3. RASTERIZÁCIA

Jednou z najdôležitejších úloh počítačovej grafiky je zobrazovanie modelov objektov zo spojitého priestoru E^3 do dvojrozmerného diskrétneho priestoru, ktorý predstavuje obrazovka počítača.

Prakticky všetky súčasné grafické displeje používajú metódu rastrového skenovania. Jej podstata spočíva v tom, že počítačový program pripravuje celý obraz v pamäťovom bufri (frame buffer) nazývanom aj bitmapa, pričom operuje s diskrétnymi bodmi. Aj samotný obraz pozostáva z tzv. obrazovkových bodov známych ako pixle (picture elements), ktoré môžu byť zapnuté, alebo vypnuté. Pixel je najmenší adresovateľný element obrazovky, jej najmenšia časť, ktorú môžeme nezávisle ovládať, čiže jej môžeme priradiť napríklad určitú farbu príp. odtieň - hodnotu šedej, ak zariadenie je monochromatické. Také jednoduché útvary, akými sú napr. krivky, špeciálne úsečky, potom zobrazujeme tak, že rozsvietime určitou intenzitou (jasom) reťazec pixlov medzi jej začiatočným a koncovým bodom. Dôležitú úlohu pri tvorbe obrazov dvoj a viac- rozmerných útvarov hrá vypĺňanie oblastí takýmito pixlami. Keďže pixle nie sú ideálne geometrické body, ale isté podmnožiny (najčastejšie štvorčeky) konečnej roviny, netvoria takto skonštruované objekty krivky, úsečky, mnohouholníky v matematickom zmysle slova, ale my ich budeme za krivky, úsečky resp. mnohouholníky považovať, pričom sa budeme snažiť, aby sa im čo najviac podobali.

Ako sme vyššie povedali každému pixlu je v bitmape priradené určité číslo špecifikujúce jeho farbu, odtieň šedej a pod. Z praktických dôvodov je užitočné predstavovať si obrazovku ako pravouholníkovú mriežku, ktorá má r- riadkov očíslovaných zdola nahor nezápornými celými číslami $0,1,...,r-1$ a s- stĺpcov očíslovaných zľava doprava číslami $0,1,...,s-1$. Potom každý pixel bude jednoznačne určený dvojicou tzv. obrazovkových súradníc $(x,y)\in\{0,1,...,r-1\}\times\{0,1,...,s-1\}$.

Na druhej strane bitmapa býva uložená v pamäti počítača ako jednorozmerné pole, ktorého indexy $i\in\{0,1,...,r\times s-1\}$ sú také, že pixlu $(x,y)\rightarrow$ index $i = x + ys$. Napr. pre $r = 5$ a $s = 7$ je všetkých pixlov 35 a pixlu $(5,3)\rightarrow i = 5+3.7 = 26$. Pozri obrázok A, z ktorého vidno predovšetkým vzájomne jednoznačné priradenie medzi množinou pixlov (DC) a frame bufrom (bitmapou).

	(0,*y*); *i*	**(1,*y*); *i***	**(2,*y*); *i***	**(3,*y*); *i***	**(4,*y*); *i***	**(5,*y*); *i***	**(6,*y*); *i***
4	(0,4); 28	(1,4); 29	(2,4); 30	(3,4); 31	(4,4); 32	(5,4); 33	(6,4); 34
3	(0,3); 21	(1,3); 22	(2,3); 23	(3,3); 24	(4,3); 25	(5,3); 26	(6,3); 27
2	(0,2); 14	(1,2); 15	(2,2); 16	(3,2); 17	(4,2); 18	(5,2); 19	(6,2); 20
1	(0,1); 7	(1,1); 8	(2,1); 9	(3,1); 10	(4,1); 11	(5,1); 12	(6,1); 13
0	(0,0); 0	(1,0); 1	(2,0); 2	(3,0); 3	(4,0); 4	(5,0); 5	(6,0); 6
	0	**1**	**2**	**3**	**4**	**5**	**6**

Obr.A

Obrazovka s rozlíšiteľnosťou napr. 512 na 512 pixlov (staré boli aj také) sa teda skladá z 262144 pixlov uložených do 512 riadkov a 512 stĺpcov očíslovaných číslami 0,1,…,511. Tieto vytvárajú množinu usporiadaných dvojíc nezáporných celých čísel $\{0,1,...,511\}\times\{0,1,...,511\}$, ktorá sa nazýva súradnicovým priestorom obrazovky – alebo DC-priestorom (Device Coordinates Space). Ak by aplikačný program pracoval len v malých celočíselných užívateľských súradniciach mohli by sme tento priestor považovať za vyhovujúci pracovný priestor PG. Existujúce grafické systémy však nemôžu aplikačné programy takto obmedzovať a preto je potrebné aby DC-priestor akceptoval aj reálne súradnice. Treba ho teda rozšíriť na priestor reálnych súradníc a to tak, aby pixle zodpovedali práve tým bodom rozšíreného DC, ktoré majú celočíselné súradnice. Toto možno dosiahnuť napr. tak, že ideálnemu bodu obrazovky s reálnymi súradnicami (x, y); $x \geq 0, y \geq 0$ priradíme pixel so súradnicami (round *x*,round *y*) [v niektorých aplikáciách to však môže byť (trunc *x*, trunc *y*)] , kde napríklad: round *x* = *k* $\Leftrightarrow$ ak *k* je to jediné nezáporné celé číslo, pre ktoré platí : $x \in \langle k-0.5, k+0.5)$.

Pripomenieme ešte, že dĺžky na obrazovke sa nemerajú v cm ani v palcoch, ale v <u>pixloch</u>, čo predstavuje dĺžku jednej strany pixla.

Keďže každému ideálnemu bodu $(x, y); x, y \in R$ (presnejšie z istej podmnožiny *R*) vieme jednoznačne priradiť pixel a pixlu miesto vo frame bufri, prislúcha toto miesto aj bodu (*x*,*y*). V ďalšom budeme stále predpokladať, že príkaz: FRAME (I,J):=INTENSITY; spôsobí okamžité vysvietenie pixla (I,J) na obrazovke intenzitou prislúchajúcou zadanej hodnote premennej INTENSITY.

