Obsah

Zoznam obrázkov

1 Rasterizácia

Podľa typu zobrazovacieho zariadenia sú výsledkami algoritmov buď postupnosti bodov – pixely alebo postupnosti úsečiek. V prvom prípade dostaneme rastrový obraz – raster a v tom druhom obraz vektorový [2].

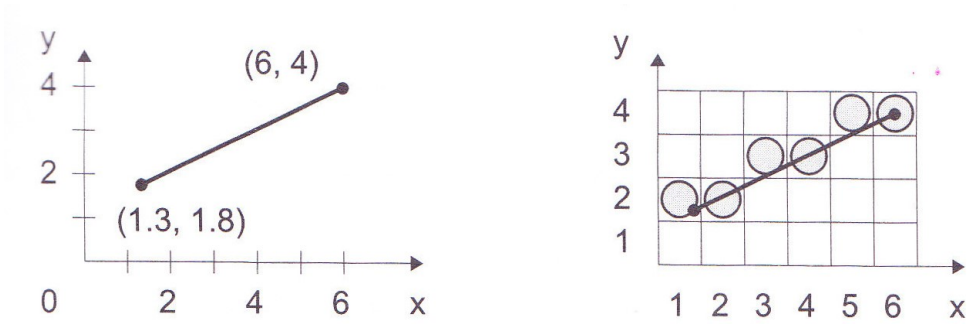
Rastrový obraz alebo raster si môžeme predstaviť ako celočíselnú sieť, ktorej každý uzol je stred kruhu o polomere $\frac{1}{2}$, predstavujúceho pixel. Cieľom je zobraziť (vysvietiť na obrazovke) množinu pixlov, ktorých geometrické stredy ležia na úsečke alebo blízko nej (pozdĺž danej úsečky). Tento proces sa nazýva **rasterizácia** a môžeme si ho predstaviť ako rozsvecovanie jednotlivých bodov rastra. Pod rasterizáciou vektora budeme rozumieť rasterizáciu orientovanej úsečky.

Za základné dvojrozmerné objekty považujeme úsečky, lomené čiary, kružnice, elipsy, mnohouholníky, krivky a textové reťazce [1]. Tieto objekty nazývame **základné grafické výstupné prvky (output primitives)** [2]. Počítačová grafika je orientovaná hlavne na tvorbu rastrového obrazu, a teda pri tvorbe rastrového obrazu je potrebné nájsť všetky pixely, ktoré reprezentujú daný grafický prvok [2].

V nasledujúcej kapitole ukážeme, ako je možné zobraziť pomocou rasterizácie úsečku, kružnicu a elipsu využitím troch základných algoritmov.

1.1 Algoritmy na rasterizáciu úsečky

Úsečka je vo všeobecnosti vyjadrená neceločíselnými súradnicami koncových bodov. Algoritmy na vykresľovanie do rastra ale počítajú s celočíselnými súradnicami. Preto sa na vstupe algoritmu koncové body zaokrúhľujú do celočíselnej aritmetiky. Chyba, ku ktorej v dôsledku zaokrúhlenia dôjde, je považovaná za bezvýznamnú (Obr. 1).



Obr. 1: Spojité a rastrové zobrazenie úsečky [2]

Úsečka je segment priamky a uvedieme tri spôsoby opisu:

- **Smernicová rovnica priamky:**

$$y = mx + b, \tag{1.1}$$

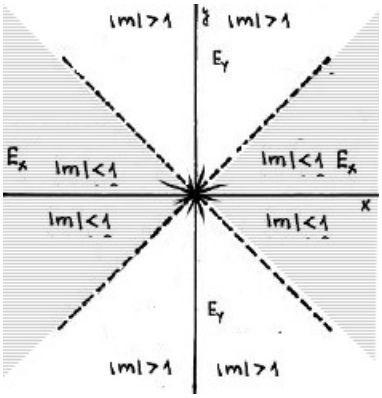
  kde m je smernica priamky a b je y-ová súradnica priesečníka priamky s osou y. Smernica priamky vyjadruje tangens uhla, ktorý zviera priamka s kladnou časťou osi x.
  Ak priamka prechádza bodom O (začiatok súradnicovej sústavy), tak $b = 0$, t.j.:

$$y = mx \tag{1.2}$$

- **Priamka určená dvoma bodmi:**
  Máme dva krajné body úsečky AB, $A = [x_A, y_A]$, $B = [x_B, y_B]$, $x_A \neq x_B$, $x_i, y_i \in \mathbb{N} \cup \{0\}, x_i \in \langle x_A, x_B \rangle, y_i \in \langle y_A, y_B \rangle$. Priamku určenú týmito dvoma bodmi dostaneme zo smernicového tvaru rovnice:
  $$y - y_A = \frac{y_B - y_A}{x_B - x_A}(x - x_A) \tag{1.3}$$
  Po úprave $y = mx + b$, kde $m = \frac{y_B - y_A}{x_B - x_A}$ a $b = y_B - mx_A$. Hodnoty $x_B - x_A$ a $y_B - y_A$ vyjadrujú prírastok v smere osi x a y a môžeme ich označiť ako $dx = x_B - x_A$ a $dy = y_B - y_A$. V literatúre sa dx označuje aj ako Δx a dy ako Δy.

- **Všeobecná rovnica priamky:**
  Rovnicu (1.1) vieme upraviť na tvar
  $$ax + by + c = 0, \tag{1.4}$$
  kde $a, b, c \in \mathbb{R}$. Toto vyjadrenie nazývame všeobecná rovnica priamky.

Ďalej nás budú zaujímať priamky $y = x$ a $y = -x$. Tie rozdelia rovinu na dve oblasti E_x a E_y, z ktorých každá je zjednotením dvoch protiľahlých vrcholových uhlov (Obr. 2).



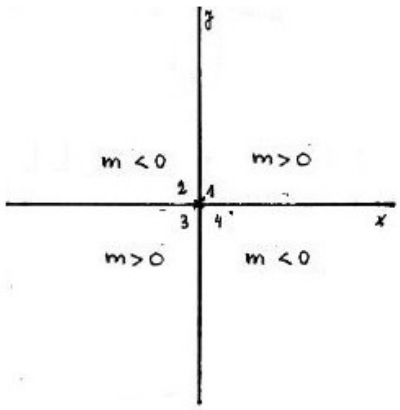
Obr. 2: Rozdelenie roviny na E_x a E_y [6]

- Oblasť E_x obsahuje os x-ovú. Je zrejmé, že do oblasti E_x patria tie a len tie priamky, ktorých smernice spĺňajú podmienku $|m| \leq 1$ a teda $\mid dy \mid \leq \mid dx \mid$. Sú to priamky s miernym stúpaním resp. klesaním vzhľadom k osi x. Ak úsečka patrí do oblasti E_x hovoríme, že má dominantný smer x.

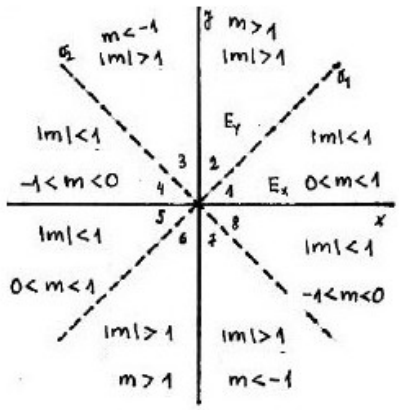
- Oblasť E_y obsahuje os y. Do oblasti E_y patria tie a len tie priamky, ktorých smernice spĺňajú podmienku $|m| \geq 1$ a zároveň $\mid dy \mid \geq \mid dx \mid$. To sú priamky so strmým stúpaním resp. klesaním vzhľadom na os x, čiže s miernym stúpaním resp. klesaním vzhľadom na os y. Ak úsečka patrí do oblasti E_y má dominantný smer y.

Hraničné priamky $y = x$ a $y = -x$ môžeme zaradiť do ktorejkoľvek z týchto oblastí, dohodnime sa, že ich zaradíme napr. do oblasti E_x.

Súradnicové osi x a y rozdelia rovinu na štyri kvadranty (Obr. 3a). Ak k nim pridáme ešte priamky $y = x$ a $y = -x$, dostaneme rozdelenie roviny na osem oktantov (Obr. 3b). Úsečke AB vieme podľa hodnoty smernice m priradiť príslušnosť do kvadrantu (Obr. 3a) alebo oktantu (Obr. 3b). Medzi základné algoritmy rasterizácie úsečky patria tri algoritmy: DDA, Bresenhamov a Midpoint algoritmus.
Pri všetkých algoritmoch platí, že vstupom je začiatočný bod $A = [x_A, y_A]$ a koncový bod úsečky $B = [x_B, y_B]$ a výstupom množina bodov rastra, ktoré aproximujú danú úsečku AB.



(a) Rozdelenie roviny do 4 kvadrantov



(b) Rozdelenie roviny do 8 oktantov

Obr. 3: Rozdelenie roviny

1.1.1 DDA algoritmus (Digital Differential Analyzer Algorithm)

História

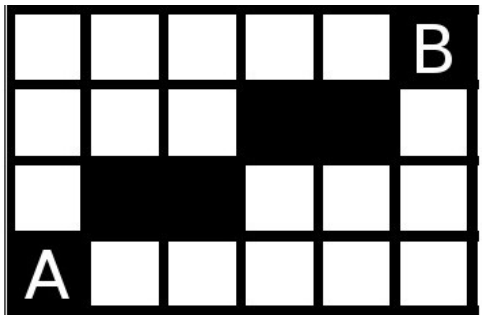
Je to jeden z prvých algoritmov používaných v počítačovej grafike.

Princíp algoritmu

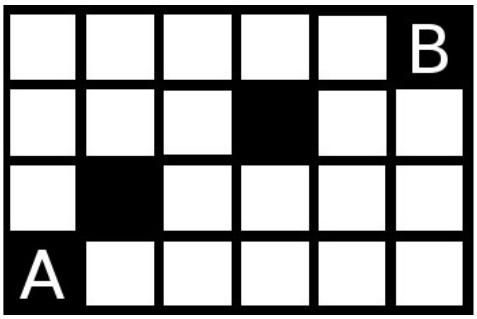
DDA algoritmus je rýchlejšia metóda pre výpočet polôh pixlov ako použiť smernicovú rovnicu priamky (1.1).

V tomto iteratívnom krokovom algoritme musíme pracovať v reálnej aritmetike a robiť zaokrúhlenia. Nepoužívame pri ňom násobenie, ale výpočet súradníc pixlov pozdĺž priamky v reálnej aritmetike. Chyba zaokrúhlenia sa napriek tomu považuje za nepodstatnú.

DDA (Digital Differential Analyzer) je prírastkový algoritmus. Vychádza z rovnice $dy = mdx$, ktorú dostaneme ako formálny prepis smernicovej rovnice priamky. Úsečka je krokovaná jednotkovým krokom v smere jednej z osí x alebo y a druhá súradnica sa vyjadrí zo smernice priamky $m = \frac{dy}{dx}$. Os, v ktorej smere prebieha vzorkovanie, sa nazýva riadiaca/hlavná os a druhá je vedlajšia os. Aby sme sa pri tomto postupe vyhli výskytu medzier na zobrazených úsečkách, nemôže zobrazovaná priamka prudko stúpať alebo klesať od riadiacej osi (Obr. 4a a Obr. 4b). Ak tento prípad nastane, vymeníme úlohu súradnicových osí, t.j. jednotkové krokovanie budeme realizovať v smere druhej súradnicovej osi.



(a) Vzorkovanie v smere osi x



(b) Vzorkovanie v smere osi y

Obr. 4: Úsečka z bodu A do bodu B vykreslená pomocou dvoch vzorkovaní

Nachádzame sa v bode úsečky (x_i, y_i) a potrebujeme určiť nasledujúci bod (x_{i+1}, y_{i+1}), ktorý vieme vyjadriť ako

$$x_{i+1} = x_i + incr_x,$$
$$y_{i+1} = y_i + incr_y,$$

kde $incr_x$ a $incr_y$ je prírastok (z ang. increment) v smere x a y.

