Obsah

Zoznam obrázkov

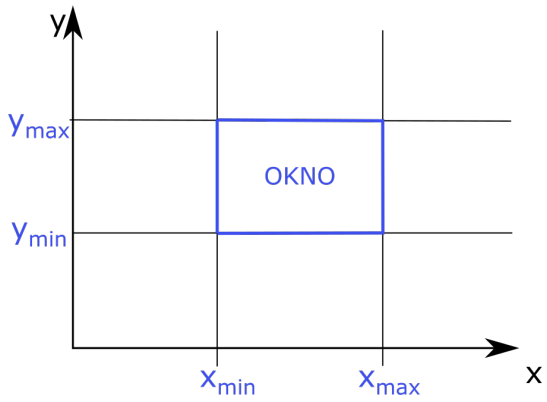
1 Orezávanie

1.1 Algoritmy na orezávanie bodu

Predpokladáme, že máme bod $A = (x_A, y_A)$ a okno, vzhľadom na ktoré orezávame je dané svojimi min-max súradnicami $x_{min}, x_{max}, y_{min}, y_{max}$.

Ak bod spĺňa všetky nerovnosti nižšie, bod leží v okne a vykreslíme ho. Inak tento bod zahadzujeme.

$$\begin{aligned} x_{min} \leq x_A \leq x_{max} \\ y_{min} \leq y_A \leq y_{max} \end{aligned} \tag{1.1}$$



Obr. 1: Orezávanie danej úsečky

1.2 Algoritmy na orezávanie úsečky do axiálneho okna

Orezávanie bodu je pomerne jednoduché. Na zložitejšie objekty však potrebujeme viac operácií, a preto sa snažíme ich počet minimalizovať. Tak je to aj s úsečkou vo všeobecnej polohe. Najprv otestujeme jej krajné body, a ak patria do okna, tak aj úsečka leží v okne.

V prípade, že úsečka neleží v okne, otestujeme jej prienik s oknom. Ak úsečka pretína okno, tak ju orezávame, a na to potrebujeme zistiť body prieniku (t.j. kde úsečka pretne hranice okna).

Podstatnou vlastnosťou pri orezávaní je rýchlosť algoritmu na nájdenie bodov prieniku (pri vykresľovaní scény môžeme mať rádovo tisíce objektov, ktoré treba orezať, za čo najkratší čas). Preto vzniklo niekoľko algoritmov na orezávanie. Najprv ale musíme urobiť testy, či naozaj treba hľadať body prieniku. Úsečky, ktoré ležia v okne, orezávať netreba, tie môžeme rovno vykresliť. Ak oba krajné body úsečky ležia nad oknom, t.j. majú súradnicu $y > y_{max}$ okna, tak úsečku nevykreslíme. Podobne ak oba krajné body úsečky ležia pod oknom, naľavo či napravo od okna, tak úsečku nevykreslíme.

V tejto kapitole si predstavíme jeden algoritmus na orezávanie úsečky.

1.2.1 Algoritmus Cohen-Sutherland

Princíp algoritmu

Algoritmus Cohen-Sutherland slúži na orezávanie úsečky do axiálneho okna, ktoré má strany rovnobežné so súradnicovými osami. Algoritmus prebieha počas behu algoritmu a minimalizuje počet orezávaní.

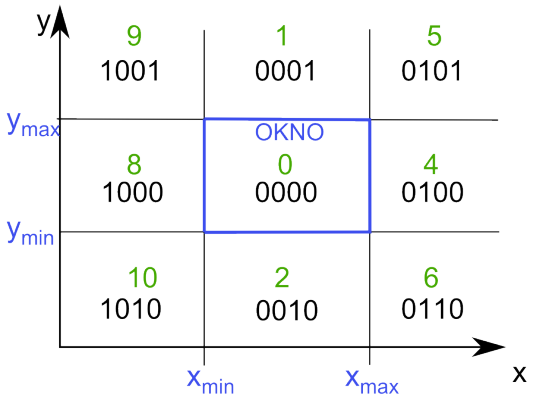
Prvá časť tohto algoritmu slúži na vylúčenie úsečiek, u ktorých netreba počítať prienik s oknom. Takáto úsečka sa buď nachádza celá v okne alebo mimo okna.

