Vypĺňanie rovinných oblastí

Definícia:

Oblasťou rozumieme množinu bodov, ktoré sú navzájom susedné.

Oblasť môžeme definovať:

1. množinou bodov patriacich do oblasti
2 hranicou, ktorá ohraničuje určitú množinu bodov
3. pomocou vytvárajúceho mnohouholníka

Základné pojmy

Uvažujme obraz zapamätaný v bitovej mape, ktorého prvky zodpovedajú jednotlivým bodom (pixlom). Vo všeobecnosti každému obrazovému bodu *P* prislúcha jeho *8* susedov, ktorí sú očíslovaní *1* . . *8*.

3	2	1
4	P	8
5	6	7

- **priami susedia** bodu *P* (***4* - susedia**) – *2, 4, 6, 8*
- **nepriami susedia** bodu *P – 1, 3, 5, 7*
- ***8* - susedia** bodu *P* – priami i nepriami susedia bodu *P*

Budú nás zaujímať dva typy oblastí: *4*-súvislé a *8*-súvislé.
Každé dva body ***4*-súvislej oblasti** je možné spojiť postupnosťou *4* - susedných bodov z tejto oblasti.
Každé 2 body ***8*-súvislej oblasti** je možné spojiť postupnosťou *8* - susedných bodov z tejto oblasti.
Algoritmy pre *8* - súvislé oblasti je možné použiť aj pre *4* - súvislé oblasti, ale naopak nie.

Oblasť zadaná svojimi vnútornými bodmi sa nazýva **vnútorne definovaná**. Všetky body oblasti majú v tomto prípade zadanú hodnotu. Algoritmy, ktoré pracujú s takýmito oblasťami, sa nazývajú **vnútorne vypĺňajúce algoritmy**.
Oblasť zadaná hranicou sa nazýva **hranične definovaná**. Body, ktoré vytvárajú hranicu, majú predpísanú hodnotu. Algoritmy, ktoré pracujú s takýmito oblasťami, sa nazývajú **hranične vypĺňajúce algoritmy.**

1 Vnútorne vypĺňajúce algoritmy

1.1 Rekurzívny algoritmus FLOOD FILL

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

Tento algoritmus priradí bodom *4* - súvislej vnútorne definovanej oblasti hodnotu *new value.*

Vstup: *old_value, new_value*, súradnice (*x, y*) vnútorného bodu oblasti, od ktorého sa má začať vypĺňať.

Postup: Pre vnútorný bod so súradnicami (*x, y*) zistíme, či sme ho ešte nevyšetrovali (preto musí mať hodnotu *old_value*). Ak je to tak, hodnota sa zmení, a potom sa vyšetria jeho ďalšie *4* - susedné body.

Dá sa modifikovať aj pre *8* - súvislé oblasti (budeme v rekurzii vyšetrovať *8* - susedov). Algoritmus sa dá použiť napr. na vykreslenie oblasti inou farbou.

2 Hraničné vypĺňajúce algoritmy

Používateľ pomocou myši alebo iného vstupného zariadenia nakreslí obrys objektu, ukáže na jeho vnútorný bod a vyberie farbu výplne. Systém potom vyplní vnútro nakresleného objektu.

2.1 Rekurzívny algoritmus SEED_FILL (Semienkové vypĺňanie)

Vstup: *vypln_farba, hran_farba*, súradnice (*x, y*) vnútorného bodu oblasti, od ktorého sa má začať vypĺňať.

Postup: Začína sa s bodom (*x, y*) a pre susedné body sa zisťuje, či majú farbu hranice. Ak nie, potom sú zafarbené farbou výplne. Takto sa spracujú všetky body oblasti až po hranicu.

```
procedure Seed_fill(x, y, vypln_farba, hran_farba: integer);
var skut_farba: integer;
begin
      skut_farba := Get_pixel;
      if (skut_farba<>hran_farba) and (skut_farba<>vypln_farba) then
            begin
                  write_pixel (x, y, vypln_farba);
                  Seed_fill (x+1, y, vypln_farba, hran_farba);
                  Seed_fill (x-1, y, vypln_farba, hran_farba);
                  Seed_fill (x, y+1, vypln_farba, hran_farba);
                  Seed_fill (x, y-1, vypln_farba, hran_farba);
            end;
end;
```

Tento algoritmus vyplňuje rekurzívnym spôsobom. Bod, od ktorého sa začína vyplňovať, sa volá *semienko*. Ak semienko neleží na hranici a nebolo doteraz zafarbené, potom je zafarbené a vyšetrujú sa ďalej jeho *4* - susedia.

