# 4. VYPĹŇANIE

**Rozklad mnohouholníka**

Pod rozkladom mnohouholníka budeme rozumieť vykreslenie mnohouholníka zadaného vrcholmi na rastrovom zariadení. Ako prvý spôsob riešenia problému, by nás mohol napadnúť nasledovný postup: rasterizáciou strán mnohouholníka vytvoríme hranične zadanú oblasť a následne použijeme algoritmus na vypĺňanie oblasti. V tomto texte si uvedieme efektívnejšiu metódu.

**Scan Line algoritmus** – úvod

Scan Line algoritmus rozkladu mnohouholníka patrí do skupiny algoritmov, ktoré používajú princíp skenovacej priamky: ak je možné vyriešiť problém, úlohu z pohľadu jednej vodorovnej alebo zvislej priamky, urobí sa tak. Úlohu takejto priamky hrá často riadok alebo stĺpec pixlov rastra. Okrem zúženia pohľadu je druhou vlastnosťou skenovacích algoritmov využitie informácií z jedného kroku, v kroku nasledujúcom.

V našom algoritme bude skenovacia priamka vodorovná s celočíselnými *y*-súradnicami. Jej rovnica teda bude $y = y_i$, kde $y_i$ nadobúda celočíselné hodnoty od najmenšej po najväčšiu *y*-ovú súradnicu vrcholov mnohouholníka.

Scan Line algoritmus na rozklad mnohouholníka funguje tak, že v každom kroku zistí prieniky skenovacej priamky so stranami mnohouholníka, tieto prieniky usporiada do dvojíc a úseky medzi dvojicami vyfarbí.

Ak urobíme prienik skenovacej priamky so stranami mnohouholníka, chceme získať párny počet bodov. Preto musíme zo zoznamu strán mnohouholníka vylúčiť vodorovné strany ( tie sa vykreslia priamo) a ešte musíme skrátiť zdola strany v neextremálnych bodoch ( vo vrcholoch, z ktorých idú strany do opačných polrovín vzhľadom na skenovaciu priamku). Konkrétna realizácia tohto skrátenia bude zrejmá z inicializácie dátovej štruktúry, ktorú algoritmus používa.

**Dátové štruktúry pre Scan Line algoritmus**

Základným kameňom dátových štruktúr bude záznam o strane mnohouholníka. Tento záznam bude obsahovať tri čísla:

1. *maximálnu y- ovú súradnicu strany*; je to väčšia z *y*-ových súradníc vrcholov strany, toto číslo použijeme pri rozhodovaní, či je nutné zisťovať prienik strany so skenovacou priamkou
2. *x-ovú súradnicu bodu s minimálnou y-ovou súradnicou* ; toto číslo je vlastne *x*-ová súradnica bodu, v ktorom skenovacia priamka prvýkrát pretne stranu a v priebehu algoritmu je táto hodnota modifikovaná
3. *prevrátenú hodnotu smernice priamky, na ktorej strana leží* ; $\frac{1}{m} = \frac{x_A - x_B}{y_A - y_B}$, kde *A,B* sú vrcholy strany.

V predchádzajúcej časti sme hovorili o skrátení strany zdola v neextremálnom vrchole. Toto skrátenie zrealizujeme tak, že druhú hodnotu záznamu pre stranu budeme inicializovať nie na *x*-ovú súradnicu vrcholu strany, ktorý má menšiu *y*- ovú súradnicu, ale ak je tento vrchol neextremálny, na jeho *x*- ovú súradnicu zväčšenú o $\frac{1}{m}$. Pri tomto skrátení vychádzame z toho,

že ak na priamke so smernicou *m* leží bod so súradnicami $[x, y]$, tak na nej leží aj bod so súradnicami $\left[x+\frac{1}{m}, y+1\right]\left(m(x+\frac{1}{m})+q = mx+q+1 = y+1\right)$.

Algoritmus používa dve dátové štruktúry: tabuľku strán (v príklade hrán) a tabuľku aktívnych strán (v príklade hrán).

*Tabuľka strán (TS)* sa vytvára pri inicializácii a má riadky označené hodnotami, ktoré skenovacia priamka nadobudne v jednotlivých krokoch (celočíselné hodnoty od najmenšej po najväčšiu *y*- ovú súradnicu vrcholov mnohouholníka).

V jednotlivých riadkoch je zoznam záznamov strán, ktoré majú túto hodnotu ako svoju minimálnu *y*- ovú súradnicu. Ak je vrchol strany s menšou *y*- ovou súradnicou neextremálny, strana bude zapísaná až v ďalšom riadku TS (pretože je zdola skrátená).

Všimnime si, že až teraz, keď je strana zaradená do TS, máme o nej úplnú informáciu.

*Tabuľka aktívnych strán (TAS)* je zoznam strán, s ktorými má aktuálna skenovacia priamka prienik a vytvára sa počas behu algoritmu vyberaním záznamov o stranách z TS.
V tejto tabuľke sa druhá položka záznamu o každej strane mení tak, aby vždy mala hodnotu *x*-ovej súradnice prieniku strany a skenovacej priamky.