Najprv rozdelíme rovinu na deväť častí hraničnými priamkami okna. Súradnice (x_{min}, x_{max}, y_{min}, y_{max}) definujú okno, vzhľadom na ktoré budeme orezávať úsečky. Každej časti priradíme 4-bitový kód, ktorý dostane aj každý bod úsečky, ktorý do tejto oblasti padne (Obr. 2).

Jednotlivé bity pre bod (x, y) sa nastavia nasledovne:

- 1. bit – bod leží naľavo od okna ($x_{min} > x$)
- 2. bit – bod leží napravo od okna ($x_{max} < x$)
- 3. bit – bod leží nižšie od okna ($y_{min} > y$)
- 4. bit – bod leží vyššie od okna ($y_{max} < y$)

Preto napríklad bod s kódom 0101 leží napravo a vyššie od okna.

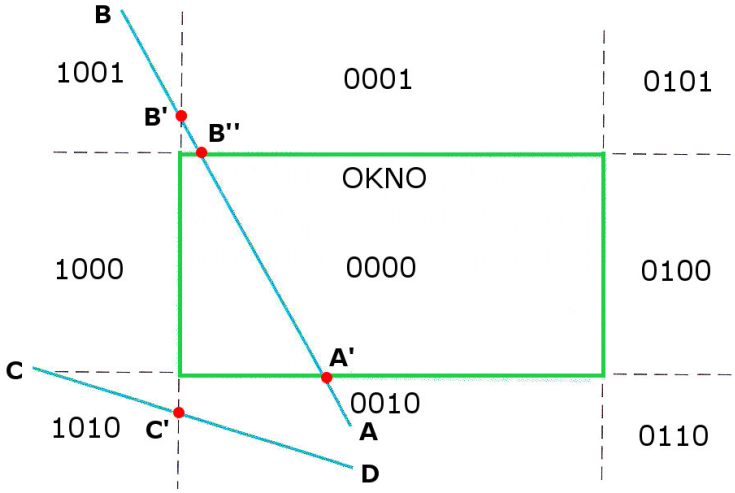


Obr. 2: Kódy oblastí určené oknom

Bod, ktorý sa nachádza vo vnútri okna, má kód 0000. Úsečka je v okne celá a môžeme ju vykresliť, ak oba jej krajné body majú kód 0000. Tento prípad nastane, ak kod(A)||kod(B) = 0000, kde A a B sú krajné body orezávanej úsečky a symbol || predstavuje binárne sčítanie. Takýto test budeme nazývať **Trivial accept**.

Ďalej môžeme vylúčiť úsečku bez vykreslenia, ak oba jej krajné body sú vľavo, vpravo, hore alebo dole od okna. Ľahko to identifikujeme práve podľa kódov krajných bodov úsečky tak, že urobíme ich logický súčin ((1 and 1)=1, inak 0). Teda kod(A)&& kod(B) $\neq$ 0000. Takýto test budeme označovať pojmom **Trivial reject**.

Ak je logický súčin rôzny od 0000, tak úsečka nepretína okno, a teda ju môžeme vylúčiť z vykresľovania. V opačnom prípade úsečku nemôžeme vylúčiť z vykresľovania a musíme ju otestovať na prienik vzhľadom na hranicu okna, v najhoršom prípade 4-krát.



Obr. 3: Orezávanie danej úsečky

Myšlienku orezávania úsečky AB vzhľadom na axiálne okno môžeme vidieť na Obr. 3.

