Literatura

[1] Zajímavé matematické úlohy, Úloha 260. Matematika–fyzika–informatika, roč. 29 (2020), č. 1, s. 21.

[2] Zajímavé matematické úlohy, Úloha 260. Matematika-fyzika–informatika, roč. 29 (2020), č. 3, s. 198–200.

[3] VRML Virtual Reality Modeling Language. [online] 1995 cit. [2023-11-01]. Dostupné z: https://www.w3.org/MarkUp/VRML/

[4] The Web3D Consortium. [online] 2023 cit. [2023-11-01]. Dostupné z: https://www.web3d.org/

[5] Official x3dom documentation. [online] 1995 cit. [2023-11-01]. Dostupné z: https://doc.x3dom.org/

[6] Projektor. [online] 2023 cit. [2023-11-01]. Dostupné z: https://projektor.geometry.cz/

[7] Slepenec 5. [online] 2023 cit. [2023-11-01]. Dostupné z: https://projektor.geometry.cz/?co=show&did=358

[8] Slepenec 7. [online] 2023 cit. [2023-11-01]. Dostupné z: https://projektor.geometry.cz/?co=show&did=359

Počítačová grafika, 4. díl

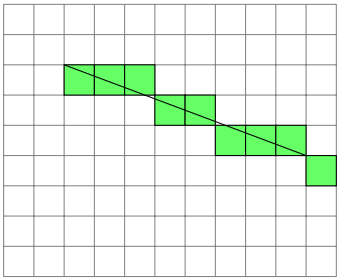
EDUARD BARTL

Přírodovědecká fakulta UP, Olomouc

Článek volně navazuje na předchozí tři díly, zabývá se rasterizací základních geometrických objektů. Podrobně se věnuje rasterizaci úsečky pomocí algoritmu DDA. Může sloužit jako pomůcka pro středoškolské učitele informatiky a výpočetní techniky.

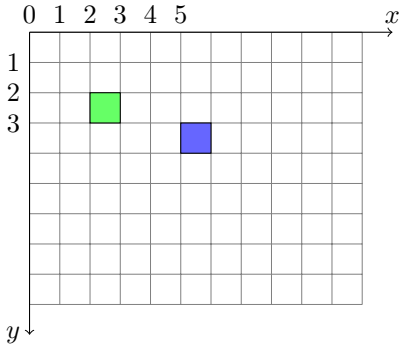
Rasterizací objektů rozumíme kreslení těchto objektů do rastru zobrazovacího zařízení, v našem případě se jedná o displej. Budeme mít k dispozici matematický popis nějakého základního geometrického objektu, například úsečky. Tento objekt je tvořen nekonečným množstvím bezrozměrných geometrických bodů (viz také první díl tohoto seriálu [1]) a naším úkolem je

ho co nejvěrněji zobrazit v rastru, který je však tvořen pouze konečným množstvím fyzických pixelů. Situace je znázorněná na obr. 1. Rasterizační algoritmy by měly navíc pracovat co nejefektivněji, to znamená, co nejrychleji s využitím malého množství paměti.



Obr. 1 Rasterizace úsečky

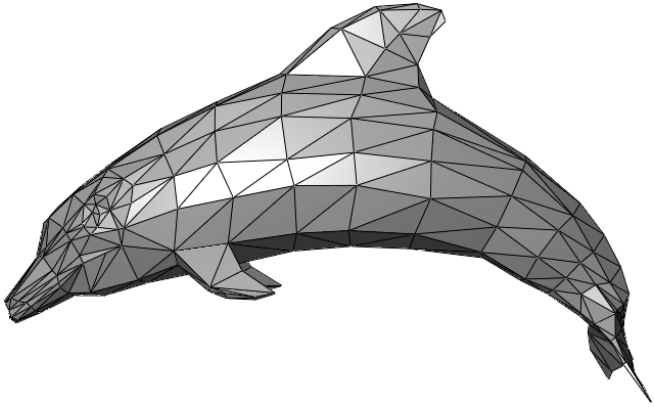
Pobavme se nejprve o souřadnicovém systému, se kterým budeme v dalším výkladu pracovat. Počátek souřadnicového systému umístíme do levého horního rohu zobrazovacího zařízení.[1)] Kladná polovina osy x tedy směřuje zleva doprava, kladná polovina osy y shora dolů. Předpokládáme dále, že pixely mají jednotkovou velikost. Logický pixel (viz [1]) umístíme do levého horního vrcholu pixelu – souřadnice daného fyzického pixelu jsou proto dány souřadnicemi jeho levého horního vrcholu, jak ukazuje obr. 2.



Obr. 2 Zelený pixel má souřadnice $\langle 2, 2\rangle$, modrý pixel má souřadnice $\langle 5, 3\rangle$

[1)] Toto nastavení je v počítačové grafice běžné.

Zabývat se budeme zejména *rasterizací úsečky*, složitější objekty totiž můžeme aproximovat lomenými čárami složenými z úseček. Stejně se tak děje dokonce i v trojrozměrné grafice. Trojrozměrné objekty se totiž reprezentují pomocí trojúhelníkové sítě, která je taktéž vykreslována pomocí úseček, příklad můžeme vidět na obr. 3.