<u>ALGORITMY PRE VYKRESĽOVANIE (úsečiek; vektorov)</u>

1° $y = mx + q$ (1) – rovnica priamky *p* rôznobežnej s osou *y*

$y = mx$ - priamka incidujúca s bodom *O* (začiatok súradnicovej sústavy).

2° Priamka určená dvomi bodmi $A = [x_1, y_1]$, $B = [x_2, y_2]$, $x_1 \neq x_2$, $x_i, y_i \in N \cup \{0\}$, má rovnicu:

$$y = y_1 + \frac{y_2 - y_1}{x_2 - x_1}(x - x_1) \quad \Rightarrow \quad m = \frac{y_2 - y_1}{x_2 - x_1}; \quad q = y_1 - mx_1$$

3° Ak $AB \equiv y = mx + q$ a $A = [x_1, y_1]$, $B = [x_1 + \Delta x, y_1 + \Delta y]$, kde $\Delta x, \Delta y$ sa chápu ako premenné, tak z rovností:

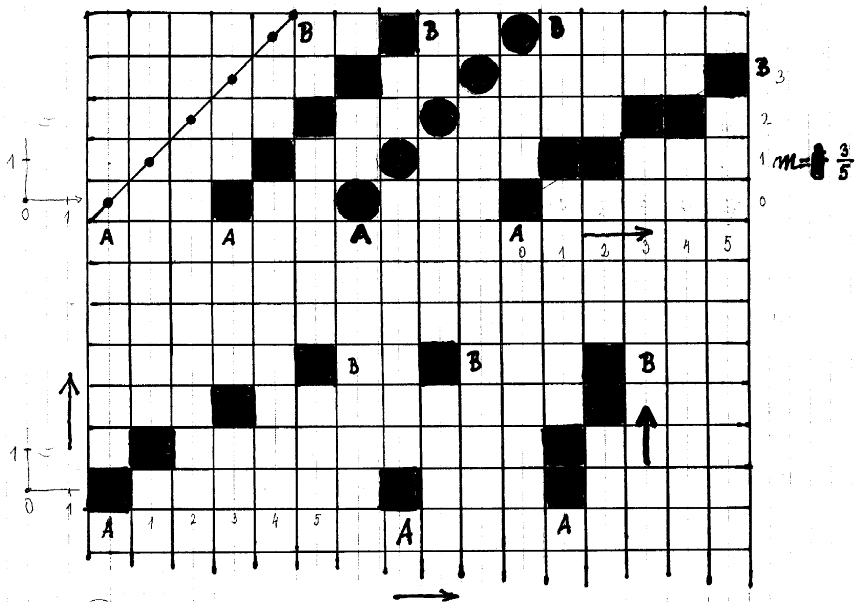
$$y_1 = mx_1 + q \quad \wedge \quad y_1 + \Delta y = m(x_1 + \Delta x) + q \quad \Rightarrow \quad \Delta y = m\Delta x \quad (1´)$$

čo je rovnica priamky idúcej začiatkom a rovnobežnej s priamkou $y = mx + q$ ((1´) sa dala získať aj diferencovaním $y = mx + q$)

4° Kresbu priamky *AB* so smernicou $m = \dfrac{y_2 - y_1}{x_2 - x_1}$ možno získať:

a) posunutím kresby priamky $\Delta y = m\Delta x$ $(y = mx)$ do bodu *A*
b) priamym vykresľovaním priamky *AB* v súradnicovej sústave $\langle A, x', y' \rangle$ kde $x' \parallel x$, $y' \parallel y$, v ktorej má $A = [0,0], B = [x_2 - x_1, y_2 - y_1]$ a $AB:$ $\Delta y = m\Delta x$, kde $\Delta x, \Delta y$ sú súradnice bodov v tejto súradnicovej sústave.

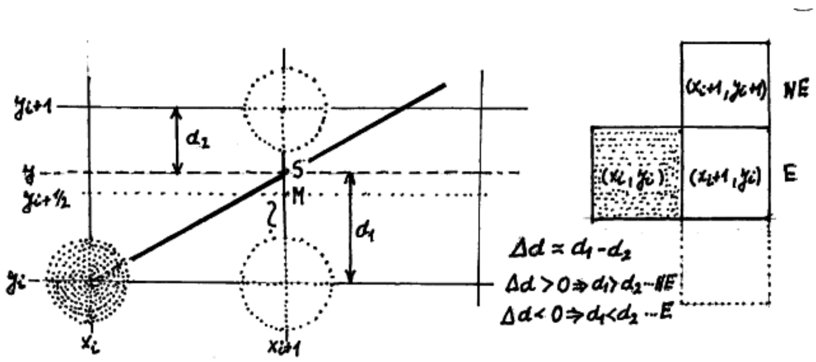
5° Rovnica $\Delta y = m\Delta x$ je východiskom pre určenie <u>odchýlky (výchylky, odklonu)</u> <u>napätia v analógovom zariadení</u>. Zmena v horizontálnej odchýlke napätia sa považuje za úmernú Δx a vo vertikálnej - Δy, ktoré sa vypočíta z rovnice (1´). Tieto výchylky sa potom využívajú na generovanie segmentu priamky (úsečky) so smernicou *m* medzi jej dvomi bodmi.