Vyberieme bod mimo okna, napr. bod A. Bitový kód bodu A je 0010, preto má zmysel orezávať úsečku AB priamkou prechádzajúcou dolnou hranou okna ($y = y_{min}$). Bod prieniku úsečky a okna označíme ako $A^{'}$. Ten predstavuje nový krajný bod orezávanej úsečky, teda úsečku $A^{'}B$. Bod $A^{'}$ sa teraz nachádza na spodnej hrane okna, teda jeho bitový kód predstavuje 0000.

Prechádzame na bod B, druhý krajný bod orezávanej úsečky. Z jeho bitového kódu vidíme, že sa nachádza vľavo od okna. Orezávame podľa priamky $x = x_{min}$ a nachádzame bod $B^{'}$. Zahadzujeme pôvodný bod B a nahradíme ho bodom $B^{'}$. Dostávame úsečku $A^{'}B^{'}$. Z bitového kódu bodu $B^{'}$ vidíme, že sa nachádza nad oknom a preto ho orezávame priamkou $y = y_{max}$. Nachádzame bod $B^{''}$, ktorého bitový kód je 0000. Úspešne sme orezali úsečku AB.

V prípade úsečky CD po prvom orezávacom kroku zistíme, že úsečka sa celá nachádza pod/vedľa axiálneho okna, preto ju celú zahadzujeme.

Matematicky je možno vyjadriť prienik orezávanej úsečky s priamkami prechádzajúcimi hranami axiálneho okna pomocou smernicovej rovnice priamky (**??**).

- Orezávanie **zľava** priamkou $x = x_{min}$ (Obr 4b):

$$x'_B = x_B + \Delta x_B = x_B + (x_{min} - x_B) = x_{min},$$
$$y'_B = y_B + \Delta y_B = y_B + \Delta x_B \mid m \mid = y_B + \mid m \mid (x_{min} - x_B)$$

- Orezávanie **sprava** priamkou $x = x_{max}$ (Obr 4a):

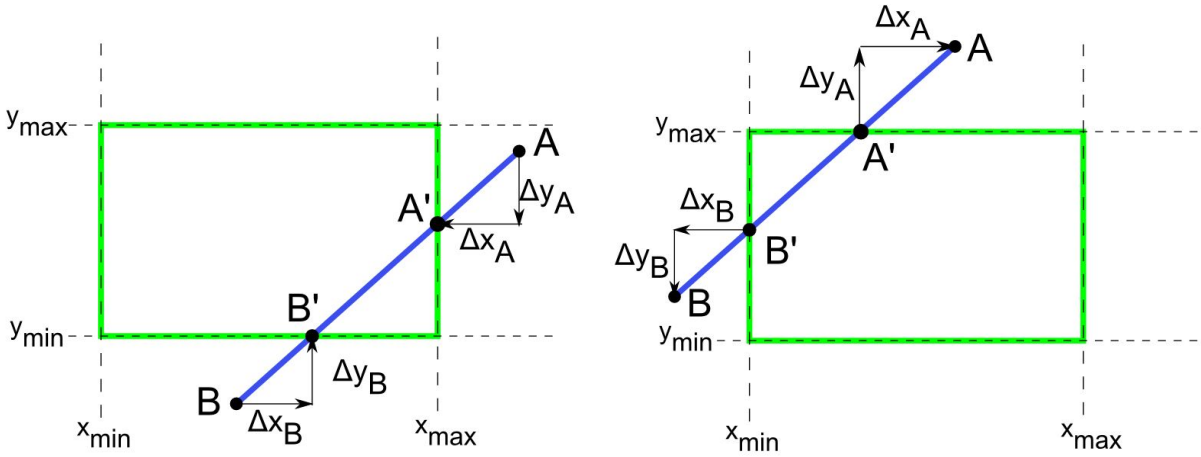
$$x'_A = x_A + \Delta x_A = x_A + (x_{max} - x_A) = x_{max},$$
$$y'_A = y_A + \Delta y_A = y_A + \Delta x_A \mid m \mid = y_A + \mid m \mid (x_{max} - x_A)$$