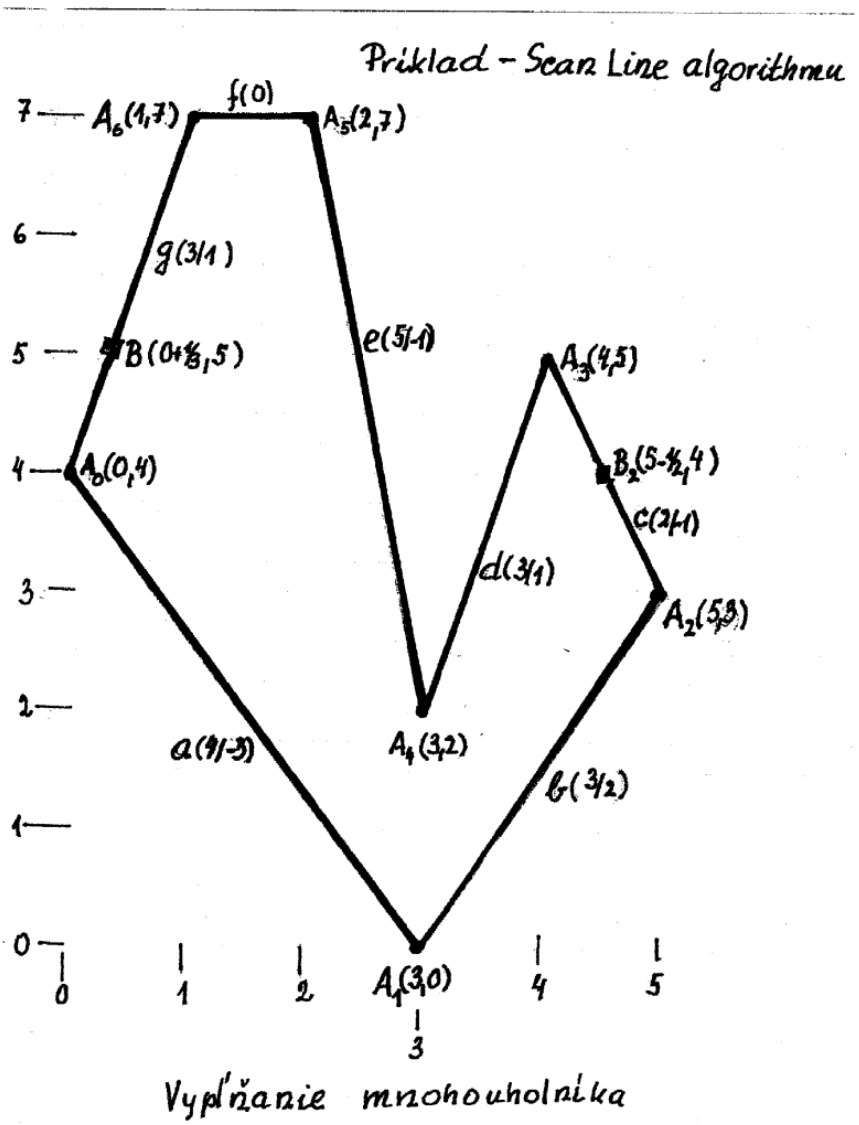
**Scan Line algoritmus**

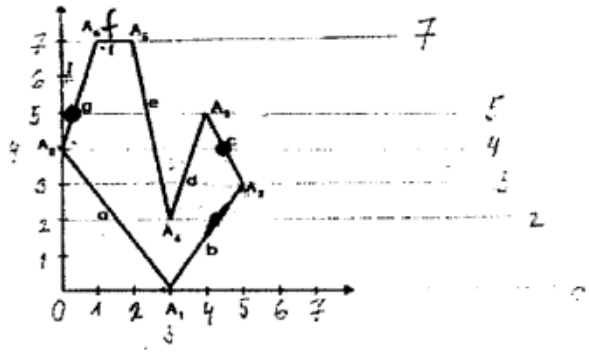
Na vstupe algoritmu nech je usporiadaný zoznam vrcholov mnohouholníka s celočíselnými súradnicami. Na výstupe algoritmu má byť tento mnohouholník rozložený do rastra t.j. v rastri majú byť vyfarbené príslušné obrazové body.

1. zisti, ktoré strany mnohouholníka sú vodorovné, ktoré vrcholy neextremálne
2. strany, ktoré nie sú vodorovné zapíš do TS, TAS inicializuj ako prázdnu, $y := y_{\min}$
3. kým TS alebo TAS sú neprázdne opakuj
   - vyber TS strany v riadku *y* a daj ich do TAS
   - usporiadaj strany v TAS podľa *x*-ovej súradnice (druhá položka v jednotlivých záznamoch)
   - vyber za sebou idúce úseky a vykresli ich
   - zruš tie strany v TAS, pre ktoré $y_{\max} = y$
   - pre strany v TAS zmeň *x* na $x+\frac{1}{m}$
   - $y := y+1$

**Príklad Scan line algoritmu**

Uvedieme ešte príklad. Pre mnohouholník na obrázku opíšeme TS (v ďalšom texte je TH, hrana) a pre jednotlivé kroky aj TAS (v ďalšom texte je TAH) a výstup algoritmu, jednotlivé vykresľované úseky.





Obrázok 1: Príklad mnohouholníka

## Príklad Scan Line algoritmu

Uveďme ešte príklad. Pre mnohouholník na obrázku, popíšeme TH a pre jednotlivé kroky aj TAH a výstup algoritmu, jednotlivé vykresľované úseky.

Mnohouholník na obrázku má jednu vodorovnú hranu $f$, tú do $TH$ nezaradíme a dva neextremálne vrcholy $A_0, A_2$.