Obr. 3 Model delfína reprezentovaný pomocí trojúhelníkové sítě (zdroj Wikipedia)

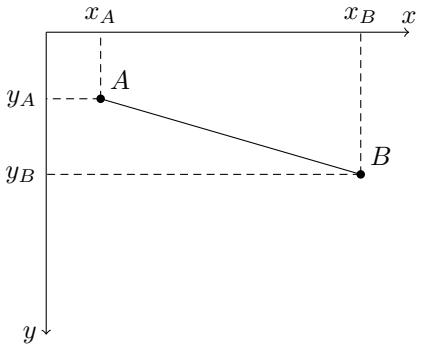
Výklad začneme základním algoritmem pro rasterizaci úsečky, v dalších dílech si pak ukážeme složitější algoritmy. Předpokládejme nejprve, že je úsečka, kterou chceme rasterizovat, určena koncovými body $A = \langle x_A, y_A \rangle$ a $B = \langle x_B, y_B \rangle$. Tyto souřadnice jsou celočíselné, přímo tedy udávají souřadnice prvního a posledního pixelu. Pro jednoduchost dále předpokládejme, že bod B bude ležet napravo a níže od bodu A, tedy $x_A < x_B$ a $y_A < y_B$, jak můžeme vidět na obr. 4.

Nejprve budeme muset najít matematickou funkci, která popisuje závislost y-ové souřadnice libovolného bodu této úsečky na jeho x-ové souřadnici. Rasterizaci začneme například v bodě A; po určitých krocích budeme měnit jednu souřadnici a s pomocí zmíněné funkce dopočítávat souřadnici druhou, dokud se nedostaneme do bodu B. Vypočtené hodnoty však budeme muset zaokrouhlit, abychom obdrželi celočíselné souřadnice pixelů.

Rozebereme si nyní tento postup podrobněji. Úsečku popíšeme pomocí *směrnicové rovnice přímky*, na které tato úsečka leží:

$$y = kx + q, \tag{1}$$

kde k je *směrnice* určující sklon přímky a q určuje průsečík přímky s osou y.



Obr. 4 Úsečka určená koncovými body A a B.

Směrnice přímky k je rovna tangentě úhlu, kterou tato přímka svírá s kladnou polovinou osy x, platí tedy:

$$k = \frac{y_B - y_A}{x_B - x_A}.$$

Rozdíl $y_B - y_A$ budeme v dalším textu značit Δy, rozdíl $x_B - x_A$ jako Δx.

Jak jsme již řekli, rasterizaci začneme v bodě A. První bod úsečky, který budeme zpracovávat proto bude mít souřadnice

$$x_1 = x_A, \qquad y_1 = y_A.$$

Souřadnice prvního pixelu, který touto rasterizací získáme, bude mít právě tyto souřadnice. V druhém kroku zvýšíme hodnotu x_1 o jedničku:[2)]

$$x_2 = x_1 + 1.$$

Ze vztahu (1) potom vypočteme příslušnou y-ovou souřadnici:

$$y_2 = kx_2 + q = k(x_1 + 1) + q = kx_1 + q + k \ = y_1 + k.$$

Souřadnice druhého pixelu pak získáme zaokrouhlením y-ové souřadnice:[3)]

$$\langle x_2, \mathrm{round}(y_2) \rangle.$$

[2)] Vzhledem k tomu, že fyzický pixel je čtverec jednotkové velikosti, přičtením jedničky se tak jistě ocitneme v pixelu, který leží napravo od daného pixelu.

[3)] x-ovou souřadnici zaokrouhlovat nemusíme, ta je totiž po celou dobu výpočtu celočíselná.

Zcela stejným způsobem pak pokračujeme v dalších krocích. Obecně tedy můžeme psát, že v i-tém kroku vypočítáme souřadnice bodu $\langle x_i, y_i \rangle$ pomocí souřadnic bodu z předchozího kroku $\langle x_{i-1}, y_{i-1} \rangle$:

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

Souřadnice příslušného pixelu pak získáme zaokrouhlením y-ové souřadnice:

$$\langle x_i, \mathrm{round}(y_i) \rangle.$$

Výpočet skončíme v okamžiku, kdy bude x-ová souřadnice větší než x_B. Tímto způsobem vypočteme souřadnice všech zelených pixelů, které jsou ukázány na obr. 1.