Je to rekurzívne riešenie – neefektívne. Každý vnútorný pixel je testovaný niekoľkokrát potom, ako už bol zafarbený. Pomer počtu rekurzívnych volaní k počtu skutočných zafarbení je 4:1. Každé vyvolanie procedúry je spojené s operáciou čítania z obrazovej pamäti.

2.2 Nerekurzívny algoritmus SEED_LINE_FILL (Riadkové semienkové vypĺňanie)

Metóda, ktorá znižuje počet prístupov do obrazovej pamäti, sa volá riadkové semienkové vypĺňanie - SEED_LINE_FILL. Namiesto rekurzie sa použije malý zásobník, do ktorého sa uloží niekoľko súradníc vnútorných bodov vyplňovanej oblasti.

Algoritmus:

1. Vlož začiatočný bod do zásobníka
2. Pokiaľ nie je zásobník prázdny
   (a) vyber bod so súradnicami (x , y) zo zásobníka
   (b) nájdi hranicu *XLeft* a *XRight* na riadku Y v najbližšom okolí bodu (x , y)
   (c) nakresli úsečku (*XLeft* , Y) – (*XRight* , Y)
   (d) na úsečke (*XLeft* , $Y + 1$) – (*XRight* , $Y + 1$) hľadaj prázdne úseky a pre každý z nich vlož do zásobníka súradnice jedného vnútorného bodu

(e) na úsečke (*XLeft , Y - 1*) – (*XRight , Y - 1*) hľadaj prázdne úseky a pre každý z nich vlož do zásobníka súradnice jedného vnútorného bodu

Metóda sa dá urýchliť tak, že do zásobníka ukladáme namiesto jednotlivých bodov informácie o voľných úsekoch nájdených v krokoch 2d a 2e algoritmu. Voľné úseky však musíme hľadať nielen v intervale (*XLeft – XRight*), ale nad a pod nakreslenou úsečkou smerom doľava a doprava až k hranici oblasti. Potom je možné vynechať v algoritme krok 2b a zmenšiť počet čítaní z obrazovej pamäti.

3 Geometricky určená hranica

Najčastejším prípadom geometricky určenej hranice je mnohouholník. Ten je definovaný postupnosťou bodov lomenej čiary, ktorá tvorí hranicu mnohouholníka.

Mnohouholník

- konvexný
- nekonvexný

a) *b)* *c)*

Obr. 1 Voľba správnej výplne pre zadanú hranicu

Hranica mnohouholníka

- môže pretínať sama seba – v tom prípade nie je jednoznačné, čo je jeho vnútrom – čo má byť vyplnené
- nemusí pretínať sama seba