Tabuľka hrán vyzerá takto:

0 $a$ | 4 | 3 | $-\frac{3}{4}$ | $b$ | 3 | 3 | $\frac{2}{3}$ |

1

2 $d$ | 5 | 3 | $\frac{1}{3}$ | $e$ | 7 | 3 | $-\frac{1}{3}$ |

3

4 $c$ | 5 | $5-\frac{1}{2}$ | $-\frac{1}{2}$ |

5 $g$ | 7 | $0+\frac{1}{3}$ | $\frac{1}{3}$ |

6

7

Uvedieme teraz jednotlivé iterácie kroku 3 algoritmu (začíname s $y = 0$):

$y=$ 0.
- $TAH$ : $a$ | 4 | 3 | $-\frac{3}{4}$ |, $b$ | 3 | 3 | $\frac{2}{3}$ |
- $TAH$ sa po usporiadení nezmení
- vykreslí sa úsek $[3,0] - [3,0]$, t.j. pixel $[3,0]$
- z $TAH$ sa žiadna hrana nezruší
- $TAH$ : $a$ | 4 | $\frac{9}{4}$ | $-\frac{3}{4}$ |, $b$ | 3 | $\frac{11}{3}$ | $\frac{2}{3}$ |
- $y := 1$

$y=$ 1.
- $TAH$ sa nezmení (žiadna hrana sa nepridá)

- $TAH$ sa po usporiadení nezmení
- vykreslí sa úsek $[\frac{9}{4}, 1] - [\frac{11}{3}, 1]$, t.j. pixle $[2,1], [3,1], [4,1]$
- z $TAH$ sa žiadna hrana nezruší
- $TAH$ : $a\,|\,4\,|\,\frac{6}{4}\,|\,-\frac{3}{4}\,|$, $b\,|\,3\,|\,\frac{13}{3}\,|\,\frac{2}{3}\,|$
- $y := 2$

2.
- $TAH$ : $a\,|\,4\,|\,\frac{6}{4}\,|\,-\frac{3}{4}\,|$, $b\,|\,3\,|\,\frac{13}{3}\,|\,\frac{2}{3}\,|$, $d\,|\,5\,|\,3\,|\,\frac{1}{3}\,|$, $e\,|\,7\,|\,3\,|\,-\frac{1}{5}\,|$
- $TAH$ : $a\,|\,4\,|\,\frac{6}{4}\,|\,-\frac{3}{4}\,|$, $d\,|\,5\,|\,3\,|\,\frac{1}{3}\,|$, $e\,|\,7\,|\,3\,|\,-\frac{1}{5}\,|$, $b\,|\,3\,|\,\frac{13}{3}\,|\,\frac{2}{3}\,|$
- vykreslia sa úseky $[\frac{6}{4}, 2] - [3, 2]$ a $[3, 2] - [\frac{13}{3}, 2]$, t.j. pixle $[2,2], [3,2], [4,2]$
- z $TAH$ sa žiadna hrana nezruší
- $TAH$ : $a\,|\,4\,|\,\frac{3}{4}\,|\,-\frac{3}{4}\,|$, $d\,|\,5\,|\,\frac{10}{3}\,|\,\frac{1}{3}\,|$, $e\,|\,7\,|\,\frac{14}{5}\,|\,-\frac{1}{5}\,|$, $b\,|\,3\,|\,\frac{13}{3}\,|\,\frac{2}{3}\,|$
- $y := 3$

3.
- $TAH$ sa nezmení (žiadna hrana sa nepridá)
- $TAH$ : $a\,|\,4\,|\,\frac{3}{4}\,|\,-\frac{3}{4}\,|$, $e\,|\,7\,|\,\frac{14}{5}\,|\,-\frac{1}{5}\,|$, $d\,|\,5\,|\,\frac{10}{3}\,|\,\frac{1}{3}\,|$, $b\,|\,3\,|\,\frac{13}{3}\,|\,\frac{2}{3}\,|$
- vykreslia sa úseky $[\frac{3}{4}, 3] - [\frac{14}{5}, 3]$ a $[\frac{10}{3}, 3] - [\frac{13}{3}, 3]$, t.j. pixle $[1,3], [2,3], [3,3], [4,3], [5,3]$
- $TAH$ : $a\,|\,4\,|\,\frac{3}{4}\,|\,-\frac{3}{4}\,|$, $e\,|\,7\,|\,\frac{14}{5}\,|\,-\frac{1}{5}\,|$, $d\,|\,5\,|\,\frac{10}{3}\,|\,\frac{1}{3}\,|$
- $TAH$ : $a\,|\,4\,|\,0\,|\,-\frac{3}{4}\,|$, $e\,|\,7\,|\,\frac{13}{5}\,|\,-\frac{1}{5}\,|$, $d\,|\,5\,|\,\frac{11}{3}\,|\,\frac{1}{3}\,|$
- $y := 4$

4.
- $TAH$ : $a\,|\,4\,|\,0\,|\,-\frac{3}{4}\,|$, $e\,|\,7\,|\,\frac{13}{5}\,|\,-\frac{1}{5}\,|$, $d\,|\,5\,|\,\frac{11}{3}\,|\,\frac{1}{3}\,|$, $c\,|\,5\,|\,\frac{9}{2}\,|\,-\frac{1}{2}\,|$
- $TAH$ sa po usporiadění nezmení
- vykreslia sa úseky $[0, 4] - [\frac{13}{5}, 4]$ a $[\frac{11}{3}, 4] - [\frac{9}{2}, 4]$, t.j. pixle $[0,4], [1,4], [2,4], [3,4], [4,4], [5,4]$
- $TAH$ : $e\,|\,7\,|\,\frac{13}{5}\,|\,-\frac{1}{5}\,|$, $d\,|\,5\,|\,\frac{11}{3}\,|\,\frac{1}{3}\,|$, $c\,|\,5\,|\,\frac{9}{2}\,|\,-\frac{1}{2}\,|$
- $TAH$ : $e\,|\,7\,|\,\frac{12}{5}\,|\,-\frac{1}{5}\,|$, $d\,|\,5\,|\,4\,|\,\frac{1}{3}\,|$, $c\,|\,5\,|\,4\,|\,-\frac{1}{2}\,|$
- $y := 5$

