Obsah

Zoznam obrázkov

1 Vypĺňanie

1.1 Vypĺňanie oblastí

V počítačovej grafike v mnohých prípadoch nie je potrebné vykresliť jedinú priamku alebo krivku, ale vyplniť určitú oblasť.

V niektorých algoritmoch sa pri hľadaní a následnom vypĺňaní pixlov predpokladá, že poznáme aspoň jeden vnútorný bod v súvislej oblasti - **semienko**. Takýto typ vypĺňania sa preto nazýva **semienkové vypĺňanie**

Pri vypĺňaní prechádzajú algoritmy všetkými pixlami súvislej oblasti a farbia ich farbou, ktorou sa má táto oblasť zafarbiť (tzv. nová farba).

Body 4- alebo 8-susedných oblastí klasifikujeme takto:

- Testovaný bod je vnútorný, ak má inú farbu ako hraničný. Vypĺňanie založené na tomto princípe sa nazýva **hraničné vypĺňanie (po hranicu)**.
- Testovaný bod je vnútorný, pokiaľ má farbu, ktorou sú zafarbené všetky body vnútra. Toto je charakteristické pre tzv. **záplavové / lavínové 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**.

Vo všeobecnosti existujú dve možnosti pre definovanie oblasti, ktorá sa má vyplniť: oblasť zadaná svojimi vnútornými bodmi alebo oblasť zadaná hranicou.

Intuitívny prístup, ktorý nás napadne, je chápať zadanú množinu hraničných pixlov ako 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.

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).

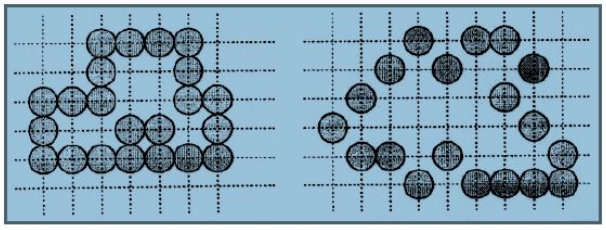
Ak hraničná krivka je 8-súvislá, tak vo všeobecnosti neplatí, že rozdeľuje nekonečnú sieť do dvoch 8-súvislých oblastí, pretože (ako vidieť na Obr.1) vnútro a vonkajšok sa dajú navzájom spojiť 8-súvislou cestou. Existuje však zaujímavý spôsob, ako upraviť tento problém. Špeciálne, ak požadujeme, aby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

V tejto kapitole si uvedieme tri algoritmy pre vypĺňanie a pre oblasť danú hranicou uvedieme jeden vylepšený algoritmus.

1.1.1 Vnútorný bod mnohouholníka

Jeden zo základných úloh počítačovej grafiky je určenie vnútorného bodu mnohouholníka.

Na objasnenie pojmov ako sú vnútro alebo vonkajšok mnohouholníka si uvedieme základné definície [8].



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

Okolie bodu je každá otvorená množina obsahujúca daný bod.

Bod A priestoru $X \subset E^n$ je vnútorný bod množiny M⊂X, ak existuje jeho okolie ležiace v M. Množina všetkých vnútorných bodov množiny M sa nazýva vnútro množiny M a označuje sa int$M$(z angl. interior = vnútro).

Bod A je hraničný bod množiny M, ak každé jeho okolie pretína množinu M aj jej doplnok (t.j. obsahuje bod patriaci množine M aj bod jej nepatriaci). Množina všetkých hraničných bodov množiny M sa nazýva hranica množiny M a označuje sa ∂M alebo bd$M$.

Bod A je vonkajší bod množiny M, ak existuje jeho okolie nepretínajúce množinu M. Množina všetkých vonkajších bodov množiny M sa nazýva vonkajšok množiny M a označuje sa ext$M$(z angl. exterior = vonkajšok).

Platí, že bod je vnútorným bodom mnohouholníka, ak polpriamka rovnobežná s osou x vychádzajúca z tohto bodu pretne mnohouholník v nepárnom počte hrán [1]. Predpokladáme pri tom, že polpriamka neprechádza žiadnym vrcholom mnohouholníka. Ak by takáto situácia nastala, môžeme mnohouholník posúvať.

1.1.2 Záplavový algoritmus (Flood fill Algorithm)

Princíp algoritmu

Na vstupe nech je 4-súvislá oblasť daná všetkými svojimi bodmi s farbou $stara - farba$.