- Orezávanie **zdola** priamkou $y = y_{min}$ (Obr 4a):

$$x'_B = x_B + \Delta x_B = x_B + \frac{1}{\mid m \mid}\Delta y_B = x_B + \frac{1}{\mid m \mid}(y_{min} - y_B),$$
$$y'_B = y_B + \Delta y_B = y_B + (y_{min} - y_B) = y_{min}$$

- Orezávanie **zhora** priamkou $y = y_{max}$ (Obr 4b):

$$x'_A = x_A + \Delta x_A = x_A + \frac{1}{\mid m \mid}\Delta y_A = x_A + \frac{1}{\mid m \mid}(y_{max} - y_A),$$
$$y'_A = y_A + \Delta y_A = y_A + (y_{max} - y_A) = y_{max}$$



(a) Orezávanie sprava a zdola

(b) Orezávanie zľava a zhora

Obr. 4: Orezávanie

Algoritmus Cohen-Sutherland v danej verzii nevie orezať úsečku rovnobežnú zo súradnicovou osou y podľa axiálneho okna, pretože pre smernicu m takejto úsečky platí $m = \frac{\Delta y}{\Delta x}$, kde $\Delta x = 0$.

Postup

Postup orezávania úsečky s krajnými bodmi $A = (x_A, y_A)$ a $B = (x_B, y_B)$ do okna určeného bodmi $M = (x_M, y_M)$ a $N = (x_N, y_N)$.

1. Zisti bitový kód bodu A kod(A) a bitový kod bodu B kod(B).
2. Otestuj Trivial accept bodov A a B. Ak vyšlo true vykresli celu úsečku, ak false pokračuj.
3. Otestuj Trivial reject bodov A a B. Ak vyšlo true zahoď celu úsečku, ak false pokračuj.
4. Vypočítaj smernicu m úsečky AB ako $m = \frac{y_B - y_A}{x_B - x_A}$ a konštanty $x_{min} = min(x_A, x_B)$, $x_{max} = max(x_A, x_B)$, $y_{min} = min(y_A, y_B)$ a $y_{max} = max(y_A, y_B)$.
5. Vyber krajný bod, ktorý sa nenachádza vnútri axiálneho okna, nech je to bod A.
6. Kým $kod1$ bodu A sa nerovná 0000 postupuj:
   (a) Orezanie zľava do bodu $A' = (x'_A, y'_A)$: $x'_A = x_{min}$ a $y'_A = y_A + \mid m \mid (x_{min} - x_A)$. Zahoď úsečku AA', teda $A = A'$
   (b) Orezanie sprava do bodu $A' = (x'_A, y'_A)$: $x'_A = x_{max}$ a $y'_A = y_A + \mid m \mid (x_{max} - x_A)$. Zahoď úsečku AA', teda $A = A'$
   (c) Orezanie zdola do bodu $A' = (x'_A, y'_A)$: $x'_A = x_A + \dfrac{1}{\mid m \mid}(y_{min} - y_A)$ a $y_A = y_{min}$. Zahoď úsečku AA', teda $A = A'$
   (d) Orezanie zhora do bodu $A' = (x'_A, y'_A)$: $x'_A = x_A + \dfrac{1}{\mid m \mid}(y_{max} - y_A)$ a $y_A = y_{max}$. Zahoď úsečku AA', teda $A = A'$
7. Vymeň bod A a bod B a opakuj bod 6.

Príklad

Na ilustráciu algoritmu Cohen-Sutherland si ukážeme orezanie úsečky s krajnými bodmi $A = (x_A, y_A) = (0, 0)$ a $B = (x_B, y_B) = (8, 7)$ do axiálneho okna určeného bodmi $M = (x_M, y_M) = (2, 1)$ a $N = (x_N, y_N) = (6, 5)$.