5.
- $TAH$ : $e\,|\,7\,|\,\frac{12}{5}\,|\,-\frac{1}{5}\,|$, $d\,|\,5\,|\,4\,|\,\frac{1}{3}\,|$, $c\,|\,5\,|\,4\,|\,-\frac{1}{2}\,|$, $g\,|\,7\,|\,\frac{1}{3}\,|\,\frac{1}{3}\,|$
- $TAH$ : $g\,|\,7\,|\,\frac{1}{3}\,|\,\frac{1}{3}\,|$, $e\,|\,7\,|\,\frac{12}{5}\,|\,-\frac{1}{5}\,|$, $d\,|\,5\,|\,4\,|\,\frac{1}{3}\,|$, $c\,|\,5\,|\,4\,|\,-\frac{1}{2}\,|$
- vykreslia sa úseky $[\frac{1}{3}, 5] - [\frac{12}{5}, 5]$ a $[4, 5] - [4, 5]$, t.j. pixle $[0,5], [1,5], [2,5], [4,5]$
- $TAH$ : $g\,|\,7\,|\,\frac{1}{3}\,|\,\frac{1}{3}\,|$, $e\,|\,7\,|\,\frac{12}{5}\,|\,-\frac{1}{5}\,|$
- $TAH$ : $g\,|\,7\,|\,\frac{2}{3}\,|\,\frac{1}{3}\,|$, $e\,|\,7\,|\,\frac{11}{5}\,|\,-\frac{1}{5}\,|$
- $y := 6$

6.
- $TAH$ sa nezmení (žiadna hrana sa nepridá)
- $TAH$ sa po usporiadení nezmení
- vykreslí sa úsek $[\frac{2}{3}, 6] - [\frac{11}{5}, 6]$, t.j. pixle $[1,6], [2,6]$
- z $TAH$ sa žiadna hrana nezruší

- prenos : g | 7 | 4/3 | 1/3 | , e | 7 | 11/5 | -1/5 |
- $TAH$ : g | 7 | 1 | $\frac{1}{3}$ |, e | 7 | 2 | $-\frac{1}{5}$ |
- $y := 7$

7.
- $TAH$ sa nezmení (žiadna hrana sa nepridá)
- $TAH$ sa po usporiadení nezmení
- vykreslí sa úsek $[1,7]-[2,7]$, t.j. pixle $[1,7],[2,7]$
- $TAH = \emptyset$

## Použitá literatúra


1. Ružický, E. - Ferko, A. 1995. Počítačová grafika a spracovanie obrazu. 1995. ISBN 80-967180-2-9. Bratislava: SAPIENTIA 1995.
2. Žára, J. a kol. 1998. Moderní počítačová grafika. ISBN 80-7226-049-9. Computer Press 1998
3. Juraj Štugel, www.pg.miesto.sk


## Poznámka

V skenovacom algoritme sa okrem iného rozkladajú do rastra aj strany mnohouholníka, pričom sa pri výbere pixlov na vykreslenie používa všeobecne známe zaokrúhľovanie. Pravda, okrem takého zaokrúhľovania sa v informatike používa aj „odsekávanie“ celočíselnej časti. Jeho použitie vedie k rasterizácii úsečiek – strán, ktorá je však menej presná ako predchádzajúca uvedená.

Porovnajme tieto dve možnosti rasterizácie úsečky *AB*, kde *A* = [0,0], *B* = [3,7]. Je zrejmé, že pre smernicu tejto úsečky platí: $m = \frac{7-0}{3-0} = \frac{7}{3} = \frac{dx}{dy}$.

Teda pre túto úsečku je *dx* = 3 a *dy* = 7.

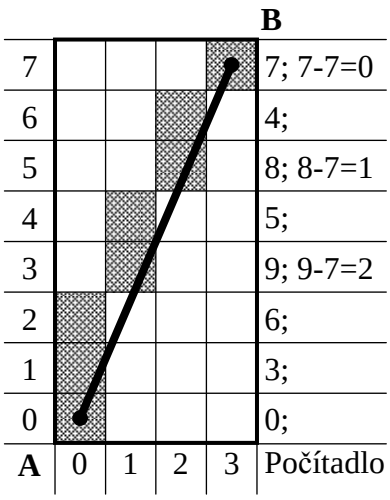
Pri riešení úlohy využijeme tzv. počítadlo P, ktoré inicializujeme na 0 a na sceločíselňovanie výsledkov začneme odsekávaním neceločíselných častí:

Ak $[x_k, y_k], [x_{k+1}, y_{k+1}]$ sú dva body úsečky na susedných skenovacích priamkach, tak platí:

$$m = \frac{y_k + 1 - y_k}{x_{k+1} - x_k} \Rightarrow x_{k+1} = x_k + \frac{1}{m} \text{ t.j. } \boxed{x_{k+1} = x_k + \frac{dx}{dy}} \qquad (1)$$

Použitím rovnosti (1), možno vďaka počítadlu realizovať celočíselné výpočty *x*. Počítadlo pri prechode k nasledujúcej skenovacej priamke inkrementujeme o hodnotu $\Delta x$ a vždy keď

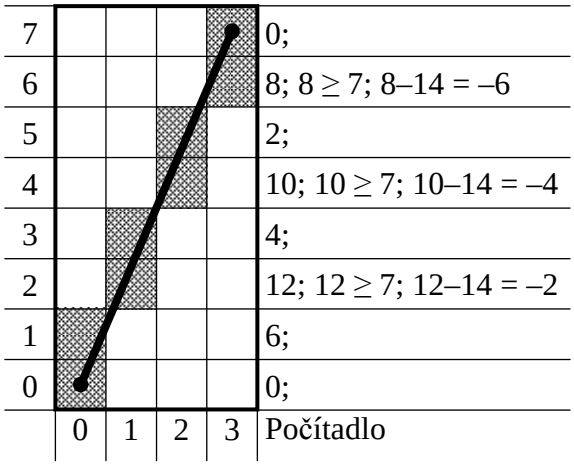
dosiahne hodnotu $\geq \Delta y$, inkrementujeme hodnotu *x* o 1 a znížime stav počítadla o hodnotu $\Delta y$. (Proces pokračuje dovtedy, pokiaľ stav počítadla nebude opäť 0). Teda:
$\Delta x = 3, \Delta y = 7$