Na výstupe má byť tá istá oblasť zafarbená farbou $nova - farba$. Procedúra ešte pozná súradnice jedného vnútorného bodu (x, y)oblasti. Je zrejmé, že ide o tzv. semienkové vypĺňanie. V algoritme sa predpokladá, že hodnoty priradené jednotlivým pixlom sa dajú priamo prečítať z obrazovej pamäte.

Tento algoritmus pracuje na princípe rekurzie. Rekurzia predstavuje využitie časti vlastnej vnútornej štruktúry, najmä definovanie funkcie pomocou seba samej resp. samotná táto funkcia.

Pseudokód

Implementácia rekurzívneho Flood-fill algoritmu pre 4-súvislú oblasť je zhrnutá v nasledovnom zápise. Na vstupe nech je 4-súvislá oblasť daná všetkými svojimi bodmi s farbou $stara - farba$.

Na výstupe má byť tá istá oblasť zafarbená farbou $nova - farba$. Procedúra, ktorá oblasť vyfarbí má okrem farieb ešte súradnice jedného vnútorného bodu oblasti (x, y).

Funkcia $zapis - pixel(x, y, color)$ má tri parametre: súradnice vykresleného bodu rastra a jeho príslušnú farbu. Funkcia $nacitaj - pixel(x, y)$ vráti farbu zadaného pixlu (x, y).

```
Flood-fill-4(int x, int y, color stara-farba, color nova-farba)
{
```

```
  if (nacitaj-pixel(x,y) = stara-farba) {
    zapis-pixel(x,y,nova-farba);
    Flood-fill-4(x,y-1,stara-farba, nova-farba);
    Flood-fill-4(x,y+1,stara-farba, nova-farba);
    Flood-fill-4(x-1,y,stara-farba, nova-farba);
    Flood-fill-4(x+1,y,stara-farba, nova-farba);
  }
}
```

Tento algoritmus sa obmedzuje síce na 4-súvislú oblasti, ale je jednoducho modifikovateľný pre 8-súvislú oblasť.

Literatúra

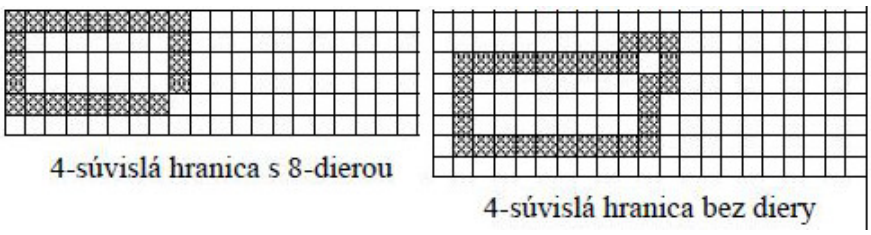
Záplavový algoritmus vieme nájsť podrobne spracovaný v literatúre [3] na strane 130. Jeho princíp je aj v [1] na 65. strane.

1.1.3 Algoritmus vypĺňania do hraničných bodov (Bound fill Algorithm)

Princíp algoritmu

Na vstupe je 4-súvislá hranica farby $hranicna - farba$. Na výstupe má byť oblasť ohraničená touto hranicou a zafarbená farbou $nova - farba$. V tomto algoritme potrebujeme hranicu, ktorá je 4-súvislá v silnejšom zmysle ako sme definovali v úvode. Hranica nesmie mať 8-dieru (Obr. 2.)

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.



Obr. 2: 4-súvislá hranica

Pseudokód

Implementácia rekurzívneho Bound-fill-4 algoritmu je zhrnutá v nasledovnom zápise. Na vstupe nech je 4-súvislá hranica farby $hranicna - farba$. Na výstupe má byť oblasť ohraničená touto hranicou a zafarbená farbou $nova - farba$. Algoritmus opäť obsahuje funkciu $zapis - pixel(x, y, color)$ a funkciu $nacitaj - pixel(x, y)$, ktorá vráti farbu zadaného pixlu (x, y). Opäť potrebujeme poznať ešte súradnice jedného vnútorného bodu oblasti (x, y).

```
Bound-fill-4(int x, int y, color hranicna − farba, color nova − farba)
{
  if (nacitaj-pixel(x,y) ≠ hranicna-farba && nacitaj-pixel(x,y)
  ≠ nova-farba) {
    zapis-pixel(x,y,nova-farba);
    Bound-fill-4(x,y-1,hranicna-farba, nova-farba);
```

```
    Bound-fill-4(x,y+1,hranicna-farba, nova-farba);
    Bound-fill-4(x-1,y,hranicna-farba, nova-farba);
    Bound-fill-4(x+1,y,hranicna-farba, nova-farba);
  }
}
```