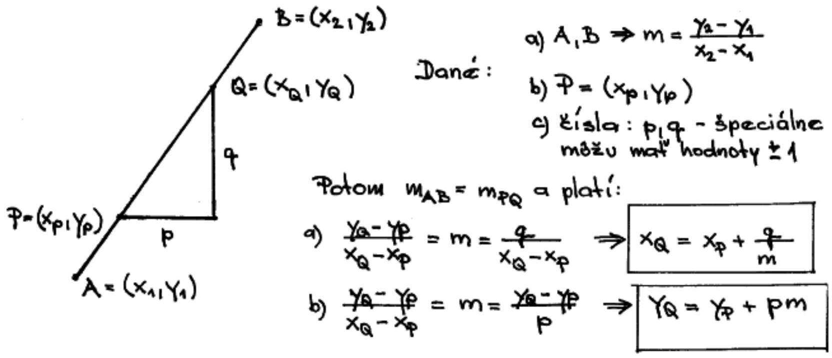


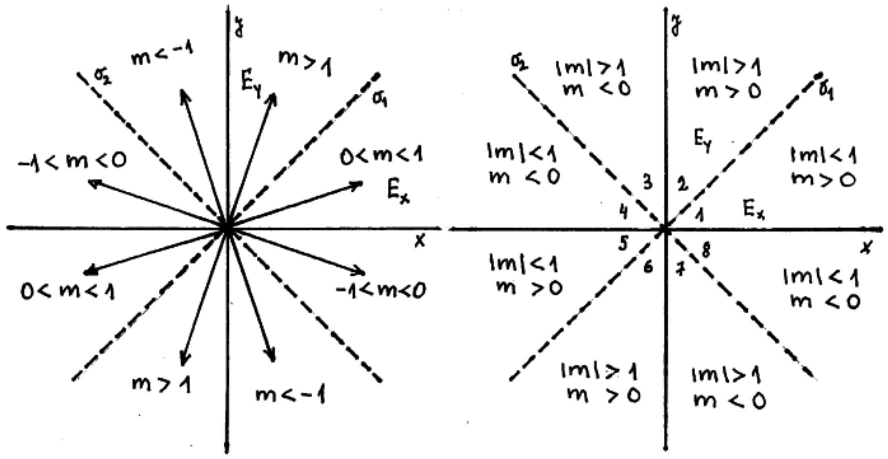
DDA- Algoritmus

Digitálny diferenciálny analyzátor (DDA) je algoritmus pre výpočet polôh pixlov pozdĺž danej priamky na základe rovnice $\Delta y = m\Delta x$, prípadne rovnice $dy = mdx$, ktorú možno chápať ako formálny prepis predchádzajúcej, alebo ako rovnicu získanú diferencovaním rovnice $y = mx + q$ (formálne $\frac{dy}{dx} = m \quad \Leftrightarrow \quad dy = mdx$).

Tento výpočet bude jednoduchý ak pre nárast jednej súradnice si zvolíme jednotkový krok (± 1 pixel) a druhú súradnicu potom z danej rovnice vypočítame. Aby sme sa však pri tomto postupe vyhli výskytu medzier na zobrazených úsečkách (nespojitosti ich pixlových obrazov a medzerám v definičných oboroch funkcií), nesmie zobrazovaná priamka (úsečka) prudko stúpať resp. klesať vzhľadom k osi, na ktorej sme si zvolili jednotkový krok. Ak tento prípad nastane, tak vymeníme úlohu súradnicových osí t.j. jednotkové krokovanie (krok= ± 1pixel) budeme realizovať na zvyšnej súradnicovej osi.







Konkrétnejšie:

1. priamky $y = x$ a $y = -x$ nám rozdelia rovinu na dve časti *Ex*, *Ey*, z ktorých každá je zjednotením dvoch protiľahlých vrcholových uhlov. Predpokladajme, že *Ex* $[Ey]$ je tá, ktorá obsahuje os *x*-ovú [*y*- ovú]. Je zrejmé, že do *Ex* patria tie a len tie priamky, ktorých smernice spĺňajú podmienku $|m| \leq 1$(priamky s miernym stúpaním resp. klesaním vzhľadom k osi *x*) a do *Ey* tie, ktorých smernice spĺňajú podmienku $|m| \geq 1$ (priamky so strmým stúpaním resp. klesaním vzhľadom na os *x*- ovú, čiže s miernym stúpaním – klesaním vzhľadom na os *y*-ovú). Hraničné priamky $y = x$ a $y = -x$ môžeme zaradiť do jednej alebo druhej triedy, dohodnime sa, že ich zaradíme napr. do triedy *Ex*.
2. Predpokladajme, že priamka *AB* (resp. rovnobežka s ňou idúca začiatkom) patrí do *Ex* t.j. $|m| \leq 1$.

• Potom pre prípad, keď bod *A* je vľavo od bodu *B*, čiže keď $x_1 < x_2$, tak prírastky Δx hodnôt *x*-ových súradníc položíme rovné 1 a hodnotu *y*-ovej súradnice nasledujúceho bodu vypočítame z *y*-ovej súradnice predchádzajúceho bodu podľa vzorca:

(5) $y_{i+1} = y_i + m$ (lebo $m = \dfrac{y_{i+1} - y_i}{\Delta x_i} \;\wedge\; \Delta x_i = 1$),

kde *i*=1 pre 1.bod *A* a postupne rastie o 1 až dosiahne koncový bod úsečky.

• Ak bod *B* je vľavo od bodu *A* t.j. $x_2 < x_1$, môžeme body *A*, *B* navzájom vymeniť alebo ekvivalentne postupovať (pre východzí bod vpravo a koncový vľavo) tak, že položíme $\Delta x = -1$ a namiesto (5) potom budeme mať:

(7) $y_{i+1} = y_i - m$ (lebo $\dfrac{y_{i+1} - y_i}{-1} = m$)

3. V prípade, že $|m|$ > 1 (priamka *AB* je „priklonená“ k osi *y*-ovej), bude situácia podobná:

• Pre priamky s kladnou smernicou *m* > 1, zameníme úlohu osí *x* a *y*. To znamená, že po osi *y*-ovej sa budeme pohybovať jednotkovými krokmi $\Delta y = 1$ a hodnotu *x*-ovej súradnice nasledujúceho bodu vypočítame z *x*-ovej predchádzajúceho bodu podľa vzorca:

(6) $x_{i+1} = x_i + \dfrac{1}{m}$, ak bod *A* je nižšie ako bod *B* ($\dfrac{1}{x_{i+1} - x_i} = m$). Ak je však bod *A* vyššie ako bod *B*, položíme $\Delta y = -1$ a vzorec (6) nahradíme vzorcom:

(8) $x_{i+1} = x_i - \dfrac{1}{m}$ ($\dfrac{-1}{x_{i+1} - x_i} = m$).

