Kapitola 4

Rasterizace objektů

Rasterizace je proces při kterém se vektorově definovaná grafika konvertuje na rastrově definované obrazy. Při zobrazení reálného modelu ve světových souřadnicích na výstupní zařízení potřebujeme zajistit co nejvěrnější podobnost reálného a zobrazovaného modelu. Omezíme se v tomto textu na rastrové výstupní zařízení, jehož nejjednodušším grafickým prvkem je bod. Protože složitější objekty jsou jen skládankou jednodušších objektů, potřebujeme mít k dispozici algoritmy na výpočet polohy bodů jednoduchých objektů jako úsečky, kružnice, elipsy, oblouků, atd... Ukážeme si některé algoritmy pro výpočet bodů úsečky a kružnice.

4.1 Úsečka a lomenná čára

Úsečka je nejjednodušším grafickým prvkem. Jejich skládáním se dají vytvářet lomené čáry a další křivočaré útvary. Vykreslení úsečky by mělo být co nejefektivnější a nejrychlejší.

Úsečka lze zapsat dvěma způsoby:

1. souřadnicemi počátečního a koncového bodu $[x_1, y_1], [x_2, y_2]$,
2. pomocí počátečního bodu $[x_1, y_1]$ a směrového vektoru

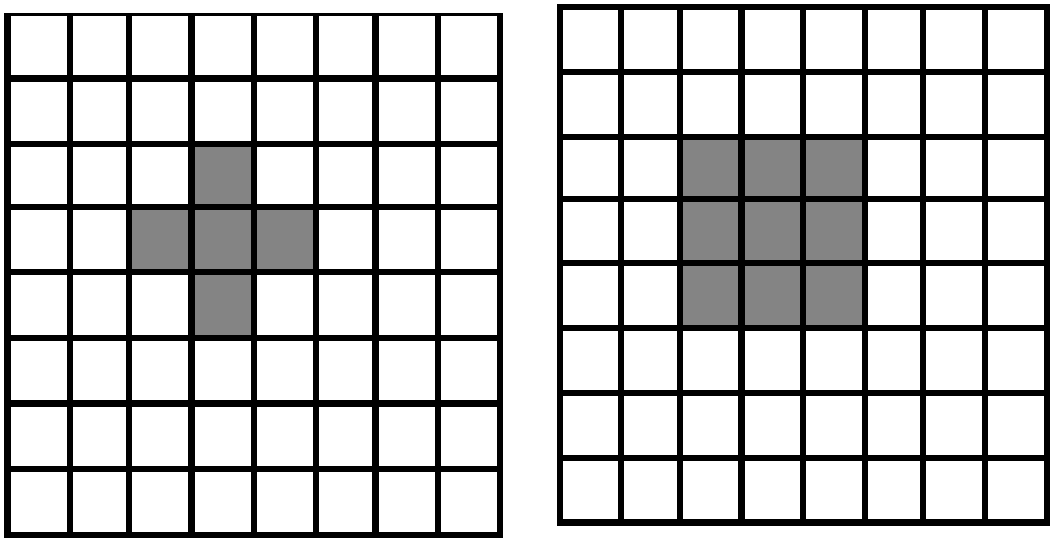
$$m = \frac{\Delta x}{\Delta y} = \frac{y_2 - y_1}{x_2 - x_1}$$

Podle velikost směrnice m se určí, vzhledem ke které ose se bude vzorkovat. Pro $|m| < 1$ je úsečka blíž k ose x a tedy bude vzorkována vzhledem k ní. Pro $|m| > 1$ je úsečka vzorkována na ose y a pro $|m| = 1$ může být řídící osa libovolná.

Rozlišujeme dva typy posloupnosti pixelů (obr. 4.1):

1. osmispojitá (8-connected) - každý pixel má osm sousedů čtyři ve vodorovném a svislém směru a čtyři úhlopříčně

2. čtyřspojitá (4-connected) - používá se zřídka, každý pixel má jen čtyři sousedy ve vodorovném s svislém směru.