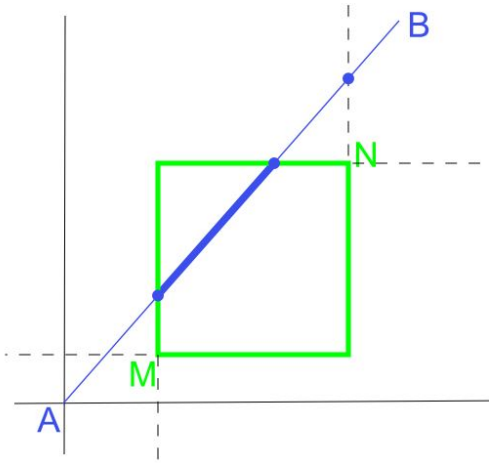
1. Bitový kód bodu A kod(A)=1010 a bitový kod bodu B kod(B)=0101.
2. Trivial accept(A, B) = false.
3. Trivial reject(A, B) = false.
4. $m = \frac{7}{8}$, $x_{min} = min(2, 6) = 2$, $x_{max} = max(2, 6) = 6$, $y_{min} = min(1, 5) = 1$ a $y_{max} = max(1, 5) = 5$.
5. Vyber krajný bod, nech je to bod A.
6. Kým kod(A) bodu A sa nerovná 0000 postupuj:

   zľava: $x'_A = x_{min} = 1$ a $y'_A = 0 + \frac{7}{8}(2 - 0)$. Teda $A = (x'_A, y'_A) = (2, \frac{7}{4})$ a kod(A)=0010.
7. Vymeň bod A a bod B, teda $A = (8, 7)$ a $B = (2, \frac{7}{4})$. V tomto prípade kod(A)=0101.

   sprava: $x'_A = x_{max} = 6$ a $y'_A = 7 + \dfrac{7}{8}(6 - 8)$. Teda $A = (x'_A, y'_A) = (6, \frac{21}{4})$ a kod(A)=0001.

   zhora: $x'_A = 6 + \dfrac{8}{7}(5 - \frac{21}{4}) = \frac{40}{7}$ a $y_A = 5$. Teda $A = (x'_A, y'_A) = (\frac{40}{7}, 5)$ a kod(A)=0000.

8. Algoritmus končí. Krajné body orezanej úsečky sú $A = (\frac{40}{7}, 5)$, $B = (2, \frac{7}{4})$. Výslednú orezanú úsečku môžete vidieť na Obr 5.



Obr. 5: Výsledná orezaná úsečka algoritmom Cohen-Sutherland

Pseudokód

Implementácia algoritmu Cohen-Sutherland je zhrnutá v nasledujúcom pseudokóde.

Na vstupe sú dva krajné body úsečky $A = (x_A, y_A)$ a $B = (x_B, y_B)$. Súradnice (x_{min}, x_{max}, y_{min}, y_{max}) definujú okno, podľa ktorého orezávame úsečku. Procedúra $ZistiKod(x, y, kod)$, priradí premennej kod 4-bitový kód bodu (x, y).

Procedúra TrivialAccept(kod1, kod2) porovná bitové kódy dvoch rôznych bodov kod1 a kod2. Vráti hodnotu true, ak sa oba body nachádzajú vnútri okna a false ak nie.