• Analogicky pre priamky so zápornou smernicou (prirodzene $|m|$ > 1) použijeme:

$\Delta y = 1 \;\wedge$ rovnice (6), ak *A* je nižšie ako *B* ($y_1 < y_2$) a

$\Delta y = -1 \;\wedge$ rovnice (8), ak *A* je vyššie ako *B* ($y_1 > y_2$).

• Tento algoritmus je zhrnutý v nasledujúcej procedúre, ktorej <u>vstupom</u> sú koncové body úsečky (x_1, y_1) a (x_2, y_2) . Rozdiely súradníc vstupných bodov sú označené ako *dx* a *dy*. Väčšia z absolútnych hodnôt rozdielov $dx = x_2 - x_!$, $dy = y_2 - y_1$ určuje hodnotu parametra <u>steps,</u> ktorý špecifikuje počet bodov, ktoré budú vykreslené pozdĺž priamky (úsečky). Štartujúc v polohe (x_1, y_1), pridávame istú „čiastku“ ku každej súradnici aktuálneho bodu, aby sme tak určili polohu - súradnice nasledujúceho bodu. Toto sa opakuje <u>steps-krát.</u> Ak veľkosť *dx* je väčšia ako veľkosť *dy* a $x_1 < x_2$, tak hodnoty prírastkov v smere osi *x* sú 1 resp. *m*. Ak väčšia zmena je v smere osi *x*-ovej ale $x_1 > x_2$, tak pridávané hodnoty pre generovanie nového bodu sú -1 resp. -*m*.

• V opačnom prípade, jednotkový prírastok (resp. úbytok) použijeme v smere osi *y*-ovej a prírastok (resp. úbytok) v smere osi *x*-ovej bude 1/*m*. Za predpokladu, že body, ktoré majú byť zobrazené sa budú zobrazovať jednotnou intenzitou tak, že príkaz <u>set-pixel</u> (špeciálny prípad FRAME) zavolá procedúru pre zaregistrovanie 1 ako hodnoty pixla (=“on“) určeného súradnicami (*x*, *y*) v súradnicovej sústave obrazovky resp. v pamäti obrazovky. Je to teda o určovaní polohy pixlov, ktoré sa majú vysvietiť-zapnúť.

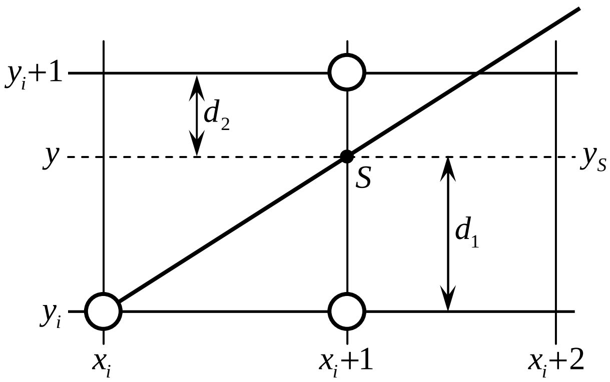
```
procedure dda(x1,y1, x2, y2: integer);
var
dx, dy, steps, k: integer;
x-increment, y-increment, x, y: real;
begin
    dx:=x2-x1;
    dy:=y2-y1;
     if abs (dx)> abs(dy) then steps:=abs(dx)
        else steps:=abs(dy);
     x-increment:=dx/steps;
     y-increment.= dy/steps;
     x:=x1; y:=y1;
     set-pixel(round(x),round(y));
      for k:=1 to steps do begin
           x:=x + x-increment;
           y:=y + y-increment;
           set-pixel (round (x),round(y))
      end { for k}
end { dda }.
```

DDA - algoritmus je silnejšia metóda pre výpočet polôh pixlov ako priame použitie rovnice priamky (1). Eliminuje násobenie v tejto rovnici tým, že využíva výhodu rastrovej charakteristiky voľbou jednotkových krokov buď v smere osi *x*-ovej alebo *y*-ovej pri prechode od jednej polohy pixla k nasledujúcej pozdĺž priamky. Avšak výpočty sú spomalené deleniami potrebnými na určenie hodnôt prírastkov (increments) a využívajúcimi aritmetiku pohyblivej čiarky a zaokruhľovacie operácie.

Rasterizácia úsečky *AB*; $A=[x_1,y_1]$, $B=[x_2,y_2]$, $\Delta x=x_2-x_1$, $\Delta y=y_2-y_1$ (predp. $A \prec B$)

<u>Bresenhamov algoritmus</u>

Tento algoritmus hľadá a vykresľuje body rastra nachádzajúce sa najbližšie ku geometrickému obrazu úsečky v <u>zvislom smere</u> a to výlučne pomocou celočíselnej aritmetiky.



Obr.1

Postup algoritmu si vysvetlíme na obr.1, načrtnutá je časť úsečky *AB* priklonenej k osi *x*-ovej (t.j. $|m| \leq 1$), čiže os *x*-ová je tzv. <u>riadiacou osou</u> , čiže osou, po ktorej sa hodnoty

x-ových súradníc bodov (pixlov) konštantne zväčšujú o hodnotu 1 a kde predpokladáme, že sme v pozícii keď bol vykreslený bod úsečky (pixel) so súradnicami $[x_i, y_i]$.Vzhľadom na jednotkové krokovanie zvolené v smere osi x-ovej, ďalší možní kandidáti na vykreslenie (v stĺpci x_i+1) sú pixle $[x_i+1, y_i]$ a $[x_i+1, y_i+1]$. Podľa uvedeného základného princípu algoritmu treba vykresliť ten z nich, ktorý je bližšie ku geometrickému priesečníku danej úsečky so spojnicou stredov týchto pixlov. Aby sme to zistili, predpokladajme, že úsečka leží na priamke o rovnici $y = mx + b$, $m = {\Delta y}/{\Delta x}$, $d_1 = y - y_i$; $d_2 = y_i + 1 - y$, kde y je y-ová súradnica bodu S. Potom: $y = m(x_i+1) + b$ a preto:

$$d_1 = m(x_i+1) + b - y_i \quad \text{a} \quad d_2 = y_i + 1 - m(x_i+1) - b \quad \Rightarrow$$