Obrázek 4.1: 4-spojitý a 8-spojitý pixel

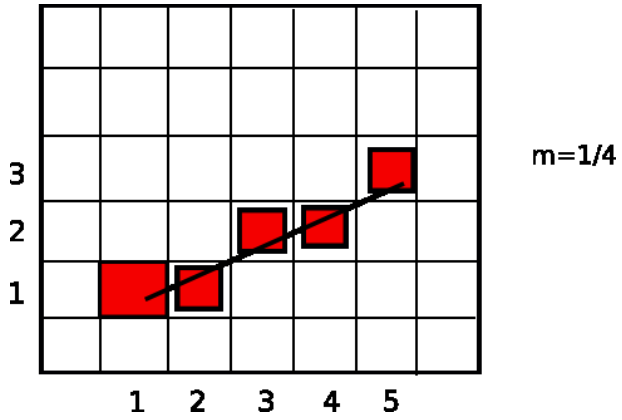
Vzhledem ke zvolenému typu pixelů se objekty vykreslují vybarvováním pixelů, které co nejvíce vystihují úsečku. Pro určení, které pixely budou vykresleny existuje několik algoritmů. My si ukážeme algoritmus DDA a Bresenhamův algoritmus pro kresbu úsečky.

4.1.1 Algoritmus DDA

Jedná se o přírůstkový algoritmus pro výpočet bodů úsečky. Postup spočívá v tom, že postupně zvedáme o jeden pixel hodnoty na x-ové souřadné ose a dopočítáváme odpovídající bod y. Jelikož jsou souřadnice pixelů vždy celá čísla, provede s zaokrouhlení hodnoty a pixel se vykreslí. Algoritmus je jednoduchý, ale relativně pomalý, protože pracuje v oboru reálných čísel.

Alogitmus pro rasterizaci ve směru osy x. Pro $|m| > 1$ je prováděna rasterizace ve směru osy y a v algoritmu se zamění operace s x a y.

1. Z koncových bodů $[x_1, y_1], [x_2, y_2]$ urči směrnici m podle vzorce výše.
2. Inicializuj bod $[x, y]$ hodnotou $[x_1, y_1]$.
3. Dokud je $x \leq x_2$ opakuj:
   (a) Vykresli bod [x, zaokrouhlené (y)]
   (b) $x = x + 1$
   (c) $y = y + m$

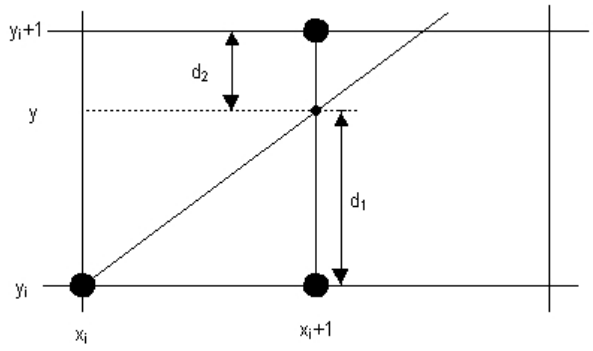


Obrázek 4.2: Vykreslení úsečky algoritmem DDA (rasterizace ve směru osy x)

4.1.2 Bresenhamův algoritmus pro kresbu úsečky

Tento algoritmus byl objeven v šedesátých letech minulého století panem Bresenhamem. Jeho výhoda je v tom, že pracuje pouze s celými čísly.

Víme, že od daného pixelu $[x, y]$ můžeme umístit další pixel (při rasterizace po ose x) pouze na dvou pozicích – $[x+1, y]$ nebo $[x+1, y+1]$. Rozdíly mezi souřadnicí y středů uvedených pixelů a skutečnou souřadnicí y označíme d_1, d_2 a můžeme je celočíselně vypočítat z parametrické rovnice kreslené úsečky. Jejich rozdíl potom vyhodnotíme. Pokud je kladný, je $d_1 > d_2$ a bližším pixelem je ten ve vzdálenosti d_2. Postup je názorně předveden na obr. 4.3.