Ako sme už povedali, táto procedúra je ekvivalentná uchovávaniu hodnoty *x* na istom celom čísle dovtedy, pokiaľ proces inkrementácie zlomkovej časti nedosiahne najbližšiu celočíselnú hodnotu. Ako vidno z obrázka, táto rasterizácia nie je príliš korektná.

Zlepšiť ju možno tým, že namiesto toho, aby sme celočíselné pozície dosahovali odsekávaním desatinných častí, budeme ich zaokrúhľovať na najbližšie celé číslo. Možno to dosiahnuť tak, že v opísanom postupe budeme stav počítadla na jednotlivých skenovacích priamkach porovnávať s hodnotou $\Delta y / 2$ (namiesto $\Delta y$). Konkrétne to možno zabezpečiť celočíselnou aritmetikou cez postupnú inkrementáciu počítadla hodnotou $2\Delta x$ a porovnávaním jeho aktuálneho stavu s hodnotou $\Delta y$.

Ak je prírastok väčší alebo sa rovná $\Delta y$, zvýšime aktuálnu hodnotu *x* o 1 a znížime stav počítadla o hodnotu $2\Delta y$. Proces iteratívne opakujeme dovtedy, pokiaľ stav počítadla nebude 0. Teda:
$2\Delta x = 6, \Delta y = 7, 2\Delta y = 14$.

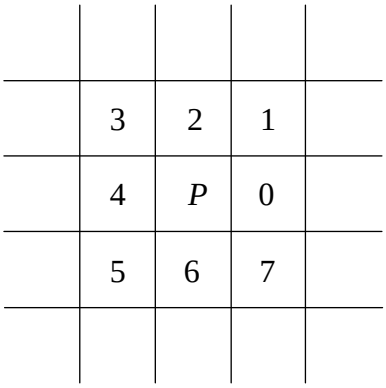


## Susednosť, vypĺňanie oblastí

Ako prvý pojem zavedieme pojem *bitová mapa*. Je to časť pamäte počítača, predstavme si ju ako štvorcovú sieť, ktorej jednotlivé prvky zodpovedajú obrazovým bodom na rastrovom zariadení. Ak je v niektorom prvku bitovej mapy zapísaná hodnota, tak je príslušný pixel rastra vysvietený farbou prislúchajúcou tejto hodnote. Pri vykresľovaní úsečky sa pre vykreslenie jednotlivých bodov volá funkcia, ktorá na určené miesto bitovej mapy zapíše hodnotu farby. V našom texte bude mať táto funkcia meno **write_pixel** a bude mať tri celočíselné parametre: súradnice a farbu. Funkcia **read_pixel** (s parametrami *x, y*) zase vráti farbu zadaného bodu.

V bitovej mape má každý vnútorný bod ôsmich susedov. Je zvykom označovať ich spôsobom, ako je uvedené na obrázku 1. Susedov s číslami 0, 2, 4, 6 nazývame *priami susedia*, alebo *4-susedia*, susedov s číslami 1, 3, 5, 7 nazývame *nepriami susedia*. Všetkých spolu nazývame *8-susedia*.

V obrázkoch v tejto prednáške budeme štvorcovou sieťou znázorňovať bitovú mapu.




Obr. 1: Susednosť v bitovej mape, resp. rastri

Oblasť je množina prvkov bitovej mapy, ktoré sú (dajú sa) súvislo pospájané. Podľa typu súvislosti rozdeľujeme oblasti na 4-súvislé a 8-súvislé. 4-súvislá oblasť je taká, ktorej každé dva prvky môžeme spojiť postupnosťou 4-susedov. 8-súvislá oblasť je taká, ktorej každé dva prvky môžeme spojiť postupnosťou 8-susedov. Prvkom oblasti hovoríme aj body.

Oblasti môžu byť zadané všetkými svojimi bodmi (používa sa aj pojem vnútorne definovaná oblasť) je zadaná farba týchto bodov a typ súvislosti. Pri oblasti danej hranicou je zadaná farba hranice.

Uvedieme dva základné algoritmy pre vypĺňanie oblastí a pre oblasť danú hranicou uvedieme neskôr ešte jeden vylepšený algoritmus.

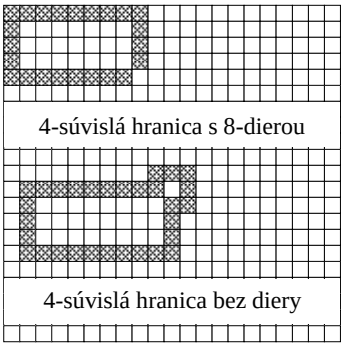
### Algoritmus vypĺňania vnútorne definovanej oblasti

Na vstupe nech je 4-súvislá oblasť daná všetkými svojimi bodmi s farbou old_color. Na výstupe má byť tá istá oblasť zafarbená farbou new_color. Procedúra, ktorá oblasť vyfarbí má okrem farieb ešte dva parametre, súradnice jedného vnútorného bodu oblasti. Uvedieme zápis procedúry v programovacom jazyku PASCAL.

```
procedure Flood_fill_4(x,y,old_color,new_color:integer)
    begin
        if read_pixel(x,y)=old_color then
            begin
                write_pixel(x,y,new_color);
                Flood_fill_4(x,y-1,old_color,new_color);
                Flood_fill_4(x,y+1,old_color,new_color);
                Flood_fill_4(x-1,y,old_color,new_color);
                Flood_fill_4(x+1,y,old_color,new_color);
            end;
    end;
```

**Algoritmus vypĺňania do hraničných bodov**

Na vstupe nech je 4-súvislá hranica farby bound_color. V tomto algoritme potrebujeme hranicu, ktorá je 4-súvislá v silnejšom zmysle ako sme definovali v úvode. Hranica nesmie mať 8-dieru (obr.2). Na výstupe má byť oblasť ohraničená touto hranicou a zafarbená farbou new_color. Parametre *x*, *y* sú opäť súradnice vnútorného bodu oblasti.