(1) $$\Delta d := d_1 - d_2 = 2m(x_i+1) - 2y_i + 2b - 1 .$$

Podľa premennej Δd môžeme jednoducho určiť, ktorý z dvoch možných pixlov je bližšie k úsečke AB. Je totiž zrejmé, že ak $\Delta d < 0$ (t.j. $d_1 < d_2$), tak bližšie k úsečke je „dolný“ pixel $[x_i+1, y_i]$ (vysvieti sa) a ak $\Delta d > 0$ (t.j. $d_1 > d_2$), tak k úsečke je bližšie „horný“ pixel $[x_i+1, y_i+1]$ (vysvieti sa). V prípade, že $\Delta d = 0$, je jedno, ktorý z týchto pixlov sa vysvieti (obyčajne horný). Z uvedeného je zrejmé, že pre určenie toho, ktorý pixel sa má vysvietiť (vykresliť) nie je dôležitá skutočná hodnota premennej Δd, ale iba jej znamienko a preto pri rozhodovaní o výbere pixlov pre vysvietenie ju možno nahradiť ľubovoľným jej kladným násobkom, konkrétne parametrom $p_i := \Delta x.\Delta d = \Delta x(d_1 - d_2)$, pomocou ktorého možno nielen rozhodovať o výbere pixlov na vysvietenie, ale súčasne aj preniesť všetky nasledujúce výpočty do celočíselnej aritmetiky, pretože vo vzťahu (1) je jedinou <u>faktickou</u> neceločíselnou veličinou smernica $m = {\Delta y}/{\Delta x}$, kde $\Delta x, \Delta y$ sú celé čísla a $\Delta x > 0$, lebo $A \prec B$. Pre zjednodušenie výpočtov je vhodné vyjadriť si parameter p_i rekurentne:

a) po vynásobení (1) číslom Δx dostaneme:

$$p_i = 2\Delta y(x_i+1) - 2\Delta x y_i + \Delta x(2b-1) \text{ resp.}$$

(2) $$p_i = 2\Delta y x_i - 2\Delta x y_i + C, \text{ kde } C = 2\Delta y + \Delta x(2b-1) \text{ je konst.}$$

Potom však $p_{i+1} = 2\Delta y x_{i+1} - 2\Delta x y_{i+1} + C = 2\Delta y x_i - 2\Delta x y_{i+1} + 2\Delta y + C$,

a preto $p_{i+1} - p_i = -2\Delta x(y_{i+1} - y_i) + 2\Delta y \;\Rightarrow$

(3) $$p_{i+1} = p_i - 2\Delta x(y_{i+1} - y_i) + 2\Delta y .$$

Aby sme mohli (3) využiť, potrebujeme ešte vypočítať hodnotu p_1:

$$p_1 = 2\Delta y x_1 - 2\Delta x(mx_1 + b) + 2\Delta y + 2\Delta x b - \Delta x = 2\Delta y x_1 - 2\Delta y x_1 - 2\Delta x b + 2\Delta y + 2\Delta x b - \Delta x$$

$\Rightarrow$ (4) $$p_1 = 2\Delta y - \Delta x .$$

Teraz už môžeme <u>iteračným</u> spôsobom počítať hodnoty každého nasledujúceho parametra p z jeho predchádzajúcej hodnoty. K tomu je možné použiť vzťah (3), alebo jeho zjednodušenie vyplývajúce zo znamienka p_i:

(3’)

a)Ak $p_i < 0$, tak vykreslíme $[x_i+1, y_i]$ t.j. $y_{i+1} = y_i \Rightarrow p_{i+1} = p_i + 2\Delta y$

b) Ak $p_i \geq 0$, tak vykreslíme $[x_i+1, y_i+1]$ t.j. $y_{i+1} = y_i + 1 \Rightarrow p_{i+1} = p_i + 2(\Delta y - \Delta x)$.

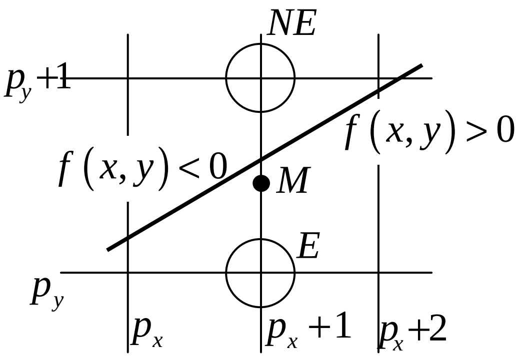
Schému algoritmu možno zapísať takto:

1. Z celočíselných koncových bodov $[x_1, y_1]$, $[x_2, y_2]$ urči konštanty:

   $c_1 = 2\Delta y,\ \ c_2 = 2(\Delta y - \Delta x)$
2. Inicializuj hodnotu parametra $p = 2\Delta y - \Delta x$
3. Inicializuj $[x, y] := [x_1, y_1]$
4. Vykresli bod $[x, y]$
5. Pokiaľ je $x \le x_2$ opakuj:
   a) zvýš hodnotu x o 1 (polož $x = x+1$)
   b) ak je parameter $p \ge 0$: polož $p = p + c_2 \wedge y = y+1$
   c) ak je parameter $p < 0$: polož $p = p + c_1$
   d) vykresli bod $[x, y]$.