Právě odvozený postup se nazývá algoritmus DDA.[4)] Jedná se o nejjednodušší algoritmus rasterizující úsečku, pseudokód tohoto algoritmu je následující:

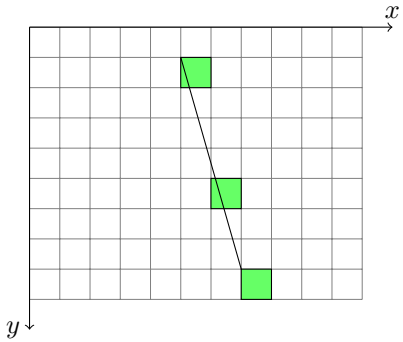
Algoritmus 1 Algoritmus DDA pro řídící osu x

Vstup: Body $A = \langle x_A, y_A \rangle$ a $B = \langle x_B, y_B \rangle$; všechny souřadnice jsou celočíselné.
Výstup: Souřadnice pixelů, které rasterizují úsečku danou koncovými body A a B.

```
k = Δy/Δx
⟨x, y⟩ = ⟨x_A, y_A⟩
while x ≤ x_B do
    obarvi pixel o souřadnicích ⟨x, round(y)⟩
    x = x + 1
    y = y + k
end while
```

Tento algoritmus má však jeden závažný háček. Uvažujme situaci zobrazenou na obr. 5. Rozdíl oproti předchozímu příkladu je zjevně v tom, že rasterizovaná úsečka má výrazně větší sklon. Popsaný postup tuto úsečku rasterizuje velmi špatný způsobem; podívejme se na zeleně vybarvené pixely na obr. 5, které tuto rasterizaci znázorňují. Z obrázku je patrné, že rasterizované pixely na sebe nenavazují, výsledek je proto velmi neuspokojivý.

[4)] DDA je zkratkou z anglického názvu *Digital Differencial Analyzer*.



Obr. 5 Rasterizace úsečky s velkým sklonem.

Je evidentní, že tento problém způsobuje právě velký sklon dané úsečky. Algoritmus 1 totiž využívá skutečnosti, že má úsečka malý sklon,[5)] což se do výpočtu promítlo neustálým zvyšováním x-ové souřadnice o jedničku. V případech, kdy je sklon úsečky malý, říkáme, že je osa x *řídící osou*.

Neustálé zvyšování x-ové souřadnice je však pro úsečky s větším sklonem nevhodné, způsobí totiž „roztržení“ pixelů, které jsme zaznamenali na obr. 5. V případech, kdy je řídící osou osa y (úsečka má velký sklon a přimyká se proto k ose y), bude vhodnější, když budeme zvyšovat o jedničku y-ovou souřadnici a dopočítávat ze vztahu (1) x-ovou souřadnici. Tento postup popisuje algoritmus 2.

Algoritmus 2 Algoritmus DDA pro řídící osu y

Vstup: Body $A = \langle x_A, y_A \rangle$ a $B = \langle x_B, y_B \rangle$; všechny souřadnice jsou celočíselné.

Výstup: Souřadnice pixelů, které rasterizují úsečku danou koncovými body A a B.

$k = \frac{\Delta x}{\Delta y}$
$\langle x, y \rangle = \langle x_A, y_A \rangle$
while $y \leq y_B$ **do**
  obarvi pixel o souřadnicích $\langle \mathrm{round}(x), y \rangle$
  $x = x + k$
  $y = y + 1$
end while

[5)] Úsečka se *přimyká* k ose x.

Před samotným výpočtem tedy musíme nejprve určit, která z os bude řídící. Učiníme tak jednoduše z hodnoty směrnice. Pokud bude totiž $k < 1$, pak se úsečka přimyká k ose x a tato osa je tedy řídící. Pokud však bude $k > 1$, úsečka se přimyká k ose y, řídící osou je tedy osa y. Pro mezní případ, $k = 1$, může být volba libovolná. Vytvoření obecného algoritmu DDA (pro libovolnou řídící osu) je pak přímočaré.

Příští díl seriálu bude opět věnovaný rasterizaci. Ukážeme si Bresenhamův rasterizační algoritmus a vysvětlíme si jeho výhody oproti algoritmu DDA. Dále si vysvětlíme, jakým způsobem se rasterizují jiné geometrické objekty (například kružnice nebo elipsa).

Literatura

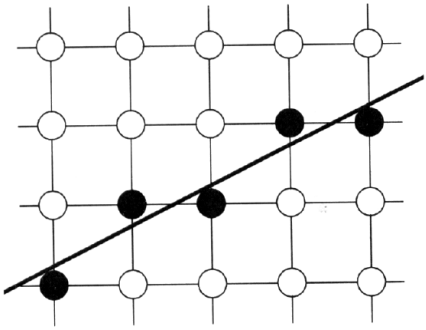
[1] *Bartl, E.*: Počítačová grafika I. Matematika-fyzika–informatika, roč. 29 (2020), č. 1, s. 138–148.

[2] *Felkel, P., Sochor, J., Žára, J., Beneš, B.*: Moderní počítačová grafika. 2. vydání. Computer Press, 2005.

[3] *Gonzalez, R. C., Woods, R. E.*: Digital Image Processing. 4. vydání. Pearson Prentice Hall, 2018.

[4] *Huges, J. F. a kol.*: Computer Graphics. Principles and Practice. 3. vydání. Addison-Wesley, 2014.

[5] *Martíšek, D.*: Matematické principy grafických systémů. Littera, 2002.



Rasterizace úsečky se sklonem $\leq 45°$, řídící osa x; zdroj: P. Strachota, Některé rastrové algoritmy pro vykreslování 2D objektů, FJFI ČVUT, 2012, [online] www.google.com/search?client=firefox-b-d&q=04.rastrove_algoritmy.pdf.