Túto procedúru vieme rozšíriť aj na osem susedných bodov:

```
Bound-fill-8(int x, int y, color hranicna-farba, color nova_color)
{
  if (nacitaj-pixel(x,y) ≠ hranicna-farba && nacitaj-pixel(x,y)
  ≠ nova-farba) {
    zapis-pixel(x,y,nova-farba);
    Bound-fill-8(x,y-1,hranicna-farba, nova-farba);
    Bound-fill-8(x,y+1,hranicna-farba, nova-farba);
    Bound-fill-8(x-1,y,hranicna-farba, nova-farba);
    Bound-fill-8(x+1,y,hranicna-farba, nova-farba);
    Bound-fill-8(x-1,y-1,hranicna-farba, nova-farba);
    Bound-fill-8(x+1,y-1,hranicna-farba, nova-farba);
    Bound-fill-8(x-1,y+1,hranicna-farba, nova-farba);
    Bound-fill-8(x+1,y+1,hranicna-farba, nova-farba);
  }
}
```

Literatúra

Algoritmus Bound fill vieme nájsť podrobne spracovaný v literatúre [3] na strane 130 ako aj v literatúre [1] na strane 66 a [2].

1.1.4 Riadkové semienkové vypĺňanie (Scan line seed fill Algorithm)

Princíp algoritmu

Rekurzívne algoritmy Flood fill a Bound fill sú 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. Body (u nekonvexných oblastí) sa ukladajú do zásobníka. Tieto algoritmy sú známe ako **riadkové semienkové vypĺňanie (scan line seed fill)**.

1.2 Rozklad mnohouholníka do rastra

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.

1.2.1 Riadkový skenovací algoritmus (Scan line Algorithm)

Princíp algoritmu

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 nasledujúcom kroku.

V algoritme bude skenovacia priamka vodorovná s celočíselnými 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$[x + \frac{1}{m}, y + 1](m(x + \frac{1}{m}) + q = mx + q + 1 = y + 1)$.

Algoritmus používa dve dátové štruktúry: tabuľku strán (TS) a tabuľku aktívnych strán (TAS).

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.

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.

Postup

Na vstupe algoritmu nech je usporiadaný zoznam vrcholov mnohouholníka s celočíselnými súradnicami. Na výstupe algoritmu musí 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, teda $y = y_{min}$
3. Kým TS alebo TAS sú neprázdne opakuj:
   (a) Vyber TS strany v riadku y a daj ich do TAS.
   (b) Usporiadaj strany v TAS podľa x-ovej súradnice (druhá položka v jednotlivých záznamoch).
   (c) Vyber za sebou idúce úseky a vykresli ich.
   (d) Zruš tie strany v TAS, pre ktoré $y_{max} = y$
   (e) Pre strany v TAS zmeň x na $x + \frac{1}{m}$
   (f) $y = y + 1$

Príklad

Pre mnohouholník $A_0A_1A_2A_3A_4A_5A_6$, kde $A_0 = (0, 4)$, $A_1 = (3, 0)$, $A_2 = (5, 3)$, $A_3 = (4, 5)$, $A_4 = (3, 2)$, $A_5 = (2, 7)$, $A_6 = (1, 7)$, opíšeme TS, jednotlivé kroky TAS ako aj výstup algoritmu a jednotlivé vykresľované úseky. Mnohouholník môžeme vidieť na Obr. 3.

1. Na obrázku môžeme vidieť, že mnohouholník má jednu vodorovnú hranu f, tú do TS nezaradíme. Tiež obsahuje dva neextremálne vrcholy A_0 a A_2.
2. $y_{min} = 0$ a $y_{max} = 7$ a TS vyzerá nasledovne:

   $0: \ a = 4 \mid 3 \mid -\frac{3}{4}, b = 3 \mid 3 \mid \frac{2}{3}$

   $1:$

   $2: \ d = 5 \mid 3 \mid \frac{1}{3}, e = 7 \mid 3 \mid -\frac{1}{5}$

   $3:$

$4: \; c = 5 \mid 5 - \frac{1}{2} \mid -\frac{1}{2}$

$5: \; g = 7 \mid 0 + \frac{1}{3} \mid \frac{1}{3}$

$6:$

$7:$

3. $i = 0:$ (a) TAS: $a = 4 \mid 3 \mid -\frac{3}{4}$, $b = 3 \mid 3 \mid \frac{2}{3}$
   (b) TAS sa po usporiadaní nezmení.
   (c) Vykresli úsek $[3, 0]$ až $[3, 0]$, t.j. pixel $[3, 0]$.
   (d) Z TAS sa žiadna hrana nezruší.
   (e) TAS: $a = 4 \mid \frac{9}{4} \mid -\frac{3}{4}$, $b = 3 \mid \frac{11}{3} \mid \frac{2}{3}$
   (f) $y = 1$