Obr.2: 4-súvislá hranica

```
procedure Bound_fill_8(x,y,bound_color,new_color:integer)
    begin
        if read_pixel(x,y)< >bound_color and read_pixel(x,y)< > new_color then
            begin
                write_pixel(x,y,new_color);
                Bound_fill_8(x,y-1,bound_color,new_color);
                Bound_fill_8(x,y+1,bound_color,new_color);
                Bound_fill_8(x-1,y,bound_color,new_color);
                Bound_fill_8(x+1,y,bound_color,new_color);
                Bound_fill_8(x-1,y-1,bound_color,new_color);
                Bound_fill_8(x+1,y-1,bound_color,new_color);
                Bound_fill_8(x-1,y+1,bound_color,new_color);
                Bound_fill_8(x+1,y+1,bound_color,new_color);
            end;
    end;
```

## Vypĺňanie oblastí

V mnohých prípadoch nie je potrebné vykresliť jedinú krivku, ale vyplniť určitú oblasť. Tento prípad sme už riešili pomocou algoritmu skenovacej priamky. Vo všeobecnosti existujú dve možnosti pre definovanie oblasti, ktorá sa má vyplniť. Jedna je mnohouholníkovo orientovaná – oblasť sa zadáva vrcholmi svojho hraničného mnohouholníka (tak to bolo u scan-line), druhá je rastrovo založená, a takou sa budeme teraz zaoberať. V tomto prípade hranica oblasti je „vyznačená“ množinou bodov rastra- pixlov a je potrebné vyplniť všetko vo vnútri oblasti.

Intuitívny prístup, ktorý nás napadne, je chápať zadanú množinu hraničných pixlov ako definitorickú pre hranicu určitej oblasti, podobne ako je to s bodmi uzavretej krivky v rovine. Takáto množina pixlov by mala byť súvislá a podobne ako uzavretá rovinná krivka by mala rozdeľovať nekonečnú mriežku na dve časti: vnútro a vonkajšok. Vyžaduje si to však viacero spresnení.

- 4-susedmi pixla *p* sú pixle, ktoré sa nachádzajú v rastri nad ním, pod ním a vpravo i vľavo od neho
- 8-susedmi pixla *p* sú okrem jeho 4-susedov aj najbližšie štyri diagonálne pixle.

Teraz už možno definovať súvislosť nasledujúcimi dvoma spôsobmi:

- **4-súvislosť:**

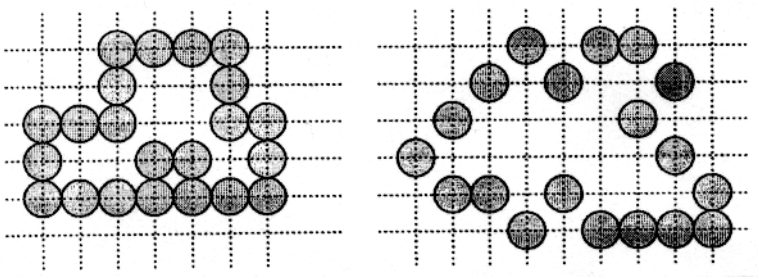
Množina pixlov je 4-súvislá, ak k ľubovoľným dvom jej pixlom existuje postupnosť pixlov patriacich tejto množine, ktorá začína v jednom pixli, končí v druhom a každé dva po sebe idúce pixle sú 4-susedia .

- **8-súvislosť:**

má rovnakú definíciu ako 4-súvislosť, len slová: 4-súvislá a 4-susedia treba nahradiť : 8-súvislá a 8-susedia.

Je zrejmé, že každá 4-súvislá množina je 8-súvislá, ale naopak to neplatí.

Podľa známej Jordanovej vety platí, že každá uzavretá krivka v rovine rozdeľuje rovinu na dve súvislé oblasti: vnútro a vonkajšok. Zatiaľ však nemáme definované, čo sa rozumie uzavretou krivkou v našom prípade, dokonca nám bez tejto definície môžu nastať určité problémy.



Obr.1: 4-súvislá (vľavo) a 8-súvislá (vpravo) množina pixlov

Všimnime si obr. 1. Ak hraničná krivka je 8-súvislá, tak vo všeobecnosti neplatí, že rozdeľuje nekonečnú sieť do dvoch 8-súvislých oblastí, pretože (ako vidieť na obr.1) vnútro a vonkajšok sa dajú navzájom spojiť 8-súvislou cestou. Existuje však zaujímavý spôsob, ako upraviť tento problém. Špeciálne, ak požadujeme, aby hraničná krivka bola 8-súvislá, tak je potrebné, aby ňou definovaná oblasť bola 4-súvislá. Podobne, ak chceme, aby hranica bola 4-súvislá je bežné požadovať, aby oblasť, ktorú definuje bola 8-súvislá.