Uvažujme úsečku AB. Prípady, ktoré môžu nastať:

- Ak $|m| < 1 \Rightarrow| \ dy \ |<| \ dx \ |$ a $x_A < x_B$ (bod A sa nachádza vľavo od bodu B), tak úsečka má dominantný smer x. Vzorkovanie bude teda prebiehať v smere osi x o konštantnú hodnotu +1. Hodnotu nasledujúcej y-ovej súradnice určíme z rovnice priamky (1.1) ako $y_{i+1} = mx_{i+1} + b = m(x_i + 1) + b = y_i + m]$. Teda platí

$$x_{i+1} = x_i + 1,$$
$$y_{i+1} = y_i + m.$$

- Ak $|m| < 1 \Rightarrow| \ dy \ |<| \ dx \ |$ a $x_A > x_B$ (bod A sa nachádza vpravo od bodu B), môžeme body A a B navzájom vymeniť alebo ekvivalentne postupovať tak, že položíme hodnotu $incr_x = -1$. Potom $y_{i+1} = my_{i+1} + b = m(x_i - 1) + b = y_i - m$. Teda

$$x_{i+1} = x_i + 1,$$
$$y_{i+1} = y_i - m.$$

- Ak $|m| > 1 \Rightarrow| \ dy \ |>| \ dx \ |$ a $y_A < y_B$ (bod A sa nachádza nižšie ako bod B), tak úsečka má dominantný smer y. Vzorkovanie bude prebiehať v smere osi y o konštantnú hodnotu +1. Hodnotu nasledujúcej x-ovej súradnice určíme z rovnice priamky (1.1) ako $x_{i+1} = (y_{i+1} - b)\frac{1}{m} = (y_i + 1 - b)\frac{1}{m} = (mx_i + b + 1 - b)\frac{1}{m} = x_i + \frac{1}{m}$. Teda platí

$$x_{i+1} = x_i + \frac{1}{m},$$
$$y_{i+1} = y_i + 1.$$

- Ak $|m| > 1 \Rightarrow| \ dy \ |>| \ dx \ |$ a $x_A > x_B$ (bod A sa nachádza vyššie ako bod B), môžeme body A a B navzájom vymeniť alebo ekvivalentne postupovať tak, že položíme hodnotu $incr_y = -1$. Potom $x_{i+1} = (y_{i+1} - b)\frac{1}{m} = (y_i - 1 - b)\frac{1}{m} = (mx_i + b - 1 - b)\frac{1}{m} = x_i - \frac{1}{m}$. Teda

$$x_{i+1} = x_i - \frac{1}{m},$$
$$y_{i+1} = y_i - 1.$$

Je zrejmé, že počet vykreslených bodov v rastri pozdĺž úsečky sa rovná maximálnej hodnote rozdielu x/y-ovej súradnice krajných bodov, teda $n = max\{| \ dy \ |, | \ dx \ |\}$. Môžeme povedať, že pre $incr_x$ a $incr_y$ platí:

- $incr_x = \frac{dx}{n}$
- $incr_y = \frac{dy}{n}$

Postup

1. Vlož dva krajné body (x_A, y_A) a (x_B, y_B) a zober ľavý (s menšou x-ovou súradnicou) ako bod (x_0, y_0).
2. Nahraj (x_0, y_0) do frame buffera, teda zobraz bod (x_0, y_0) do rastra.
3. Vypočítaj konštanty $dx = x_B - x_A$, $dy = y_B - y_A$, $m = \frac{dy}{dx}$, $n = max\{\mid dy \mid, \mid dx \mid\}$ a hodnoty $incr_x = \frac{dx}{n}$, $incr_y = \frac{dy}{n}$.
4. V každej pozícii x_i počnúc $i = 1$ až po $i = n$ pozdĺž úsečky vykonaj:
   (a) $(x_{i+1}, y_{i+1}) = (x_i + incr_x, y_i + incr_y)$.
   (b) Zaokrúhli (x_{i+1}, y_{i+1}) na celé čísla.
   (c) Vykresli bod (x_{i+1}, y_{i+1}).

Príklad

Na ilustráciu algoritmu si ukážeme rasterizáciu úsečky s krajnými bodmi $(x_A, y_A) = (0, 6)$ a $(x_B, y_B) = (4, 0)$.

1. Označíme $(x_0, y_0) = (x_A, y_A) = (0, 6)$
2. Zobrazíme tento bod do rastra.
3. $dx = 4 - 0 = 4$,
   $dy = 0 - 6 = -6$
   $m = \frac{-3}{2}$
   $n = 6$
   $incr_x = \frac{2}{3}$
   $incr_y = -1$
4. $i = 1 :$ $(x_1, y_1) = (x_0 + incr_x, y_0 + incr_y) = (\frac{2}{3}, 5)$. Zaokrúhlenie $(x_1, y_1) = (1, 5)$. Vykresli bod (x_1, y_1).

   $i = 2 :$ $(x_2, y_2) = (x_1 + incr_x, y_1 + incr_y) = (\frac{4}{3}, 4)$. Zaokrúhlenie $(x_2, y_2) = (1, 4)$. Vykresli bod (x_2, y_2).

   $i = 3 :$ $(x_3, y_3) = (x_2 + incr_x, y_2 + incr_y) = (\frac{6}{3}, 3)$. Zaokrúhlenie $(x_3, y_3) = (2, 3)$. Vykresli bod (x_3, y_3).

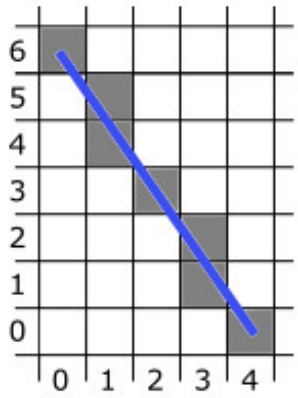
   $i = 4 :$ $(x_4, y_4) = (x_3 + incr_x, y_3 + incr_y) = (\frac{8}{3}, 2)$. Zaokrúhlenie $(x_4, y_4) = (3, 2)$. Vykresli bod (x_4, y_4).

   $i = 5 :$ $(x_5, y_5) = (x_4 + incr_x, y_4 + incr_y) = (\frac{10}{3}, 1)$. Zaokrúhlenie $(x_5, y_5) = (3, 1)$. Vykresli bod (x_5, y_5).

   $i = 6 :$ $(x_6, y_6) = (x_5 + incr_x, y_5 + incr_y) = (\frac{12}{3}, 0)$. Zaokrúhlenie $(x_6, y_6) = (4, 0)$. Vykresli bod (x_6, y_6).Hodnota $i = n$ a dostali sme sa do koncového bodu (x_B, y_B). Výsledok môžeme vidieť na Obr. 5.

Pseudokód

Implementácia DDA algoritmu je zhrnutá v nasledujúcom pseudokóde. Na vstupe sú krajné body úsečky. Parametre dx a dy predstavujú horizontálnu a vertikálnu vzdialenosť dvoch krajných bodov úsečky. Väčšia z týchto hodnôt určuje hodnotu parametru `steps`.



Obr. 5: Výsledok

Ak absolútna hodnota dx je väčšia ako absolútna hodnota dy a x_A je menšie ako x_B, hodnoty prírastku `xIncrement` a `yIncrement`sú 1 a m. V opačnom prípade, ak $x_A > x_B$, tak hodnoty prírastkov `xIncrement` a `yIncrement` sú -1 a $-m$. Funkcia `ROUND(x)` predstavuje zaokrúhlenie hodnoty x na celé čísla.

```
lineDDA (int xA, int yA, int xB, int yB)
{

  int dx = xB − xA, dy = yB − yA, steps,k;
  float xIncrement, yIncrement, x = xA, y = yA;

  if(abs(dx) > abs(dy)) steps = abs(dx);
  else steps = abs(dy);
  xIncrement = dx/(float) steps;
  yIncrement = dy/(float) steps;

  vykresliPixel(ROUND(x), ROUND(y));
  for(k = 0; k < steps; k++){
    x += xIncrement;
    y += yIncrement;
  vykresliPixel(ROUND(x), ROUND(y));
  }
}
```

Literatúra

Algoritmus DDA je spracovaný v anglickej literatúre [3] od strany 87. S jeho odvodením sa stretneme aj v [1] na strane 55 a na strane 81 v [2].

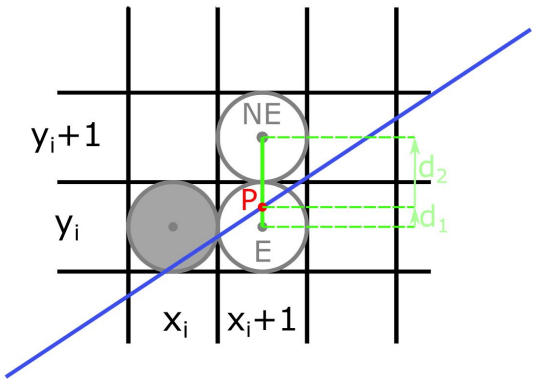
1.1.2 Bresenhamov algoritmus rasterizácie úsečky (Bresenham's Line Algorithm)

Princíp algoritmu

Tento algoritmus vykresľuje body rastra, ktoré ležia najbližšie ku geometrickému obrazu úsečky v zvislom/vodorovnom smere. Využíva sa výhradne celočíselná aritmetika.

Ilustrujeme postup pre časť úsečky AB (bod A leží naľavo od bodu B), ak riadiacou osou je x-ová súradnicová os a pre smernicu tejto úsečky AB platí $0 < m < 1$. Prírastok v smere osi x sa konštantne zväčšuje o hodnotu +1 (Obr. 6).

Predpokladajme, že sme v pozícii bodu úsečky $X = (x_i, y_i)$. Vzhľadom na jednotkové krokovanie v smere osi x, kandidátmi na ďalší bod sú pixely $E = (x_i+1, y_i)$ alebo $NE = (x_i+1, y_i+1)$. Je to ten z nich, ktorý je bližšie ku geometrickému priesečníku P danej úsečky so spojnicou týchto pixlov. Označíme $d_1 = y - y_i$ a $d_2 = (y_i + 1) - y$, kde y je y-ová súradnica bodu P na priamke a teda platí $y = m(x_i+1)+b$ (Obr. 6). Z toho $d_1 = m(x_i+1)+b-y_i$ a $d_2 = (y_i+1)-m(x_i+1)+b$. Zvolíme si premennú $\Delta d = d_1 - d_2 = 2m(x_i+1) - 2y_i + 2b - 1$, podľa ktorej vieme určiť, ktorý



Obr. 6: Časť úsečky AB

z dvoch možných pixlov je bližšie k úsečke AB. Je zrejmé, že:

- Ak $\Delta d < 0 \Leftrightarrow d_1 < d_2$, tak bližšie k úsečke je pixel $E = (x_i + 1, y_i)$.
- Ak $\Delta d > 0 \Leftrightarrow d_1 > d_2$, tak bližšie je pixel $NE = (x_i + 1, y_i + 1)$.
- V prípade, ak $\Delta d = 0 \Leftrightarrow d_1 = d_2$, je jedno, ktorý z týchto pixlov sa vykreslí, obyčajne ten s väčšou y-ovou súradnicou.

Z uvedeného postupu vyplýva, že pre určenie, ktorý pixel sa vykreslí, nie je dôležité vypočítať hodnotu premennej Δd, ale iba určiť jej znamienko. Preto pri rozhodovaní môžeme túto hodnotu nahradiť ľubovoľným jej kladným násobkom, konkrétne parametrom $p_i = dx\Delta d$. Týmto krokom prenesieme výpočet do celočíselnej aritmetiky, pretože tým eliminujeme jediný neceločíselný člen, smernicu $m = \frac{dy}{dx}$.

Pre zjednodušenie výpočtov je vhodné si vyjadriť parameter p_i rekurentne:

$p_i = 2dyx_i - 2dxy_i + C$, kde $C = 2dy + dx(2b - 1)$ je konštanta nezávislá od i.