Procedúra TrivialReject(kod1, kod2) porovná bitové kódy dvoch rôznych bodov kod1 a kod2. Vráti hodnotu true, ak sa oba body nachádzajú vľavo/vpravo/zľava alebo sprava od okna a false ak nie.

```
Cohen-Sutherland (int x_A, int y_A, int x_B, int y_B)
{
  ZistiKod(x_1, y_1, kod1)
  ZistiKod(x_2, y_2, kod2)
  while(kod≠0000) {
    if (TrivialAccept(kod1, kod2) == true) koniec;
    elseif (TrivialReject(kod1, kod2) == true) koniec;
    else {
      if(kod1 ≠ 0000) KodVonku=kod1;
      else KodVonku=kod2;
      if (KodVonku & HORE) { /bod sa nachádza nad oknom
        x = x_A + (x_B − x_A) ∗ (y_max − y_A)/(y_B − y_A);
        y = y_max;
      } else if (KodVonku & DOLU) { /bod sa nachádza pod oknom
        x = x_A + (x_B − x_A) ∗ (y_min − y_A)/(y_B − y_A);
        y = y_min;
      } else if (KodVonku & VPRAVO) { /bod sa nachádza vpravo
      od okna
```

```
      y = y_A + (y_B - y_A) * (x_max - x_A)/(x_B - x_A);
      x = x_max;
    } else if (KodVonku & VLAVO) { /bod sa nachádza vl'avo
    od okna
      y = y_A + (y_B - y_A) * (x_min - x_A)/(x_B - x_A);
      x = x_min;
    }
    // Presunieme sa na druhý koncový bod úsečky, ktorý
    sa nachádza mimo okna
    if (KodVonku=kod1){
      x_A = x;
      y_A = y;

    } else {
      x_B = x;
      y_B = y;

    }
    KodVonku=kod2
  }
 }
}
```

Literatúra

Algoritmus Cohen-Sutherland vieme nájsť podrobne spracovaný v anglickej literatúre [3] od strany 226. S jeho vysvetlením sa môžeme stretnúť aj v [1] o strany 26 a v českej literatúre [2] sú tomuto algoritmu venované dve strany 106-107.

1.3 Algoritmy na orezávanie úsečky mnohouholníkom

1.3.1 Algoritmus Liang-Barsky

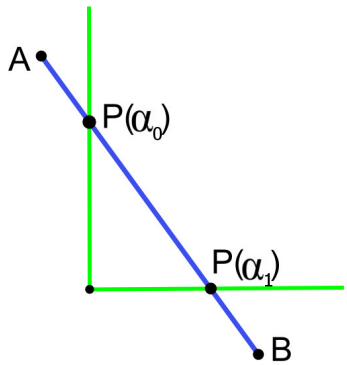
Princíp algoritmu

Algoritmus Liang-Barsky slúži na orezávanie úsečky do konvexného mnohouholníka. Bod na úsečke AB je reprezentovaný pomocou parametrického vyjadrenia úsečky,

$$P(\alpha) = (1 - \alpha)A + \alpha B, \tag{1.2}$$

kde $0 \leq \alpha \leq 1$.

Algoritmus počíta dve hodnoty parametra α_0, α_1, ktorým prislúcha výsledná orezaná úsečka $P(\alpha_0)P(\alpha_1)$, kde $\alpha_0 < \alpha_1$ (Obr 6).

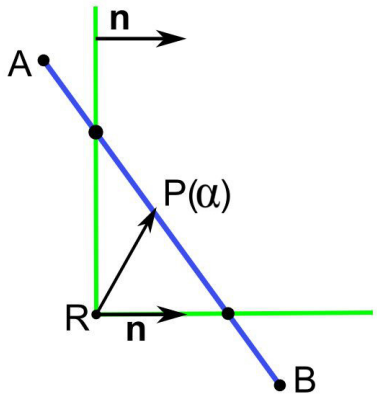


Obr. 6: Priesečníky úsečky a okna

Na začiatku algoritmu sa inicializujú hodnoty ako $\alpha_0 = 0$ a $\alpha_1 = 1$, teda $P(\alpha_0)P(\alpha_1) = AB$.

K orezávaniu úsečky pristupujeme postupne vzhľadom na polroviny mnohouholníka. Skúmame, ako oreže úsečku polrovina určená $< R, \mathbf{n} >$, kde *n* je normálový vektor hranice smerujúci dovnútra polroviny (Obr 7).

Počas priebehu algoritmu bude hodnota α_0 rásť (neklesať) a hodnota α_1 klesať (nerásť). Ak $\alpha_0 > \alpha_1$, tak orezávaná úsečka je prázdna a algoritmus končí.