Midpoint line algoritmus

• rozumná predstava rastra: celočíselná sieť, ktorej každý uzol je stredom kruhu o polomere ½, predstavujúceho pixel
• rasterizácia vektora: rasterizácia orientovanej úsečky
• cieľ: zobraziť (= vysvietiť na obrazovke) množinu pixlov, ktorých geometrické stredy ležia na úsečke alebo blízko nej (pozdĺž úsečky)
• uvažujme úsečku *AB*, $A=[x_A,y_A]$, $B=[x_B,y_B]$, kde $x_A < x_B$ (*A* je vľavo od *B*) a pre smernicu *m* úsečky *AB* platí $0 \le m \le 1$
(zrejme $m=dy/dx$, kde $dx = x_B - x_A$ a $dy = y_B - y_A$ $q = (dx.y_B - dy.x_B)$)
• rovnicu priamky *AB* potom možno zapísať v tvare: $2ax + 2by + 2c = 0$ kde
$a = dy,\ \ b = -dx,\ \ c = dx.y_B - dy.x_B$
• keďže priamka *AB* pretína os *y* v bode $Q = [0, (dx.y_B - dy.x_B)/dx]$, je zrejmé, že pre funkciu $f(x, y) = 2ax + 2by + 2c = 2dy.x - dx.y + 2(dx.y_B - dy.x_B)$ platí $f(0,0) = 2(dx.y_B - dy.x_B) > 0$ $\Leftrightarrow$ bod *Q* je nad bodom *O* $\Leftrightarrow$ bod *O* je pod priamkou *AB*. Pretože priamka *AB* určuje rozklad roviny E^2 na hornú a dolnú časť, môžeme povedať, že bod $X = (x, y)$ leží pod priamkou $AB \ \Leftrightarrow\ f(x, y) > 0$ a leží na priamke *AB* alebo nad ňou $\Leftrightarrow\ f(x, y) \le 0$.
• skôr než prejdeme k vykresľovaniu úsečky *AB*, pripomeňme si, že vyššie uvedené obmedzenia sú ľahko odstrániteľné, lebo ak by $|m| > 1$, tak zámena $x \leftrightarrow y$ vedie k úsečke s prevrátenou hodnotou smernice; ak $m < 0$, algoritmus sa dá veľmi ľahko modifikovať zámenou prírastok ↔ úbytok, resp. stúpanie ↔ klesanie; konečne výmenou bodov $A \leftrightarrow B$ možno zabezpečiť možnosť kreslenia zľava doprava vždy.
• predpokladajme, že sme kreslenie úsečky *AB* začali vykreslením bodu $A=[x_A,y_A]$ a dosiahli sme pixel (p_x, p_y).



Obr.B

Treba sa nám rozhodnúť, ktorý z pixlov $E = (p_x +1, p_y)$ a $NE= (p_x +1, p_y+1)$ vykreslíme ako nasledujúci (lebo: $0 < m < 1$). Pomôže nám pritom veličina $D = f(M)$, kde

$M = \text{stred}(E, NE) = (p_x + 1, p_y + 1/2)$. Teda

$D = f(p_x + 1, p_y + 1/2) = 2a(p_x + 1) + 2b(p_y + 1/2) + 2c = 2ap_x + 2bp_y + (2a + b + 2c) \in N$ (*)

• je zrejmé, že ak $D \geq 0$, tak bod M je pod priamkou AB alebo na nej a z uvažovaných dvoch pixlov je k nej bližšie pixel NE, ktorý sa vysvieti a ak $D < 0$, tak bod M je nad priamkou, v dôsledku čoho je k AB bližšie pixel E a ten sa vysvieti.

• je dobré, že D je celočíselné, ale nie je až také dobré, že jeho výpočet si vyžaduje dve násobenia a dve sčítania, ak nemeniaca zložka v zátvorke je prepočítaná.

• jedným z veľmi šikovných trikov tohto algoritmu však je, že D sa počíta prírastkovo. Predpokladajme teda, že poznáme aktuálnu hodnotu D a chceme vypočítať jeho nasledujúcu hodnotu.

• ak sme ako nový vysvietený pixel vybrali pixel $E = (p_x +1, p_y)$, tak v nasledujúcom kroku k nemu prislúcha nový stred $M = ((p_x + 1) + 1, p_y + \frac{1}{2})$ a nové

$D_{new} = 2a(p_x + 1) + 2bp_y + (2a + b + 2c) = D + 2a$ t.j. $D_{new} = D + 2dy$

• ak sme však ako nový vysvietený pixel vybrali bod $NE= (p_x +1, p_y+1)$, tak k nemu prislúcha nový stred

$M = ((p_x + 1) + 1, (p_y + 1) + \frac{1}{2})$ a k nemu

$D_{new} = 2a(p_x + 1) + 2b(p_y + 1) + (2a + b + 2c) = D + 2a + 2b = D + 2(dy - dx)$

• Teda $D_{new} = D + 2dy$, ak $D < 0$ (E)

$D_{new} = D + 2(dy - dx)$, ak $D \geq 0$ (NE).

Z tohto výsledku vyplýva, že parameter D je ekvivalentný parametru p z predchádzajúceho algoritmu a možno ho teda považovať za rozhodovací parameter Bresenham-line algoritmu. Autori ho často uprednostňujú pred p, lebo princíp jeho konštrukcie možno použiť aj v ďalších algoritmoch napr. v algoritme Bresenham-circle.

• z ekvivalencie p a D vyplýva, že algoritmus možno zúplniť (dokončiť) tak ako Bresenham-line algoritmus.

Cieľom našich ďalších snáh sú ešte algoritmy na efektívne vykreslenie kružnice: minimalizácia výpočtov a pokiaľ je to možné s využitím len celočíselnej aritmetiky.

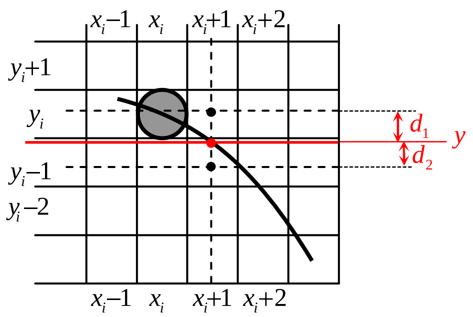
<u>Bresenhamov algoritmus</u> (pre generovanie kružnice)

Na vykresľovanie kružnice nám stačí vygenerovať jej časť, ktorá sa nachádza v jednom oktante. Zvolíme si 2. oktant. Ostatné časti kružnice dostaneme súmernosťou podľa súradnicových osí a priamok, ktoré rozpoľujú nimi určené kvadranty ($y = x$ a $y = -x$).

Výber pixlov sa uskutočňuje na podobnom princípe ako u úsečky, z dvoch možných kandidátov na vysvietenie vyberá ten, ktorý (jeho stred) je bližšie k danej kružnici. Tento výber sa uskutočňuje na základe istého parametra $p_i = d_1 - d_2$, ktorý sa opäť vypočítava rekurentne, ale hodnoty d_1, d_2 sú <u>štvorcami</u> rozdielov y-ových súradníc.