$p_{i+1} = 2dyx_{i+1} - 2dxy_{i+1} + C$

Z toho: $p_{i+1} - p_i = -2dx(y_{i+1} - y_i) + 2dy \Rightarrow p_{i+1} = p_i - 2dx(y_{i+1} - y_i) + 2dy$. Aby sme mohli tento predpis využiť, potrebujeme ešte hodnotu $p_0 = 2dy - dx$, kde sme pri výpočte využili, že y_0 je bod na priamke a platí $y_0 = mx_0 + b$. Teraz môžeme iteračným spôsobom počítať hodnoty každého nasledujúceho parametra p z jeho predchádzajúcej hodnoty. Teda:

- Ak $p_i < 0 \Leftrightarrow d_1 < d_2$, tak vykreslíme bod na pozícií $E = (x_i + 1, y_i)$, t.j. $y_{i+1} = y_i$ a teda $p_{i+1} = p_i + 2dy$
- Ak $p_i \geq 0 \Leftrightarrow d_1 > d_2$, tak vykreslíme bod na pozícií $NE = (x_i + 1, y_i + 1)$, t.j. $y_{i+1} = y_i + 1$ a teda $p_{i+1} = p_i + 2(dy - dx)$

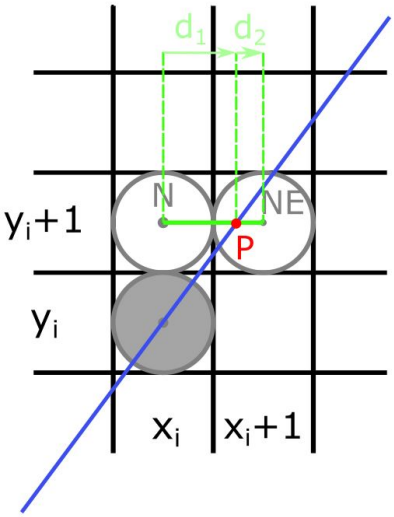
<u>Všeobecne:</u>

Vo vyššie opísanom postupe sme predpokladali, že smernica $0 < m < 1$ a teda $x_{i+1} = x_i + 1$. Ak pre smernicu m platí $-1 < m < 0$, tak sa jednotkový krok na x-ovej osi zmení na -1, teda $x_{i+1} = x_i - 1$.

Princíp algoritmu je rovnaký aj pre úsečku AB so smernicou $m > 1$.

Vtedy sa bod A nachádza nižšie ako bod B, t.j. $y_A < y_B$ (Obr. 7). Úsečka je priklonená k osi y. Preto zvolíme vzorkovanie v smere osi y o hodnotu +1. Ak sa nachádzame v pixli $X = (x_i, y_i)$, tak kandidáti na vykreslenie sú na pozícii $N = (x_i, y_i + 1)$ a $NE = (x_i + 1, y_i + 1)$. Označíme P geometrický priesečník danej úsečky so spojnicou týchto pixlov. Hodnoty d_1 a d_2 reprezentujú vzdialenosť medzi priesečníkom P a stredom pixlov N a NE (Obr. 7). Teda $d_1 = x - x_i$ a $d_2 = x_{i+1} - x$, kde x je x-ová súradnica bodu P a z rovnice priamky (1.1) pre ňu dostaneme $x = \frac{1}{m}(y_{i+1} - b)$.

Parameter p_i je možné určiť ako $p_i = dy(d_1 - d_2) = 2dxy_i - 2dyx_i + C$, kde $C = 2dx(1-b) - dy$ je konštanta nezávislá od i. Potom $p_{i+1} = p_i + 2dx - 2dy(x_{i+1} - x_i)$ a $p_0 = 2dx - dy$



Obr. 7: Časť úsečky AB

Teda pre vzorkovanie v smere osi y platí:

- Ak $p_i < 0 \Leftrightarrow d_1 < d_2$, tak vykreslíme bod na pozícií $N = (x_i, y_i + 1)$, t.j. $x_{i+1} = x_i$ a $p_{i+1} = p_i + 2dx$
- Ak $p_i \geq 0 \Leftrightarrow d_1 > d_2$, tak vykreslíme bod na pozícií $NE = (x_{i+1}, y_{i+1})$, t.j. $x_{i+1} = x_i + 1$ a $p_{i+1} = p_i + 2(dx - dy)$

Postup pre $0 < m < 1$

1. Vlož dva krajné body (x_A, y_A) a (x_B, y_B) a zober ľavý (s menšou x-ovou súradnicou) ako bod (x_0, y_0).
2. Nahraj (x_0, y_0) do frame buffera, teda zobraz bod (x_0, y_0) do rastra.
3. Vypočítaj konštanty dx, dy, $2dy$, $2(dy - dx)$ a urči hodnotu parametra p_0.
4. V každej z pozícii x_i počnúc $i = 0$ pozdĺž úsečky vykonaj test:
   (a) Ak $p_i < 0$: nasledujúci bod je $(x_{i+1}, y_{i+1}) = (x_i + 1, y_i)$ a $p_{i+1} = p_i + 2dy$.
   (b) Inak je nasledujúci bod $(x_{i+1}, y_{i+1}) = (x_i + 1, y_i + 1)$ a $p_{i+1} = p_i + 2dy - 2dx$.
   (c) Vykresli bod (x_{i+1}, y_{i+1}).
5. Opakuj krok 4. dx-krát.

Príklad

Na ilustráciu algoritmu si ukážeme rasterizáciu úsečky s krajnými bodmi $(x_A, y_A) = (0, 0)$ a $(x_B, y_B) = (7, 4)$.

1. Označíme $(x_0, y_0) = (x_A, y_A) = (0, 0)$
2. Zobrazíme tento bod do rastra.
3. $dx = 7 - 0 = 7$,
   $dy = 4 - 0 = 4$
   (smernica $m = \frac{4}{7} < 1$ a teda sa priamka leží v I. oktante a riadiaca os je x)
   $2dy = 2.4 = 8$
   $2(dy - dx) = 2.(4 - 7) = -6$
   $p_0 = 2.4 - 7 = 1$
4. $i = 0 :\ p_0 > 0 \Rightarrow (x_1, y_1) = (x_0 + 1, y_0 + 1) = (1, 1)$ a $p_1 = p_0 + 2(dy - dx) = -5$. Vykresli bod (x_1, y_1).

   $i = 1 :\ p_1 < 0 \Rightarrow (x_2, y_2) = (x_1 + 1, y_1) = (2, 1)$ a $p_2 = p_1 + 2dy = 3$. Vykresli bod (x_2, y_2).

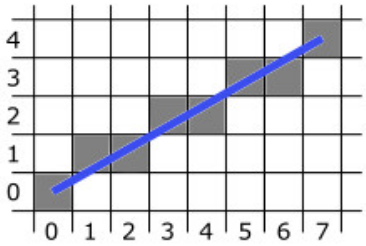
   $i = 2 :\ p_2 > 0 \Rightarrow (x_3, y_3) = (x_2 + 1, y_2 + 1) = (3, 2)$ a $p_3 = p_2 + 2(dy - dx) = -3$. Vykresli bod (x_3, y_3).

   $i = 3 :\ p_3 < 0 \Rightarrow (x_4, y_4) = (x_3 + 1, y_3) = (4, 2)$ a $p_4 = p_3 + 2dy = 5$. Vykresli bod (x_4, y_4).

   $i = 4 :\ p_4 > 0 \Rightarrow (x_5, y_5) = (x_4 + 1, y_4 + 1) = (5, 3)$ a $p_5 = p_4 + 2(dy - dx) = -1$. Vykresli bod (x_5, y_5).

   $i = 5 :\ p_5 < 0 \Rightarrow (x_6, y_6) = (x_5 + 1, y_5) = (6, 3)$ a $p_6 = p_5 + 2dy = 7$. Vykresli bod (x_6, y_6).

   $i = 6 :\ p_6 > 0 \Rightarrow (x_7, y_7) = (x_6 + 1, y_6 + 1) = (7, 4)$. Príkaz sme vykonali dx-krát a dostali sme sa do krajného bodu (x_B, y_B). Vykresli bod (x_7, y_7). Výsledok je zobrazený na Obr. 8.



Obr. 8: Výsledok rasterizácie úsečky AB

Pseudokód

Implementácia Bresenhamovho algoritmu pre smernicu úsečky $0 < m < 1$ je zhrnutá v nasledujúcom pseudokóde. Na vstupe sú dva krajné body úsečky $A = (x_A, y_A)$ a $B = (x_B, y_B)$.

```
lineBres (int xA, int yA, int xB, int yB)
{

  int dx = abs(xB − xA), dy = abs(yB − yA);
  int p=2 * dy - dx;
  int x, y, xKoncovy;

  /* Rozhodnutie, ktorý krajný bod sa použije ako začiatočný a koncový
  if (xA > xB) {
    x = xB;
    y = yB;
    xKoncovy = xA;
  }
  else {
    x = xA;
    y = yA;
    xKoncovy = xB;
  }
  vykresliPixel(x,y);

  while(x < xKoncovy) {
    x++;
    if(p < 0) p += 2 * dy;
    else {
      y++;
      p += 2 * (dy - dx);
    }
  vykresliPixel(x, y);
  }
}
```

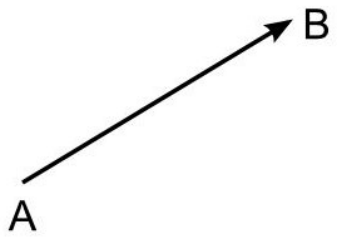
Literatúra

Bresenhamov algoritmus je podrobne spracovaný v literatúre [3] od strany 88. V slovenskej/českej literatúre ho nájdeme odvodený v [1] na strane 58 a taktiež v [2] od strany 82.

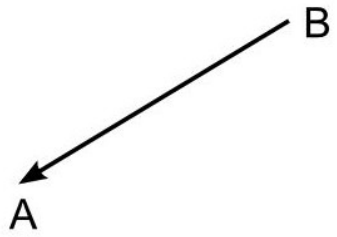
1.1.3 Bresenhamov stredový algoritmus rasterizácie úsečky (Bresenham's Midpoint Line Algorithm)

Princíp algoritmu

Orientovaná úsečka je nenulová úsečka, ktorej krajné body sú usporiadané. Hovoríme o začiatočnom a koncovom bode orientovanej úsečky (Obr. 9. a Obr. 10). Hovoríme tiež, že orientovaná úsečka zo svojho začiatočného bodu vychádza a do koncového bodu vchádza. Orientovaná úsečka orientuje prirodzeným spôsobom priamku, na ktorej leží a naopak. Orientáciu úsečky môžeme intuitívne vnímať ako jeden z dvoch smerov pohybu po nej [8].

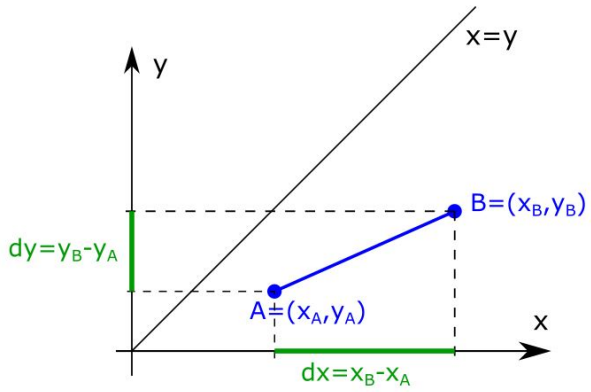


Obr. 9: Orientovaná úsečka AB



Obr. 10: Orientovaná úsečka BA

Uvažujeme orientovanú úsečku AB, kde $x_A < x_B$ (bod A je vľavo od bodu B) a smernica m úsečky AB je $0 \leq m \leq 1$ (Obr. 11.).



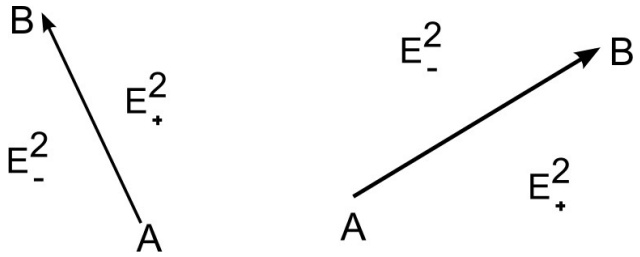
Obr. 11: Úsečka AB

Rovnicu priamky potom možno napísať v implicitnom tvare $f(x, y) = ax + by + c = 0$, kde $a = dy, b = -dx, c = dx.y_B - dy.x_B$. Všimnime si, že všetky koeficienty sú celočíselné. Budeme používať ekvivalentnú reprezentáciu priamky $f(x, y) = 2ax + 2by + 2c = 0$. Kvôli vhodne zvolenej reprezentácii sa presunieme do celočíselnej aritmetiky.

