare } \nearrow \end{cases}$$

Elipse oarecare (3)

$k_1 = 2a, \; k_2 = 2a + b, \; k_3 = 2a + 2b + 2c$

Trecere din octantul 1 în octantul 2 la schimbare semn pt. $\left(\frac{\partial G}{\partial x} + \frac{\partial G}{\partial y}\right)(x_i, y_i) = (2ax_i + by_i + d) + (bx_i + 3cy_i + e) = v_i - \frac{k_2}{2}$

Curbe de grad doi (1)

$f(x,y) = a_1x^2 + a_2xy + a_3y^2 + a_4x + a_5y + a_6 = 0,\ \ a_1 > 0,\ \ a_i \in Z$

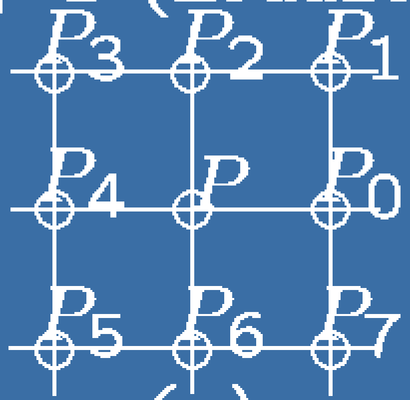
Fie $\delta = a_2^2 - 4a_1a_3$; $\delta \begin{cases} < 0: & \text{elipsă (sau cerc),} \\ = 0: & \text{parabolă,} \\ > 0: & \text{hiperbolă.} \end{cases}$

Metoda I – matematică: rezolvarea ecuaţiei de grad 2 într-o variabilă

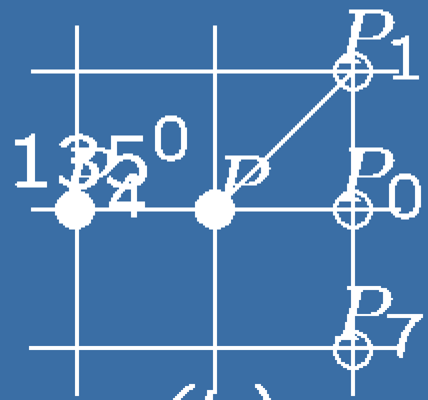
Metoda II – incrementală: operaţii numai în nr.întregi

Curbe de grad doi (2)

Aplic. regula de conexiune discretă: pas k deplasare în direcţia i $\Rightarrow$ pas $k+1$ deplasare în 1 din direcţiile $i-1$, i sau $i+1$ $(\mathtt{modulo}\ 8)$

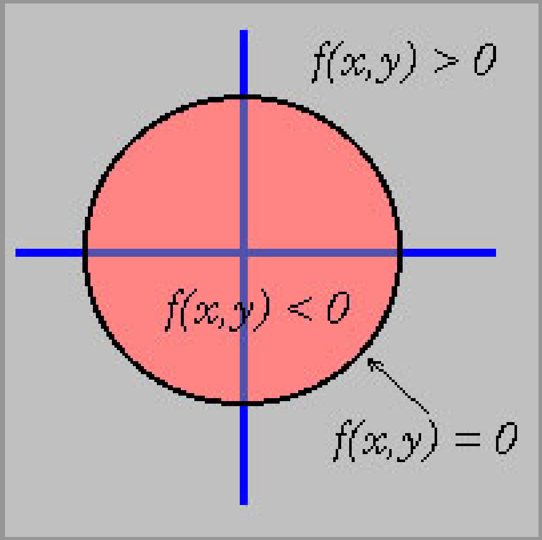


(a)



(b)

Curbe de grad doi



Curbe de grad doi (3)

Pp. sens parcurs a.î $f(x,y) < 0$, i.e. Int, (> 0, i.e. Ext) la stânga (respectiv dreapta) direcției de înaintare $\Rightarrow P_{i+1} \in$Int ($f(P_{i+1}) < 0$), $P_{i-1} \in$Ext ($f(P_{i-1}) > 0$)

? $P_i \in$Int sau $P_i \in$Ext

Criteriu decizional:

$$\begin{cases} f(P_{i-1}) + f(P_i) < 0 : & \text{aleg } P_{i-1} \\ f(P_{i-1}) + f(P_i) \geq 0 \wedge f(P_i) + f(P_{i+1}) \leq 0 : & \text{aleg } P_i \\ f(P_i) + f(P_{i+1}) > 0 : & \text{aleg } P_{i+1} \end{cases}$$

Descrierea curbelor plane

1. *ecuaţii explicite:* $y = f(x)$;

2. *ecuaţii polare:* (r, t), $t \in [a, b]$ şi $r = f(t)$,

Polare $\to$ cartez. $x = r\cos t$, $y = r\sin t$;

3. *ecuaţii parametrice:* $x = f(t)$, $y = g(t)$, $t \in [a, b]$;

4. *ecuaţii implicite:* $F(x, y) = 0$; anumite condiţii $F \overset{\text{T.fcţ.implicite}}{\Longrightarrow}$ ec.explicite.

Trasarea curbelor plane

– caz 1: necesar determinarea scării, în urma analizei valorilor extreme

– caz 2: transformare în coordonate carteziene

– caz 3: trasez curba parametrică $x = x(t), y = y(t),\ t \in [0,1],\ x, y \in C^0 \ldots$

Trasarea curbelor parametrice

Metoda I: aleg (?!) *rată constantă* Δ pt. creştere t şi calculez $\{(x(k\Delta), y(k\Delta)) \mid k = 0, 1, \ldots, [1/\Delta]\}$ a.î:

(a) un punct să nu fie selectat de 2 ori;

(b) să fie generate pct.consecutive adiacente în mediu de afişare

Metoda II (înjumătăţirea intervalului): pp. că fcţ e continuă, probl. la trasarea cu şablon

Înjumătăţirea intervalului

Pas 0: $(x(0), y(0))$, $(x(1), y(1))$

Ciclu:

dacă $(x(t_1), y(t_1)), (x(t_2), y(t_2))$ adiacente,

atunci trasează;

altfel $t_3 = (t_1 + t_2)/2$

$(x(t_1), y(t_1))$, $(x(t_3), y(t_3))$ în stivă,

tratează $(x(t_3), y(t_3))$, $(x(t_2), y(t_2))$

apoi tratează $(x(t_1), y(t_1))$, $(x(t_3), y(t_3))$