<u>Konkrétny postup</u>:

1. pre zjednodušenie volíme kružnicu so stredom v začiatku súradnicovej sústavy a zobrazíme jej segment, ktorý je v 2.oktante
2. štartovať budeme v bode $(0, r) \in y$, jednotkové kroky budeme voliť v smere osi x a skončíme v bode, pre ktorý $x = y$.
3. obrázok C reprezentuje situáciu v jednom kroku algoritmu:



Obr.C

- (x_i, y_i) je dosiahnutá pozícia pre vysvietenie v i-tom kroku algoritmu
- v nasledujúcom kroku sa treba rozhodnúť pre pozíciu $(x_i + 1, y_i)$ alebo $(x_i + 1, y_i - 1)$
- obom možnostiam voľby zodpovedá na kružnici aktuálna poloha $(x_i + 1, y)$, kde y je určené rovnosťou $y^2 = r^2 - (x_i + 1)^2$
- zvislú vzdialenosť medzi $(x_i + 1, y_i)$ a $(x_i + 1, y)$ budeme reprezentovať číslom
  $$d_1 = y_i^2 - y^2 = y_i^2 - r^2 + (x_i + 1)^2$$
  a medzi polohami $(x_i + 1, y)$ a $(x_i + 1, y_i - 1)$ číslom
  $$d_2 = y^2 - (y_i - 1)^2 = r^2 - (x_i + 1)^2 - (y_i - 1)^2$$
- parameter p_i pre určenie „správnej“ polohy pre vysvietenie zvolíme ako rozdiel d_1- d_2 t.j.
  (*) $p_i := d_1 - d_2 = 2(x_i + 1)^2 + y_i^2 + (y_i - 1)^2 - 2r^2$,
  špeciálne, ak položíme $(x_1, y_1) = (0, r)$ dostaneme: $p_1 = 3 - 2r$
- ak je $p_i < 0$, tak je $d_1 < d_2 \Rightarrow$ $(x_i + 1, y_i)$ je bližšie k $(x_i + 1, y)$ a preto vyberieme na vysvietenie pixel $(x_i + 1, y_i)$
- ak je však $p_i \geq 0$ je $d_2 < d_1 \Rightarrow$ $(x_i + 1, y_i - 1)$ je bližšie k $(x_i + 1, y)$ a preto vysvietime pixel $(x_i + 1, y_i - 1)$
- aby sme zjednodušili výpočet parametra p_i odvodíme rekurentný vzorec:

$$p_{i+1} = 2\left[(x_i+1)+1\right]^2 + {y_{i+1}}^2 + (y_{i+1}-1)^2 - 2r^2 =$$
$$= 2(x_i+1)^2 + {y_i}^2 + (y_i-1)^2 - 2r^2 - {y_i}^2 - (y_i) + 4(x_i+1) + 2 + 2{y_{i+1}}^2 - 2y_{i+1} + 1$$
$$\Rightarrow \qquad p_{i+1} = p_i + 4x_i + 6 + 2({y_{i+1}}^2 - {y_i}^2) - 2(y_{i+1} - y_i)$$

- tento vzťah ešte možno upraviť v závislosti od znamienka p_i:
  a) ak $p_i < 0$, tak $y_{i+1} = y_i$ a preto $p_{i+1} = p_i + 4x_i + 6$
  b) ak $p_i \geq 0$, tak $y_{i+1} = y_i - 1$ a preto $p_{i+1} = p_i + 4(x_i - y_i) + 10$

Tento opis možno zhrnúť do nasledujúcich štyroch krokov:

1. voľba prvého pixla pre vykreslenie: $(x_1, y_1) = (0, r)$
2. výpočet prvého parametra: $p_1 = 3 - 2r$. Ak $p_1 < 0$ vyber ako nasledujúcu pozíciu (x_1+1, y_1) . V opačnom prípade ($p_1 \geq 0$) vyber (x_1+1, y_1-1) .
3. Pokračuj zvýšením *x*-ovej súradnice o 1-kový krok a vypočítaj nasledujúci parameter z predchádzajúceho. Ak pre predchádzajúci parameter platilo $p_i < 0$, tak polož $p_{i+1} = p_i + 4x_i + 6$. V opačnom prípade ($p_i \geq 0$) polož $p_{i+1} = p_i + 4(x_i - y_i) + 10$. Ak $p_{i+1} < 0$, tak ako nasledujúci vyber bod $(x_i + 2, y_{i+1})$. V opačnom prípade ($p_{i+1} \geq 0$) ako nasledujúci vykresli bod $(x_i + 2, y_{i+1} - 1)$. Pritom pre y_{i+1} platí: $y_{i+1} = y_i$, ak $p_i < 0$ a $y_{i+1} = y_i - 1$, ak $p_i \geq 0$.
4. Opakuj procedúry kroku 3 pokiaľ sa *x*, *y* nebudú rovnať.

Hoci pri výpočte parametra sa vyžaduje násobenie, príslušný násobok je mocninou 2 a preto ho možno implementovať cez logické operácie. Ostatné operácie sú celočíselné sčítanie a odčítanie.

Nasledujúca procedúra je zápisom opísaného kružnicového algoritmu kde vstupom sú súradnice stredu kružnice a jej polomer. Procedúra plní pole obrazovkovej pamäte bodmi pozdĺž obvodu kružnice volaním operácie set-pixel.