Táto priamka rozdeľuje rovinu E^2 na dve polroviny (Obr. 12.):

- $E^2_- = \{(x, y); f(x, y) < 0\}$. Je to otvorená polrovina, ktorú nazývame ľavá polrovina.
- $E^2_+ = \{(x, y); f(x, y) \geq 0\}$. Je uzavretá polrovina a nazývame ju pravá polrovina.

Z toho môžeme povedať, že:



Obr. 12: Polroviny vzhľadom na orientovanú úsečku AB

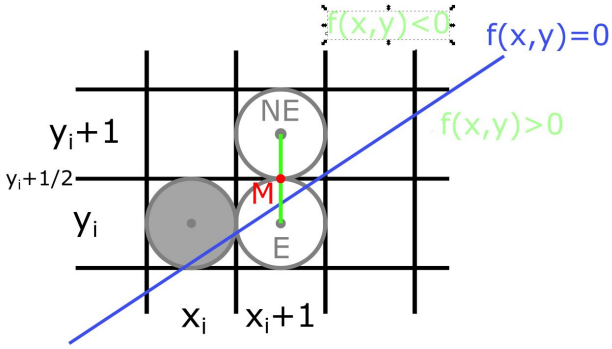
- bod $X = (x, y)$ leží vpravo od priamky $AB \Leftrightarrow f(x, y) > 0$
- bod $X = (x, y)$ leží vľavo od priamky $AB \Leftrightarrow f(x, y) < 0$
- bod $X = (x, y)$ leží na priamke $AB \Leftrightarrow f(x, y) = 0$.

Predpoklad, že pre smernicu vykresľovanej úsečky platí $0 \leq m \leq 1$, môžeme odstrániť. Ak $|m| > 1$, tak zámena $x \leftrightarrow y$ vedie k úsečke s prevrátenou hodnotou smernice. Ak $m < 0$, algoritmus vieme modifikovať zámenou prírastok $\leftrightarrow$ úbytok alebo stúpanie $\leftrightarrow$ klesanie. Výmenou vstupných bodov $A \leftrightarrow B$ je možné vždy zabezpečiť kreslenie zľava doprava.

Opäť uvažujme, že pre smernicu platí $0 < m < 1$. Pri rasterizácii sme dosiahli pixel (x_i, y_i). Potrebujeme rozhodnúť, ktorý z nasledujúcich pixlov $E = (x_i+1, y_i)$ alebo $NE = (x_i+1, y_i+1)$ vykreslíme.

Použijeme bod $M = stred(E, NE) = (x_i + 1, y_i + \frac{1}{2})$ a jeho polohu vzhľadom na priamku $f(x, y)$ (Obr. 13.). Označíme $D = f(M) = 2a(x_i + 1) + 2b(y_i + \frac{1}{2}) + 2c = 2ax_i + 2by_i + (2a + b + 2c) \in \mathbb{N}$. Zrejme platí:

- Ak $D > 0$, tak bod M leží v E^2_+ a z uvažovaných pixlov je bližšie NE, ktorý sa vysvieti.
- Ak $D < 0$, tak bod M leží v E^2_- a bližšie je bod E, ten sa rozsvieti.
- Ak nastáva $D = 0$ vysvietime hociktorý z bodov NE, E, zvyčajne NE.



Obr. 13: Výber NE resp. E

Je výhodné, že D je celočíselné, ale jeho výpočet si vyžaduje dve násobenia a dve sčítania, ak nemeniaca sa zložka v zátvorke je prepočítaná. Jedným z veľmi šikovných trikov je však, že hodnota D sa počíta prírastkovo.

Predpokladajme, že poznáme aktuálnu hodnotu D a chceme vypočítať jeho nasledujúcu hodnotu:

1. Ak sme ako nový vysvietený pixel vybrali $E = (x_i + 1, y_i)$, tak v nasledujúcom kroku k nemu prislúcha nový stred $M_{new} = ((x_i + 1) + 1, y_i + \frac{1}{2})$ a nové $D_{new} = f(M_{new}) = 2a(x_i + 2) + 2b(y_i + \frac{1}{2}) + 2c = 2a(x_i + 1) + 2by_i + (2a + b + 2c) = D + 2a = D + 2dy$

2. Ak sme však ako nový vysvietený pixel vybrali bod $NE = (x_i + 1, y_i + 1)$, tak k nemu prislúcha nový stred $M_{new} = ((x_i+1)+1, (y_i+1)+\frac{1}{2})$ a $D_{new} = f(M_{new}) = 2a(x_i+2)+2b(y_i+1+\frac{1}{2})+2c = 2a(x_i+1)+2b(y_i+1)+(2a+b+2c) = D+2a+2b = D+2(dy-dx)$.

Platí teda:

- Ak $D < 0$, tak $D_{new} = D + 2dy$ a vysvietime bod E
- Ak $D \geq 0$, tak $D_{new} = D + 2(dy - dx)$ a vysvietime bod NE

Z toho vyplýva, že parameter D je ekvivalentný parametru p z predchádzajúceho algoritmu a možno ho považovať za rozhodovací parameter Bresenhamovho algoritmu rasterizácie úsečky. Teda algoritmus možno dokončiť ako Bresenhamov line algoritmus.

V literatúre parameter D autori často uprednostňujú pred parametrom p z predchádzajúceho algoritmu, lebo princíp jeho konštrukcie je možno použiť aj v ďalších algoritmoch ako napr. v Bresenhamovom algoritme rasterizácie kružnice.

Postup pre $|m| < 1$

Kroky algoritmu vieme zhrnúť rovnakým spôsobom ako v predošlom algoritmu, keďže parameter D je ekvivalentný parametru p.

1. Vlož dva krajné body (x_A, y_A) a (x_B, y_B) a zober ľavý (s menšou x-ovou súradnicou) ako bod (x_0, y_0).
2. Nahraj (x_0, y_0) do frame buffera, teda zobraz bod (x_0, y_0) do rastra.
3. Vypočítaj konštanty dx, dy, $2dy$, $2(dy - dx)$ a urči počiatočnú hodnotu parametra $D = f(M) = 2ax_i + 2by_i + (2a + b + 2c)$, kde $M = (x_i + 1, y_i + \frac{1}{2})$.
4. V každej pozícií x_i začínajúc s $i = 0$ pozdĺž úsečky vykonaj test:
   (a) Ak $D < 0$: nasledujúci bod je $(x_{i+1}, y_{i+1}) = (x_i + 1, y_i)$ a $D_{new} = D + 2dy$.
   (b) Inak je nasledujúci bod $(x_{i+1}, y_{i+1}) = (x_i + 1, y_i + 1)$ a $D_{new} = D + 2dy - 2dx$.
   (c) Vykresli bod (x_{i+1}, y_{i+1}) a $D = D_{new}$.
5. Opakuj krok 4. dx-krát.

Príklad

Ako sme uviedli parameter D je ekvivalentný parametru p z predchádzajúceho algoritmu, postup výpočtu príkladu je rovnaký ako v predchádzajúcom príklade.

Pseudokód

Keďže parameter D je ekvivalentný parametru p z predchádzajúceho algoritmu, implementácia Bresenhamovho stredového algoritmu pre smernicu úsečky $0 < m < 1$ je zhrnutá v predchádzajúcej kapitole.

Literatúra

Bresenhamov stredový algoritmus sa nenachádza v [3] ani v [7]. Zo slovenskej/českej literatúry sa taktiež nenachádza v [1] ani [2].

1.2 Algoritmy na rasterizáciu kružnice

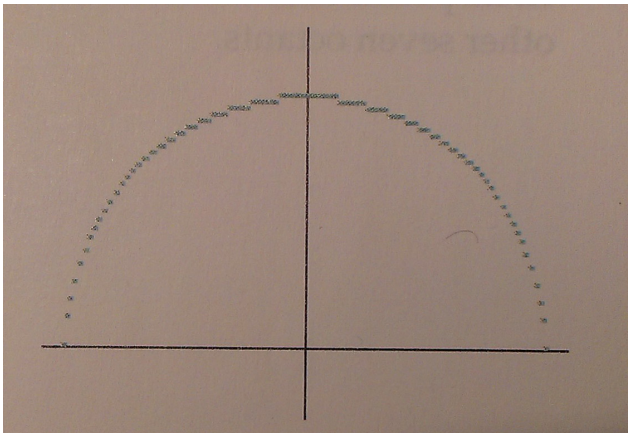
Kružnice sú často používané komponenty v obrázkoch a grafoch, a preto sú procedúry na ich generovanie zahrnuté v grafických knižniciach. Kružnica v E^2 je definovaná ako množina všetkých bodov v rovine, ktoré sú vo vzdialenosti r od stredu kružnice (x_C, y_C). Číslo r nazývame polomerom kružnice. Implicitnú rovnicu kružnice v karteziánskej súradnicovej sústave môžeme zapísať ako množinu bodov (x, y), ktoré vyhovujú rovnici

$$(x - x_C)^2 + (y - y_C)^2 - r^2 = 0, \tag{1.5}$$

kde (x_C, y_C) je stred kružnice a r je polomer danej kružnice. Túto rovnicu je možné upraviť na explicitné vyjadrenie

$$y = y_C \pm \sqrt{r^2 - (x_C - x)^2}, \tag{1.6}$$

v ktorom môžeme využiť jednotkový krok na osi x, a $x \in$
$\langle x_C - r, x_C + r \rangle$ Táto metóda však nie je najvýhodnejšia pre generovanie kružníc. Problémom je veľa výpočtov v každom kroku a pri vykresľovaní sa objavujú medzery medzi jednotlivými pixelmi (Obr. 14). Môžeme zameniť úlohy x a y a postupovať jednotkovým krokom po y-ovej



Obr. 14: Kružnica vykreslená v kladnej polrovine pomocou rovnice (2.6) [3]

súradnici, čím však zvýšime počet potrebných výpočtov a predĺžime čas behu algoritmu. Určitú eliminácia nerovnomerne zobrazených pixlov (na Obr. 14) môžeme zabezpečiť pomocou parametrizovaného vyjadrenia kružnice.

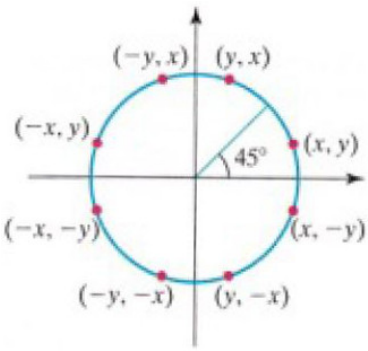
Parametrizovaná rovnica kružnice:

$$\begin{aligned} x(t) &= x_C + r.\cos(t), \\ y(t) &= y_C + r.\sin(t), \end{aligned} \tag{1.7}$$

kde parameter $t \in \langle 0, 2\pi)$, r je polomer kružnice a (x_C, y_C) je stred danej kružnice.

Ak generujeme kružnicu pomocou týchto rovníc pri rovnomernom uhlovom kroku, kružnica je zobrazená s rovnomerne vzdialenými bodmi pozdĺž kružnice. Väčšie medzery medzi bodmi kružnice môžu byť spojené úsečkami, aby aproximovali kružnicový tvar. Pre spojité zobrazenie kružnice môžeme použiť krokovanie o veľkosti $\frac{1}{r}$. Získané body následne budú vzdialené približne jeden pixel od seba.

Generovanie bodov kružnice môže byť zjednodušené využitím osových a stredových súmerností kružnice (Obr. 15). Segmenty kružnice sú v jednotlivých oktantoch 1-8 rovnaké. Preto stačí



Obr. 15: Symetrie bodu (x, y) na kružnici[3]

generovať iba segment kružnice v jednom oktante a ten pomocou súmerností zobraziť do ostatných.

Generovanie kružnice pomocou vyššie opísaných postupov vyžaduje značné množstvo výpočtov a teda dlhší čas behu algoritmu. V nasledujúcich podkapitolách opíšeme algoritmy na efektívne vykreslenie kružnice: minimalizácia výpočtov a s využitím len celočíselnej aritmetiky.

Pri všetkých algoritmoch máme na vstupe polomer r a stred (x_C, y_C) kružnice K. Na výstupe dostaneme množinu bodov rastra, ktoré aproximujú danú kružnicu K.