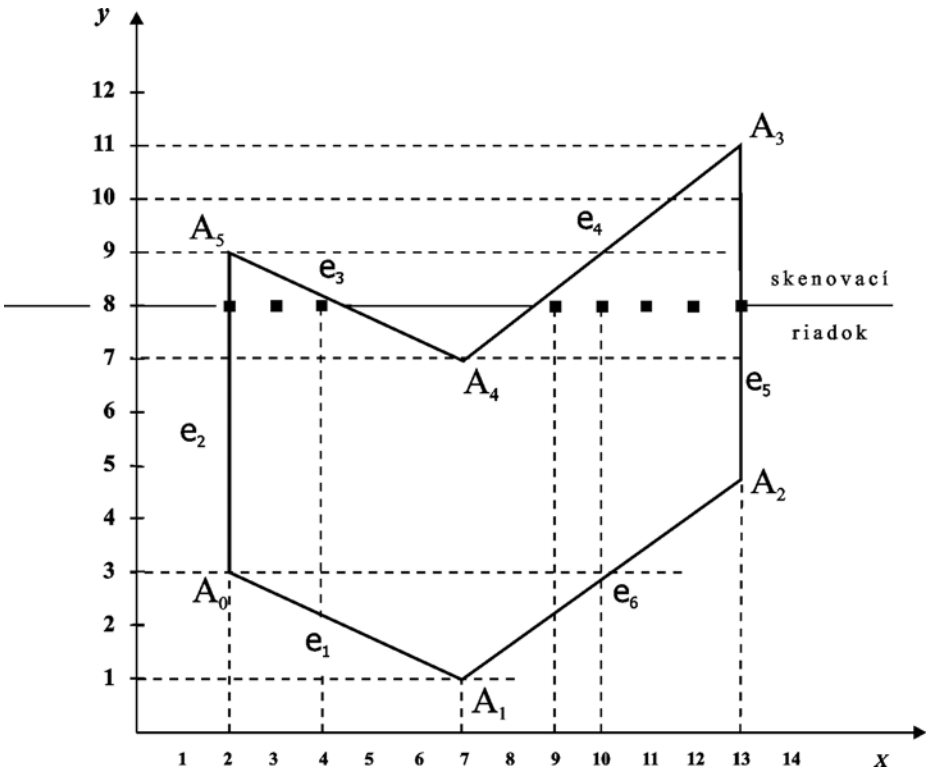
Na obr. 1 a) je nakreslená hviezda, určená lomenou čiarou tak, že jej hranica pretína sama seba. Pri vyplňovaní môže používateľ očakávať jeden z dvoch výsledkov b) a c). Ak vychádzame z predpokladu, že každá hranica oddeľuje priestor nevyplnený od vyplneného, potom správny výsledok je c). Ak chce mať používateľ vyplnené celé vnútro hviezdy, tak ju musí zadať iným spôsobom.

Rastrový rozklad mnohouholníka je rozklad mnohouholníka po riadkoch prenesený do pamäti obrazu. V každom kroku sa vyplní len jeden riadok rozkladu. Algoritmy založené na

tomto princípe - **skenovacie algoritmy.** Pri rastrovom rozklade mnohouholníka je oblasť definovaná vrcholmi, nepredpokladá sa zadanie hodnôt v pamäti obrazu.

3.1 Skenovací algoritmus

Na obr. 2 je zobrazený mnohouholník s jedným skenovacím riadkom. Potrebujeme zistiť, ktoré body sa nachádzajú vo vnútri mnohouholníka pre daný skenovací riadok (na obr. 2 sú to body *2 - 4* a *9 - 13*) a definujeme im príslušnú vnútornú hodnotu. Tento proces opakujeme pre každý skenovací riadok prechádzajúci daným mnohouholníkom. Takto dostaneme rastrový rozklad mnohouholníka.



Obr. 2 Mnohouholník a skenovací riadok

Algoritmus:

1. Nájdeme všetky priesečníky skenovacej priamky so všetkými stranami mnohouholníka.
2. Usporiadame priesečníky podľa *x* - ovej súradnice.
3. Zafarbíme všetky body, ktoré sa nachádzajú vždy medzi dvojicami za sebou usporiadaných priesečníkov.

Napr. pre riadok *8* na obr. 2 sa zafarbia body *2 - 4* a *9 - 13*, pre riadok *6* body *2 - 13*.

Horizontálne strany nemôžu mať priesečník *1* bod, preto ich vynecháme zo zisťovania priesečníkov so skenovacími priamkami, ale ich hneď vykreslíme.

Pri zisťovaní priesečníkov so skenovacou priamkou môže vzniknúť problém – nepárny počet prienikov. Napr. skenovací riadok $y = 3$ pretína stranu e_1 vo vrchole A_0 ($x = 2$), stranu e_2 vo vrchole A_0 ($x = 2$), stranu e_6 v bode $x = 10$, čo sú tri prieniky. Podľa obrázka sa zdá, že bod prieniku A_0 (*2, 3*) sa má počítať len jedenkrát, aj keď je priesečníkom skenovacej priamky s dvoma stranami mnohouholníka.