## Semienkové vypĺňanie

- V algoritmoch tohto typu sa predpokladá, že poznáme aspoň jeden bod v súvislej oblasti. Pri vypĺňaní prechádzajú algoritmy všetkými pixlami súvislej oblasti a farbia ich farbou, ktorou sa má táto oblasť zafarbiť (tzv. nová farba).
- Ako sme už povedali, oblasť tvorená diskrétnymi bodmi rastrovej mriežky je 4-súvislá [8-súvislá], ak medzi každými dvomi jej vnútornými bodmi existuje cesta zložená len z horizontálnych a vertikálnych [horizontálnych, vertikálnych a diagonálnych] krokov po jednotlivých vnútorných bodoch oblasti.
- Je zrejmé, že požiadavka na 4-súvislosť je prirodzenejšia ako na 8-súvislosť, lebo bráni prepojeniu dvoch oblastí, ktoré sa dotýkajú v rohových pixloch. Ako už vieme, každá 4-súvislá oblasť je aj 8-súvislá. Naše algoritmy sa obmedzujú síce na 4-súvislé oblasti, ale sú dosť jednoducho modifikovateľné na 8-súvislé. V algoritmoch sa predpokladá, že hodnoty priradené jednotlivým pixlom sa dajú priamo prečítať z obrazovej pamäte.
- Pri hľadaní pixlov a následnom vypĺňaní pixlov vychádzame zo známeho vnútorného bodu – **semienka** a klasifikujeme body 4- alebo 8-susedných oblastí takto:

∗ testovaný bod je vnútorný, ak má inú farbu ako hraničný. Vypĺňanie založené na tomto princípe sa nazýva hraničné (po hranicu).

∗ testovaný bod je vnútorný, pokiaľ má farbu, ktorou sú zafarbené všetky body vnútra. Toto je charakteristické pre tzv. záplavové resp. lavínové (niekedy aj vlnové) vypĺňanie, pri ktorom sa súvislá oblasť prefarbuje novou farbou (všetky jej pixle tou istou).

∗ testovaný bod je vnútorný, ak má farbu, ktorá sa výrazne líši od farby hranice. Je to špeciálny prípad hraničného vypĺňania, v prípade, že hranica má len približne určený farebný odtieň a jas, čo býva obyčajne dôsledkom vyhladzovania obrazu a špeciálne hranice. V tomto prípade hovoríme o mäkkom vypĺňaní.

Týmto trom typom prislúchajú tri typy algoritmov pre semienkové vypĺňanie 4-súvislých oblastí. Načrtneme si ich:

## Záplavový algoritmus.

Rekurzívne priradí bodom 4-súvislej vnútorne definovanej oblasti (napr. farbou oldcolour) hodnotu newcolour (farba, ktorou sa má oblasť vyplniť). Procedúra read pixel zistí farbu bodu (*x*,*y*).

Algoritmus záplavového vypĺňania

```
procedure Flood_fill_4 (x,y, old_colour, new_colour: integer)     {začiatočný vnútorný bod
begin                                                              vypĺňania oblasti-semienko}
   if read_pixel (x,y) = old_colour then
      begin
         write_pixel(x,y, new_colour);
         Flood_fill_4 (x, y-1, old_colour);
         Flood_fill_4 (x, y+1, old_colour);
         Flood_fill_4 (x-1, y, old_colour);
         Flood_fill_4 (x+1, y, old_colour);
      end
end
```

Algoritmus zisťuje, či sme už vyšetrovali vnútorný bod (*x*,*y*). Ak nie, čiže ak má hodnotu old_colour, tak ju zmení na new_colour a rekurzívne vyšetrí jeho 4-susedov. Ak bod (*x*,*y*) nemá hodnotu old_colour , tak neurobí nič. Algoritmus možno modifikovať na 8-súvislé oblasti tak, že sa vyšetria všetci 8-susedia bodu (*x*,*y*).

## **Algoritmus vypĺňania po hraničné body**: Bound_fill_4

```
procedure Bound_fill_4(x, y,                    {začiatočný bod vypĺňania oblasti}
                       bound_colour,            {farba hranice oblasti}
                       new_colour: integer);    {nová farba oblasti}
begin
if read pixel (x,y) ≠ bound_colour and read_pixel(x,y) ≠ new_colour then
  begin
    write_pixel(x,y, new_colour);
      Bound_fill_4 (x, y-1, old_colour ,new_colour);
      Bound_fill_4 (x, y+1, old_colour, new_colour);
      Bound_fill_4 (x-1, y, old_colour, new_colour);
      Bound_fill_4 (x+1, y, old_colour, new_colour);
    end
end
```