1.2.1 Bresenhamov kružnicový algoritmus (Bresenham's Circle Algorithm)

Princíp algoritmu

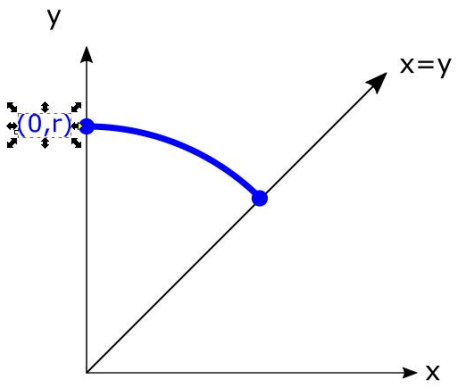
Tento algoritmus predstavuje zovšeobecnenie Bresenhamovho algoritmu rasterizácie úsečky na kružnicu. Rozklad zrýchlime tým, že budeme generovať len segment kružnice v jednom oktante. Ostatné časti kružnice dostaneme súmernosťou podľa súradnicových osí x, y a priamok $x = y$ a $y = -x$.

Výber pixlov je založený na rovnakom princípe ako pri úsečke, kde z dvoch možných kandidátov na vysvietenie vyberáme toho, ktorého stred je bližšie ku danej kružnici. Využívame parameter $p_i = d_1 - d_2$, ktorý sa vyčísľuje rekurentne, kde hodnoty d_1 a d_2 sú štvorcami rozdielov y-ových súradníc.

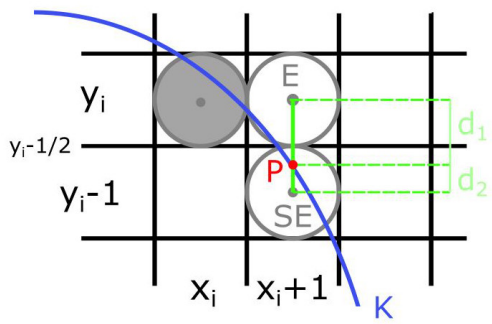
Pre ilustráciu algoritmu volíme kružnicu so stredom v začiatku súradnicovej sústavy a zobrazíme jej segment, ktorý sa začína v bode $(0, r)$ a končí, v bode (x_i, y_i), kde $x_i \geq y_i$ (Obr. 16). V tomto prípade je jednotkový krok v smere súradnicovej osi x. Nachádzame sa v bode (x_i, y_i) a potrebujeme sa rozhodnúť, ktorý z pixlov $E = (x_i + 1, y_i)$ alebo $SE = (x_i + 1, y_i - 1)$ si vyberieme (Obr. 17).

Označíme si $d_1 = y_i^2 - y^2$ a $d_2 = y^2 - (y_i - 1)^2$, kde y je hodnota určená rovnicou $y^2 = r^2 - (x_i + 1)^2$ z (1.5). Hodnota d_1 vyjadruje vzdialenosť medzi bodmi E a $(x_i + 1, y)$ a hodnota d_2 medzi $(x_i + 1, y)$ a SE. Určíme $p_i = d_1 - d_2 = 2(x_i + 1)^2 + y_i^2 + (y_i - 1)^2 - 2r^2$. Zrejme platí

- Ak $p_i < 0 \Leftrightarrow d_1 < d_2$, tak bod E je bližšie k bodu na kružnici $(x_i + 1, y_i)$ a preto vysvietime pixel E.
- Ak $p_i \geq 0 \Leftrightarrow d_1 \geq d_2$, tak je bližšie bod SE, ktorý následne vysvietime. Pri rovnosti môžeme vybrať ľubovoľný z bodov E, SE, no zvyčajne sa vyberá vyššie položený bod.



Obr. 16: Generovaný segment



Obr. 17: Bresenhamov algoritmus generovania kružnice

Teraz vyjadríme parameter p_i rekurentne pomocou p_{i+1}
$p_{i+1} = 2(x_{i+1}+1)^2 + y_{i+1}^2 + (y_{i+1}-1)^2 - 2r^2$ a rozdielu
$p_{i+1} - p_i = 4x_i + 6 + 2(y_{i+1}^2 - y_i^2) - 2(y_{i+1} - y_i)$.
Možno zapísať rekurentný vzťah $p_{i+1} = p_i + 4x_i + 6 + 2(y_{i+1}^2 - y_i^2) - 2(y_{i+1} - y_i)$. Teda:

- Ak $p_i < 0 \Leftrightarrow d_1 < d_2$, nasledujúci bod je $(x_{i+1}, y_{i+1}) = E = (x_i+1, y_i)$ a $p_{i+1} = p_i + 4x_i + 6$
- Ak $p_i \geq 0 \Leftrightarrow d_1 \geq d_2$, nasledujúci bod je $(x_{i+1}, y_{i+1}) = SE = (x_i + 1, y_i - 1)$, a preto $p_{i+1} = p_i + 4x_i + 6 + 2(y_i - y_i^2)$

Potom pre začiatočnú hodnotu p_0 v bode $(x_C, y_C) = (0, r)$ dostaneme $p_0 = 3 - 2r$. Hoci pri výpočte parametra sa používa násobenie, príslušný násobok je druhou mocninou a je možná implementácia cez logické operácie. Ostatné operácie sú celočíselné sčítanie a odčítanie.

Postup

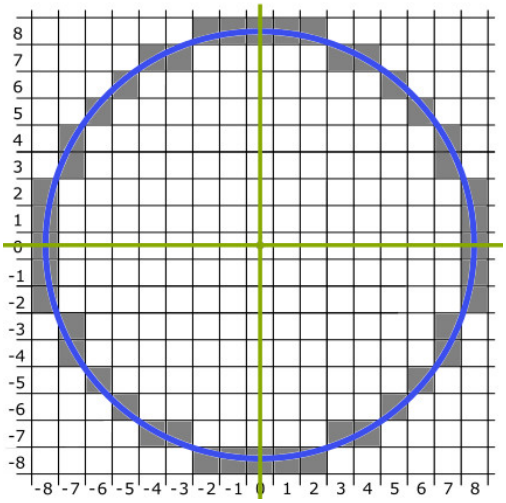
Kroky algoritmu vieme zhrnúť nasledovne:

1. Zadaj polomer r a stred kružnice (x_C, y_C) a urči prvý bod na kružnici so stredom v začiatku ako $(x_0, y_0) = (0, r)$.
2. Vypočítaj začiatočnú hodnotu rozhodovacieho parametra ako $p_0 = 3 - 2r$
3. V každej pozícii x_i, štartujúc v $i = 0$, vykonaj nasledujúci test:
   (a) Ak $p_i < 0$: nasledujúci bod pozdĺž kružnice so stredom $(0, 0)$ bude $(x_{i+1}, y_{i+1}) = (x_i + 1, y_i)$ a $p_{i+1} = p_i + 4x_i + 6$.
   (b) Inak je nasledujúci bod $(x_{i+1}, y_{i+1}) = (x_i + 1, y_i - 1)$ a $p_{i+1} = p_i + 4(x_i - y_i) + 10$.
4. Urči súmerne položené body v ostatných siedmich oktantoch.
5. Posuň každú vypočítanú pixlovú pozíciu (x, y) na kružnicu so stredom (x_C, y_C) a zobraz bod so súradnicami: $x = x + x_C$,$y = y + y_C$.
6. Opakuj kroky 3. až 5. pokiaľ $x_i \geq y_i$.

Príklad

Na ilustráciu algoritmu si ukážeme rasterizáciu kružnice s polomerom $r = 8$ a stredom kružnice $(x_C, y_C) = (1, 2)$.

1. $(x_0, y_0) = (0, r) = (0, 8)$
2. $p_0 = 3 - 2r = -13$
3. $i = 0: \; p_0 < 0 \Rightarrow (x_1, y_1) = (x_0 + 1, y_0) = (1, 8)$ a $p_1 = p_0 + 4x_0 + 6 = -7$
   $i = 1: \; p_1 < 0 \Rightarrow (x_2, y_2) = (x_1 + 1, y_1) = (2, 8)$ a $p_2 = p_1 + 4x_1 + 6 = 3$
   $i = 2: \; p_2 > 0 \Rightarrow (x_3, y_3) = (x_2 + 1, y_2 - 1) = (3, 7)$ a $p_3 = p_2 + 4(x_2 - y_2) + 10 = -11$
   $i = 3: \; p_3 < 0 \Rightarrow (x_4, y_4) = (x_3 + 1, y_3) = (4, 7)$ a $p_4 = p_3 + 4x_3 + 6 = 7$
   $i = 4: \; p_4 > 0 \Rightarrow (x_5, y_5) = (x_4 + 1, y_4 - 1) = (5, 6)$ a $p_5 = p_4 + 4(x_4 - y_4) + 10 = 5$
   $i = 5: \; p_5 > 0 \Rightarrow (x_6, y_6) = (x_5 + 1, y_5 - 1) = (6, 5)$ a $x_6 \geq y_6$ teda algoritmus končí.
4. Určíme súmerne položené body v ostatných oktantoch (Obr. 18).
5. Posunieme každú hodnotu pixla do stredu $(1, 2)$. Teda body: $(2, 10), (3, 10), (4, 9), (5, 9), (6, 8), ...$



Obr. 18: Výsledná kružnica so stredom v bode $(0, 0)$

Pseudokód

Implementácia Bresenhamovho kružnicového algoritmu je zhrnutá v nasledujúcom pseudokóde. Na vstupe je stred (x_C, y_C) a polomer kružnice r.

```
circleBres (int xC, int yC, int r)
{
  int x = 0;
  int y = r;
  int p = 3 - (2 * r);
  void vykresliBodKruznice(int, int, int, int);

  /* Vykreslí prvý bod kružnice
```

```
  vykresliBodKruznice(xC, yC, x, y);

  while(x < y) {
    x++;
    if(p < 0) p = p + (4 * x) + 6;
    else {
      y--;
      p = p + 4 * (x - y) + 10;
    }
  vykresliBodKruznice(xC, yC, x, y);
  }
}

void vykresliBodKruznice(int xC, int yC, int x, int y);
{
  vykresliPixel(xC + x, yC + y);
  vykresliPixel(xC - x, yC + y);
  vykresliPixel(xC + x, yC - y);
  vykresliPixel(xC - x, yC - y);
  vykresliPixel(xC + y, yC + x);
  vykresliPixel(xC - y, yC + x);
  vykresliPixel(xC + y, yC - x);
  vykresliPixel(xC - y, yC - x);
}
```

Literatúra

Bresenhamov kružnicový algoritmus sa nachádza podrobne spracovaný v literatúre [3]. Jeho myšlienka je načrtnutá aj v [1] na strane 62 a taktiež jeho vysvetlenie sa nachádza v [2] od strany 88.

1.2.2 Bresenhamov stredový kružnicový algoritmus (Midpoint Circle Algorithm)

Princíp algoritmu

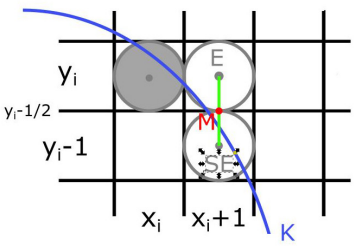
Algoritmus predstavuje úpravu Bresenhamovho midpoint line algoritmu pre kružnicu. Algoritmus vychádza z polomeru r kružnice K a jej stredu v začiatku súradnicovej sústavy, t.j. v bode $(x_C, y_C) = (0, 0)$.

Ak sa stred (x_C, y_C) nenachádza v začiatku, tak pozíciu bodu kružnice (x, y) vieme vypočítať pričítaním x_C k x-ovej súradnici a y_C k y-ovej. Vychádzame z rovnice kružnice: $f(x, y) = x^2 + y^2 - r^2 = 0$.

Platí:

- Ak $f(x, y) < 0$, tak bod (x, y) sa nachádza vnútri kružnice K. Takýto bod nazývame vnútorný bod
- Ak $f(x, y) = 0$, tak bod (x, y) sa nachádza na kružnici K
- Ak $f(x, y) > 0$, tak bod (x, y) sa nachádza mimo kružnice K. Takýto bod nazývame vonkajší bod.

Opäť sa nachádzame na segmente kružnice od bodu $(0, r)$ až po bod, pre ktorý je $x_i \geq y_i$ Nech sme v bode (x_i, y_i) a hľadáme nasledujúci bod. Rozhodujeme sa medzi dvoma kandidátmi $E = (x_i + 1, y_i)$ a $SE = (x_i + 1, y_i - 1)$ (Obr. 19).