   $i = 1:$ (a) TAS sa nezmení, t.j. TAS: $a = 4 \mid 3 \mid -\frac{3}{4}$, $b = 3 \mid 3 \mid \frac{2}{3}$
   (b) TAS sa po usporiadaní nezmení.
   (c) Vykresli úsek $[\frac{9}{4}, 1]$ až $[\frac{11}{3}, 1]$, t.j. pixle $[2, 1]$,$[3, 1]$,$[4, 1]$.
   (d) Z TAS sa žiadna hrana nezruší.
   (e) TAS: $a = 4 \mid \frac{6}{4} \mid -\frac{3}{4}$, $b = 3 \mid \frac{11}{3} \mid \frac{2}{3}$
   (f) $y = 2$

   $i = 2:$ (a) TAS: $a = 4 \mid \frac{6}{4} \mid -\frac{3}{4}$, $b = 3 \mid \frac{11}{3} \mid \frac{2}{3}$, $d = 5 \mid 3 \mid \frac{1}{3}$, $e = 7 \mid 3 \mid -\frac{1}{3}$
   (b) TAS: $a = 4 \mid \frac{6}{4} \mid -\frac{3}{4}$, $d = 5 \mid 3 \mid \frac{1}{3}$, $e = 7 \mid 3 \mid -\frac{1}{3}$, $b = 3 \mid \frac{11}{3} \mid \frac{2}{3}$.
   (c) Vykreslia sa úseky $[\frac{6}{4}, 2]$ až $[3, 2]$ a $[3, 2]$ až $[\frac{13}{3}, 2]$, t.j. pixle $[2, 2]$,$[3, 2]$,$[4, 2]$.
   (d) Z TAS sa žiadna hrana nezruší.
   (e) TAS: $a = 4 \mid \frac{3}{4} \mid -\frac{3}{4}$, $d = 5 \mid \frac{10}{3} \mid \frac{1}{3}$, $e = 7 \mid \frac{14}{5} \mid -\frac{1}{5}$, $b = 3 \mid 5 \mid \frac{2}{3}$.
   (f) $y = 3$

   $i = 3:$ (a) TAS sa nemení.
   (b) TAS: $a = 4 \mid \frac{3}{4} \mid -\frac{3}{4}$, $e = 7 \mid \frac{14}{5} \mid -\frac{1}{5}$, $d = 5 \mid \frac{10}{3} \mid \frac{1}{3}$, $b = 3 \mid 5 \mid \frac{2}{3}$.
   (c) Vykreslia sa úseky $[\frac{3}{4}, 3]$ až $[\frac{14}{3}, 3]$ a $[\frac{10}{3}, 3]$ až $[5, 3]$, t.j. pixle $[1, 3]$,$[2, 3]$, $[3, 3]$, $[4, 3]$, $[5, 3]$.
   (d) TAS: $a = 4 \mid \frac{3}{4} \mid -\frac{3}{4}$, $e = 7 \mid \frac{14}{5} \mid -\frac{1}{5}$, $d = 5 \mid \frac{10}{3} \mid \frac{1}{3}$.
   (e) TAS: $a = 4 \mid 0 \mid -\frac{3}{4}$, $e = 7 \mid \frac{13}{5} \mid -\frac{1}{5}$, $d = 5 \mid \frac{11}{3} \mid \frac{1}{3}$.
   (f) $y = 4$