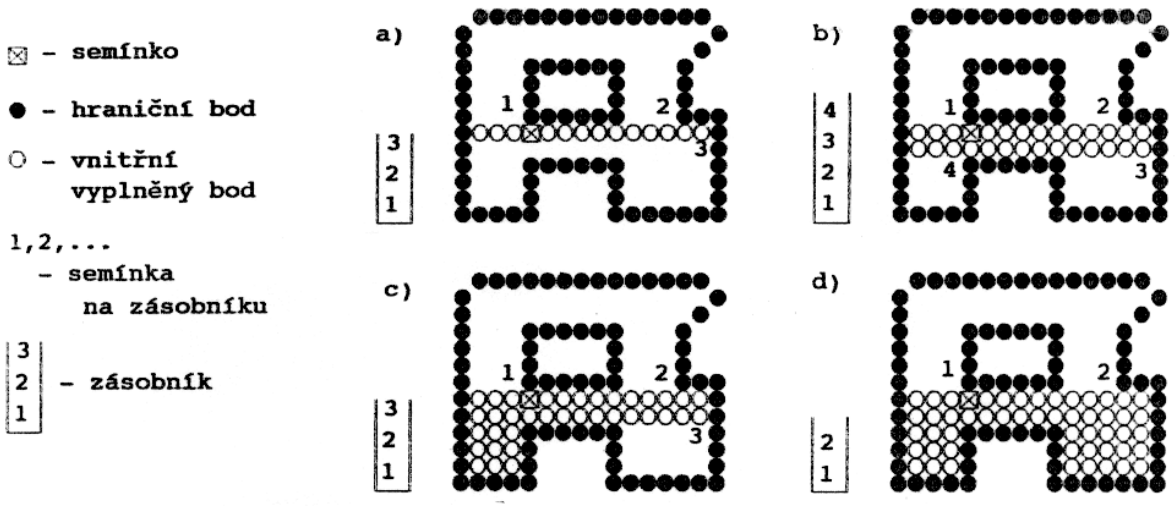
- Ako vidno, myšlienka tohto algoritmu sa podobá na Flood_fill_4. Nestačí však testovať, či bod (*x*,*y*) patrí oblasti. Potrebné sú dva testy, či je vo vnútri oblasti a či mu už skôr nebola pridelená nová farba. Aj modifikácia tohto algoritmu na 8-súvislé oblasti je pomerne jednoduchá.

Tento algoritmus sa tiež dá veľmi jednoducho upraviť na algoritmus mäkkého vypĺňania. Stačí upraviť podmienku tak, aby sa v nej akceptovala aj prahová hodnota.

- Tieto rekurzívne procedúry sú veľmi jednoduché, no nie príliš vhodné pre implementáciu. Okrem veľkých pamäťových nárokov (väčšinou sa používa zásobník pre rekurziu ), navyše pracujú dosť neefektívne, lebo v každom kroku vypĺňajú len štyri susedné body, ktoré sú naviac rozmiestnené v rôznych smeroch. Preto sa pre implementáciu semienkového vypĺňania používajú rekurzívne algoritmy, ktoré pracujú po riadkoch a vypĺňajú postupne všetky body napravo i naľavo až po nájdenie hraničných bodov. Pokračovacie body (u nekonvexných oblastí) sa ukladajú do zásobníka. Tieto algoritmy sú známe ako riadkové semienkové vypĺňanie (scan line seed fill). Príklad semienkového vypĺňania je na obr. 2, z učebného textu Aplikácie PG od J. Sochora a J. Žáru, kde je zachytený stav vyplnenia oblasti, poloha novo pripravených semienok a ich uloženie v zásobníku. Odtiaľ je prevzatý aj algoritmus riadkového semienkového vypĺňania. Tu sa možno dočítať aj o ďalších algoritmoch vypĺňania oblastí napr. o inverznom vypĺňaní.

Alg: Řádkové semínkové vyplňování

```
procedure Vypln_radku(x,y:integer);
var xvlevo,xvpravo,xhorni,xdolni,xvstup:integer; pokracovani:boolean;
begin
  xvpravo:=x;                                          { vyplňuj vpravo }
  repeat PIXEL[xvpravo,y]:=Barva_oblasti; xvpravo:=xvpravo+1
  until PIXEL[xvpravo,y]=Barva_hranice;
  xvlevo:=x+1;                                         { vyplňuj vlevo }
  repeat xvlevo:=xvlevo-1; PIXEL[xvlevo,y]:=Barva_oblasti
  until PIXEL[xvlevo-1,y]=Barva_hranice;
  xhorni:=xvlevo;                                { pokračuj na horní řádce }
  while (xhorni<xvpravo) do
    begin
      pokracovani:=false;
      while (PIXEL[xhorni,y+1]<>Barva_hranice) and
            (PIXEL[xhorni,y+1]<>Barva_oblasti) and (xhorni<xvpravo)
      do begin pokracovani:=true;xhorni:=xhorni+1; end;
    if pokracovani then { nové semínko = nejpravější nevyplněný pixel }
      begin
        if (xhorni=xvpravo) and (PIXEL[xhorni,y+1]<>Barva_hranice)
            and (PIXEL[xhorni,y+1]<>Barva_oblasti)
        then Vypln_radku(xhorni,y+1) else Vypln_radku(xhorni-1,y+1)
      end;
    xvstup:=xhorni;
    while ((PIXEL[xhorni,y+1]=Barva_hranice) or
          (PIXEL[xhorni,y+1]=Barva_oblasti)) and (xhorni<xvpravo)
    do xhorni:=xhorni+1;
    if xhorni=xvstup then xhorni:=xhorni+1
  end;
xdolni:=xvlevo;                                  { pokračuj na dolní řádce }
while (xdolni<xvpravo) do
  begin
    pokracovani:=false;
    while (PIXEL[xdolni,y-1]<>Barva_hranice) and
          (PIXEL[xdolni,y-1]<>Barva_oblasti) and (xdolni<xvpravo)
    do begin pokracovani:=true;xdolni:=xdolni+1; end;
    if pokracovani then { nové semínko = nejpravější nevyplněný pixel }
      begin
        if (xdolni=xvpravo) and
          (PIXEL[xdolni,y-1]<>Barva_hranice) and
          (PIXEL[xdolni,y-1]<>Barva_oblasti)
        then  Vypln_radku(xdolni,y-1) else Vypln_radku(xdolni-1,y-1)
      end;
    xvstup:=xdolni;                          { pokračuj v přerušeném úseku }
    while ((PIXEL[xdolni,y-1]=Barva_hranice) or
          (PIXEL[xdolni,y-1]=Barva_oblasti)) and (xdolni<xvpravo)
    do xdolni:=xdolni+1;
    if xdolni=xvstup then xdolni:=xdolni+1
  end
end;
```




Obr.2 Riadkové semienkové vypĺňanie