Obr. 19: Midpoint algoritmus

Určíme $M = stred(E, SE) = (x_i + 1, y_i - \frac{1}{2})$. Budeme zisťovať, či tento bod je vnútorný alebo vonkajší. Definujeme rozhodovací parameter p_i: $p_i = f(M) = (x_i + 1)^2 + (y_i - \frac{1}{2})^2 - r^2$ a odvodíme z neho rekurentný vzorec:
$p_{i+1} = f((x_{i+1} + 1, y_{i+1} - \frac{1}{2}) = (x_{i+1} + 1)^2 + (y_{i+1} - \frac{1}{2})^2 - r^2$
Potom $p_{i+1} - p_i = 2x_{i+1} + (y_{i+1}^2 - y_{i+1}) - (y_i^2 - y_i) + 1$ rekurentný predpis má tvar:
$p_{i+1} = p_i + 2x_{i+1} + (y_{i+1}^2 - y_{i+1}) - (y_i^2 - y_i) + 1$

Zrejme platí:

- Ak $p_i < 0$, vyberáme ako nasledujúci pixel $(x_{i+1}, y_{i+1}) = E = (x_i + 1, y_i)$ a $p_{i+1} = p_i + 2x_i + 3$
- Ak $p_i \geq 0$, vyberáme bod $(x_{i+1}, y_{i+1}) = SE = (x_i + 1, y_i - 1)$ a teda $p_{i+1} = p_i + 2(x_i - y_i) + 5$

Potrebujeme ešte poznať hodnotu začiatočného parametra $p_0 = (x_0 + 1)^2 + (y_0 - \frac{1}{2})^2 - r^2$, ktorý pre bod $(x_0, y_0) = (0, r)$ sa rovná $p_0 = \frac{5}{4} - r$. Ak je polomer špecifikovaný ako celé číslo, môžeme hodnotu p_0 zaokrúhliť ako $p_0 = 1 - r$, lebo všetky prírastky sú celočíselné.

Na rozdiel od lineárnych algoritmov rasterizácie úsečky, sme v prípade kružnicových algoritmov nezískali rovnaké rozhodovacie parametre.

Postup algoritmu sme ilustrovali pre kružnicový oblúk, ktorý sa nachádza v II. oktante, avšak algoritmus je možné rovnakým spôsobom odvodiť aj pre ktorýkoľvek z ostatných oktantov. Podobne ako Bresenhamov line algoritmus, aj tento vypočítava pixely pozdĺž kružnice, používajúc pri tom len celočíselné sčítania a odčítania za predpokladu, že stred a polomer kružnice sú špecifikované v celočíselných obrazových súradniciach.

Postup

Kroky algoritmu vieme zhrnúť nasledovne:

1. Zadaj polomer r a stred kružnice (x_C, y_C) a urči prvý bod na kružnici so stredom v začiatku $(x_0, y_0) = (0, r)$.
2. Vypočítaj začiatočnú hodnotu rozhodovacieho parametra ako $p_0 = 1 - r$ (pretože predpokladáme, že r je celočíselné).
3. V každej pozícii x_i, štartujúc v $i = 0$, vykonaj nasledujúci test:
   (a) Ak $p_i < 0$: nasledujúci bod pozdĺž kružnice so stredom $(0, 0)$ bude $(x_{i+1}, y_{i+1}) = (x_i + 1, y_i)$ a $p_{i+1} = p_i + 2x_i + 3$.
   (b) Inak je nasledujúci bod $(x_{i+1}, y_{i+1}) = (x_i + 1, y_i - 1)$ a $p_{i+1} = p_i + 2(x_i - y_i) + 5$.
4. Urči súmerné body v ostatných siedmich oktantoch.
5. Posuň každú vypočítanú pixlovú pozíciu (x, y) na kružnicu so stredom (x_C, y_C) a zobraz bod so súradnicami: $x = x + x_C$,$y = y + y_C$.
6. Opakuj kroky 3. až 5. pokiaľ $x_i \geq y_i$.

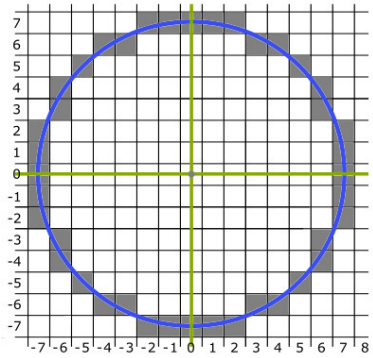
Príklad

Na ilustráciu algoritmu si ukážeme rasterizáciu kružnice so stredom v bode $(x_C, y_C) = (1, 2)$ a polomerom $r = 7$.

1. $(x_0, y_0) = (0, r) = (0, 7)$
2. $p_0 = 1 - r = -6$
3. $i = 0 :\ p_0 < 0 \Rightarrow (x_1, y_1) = (x_0 + 1, y_0) = (1, 7)$ a $p_1 = p_0 + 2x_0 + 3 = -3$
   $i = 1 :\ p_1 < 0 \Rightarrow (x_2, y_2) = (x_1 + 1, y_1) = (2, 7)$ a $p_2 = p_1 + 2x_1 + 3 = 2$
   $i = 2 :\ p_2 > 0 \Rightarrow (x_3, y_3) = (x_2 + 1, y_2 - 1) = (3, 6)$ a $p_3 = p_2 + 2(x_2 - y_2) + 5 = -3$
   $i = 3 :\ p_3 < 0 \Rightarrow (x_4, y_4) = (x_3 + 1, y_3) = (4, 6)$ a $p_4 = p_3 + 2x_3 + 3 = 6$
   $i = 4 :\ p_4 > 0 \Rightarrow (x_5, y_5) = (x_4 + 1, y_4 - 1) = (5, 5)$ a $x_5 \geq y_5$ teda algoritmus končí.
4. Určíme súmerné body v ostatných oktantoch (Obr. 20).
5. Posunieme každú hodnotu pixla do stredu $(1, 2)$. Teda body: $(2, 9), (3, 9), (4, 8), (5, 8), (6, 8), ...$

Pseudokód

Implementácia Bresenhamovho stredového kružnicového algoritmu je zhrnutá v nasledujúcom pseudokóde. Na vstupe je stred kružnice (x_C, y_C) a polomer kružnice r.



Obr. 20: Výsledná kružnica so stredom v bode $(0, 0)$

```
circleMidpoint (int xC, int yC, int r){
  int x = 0;
  int y = r;
  int p = 1 - r;
  void vykresliBodKruznice(int, int, int, int);

  /* Vykreslí prvý bod kružnice
  vykresliBodKruznice(xC, yC, x, y);

  while(x < y) {
    x++;
    if(p < 0) p += 2 * x + 3;
    else {
      y--;
      p += 2 * (x - y) + 5;
    }
  vykresliBodKruznice(xC, yC, x, y);
  }
}
void vykresliBodKruznice(int xC, int yC, int x, int y); {

  vykresliPixel(xC + x, yC + y);
  vykresliPixel(xC - x, yC + y);
  vykresliPixel(xC + x, yC - y);
  vykresliPixel(xC - x, yC - y);
  vykresliPixel(xC + y, yC + x);
  vykresliPixel(xC - y, yC + x);
  vykresliPixel(xC + y, yC - x);
  vykresliPixel(xC - y, yC - x);
}
```

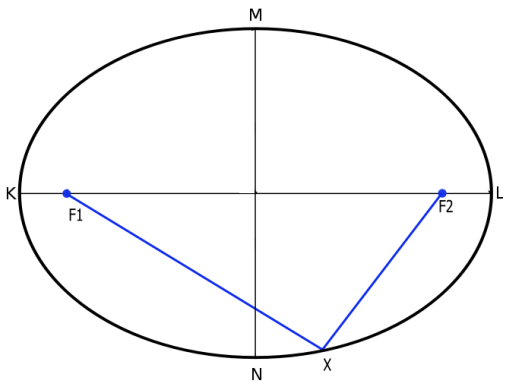
Literatúra

Bresenhamov stredový kružnicový algoritmus je detailne spracovaný v knihe [3] od strany 98. Zo slovenskej/českej literatúry nie je uvedený ani v [1] ani v [2].

1.3 Algoritmy na rasterizáciu elipsy

Elipsa je rovinná krivka, ktorá patrí do triedy kužeľosečiek.

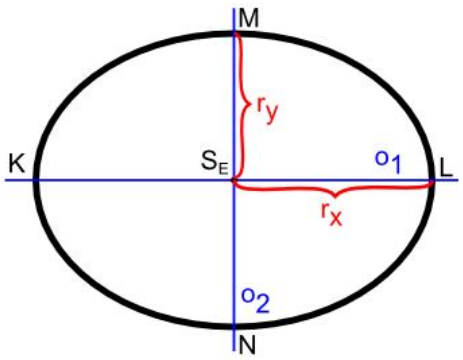
Elipsa E je definovaná ako množina všetkých bodov roviny, ktoré majú od dvoch rôznych pevných bodov F_1, F_2 rovnaký súčet vzdialeností, ktorý je väčší ako vzdialenosť týchto bodov (Obr. 21.).



Obr. 21: Elipsa

Body F_1 a F_2 nazývame ohniská elipsy. Priamka prechádzajúca týmito dvoma bodmi sa nazýva hlavná os elipsy a označíme ju o_1. Stred úsečky F_1F_2 nazývame stred elipsy a označíme ho ako $S_E = (x_C, y_C)$. Priamku kolmú na os o_1 prechádzajúcu bodom S_E nazývame vedľajšia os elipsy a označíme ju o_2. Body, v ktorých elipsa E pretne os o_1, nazývame hlavné vrcholy (body K a L na Obr. 22.) a body prieniku osi o_2 s elipsou E nazývame vedľajšie vrcholy elipsy (body M,N na Obr. 22.).

Dĺžku úsečky S_EL označíme ako r_x a nazývame ju hlavná poloos elipsy. Analogicky dĺžku úsečky S_EM označujeme r_y a nazývame vedľajšia poloos.

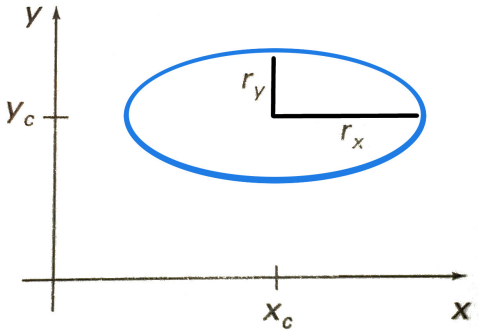


Obr. 22: Časti elipsy

Ďalej sa budeme zaoberať iba špeciálnou polohou elipsy, v ktorej je hlavná a vedľajšia os rovnobežná so súradnicovými osami. Túto polohu nazývame základná pozícia elipsy (Obr.23.).

V tomto prípade môžeme rovnicu elipsy zapísať ako

$$\left(\frac{x - x_C}{r_x}\right)^2 + \left(\frac{y - y_C}{r_y}\right)^2 = 1 \tag{1.8}$$



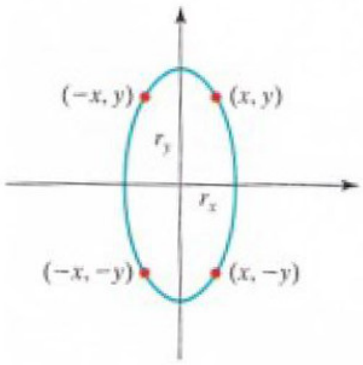
Obr. 23: Elipsa so stredom v bode (x_C, y_C) v základnej pozícii [3]

Pomocou parametrických súradníc vieme elipsu v základnej pozícií vyjadriť pomocou rovníc

$$\begin{aligned} x(t) &= x_C + r_x \cos(t), \\ y(t) &= y_C + r_y \sin(t), \end{aligned} \tag{1.9}$$

kde parameter $t \in \langle 0, 2\pi)$.

Na zrýchlenie výpočtov pixlov pozdĺž elipsy môžeme využiť súmernosti elipsy. Elipsa v základnej pozícii na rozdiel od kružnice je súmerná iba podľa súradnicových osí x a y. Preto môžeme vyčísliť pozície pixelov v jednom kvadrante a ostatné dostaneme zo súmerností (Obr. 24.).