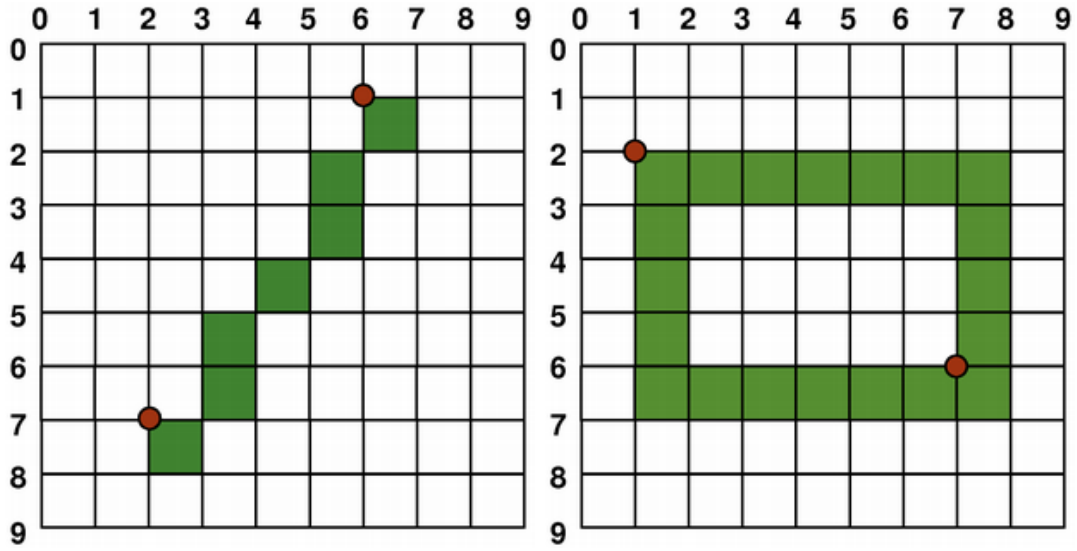


Obrázek 4.3: Princip Bresenhamova algoritmu

Na obr. 4.4 je ukázka rasterizace obecné úsečky a obdélníka.

4.1.3 Kresba silné a přerušované čáry

Silnou čáru můžeme kreslit pomocí dvou typů algoritmů. Jednoduchý postup je rychlý, ale dochází v něm k nepřesnostem. Jedná se o Bresenhamův algoritmus, kdy se v každém kroku vykresluje několik pixelů nad sebou či vedle sebe. Má však



Obrázek 4.4: dd

nevýhody. První z nich je, že se tloušťka čáry liší podle jejího sklonu (vzhledem k metrice rastru) a druhou nevýhodou je zakončení čar, které je rovnoběžné s některou souřadnou osou.

Přesný postup je však početně náročnější. Pracuje na principu vyplňování plochy, která určuje silnou čáru. Jako nejjednodušší tvar se používá obdélník, u něj je problém při napojování lomené čáry. Vhodnější je kresba pomocí obdélníků se zakulacenými rohy.

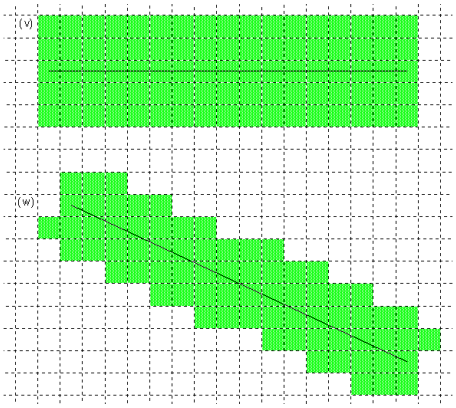
Pro kresbu přerušované čáry opět uvedeme dva algoritmy. První je založen na známém Bresenhamově algoritmu, kdy přidáme informaci o tom, zda jsme v úseku, který má být zobrazen či ne. Nevýhodou je opět metrika na rastrovém poli. Při různých sklonech úsečky budou dělící dílky nestejně dlouhé, což opticky působí nepřirozeně.

V praxi se většinou používá přesnější postup, založený na výpočtu délek dělících úseček. Potom se teprve použije známý algoritmus pro každý dílek samostatně. V závislosti na kvalitě programu je možné pozorovat různé zobrazení koncových dílku. Některé algritmy umí rozdělit úsečku tak, aby poslední dílek byl vždy viditelný.