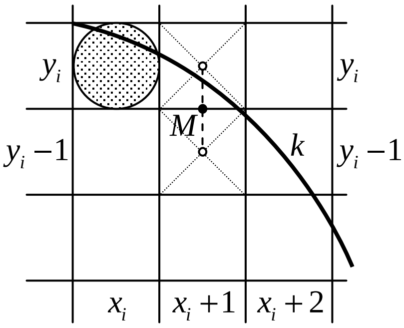
```
procedure bres_cirle(x_center,y_center,radius: integer);
var
p, x, y: integer;
procedure plot_circle_points
begin
    set_pixel(x_center +x,y_center+y);
    set_pixel(x_center -x,y_center+y);
    set_pixel(x_center +x,y_center-y);
    set_pixel(x_center -x,y_center-y);
    set_pixel(x_center +y,y_center+x);
    set_pixel(x_center -y,y_center+x);
    set_pixel(x_center +y,y-_enter-x);
    set_pixel(x_center -y,y_center-x);
end; {plot_circle_points}
begin {bres_circle}
   x:=0;
   y:=radius;
   p:=3 – 2*radius;
   while x < y do begin
     plot_circle_points;
```

```
        if p < 0 then p:= p+4*x+6
        else begin
          p:= p+4*(x-y)+10;
          y:= y – 1
          end; {if p not < 0}
       x:= x  + 1
     end; {while x < y}
     if x=y then plot_circle_points
end; { bres_circle}
```

Midpoint circle algoritmus

Ide o úpravu Bresenhamovho algoritmu v otázke výberu kandidáta na vysvietenie jedného z dvojice pixlov (x_i+1, y_i) a (x_i+1, y_i-1). Vyjdeme z funkcie $f(x,y) = x^2 + y^2 - r^2$. Platí:

$$f(x,y) < 0 \Leftrightarrow (x,y) \in \text{ vnútra } k$$
$$f(x,y) = 0 \Leftrightarrow (x,y) \in k \qquad .$$
$$f(x,y) > 0 \Leftrightarrow (x,y) \in \text{ vonk. } k$$



Obr.D

Je zrejmé, že ak *M*= stred$\left[(x_i+1, y_i), (x_i+1, y_i-1)\right]$, tak $M = [x_i+1, y_i - 1/2]$.

Definujme:

$$p_i := f(x_i+1, y_i - 1/2) = (x_i+1)^2 + (y_i - 1/2)^2 - r^2 \qquad (1)$$

Je zrejmé, že ak

a) $p_i = f(M) < 0$, bližším ku *k* je pixel $\left[x_i+1, y_i\right]$

b) $p_i = f(M) > 0$, bližším ku *k* je pixel $\left[x_i+1, y_i-1\right]$.

Teda p_i bude rozhodovacím parametrom pre vysvietenie jedného z týchto dvoch pixlov. Odvodíme si rekurentný vzorec:

$$p_{i+1} = f((x_i+1)+1), y_{i+1} - 1/2) = \left[(x_i+1)+1\right]^2 + \left[y_{i+1} - 1/2\right]^2 - r^2 \;.....\Rightarrow$$
$$p_{i+1} = p_i + 2(x_i+1) + (y_{i+1}{}^2 - y_i{}^2) - (y_{i+1} - y_i) + 1 \qquad (2).$$

Aj tento vzťah možno zjednodušiť v závislosti od znamienka p_i:

a) Ak $p_i < 0$ vyberáme pixel $\left[x_i+1, y_i\right]$ t.j. $y_{i+1} = y_i$. Ak toto dosadíme do (1), tak dostaneme:

$p_{i+1} = p_i + 2(x_i+1) + 1 = p_i + 2x_{i+1} + 1$, kde $2x_{i+1} = 2x_i + 2$ resp. $p_{i+1} = p_i + 2x_i + 3$

b) Ak $p_i \geq 0$, vyberáme pixel $\left[x_i+1, y_i-1\right]$ takže $y_{i+1} = y_i - 1$ a preto

$$p_{i+1} = p_i + 2(x_i+1) + 1 + (y_i-1)^2 - y_i{}^2 - (y_i - 1 - y_i) =$$

$= p_i + 2(x_i + 1) + 1 - 2y_i + 1 + 1 = p_i + 2(x_i - y_i) + 5 \Rightarrow$

$p_{i+1} = p_i + 2x_{i+1} - 2y_{i+1} + 1 = p_i + 2(x_{i+1} - y_{i+1}) + 1$.

Aby sme tieto vzťahy mohli využiť potrebujeme ešte poznať začiatočný rozhodovací parameter p_0. Dostaneme ho pre bod $(x_0, y_0) = (0, r)$ z (1): $p_0 = 5/4 - r$.

Ak bol však polomer r inicializovaný ako celočíselný môžeme p_0 zaokrúhliť takto: $p_0 = 1 - r$ (pre $r \in N$), lebo všetky prírastky sú celočíselné. Na rozdiel od lineárnych algoritmov sme v prípade kružnicových nezískali ekvivalentné rozhodovacie parametre.

Podobne ako Bresenhamov aj tento algoritmus však vypočítava pixle pozdĺž obvodu kruhu používajúc pritom len celočíselné sčítania a odčítania za predpokladu, že kružnicové parametre sú špecifikované v celočíselných obrazovkových súradniciach. Jednotlivé kroky algoritmu možno zhrnúť takto:

<u>Midpoint Circle Algorithm</u>

1. Zadaj polomer r a stred kružnice (x_S, y_S) a urči prvý bod na obvode kružnice so stredom v začiatku: $(x_0, y_0) = (0, r)$
2. Vypočítaj začiatočnú hodnotu rozhodovacieho parametra ako:
   $$p_0 = 5/4 - r$$
3. V každej pozícii x_k, štartujúc v $k = 0$, vykonaj nasledujúci test:
   Ak $p_k < 0$, tak nasledujúci bod pozdĺž obvodu kružnice so stredom (0,0) bude $(x_k + 1, y_k)$ a $p_{k+1} = p_k + 2x_{k+1} + 1$.
   V opačnom prípade nasledujúci bod pozdĺž obvodu tejto kružnice bude $(x_k + 1, y_k - 1)$ a $p_{k+1} = p_k + 2x_{k+1} + 1 - 2y_{k+1}$, kde $2x_{k+1} = 2x_k + 2$ a $2y_{k+1} = 2y_k - 2$
4. Urči symetrické body v ostatných siedmich oktantoch
5. Posuň každú vypočítanú pixlovú pozíciu (x, y) na kružnicu so stredom (x_S, y_S) a zobraz bod so súradnicami:
   $$x = x + x_S, \quad y = y + y_S$$
6. Opakuj kroky 3. až 5. pokiaľ x ≥ y.