Obr. 24: Súmerností elipsy [3]

V nasledujúcej podkapitole si ukážeme jeden algoritmus na zobrazenie elipsy do rastra. Na vstupe algoritmu máme polomer stred (x_C, y_C) elispy E a hodnoty r_x a r_y. Na výstupe dostaneme množinu bodov rastra, ktoré aproximujú danú elipsu E.

1.3.1 Bresenhamov stredový elipsový algoritmus (Midpoint Ellipse Algorithm)

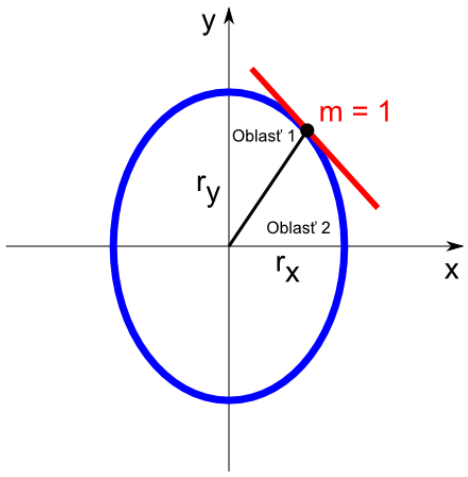
Princíp algoritmu

Bresenhamov stredový elipsový algoritmus je založený na rovnakom princípe ako Bresenhamov kružnicový algoritmus. Dané sú parametre r_x, r_y a stred elipsy (x_C, y_C). Vyčísľujeme body elipsy v základnej pozícii a to so stredom v bode $(0, 0)$. Následne sa všetky body elipsy posunú tak, že bod (x_C, y_C) je jej stredom.

Ak chceme vykresliť elipsu v inej ako základnej polohe (ak hlavná a vedľajšia os elipsy nie sú rovnobežné so súradnicovými osami), môžeme elipsu otočiť okolo tredu otáčania (viď Kapitola 1. rotácia).

V nasledujúcej časti ukážeme rasterizáciu elipsy so stredom v bode $(0, 0)$ a základnej pozícii. Algoritmus budeme aplikovať v I. kvadrante v dvoch oblastiach (Obr. 25.). Tieto oblasti vzniknú rozdelením I. kvadrantu na dva v bode elipsy, v ktorom dotyčnica k elipse má smernicu $m = -1$.

Hranicou medzi oblasťami je priamka spájajúca stred elipsy s dotykovým bodom.



Obr. 25: I. kvadrant elipsy rozdelený na dve oblasti [3]

Začíname v bode elipsy $(0, r_y)$ a postupujeme v smere chodu hodinových ručičiek. Najprv postupujeme jednotkovým krokom v smere osi x v prvej oblasti, až pokým sa nedostaneme na hranicu medzi oblasťami. Vtedy prejdeme na jednotkový krok v smere osi y, až kým neprejdeme celý I. kvadrant.Tento postup je potrebný, pretože ak by sme postupovali jednotkovým krokom iba v smere jednej zo súradnicových osí, pri rasterizácii by vznikali medzery medzi zobrazenými pixlami a výsledná elipsa by bola nespojitá.

Pre smernicu m dotyčnice v bode elipsy $X = (x, y)$ na hranici oblasti platí $m = -1 \Rightarrow \frac{dy}{dx} = \frac{2r_y^2 x}{2r_x^2 y} \Rightarrow 2r_y^2 x = 2r_x^2 y$. Teda pre smernicu m dotyčnice v bode elipsy $X = (x, y)$ v prvej oblasti I. kvadrantu platí $m > -1$ a v druhom $m < -1$. Preto môžeme povedať, že bod elipsy $X = (x, y)$ sa nachádza v prvej oblasti, ak platí $2r_y^2 x \geq 2r_x^2 y$, t.j. vzorkujeme jednotkovým krokom v smere osi x, inak v smere osi y.

Samozrejme, modifikáciou algoritmu je možné postupovať zo začiatočného bodu $(r_x, 0)$ proti smeru hodinových ručičiek.

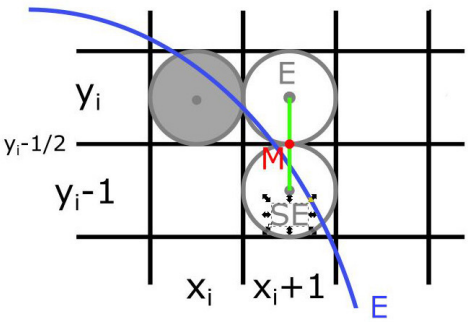
Zapíšeme rovnicu elipsy vyjadrenú v rovnici (1.8) so stredom $(x_C, y_C) = (0, 0)$ ako $f(x, y) = r_y^2 x^2 + r_x^2 y^2 - r_x^2 r_y^2$.

Je zrejmé, že platí:

- Ak $f(x, y) < 0$ tak bod $X = (x, y)$ je vnútorný bod elipsy
- Ak $f(x, y) = 0$ tak bod $X = (x, y)$ je bodom elipsy
- Ak $f(x, y) > 0$ tak bod $X = (x, y)$ je vonkajším bodom elipsy

Napr. nachádzame sa v bode elipsy (x_i, y_i) a rozhodujeme sa, ktorý z nasledujúcich bodov pozdĺž elipsy vykreslíme. Môžu nastať dve situácie:

1. Nachádzame sa v prvej oblasti. Teda platí $2r_y^2x_i \leq 2r_x^2y_i$ a kandidáti na nasledujúci bod sú body na mieste $E = (x_i + 1, y_i)$ a $SE = (x_i + 1, y_i - 1)$ (Obr. 26.). Ako v oboch



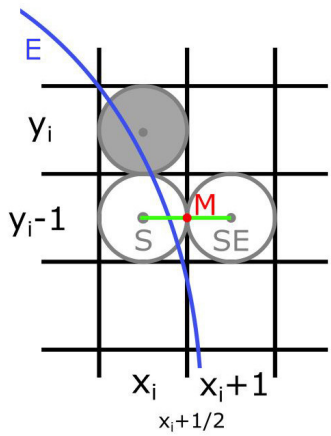
Obr. 26: Dvaja kandidáti na vykreslenie

Midpoint algoritmoch, použijeme bod $M = stred(E, SE) = (x_i + 1, y_i - \frac{1}{2})$. Vyjadríme si rozhodovací parameter pre prvú oblasť ako $p1_i = f(M) = f(x_i + 1, y_i - \frac{1}{2}) = r_y^2(x_i + 1)^2 + r_x^2(y_i - \frac{1}{2})^2 - r_x^2r_y^2$ a odvodíme rekurentný vzorec.
$p1_{i+1} = f(x_{i+1} + 1, y_{i+1} - \frac{1}{2}) = r_y^2\left(x_{i+1} + 1\right)^2 + r_x^2\left(y_{i+1} - \frac{1}{2}\right)^2 - r_x^2r_y^2$. Z toho
$p1_{i+1} = p1_i + 2r_y^2(x_i + 1) + r_y^2 + r_x^2\left((y_{i+1} - \frac{1}{2})^2 - (y_i - \frac{1}{2})^2\right)$.
Zrejme platí:

- Ak $p1_i < 0$, tak vyberáme za nasledujúci pixel $(x_{i+1}, y_{i+1}) = E = (x_i + 1, y_i)$ a $p1_{i+1} = p1_i + 2r_y^2x_{i+1} + r_y^2$.
- Ak $p1_i \geq 0$, tak vyberáme za nasledujúci pixel $(x_{i+1}, y_{i+1}) = SE = (x_i + 1, y_i - 1)$ a $p1_{i+1} = p1_i + 2r_y^2x_{i+1} + r_y^2 - 2r_x^2y_{i+1}$.

Potrebujeme ešte vyčísliť hodnotu začiatočného parametra v začiatočnej pozícii pre $(x_0, y_0) = (0, r_y)$, teda $p1_0 = f(1, r_y - \frac{1}{2}) = r_y^2 - r_x^2r_y + \frac{1}{4}r_x^2$

2. Ak $2r_y^2x_i \geq 2r_x^2y_i$ nachádzame sa v druhej oblasti. Teda kandidáti na nasledujúci bod sú body na mieste $S = (x_i, y_i - 1)$ a $SE = (x_i + 1, y_i - 1)$ (Obr. 27.).V tomto prípade použijeme bod $M = stred(S, SE) = (x_i + \frac{1}{2}, y_i - 1)$. Vyjadríme si rozhodovací parameter pre druhú



Obr. 27: Dvaja kandidáti na vykreslenie

oblasť ako $p2_i = f(M) = f(x_i + \frac{1}{2}, y_i - 1) = r_y^2(x_i + \frac{1}{2}) + r_x^2(y_i - 1) - r_x^2 r_y^2$ a odvodíme rekurentný vzorec.
$p2_{i+1} = f(x_{i+1} + \frac{1}{2}, y_{i+1} - 1) = r_y^2(x_{i+1} + \frac{1}{2})^2 + r_x^2\,(y_i - 1)^2 - r_x^2 r_y^2$. Z toho
$p2_{i+1} = p2_i + 2r_x^2(y_i - 1) + r_x^2 + r_y^2\left((x_{i+1} + \frac{1}{2})^2 - (x_i + \frac{1}{2})^2\right)$.
Zrejme platí:

- Ak $p2_i < 0$, tak vyberáme za nasledujúci pixel $(x_{i+1}, y_{i+1}) = S = (x_i, y_i - 1)$ a $p2_{i+1} = p2_i - 2r_x^2 y_{i+1} + r_x^2$.
- Ak $p2_i \geq 0$, tak vyberáme za nasledujúci pixel $(x_{i+1}, y_{i+1}) = SE = (x_i + 1, y_i - 1)$ a $p2_{i+1} = p2_i + 2r_y^2 x_{i+1} - 2r_x^2 y_{i+1} + r_x^2$.

Potrebujeme ešte vyčísliť hodnotu začiatočného parametra v poslednom bode z prvej oblasti, ktorý je vlastne bod (x_0, y_0) v oblasti dva. Teda $p2_0 = f(x_0 + \frac{1}{2}, y_0 - 1) = r_y^2(x_0 + \frac{1}{2})^2 + r_x^2(y_0 - 1)^2 - r_x^2 r_y^2$.

Pre zjednodušenie môžeme pixely v druhej oblasti I. kvadrantu vyčísľovať taktiež z bodu $(r_x, 0)$ proti smeru chodu hodinových ručičiek, až kým sa nedostaneme do posledného vyčísleného bodu prvej oblasti.

Tak ako v kružnicovom algoritme, pri výpočte $p1_{i+1}$ a $p2_{i+1}$ používame iba operácie sčitovania a odčitovania. Hodnoty $2r_y^2 x_{i+1}$ a $2r_x^2 y_{i+1}$ je možno počítať s pomocou konštantného prírastku. Pre hodnotu začiatočného parametra v bode $(x_0, y_0) = (0, r_y)$ platí $2r_y^2 x_0 = 0$ a $2r_x^2 y_0 = 2r_x^2 r_y$. Ako sa budú hodnoty x_i a y_i zvyšovať, budú sa tieto parametre meniť v I kvadrante takto:

- Ak $x_{i+1} > xi$ platí $2r_y^2 x_{i+1} = 2r_y^2 x_i + 2r_y^2$, inak $2r_y^2 x_{i+1} = 2r_y^2 x_i$.
- Ak $y_{i+1} < yi$ platí $2r_x^2 y_{i+1} = 2r_x^2 x_i - 2r_x^2$, inak $2r_x^2 y_{i+1} = 2r_x^2 y_i$.