4.2 Kružnice a elipsa

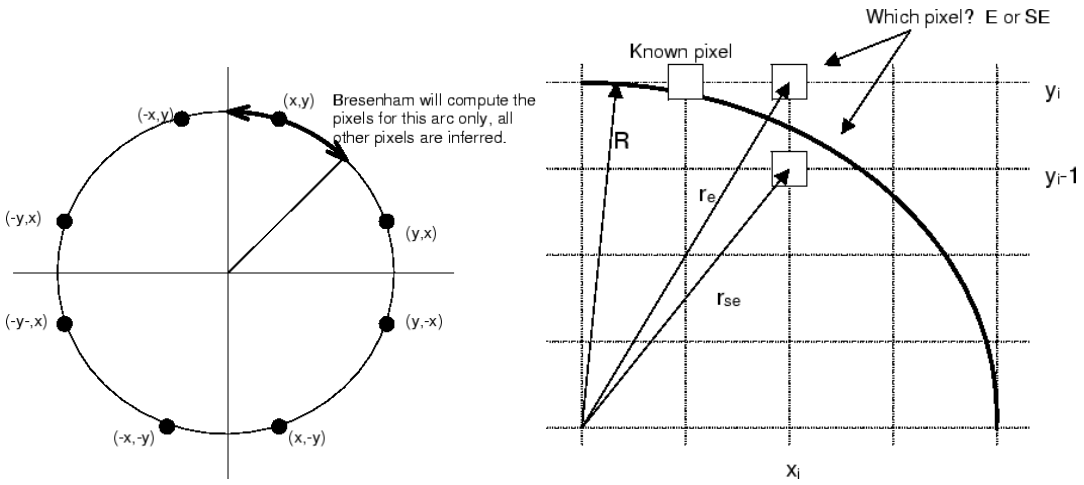
Kružnice se klasicky definuje středem a poloměrem. Elipsa je zase dána středem a velikostí hlavní a vedlejší poloosy. Otočením ji lze umístit do libovolné polohy. Při kresbě eliptických či kružnicových oblouků je navíc nutné definovat počáteční a koncový bod oblouku.

Při výpočtu bodů na kružnici stačí nalézt souřadnice oblouku 45 stupňů (obr. 4.6 vlevo), neboť všechny další body dostaneme záměnou souřadnic a změnou znamének.



Obrázek 4.5: Kresba silné čáry v rastru

Princip je opět na bázi Bresenhamova algoritmu pro úsečky. Jsme ve výchozím pixelu a ptáme se, kam teď můžu jít a testujeme, která pozice je blíž vykreslované kružnici (obr. 4.6 vpravo). Výhodu tohoto postupu je opět rychlost, neboť vždy pracujeme pouze s celými čísly.



Obrázek 4.6: Rasterizace kružnice

Elipsa se na rozdíl od kružnice musí vygenerovat pro celý kvadrant, tj. devadesát stupňů. Opět se použije Bresenhamův algortimus pouze s celočíselnou aritmetikou. Jediný rozdíl je ve tvaru testovací funkce, který z možných kandidátů na další bod leží blíž elipse.

4.3 Kontrolní otázky

1. Popište Bresenhamův algoritmus kreslení úsečky.
2. Co je algoritmus DDA?

3. Jaký je rozdíl mezi 8-spojitým a 4-spojitým pixelem.
4. Jaký je princip kreslení přerušované čáry?
5. Jaký je princip kreslení tlusté čáry?
6. Popište myšlenku kreslení kružnice.

4.4 Literatura

Bresenham algoritmus

`http://slady.cz/java/Bresenham/ (česky s Javou)`
`http://graphics.idav.ucdavis.edu/education/`
`GraphicsNotes/Bresenhams-Algorithm/Bresenhams-Algorithm.html`
`http://www.cs.helsinki.fi/group/goa/mallinnus/lines/bresenh.html`
`http://en.wikipedia.org/wiki/Bresenham's_line_algorithm`

Kresba tlusté čáry a kružnice

`http://www.acm.uiuc.edu/bug/Be%20Book/`
`The%20Interface%20Kit/3_CoordinateSpace.html`
`http://rich12345.tripod.com/circles/ // algoritmus pro kružnici`