Teda prienik skenovacej priamky so stranou mnohouholníka počítame takto:

1. Ak skenovací riadok pretína stranu mnohouholníka, ale nie v jej koncovom bode, počítame jeden prienik.
2. Ak skenovací riadok pretína stranu mnohouholníka v koncovom bode A_i a

a) ak sa predchádzajúci vrchol A_{i-1} a nasledujúci vrchol A_{i+1} nachádzajú v tej istej polrovine určenej skenovacou priamkou, tak počítame 2 prieniky

b) ak sa predchádzajúci vrchol A_{i-1} a nasledujúci vrchol A_{i+1} nachádzajú v rôznych polrovinách určených skenovacou priamkou, tak počítame 1 prienik

Ak sme už bod A_i vyšetrovali, označíme ho ako vyšetrený, a už ho vyšetrovať nebudeme, aby sme počet prienikov skenovacej priamky s daným bodom nepočítali dvakrát.

3.2 Koherentnosť strán a algoritmus skenovania

V predchádzajúcom algoritme sa najprv nájdu priesečníky skenovacej priamky so všetkými stranami mnohouholníka. Tento výpočet je pomalý, pretože treba každú stranu porovnať so skenovacou priamkou.

Pri hľadaní priesečníkov so skenovacou priamkou však nemusíme s ňou porovnávať všetky strany. Ak strana pretína i - ty skenovací riadok, potom obvykle pretína aj ($i + 1$) - vý skenovací riadok. Pri prechode od jedného riadku k druhému môžeme vypočítať novú x - ovú súradnicu priesečníka so skenovacím riadkom takto:

$x(i + 1) = x(i) + 1 / m,$ kde m je tangens uhla strany s osou x – smernica priamky, na ktorej úsečka leží

$m = dy / dx$, $dy = 1$, $m = 1 / dx$, potom $dx = 1 / m$

Vytvoríme **tabuľku strán** TS

- v nej budú uložené všetky strany usporiadané podľa y - ovej súradnice
- pre každý skenovací riadok je vytvorená množina strán začínajúcich v danom skenovacom riadku

- každá množina sa usporiada do zoznamu podľa x - ovej súradnice, zodpovedajúcej bodu s najmenšou y - ovou súradnicou

Každý prvok tabuľky strán TS obsahuje:

- maximálnu y - ovú súradnicu strany y_{max}
- x - ovú súradnicu bodu s najmenšou y - ovou súradnicou
- zodpovedajúcu konštantu ($1 / m$)

TAS – **tabuľka aktívnych strán** – obsahuje zoznam strán, ktoré sa s daným riadkom pretínajú.

Vytvorenie tabuľky aktívnych strán TAS:

- pre každý skenovací riadok budeme brať do úvahy len tie strany, ktoré sa s ním pretínajú
- pri prechode k nasledujúcemu riadku nové x - ové súradnice vypočítame podľa vzorca
- všetky nové strany, ktoré sa pretínajú s daným riadkom budú pridané do TAS
- všetky tie hrany, ktoré sa nepretínajú s riadkom budú vylúčené z TAS

Algoritmus:

1. Vyberieme minimálnu hodnotu y z tabuľky strán s neprázdnym zoznamom.
2. Inicializujeme tabuľku aktívnych strán TAS = 0, t.j. prázdny zoznam.
3. Opakujeme krok 3 dovtedy, kým nebude TAS a TS prázdna.

   3.1. Pre danú hodnotu y pridáme zodpovedajúci zoznam do tabuľky TAS, pričom musíme zachovať usporiadanie podľa x - súradníc.

   3.2. Vyberieme z tabuľky TAS dvojice za sebou a medzi nimi vypĺňame pre dané x-ové úseky zodpovedajúcich bodov.

   3.3. Zrušíme tie strany z TAS, pre ktoré nastala rovnosť $y = y_{max}$.

   3.4. Pre všetky hodnoty x z tabuľky TAS musíme vypočítať nové hodnoty x (upravíme $x := x + 1/m$).

   3.5. Urobíme usporiadanie podľa x – ových súradníc v tabuľke TAS.

   3.6. Zvýšime aktuálnu hodnotu y na $y + 1$ a vrátime sa k bodu 3.