Postup

Kroky algoritmu vieme zhrnúť nasledovne:

1. Zadaj r_x a r_y a stred elipsy (x_C, y_C) a urči prvý bod na elipse so stredom v začiatku $(x_0, y_0) = (0, r_y)$.
2. Vypočítaj začiatočnú hodnotu rozhodovacieho parametra v prvej oblasti ako $p1_0 = r_y^2 - r_x^2 r_y + \frac{1}{4} r_x^2$, urči $2r_y^2 x_0 a 2r_x^2 y_0$.
3. V každej pozícii x_i v prvej oblasti, štartujúc v $i = 0$, vykonaj nasledujúci test:
   (a) Ak $p1_i < 0$: nasledujúci bod pozdĺž elipsy so stredom $(0, 0)$ bude $(x_{i+1}, y_{i+1}) = (x_i + 1, y_i)$ a $p1_{i+1} = p1_i + 2r_y^2 x_{i+1} + r_y^2$.
   (b) Inak je nasledujúci bod $(x_{i+1}, y_{i+1}) = (x_i + 1, y_i - 1)$ a $p1_{i+1} = p1_i + 2r_y^2 x_{i+1} - 2r_x^2 y_{i+1} + r_y^2$.

   a urči $2r_y^2 x_{i+1} = 2r_y^2 x_i + 2r_y^2$ a $2r_x^2 y_{i+1} = 2r_x^2 y_i - 2r_x^2$
4. Opakuj krok 3. až pokým $2r_y^2 x \leq 2r_x^2 y$.
5. Urči začiatočnú hodnotu rozhodovacieho parametra v druhej oblasti využitím posledného bodu z prvej oblasti ako $p2_0 = r_y^2(x_0 + \frac{1}{2})^2 + r_x^2(y_0 - 1)^2 - r_x^2 r_y^2$
6. V každej pozícii x_i v druhej oblasti, štartujúc v $i = 0$, vykonaj nasledujúci test:
   (a) Ak $p2_i > 0$: nasledujúci bod pozdĺž elipsy so stredom $(0, 0)$ bude $(x_{i+1}, y_{i+1}) = (x_i, y_i - 1)$ a $p2_{i+1} = p2_i - 2r_x^2 y_{i+1} + r_x^2$.
   (b) Inak je nasledujúci bod $(x_{i+1}, y_{i+1}) = (x_i + 1, y_i - 1)$ a $p2_{i+1} = p2_i + 2r_y^2 x_{i+1} - 2r_x^2 y_{i+1} + r_x^2$.

   a urči $2r_y^2 x_{i+1} = 2r_y^2 x_i + 2r_y^2$ a $2r_x^2 y_{i+1} = 2r_x^2 y_i - 2r_x^2$.
7. Urči súmerné body v ostatných troch kvadrantoch.
8. Posuň každú vypočítanú pixlovú pozíciu (x, y) na elipsu so stredom (x_C, y_C) a zobraz bod so súradnicami: $x = x + x_C$,$y = y + y_C$.

Príklad

Na ilustráciu algoritmu si ukážeme rasterizáciu elipsy s parametrami $r_x = 8$ a $r_y = 6$ a stredom elipsy v bode $(0, 7)$

1. $(x_0, y_0) = (0, r_y) = (0, 6)$.
2. $p1_0 = -332$, $2r_y^2 x_0 = 0$ a $2r_x^2 y_0 = 2.8^2.6 = 768$. Výraz $2r_y^2 x$ sa bude zvyšovať o hodnotu $2r_y^2 = 72$ a výraz $2r_x^2 y$ o hodnotu $-2r_x^2 = -128$.
3. $i = 0 :\ p1_0 < 0 \Rightarrow (x_1, y_1) = (x_0 + 1, y_0) = (1, 6)$, $p1_1 = p1_0 + 2r_y^2 x_1 + r_y^2 = -224$, $2r_y^2 x_1 = 72$ a $2r_x^2 y_1 = 768$.

   $i = 1 :\ p1_1 < 0 \Rightarrow (x_2, y_2) = (x_1 + 1, y_1) = (2, 6)$, $p1_2 = p1_1 + 2r_y^2 x_2 + r_y^2 = -44$, $2r_y^2 x_2 = 72 + 72 = 144$ a $2r_x^2 y_2 = 768$.

   $i = 2 :\ p1_2 < 0 \Rightarrow (x_3, y_3) = (x_2 + 1, y_2) = (3, 6)$, $p1_3 = p1_2 + 2r_y^2 x_3 + r_y^2 = 208$, $2r_y^2 x_3 = 144 + 72 = 216$ a $2r_x^2 y_3 = 768$.

$i = 3:\ p1_3 > 0 \Rightarrow (x_4, y_4) = (x_3 + 1, y_3 - 1) = (4, 5), p1_4 = p1_3 + 2r_y^2 x_4 - 2r_x^2 y_4 + r_y^2 = -108$, $2r_y^2 x_4 = 216 + 72 = 288$ a $2r_x^2 y_4 = 768 - 128 = 640$.

$i = 4:\ p1_4 < 0 \Rightarrow (x_5, y_5) = (x_4 + 1, y_4) = (5, 5)$, $p1_5 = p1_4 + 2r_y^2 x_5 + r_y^2 = 288$, $2r_y^2 x_5 = 288 + 72 = 360$ a $2r_x^2 y_5 = 640$.

$i = 5:\ p1_5 > 0 \Rightarrow (x_6, y_6) = (x_5 + 1, y_5 - 1) = (6, 4), p1_6 = p1_5 + 2r_y^2 x_6 - 2r_x^2 y_6 + r_y^2 = 244$, $2r_y^2 x_6 = 360 + 72 = 432$ a $2r_x^2 y_6 = 640 - 128 = 512$.

$i = 6:\ p1_6 > 0 \Rightarrow (x_7, y_7) = (x_6 + 1, y_6 - 1) = (7, 3)$, $2r_y^2 x_7 = 432 + 72 = 504$ a $2r_x^2 y_7 = 512 - 128 = 384$. Platí $2r_y^2 x_7 \geq 2r_x^2 y_7$, preto prechádzame do druhej oblasti.

4. $(x_0, y_0) = (7, 3)$ a teda $p2_0 = r_y^2 (x_0 + \frac{1}{2})^2 + r_x^2 (y_0 - 1)^2 - r_x^2 r_y^2 = -151$.

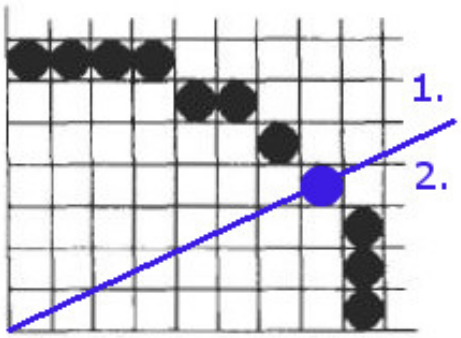
5. $i = 0:\ p2_0 < 0 \Rightarrow (x_1, y_1) = (x_0 + 1, y_0 - 1) = (8, 2), p2_1 = p2_0 + 2r_y^2 x_1 - 2r_x^2 y_1 + r_x^2 = 233$, $2r_y^2 x_1 = 504 + 72 = 576$ a $2r_x^2 y_1 = 384 - 128 = 256$.

   $i = 1:\ p2_1 > 0 \Rightarrow (x_2, y_2) = (x_1, y_1 - 1) = (8, 1)$, $p2_2 = p2_1 + 2r_x^2 y_2 + r_x^2 = 169$, $2r_y^2 x_2 = 576$ a $2r_x^2 y_2 = 256 - 128 = 128$.

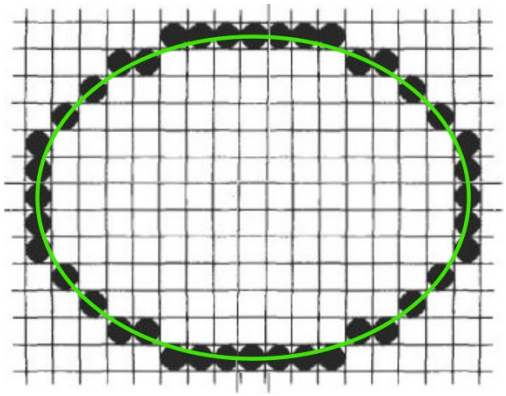
   $i = 2:\ p2_2 > 0 \Rightarrow (x_3, y_3) = (x_2, y_2 - 1) = (8, 0)$, algoritmus končí, pretože sme sa dostali na hranicu I. oktantu. Výsledné body možno vidieť na Obr. 28.

6. Určíme súmerné body v ostatných oktantoch. Výslednú elipsu možno vidieť na Obr. 29.

7. Posunieme každú hodnotu pixla do stredu $(0, 7)$. Teda dostaneme body: $(0, 13)$,$(1, 13)$,$(2, 13)$, $(3, 13)$, $(4, 12)$, ...



Obr. 28: Body elipsy v 1. kvadrante



Obr. 29: Výsledná elipsa

Pseudokód

Implementácia Bresenhamovho stredového elipsového algoritmu je zhrnutá v nasledujúcom pseudokóde. Na vstupe sú parametre r_x, r_y a stred elipsy(x_C, y_C).

```
ellipseMidpoint (int xC, int yC, int rx, int ry)
{
  int p;
  int x = 0;
  int y = ry;
  int px = 0;
  int py = 2 * rx * rx * y;
```

```
  void vykresliBodElipsy(int, int, int, int);

  /* Vykreslí prvý bod elipsy
  vykresliBodElipsy(xC, yC, x, y);

  /* Región 1
  p=ROUND(ry * ry -(rx * rx * ry) + (0.25 * rx * rx));
  while(px <py) {
    x++;
    px += 2 * ry * ry;
    if(p < 0) p += ry * ry + px;
    else {
      y--;
      py -= 2 * rx * rx;
      p += ry * ry + px - py;
    }
  vykresliBodElipsy(xC, yC, x, y);
  }

  /* Región 2
  p=ROUND(ry * ry * (x+0.5) * (x+0.5) + rx * rx * (y - 1) * (y - 1)
  - rx * rx * ry * ry);
  while(y > 0) {
    y--;
    py += 2 * rx * rx;
    if(p < 0) p += rx * rx + py;
    else {
      x++;
      px -= 2 * ry * ry;
      p += rx * rx + px;
    }
  vykresliBodElipsy(xC, yC, x, y);
  }
}

void vykresliBodElipsy(int xC, int yC, int x, int y);
{
  vykresliPixel(xC + x, yC + y);
  vykresliPixel(xC - x, yC + y);
  vykresliPixel(xC + x, yC - y);
  vykresliPixel(xC - x, yC - y);
}
```

Literatúra

Bresenhamov stredový elipsový algoritmus je podrobne spracovaný v [3] od strany 102. Je uvedený aj v [2] od strany 88, avšak je uvedený iba výsledný vzorec na výpočet bodov pozdĺž elipsy, bez odvodenia.

2 Literatúra

[1] **RUŽICKÝ, E., FERKO, A.**, 1995. *Počítačová grafika a spracovanie obrazu*, Bratislava: Sapientia, 1995. ISBN 80-967180-2-9. Dostupné na internete:< http://www.sccg.sk/ ferko/PGASO2012-bookmarks.pdf>.
[2] **ŽÁRA, J. et al.** 2004. *Moderní počítačová grafika*, druhé vydání 2004. Brno: Computer Press, 2004. ISBN 80-251-0454-0
[3] **HEARN, D., BAKER, M. P.**, 1997. *Computer graphics*, C version, second edition, USA: Prentice Hall, 1997, ISBN 0-13-578634-7
[4] **HEARN, D., BAKER, M. P.**, 2004. *Computer graphics with OpenGl*, third edition, USA: Prentice Hall, 2004, ISBN 0-13-120238-3
[5] **HUGHES, J., F. et al.** 2013. *Computer Graphics Principles and Fundamentals*. third edition, USA: Addison-Wesley. 2014, ISBN 0-132-39952-8
[6] **ZAŤKO, V.**, 2014. *Poznámky z prednášok Počítačová grafika (1): 2-MPG-101*. [online]. 04/2014. [cit. 2.1.2015]. Dostupné na internete:< http://flurry.dg.fmph.uniba.sk/webog/sk/zatko-vyucba/389-pocitacova-grafika-1.html>.
[7] **WATT, A.** 2000. *3D Computer Graphics*. third edition, USA: Addison-Wesley. 2000, ISBN 0-201-39855-9
[8] **BOŽEK, M.**, 2014. *Učebné texty ku predmetu Geometria (1) – Mnohouholníky.*
[9] **FOLEY, J. D., VAN DAM, A.**, 1982. *The Fundamentals of Interactive Computer Graphics*. first edition, USA: Addison-Wesley. 1982, ISBN 0-201-14468-9
[10] **RUŽICKÝ, E.**, 1991. *Úvod do počítačovej grafiky*, Bratislava: Polygrafické stredisko UK, 1991. ISBN 80-223-0375-5