   $i = 4:$ (a) TAS: $a = 4 \mid 0 \mid -\frac{3}{4}$, $e = 7 \mid \frac{13}{5} \mid -\frac{1}{5}$, $d = 5 \mid \frac{11}{3} \mid \frac{1}{3}$, $c = 5 \mid \frac{9}{2} \mid -\frac{1}{2}$.
   (b) TAS sa po usporiadaní nezmení.
   (c) Vykreslia sa úseky $[0, 4]$ až $[\frac{13}{5}, 4]$ a $[\frac{11}{3}, 4]$ až $[\frac{9}{2}, 4]$, t.j. pixle $[0, 4]$,$[1, 4]$, $[2, 4]$, $[3, 4]$, $[4, 4]$, $[5, 4]$.
   (d) TAS: $e = 7 \mid \frac{13}{5} \mid -\frac{1}{5}$, $d = 5 \mid \frac{11}{3} \mid \frac{1}{3}$, $c = 5 \mid \frac{9}{2} \mid -\frac{1}{2}$.
   (e) TAS: $e = 7 \mid \frac{12}{5} \mid -\frac{1}{5}$, $d = 5 \mid 4 \mid \frac{1}{3}$, $c = 5 \mid 4 \mid -\frac{1}{2}$.
   (f) $y = 5$

   $i = 5:$ (a) TAS: $e = 7 \mid \frac{12}{5} \mid -\frac{1}{5}$, $d = 5 \mid 4 \mid \frac{1}{3}$, $c = 5 \mid 4 \mid -\frac{1}{2}$, $g = 7 \mid \frac{1}{3} \mid \frac{1}{3}$.
   (b) TAS: $g = 7 \mid \frac{1}{3} \mid \frac{1}{3}$, $e = 7 \mid \frac{12}{5} \mid -\frac{1}{5}$, $d = 5 \mid 4 \mid \frac{1}{3}$, $c = 5 \mid 4 \mid -\frac{1}{2}$.
   (c) Vykreslia sa úseky $[\frac{1}{3}, 5]$ až $[\frac{12}{5}, 5]$ a $[4, 5]$ až $[4, 5]$, t.j. pixle $[0, 5]$,$[1, 5]$, $[2, 5]$, $[4, 5]$.
   (d) TAS: $g = 7 \mid \frac{1}{3} \mid \frac{1}{3}$, $e = 7 \mid \frac{12}{5} \mid -\frac{1}{5}$.

(e) TAS: $g = 7 \mid \frac{2}{3} \mid \frac{1}{3}$, $e = 7 \mid \frac{11}{5} \mid -\frac{1}{5}$.

(f) $y = 6$

$i = 6:$ (a) TAS sa nemení.

(b) TAS sa po usporiadaní nezmení.

(c) Vykreslí sa úsek $[\frac{2}{3}, 6]$ až $[\frac{11}{5}, 6]$, t.j. pixle $[1, 6]$, $[2, 6]$.

(d) Z TAS sa žiadna hrana nezruší.

(e) TAS: $g = 7 \mid 1 \mid \frac{1}{3}$, $e = 7 \mid 2 \mid -\frac{1}{5}$.

(f) $y = 7$

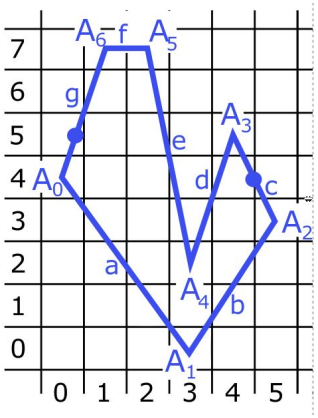
$i = 7:$ (a) TAS sa nemení.

(b) TAS sa po usporiadaní nezmení.

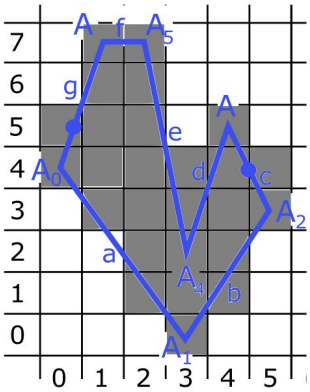
(c) Vykreslí sa úsek $[1, 7]$ až $[2, 7]$, t.j. pixle $[1, 7]$, $[2, 7]$.

(d) TAS = $\emptyset$.

(e) $y = y_{max}$, algoritmus končí. Výsledok je zobrazený na Obr. 4.



Obr. 3: Príklad mnohouholníka



Obr. 4: Výsledný vyplnený mnohouholník

Pseudokód

Pseudokód Scan-line algoritmu môžeme nájsť na strane 122 v literatúre [3]. Pre jeho veľký rozsah a zložitosť ho tu neuvádzame.

Literatúra

Algoritmus Scan-line vieme nájsť podrobne spracovaný v [3] na strane 117 ako aj v slovenskej literatúre [1] na strane 70 a [2]. Je v krátkosti spomenutý aj v kapitole o rasterizovaní polygónov v [7].

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