Obr. 7: Polrovina určená bodom R a vektorom **n** [6]

Potrebujeme zistiť hodnotu α (ak existuje), pre ktorú priamka, na ktorej leží úsečka, pretína hranicu polroviny. Presnejšie potrebujeme nájsť podmienku, aby $P(\alpha)$ patrí polrovine. Aby sme takéto α vypočítali, dosadíme $P(\alpha)$ do podmienky patriť polrovine:

$$\begin{aligned}
(P(\alpha) - R).\mathbf{n} &\geq 0, \\
((1-\alpha)A + \alpha B) - R).\mathbf{n} &\geq 0, \\
(A + \alpha(B - A)) - R).\mathbf{n} &\geq 0, \\
\alpha(B - A).\mathbf{n} + (A - R).\mathbf{n} &\geq 0, \\
\alpha(B - A).\mathbf{n} &\geq (R - A).\mathbf{n}, \\
\alpha d_l &\geq d_r,
\end{aligned} \tag{1.3}$$

kde $d_l = (B - A).\mathbf{n}$ a $d_r = (R - A).\mathbf{n}$ (Obr 7).

Potom existuje niekoľko prípadov v závislosti od hodnoty d_l:

1. $d_l > 0$: Potom $\alpha \geq \frac{d_r}{d_l}$. Aby sme to zabezpečili, stačí položiť: $\alpha_0 = max(\alpha_0, \frac{d_r}{d_l})$.

   Ak je $\alpha_0 > \alpha_1$ ako výsledok dostaneme, že orezávaná úsečka je prázdna a algoritmus končí.

2. $d_l < 0$: Potom $\alpha \leq \frac{d_r}{d_l}$. Aby sme to zabezpečili, stačí položiť: $\alpha_1 = min(\alpha_1, \frac{d_r}{d_l})$.

   Ak ako výsledok dostaneme $\alpha_0 > \alpha_1$, tak opäť orezávaná úsečka je prázdna a algoritmus končí.

3. $d_l = 0$: Potom α nie je definované. Geometricky to znamená, že hraničná priamka a orezávaná úsečka sú rovnobežné. V tomto prípade stačí vziať ľubovoľný bod na úsečke napr. A a ak bude $(A - R).\mathbf{n} < 0$, tak bude jasné, že celá úsečka je mimo polroviny, a preto prienik s mnohouholníkom je prázdny. Inak neurobíme nič, čiže pokračujeme v orezávaní ďalšími polrovinami.

Vo všeobecnosti je algoritmus Liang-Barsky efektívnejší ako algoritmus Cohen-Sutherland, pretože netreba počítať prienik úsečky a okna. Každá aktualizácia parametru u vyžaduje iba operáciu delenia a prienik úsečky a okna sa počíta iba jedenkrát. Taktiež Cohen-Sutherland môže hľadať prienik okna s úsečkou aj keď tá sa nachádza mimo okna. Oba tieto algoritmy je možné rozšíriť do trojdimenzionálneho priestoru.

Postup

Na ilustráciu algoritmu Liang-Barsky si ukážeme orezanie úsečky s krajnými bodmi $A = (x_A, y_A)$ a $B = (x_B, y_B)$. Je dané okno, kde $x \in < x_{min}, x_{max} >$ a $y \in < y_{min}, y_{max} >$. Určenie polrovín si vyžaduje voľbu minimálne dvoch bodov, $R_0 = (x_{min}, y_{min})$ a $R_1 = (x_{max}, y_{max})$ a dvoch vektorov $\boldsymbol{e_x} = (1, 0)$ a $\boldsymbol{e_y} = (0, 1)$.

1. Vytvoríme nasledujúce štyri polroviny:

$$H_1 = \langle R_0, \boldsymbol{e_x} = (1,0)\rangle; \qquad H_3 = \langle R_1, -\boldsymbol{e_x} = (-1,0)\rangle$$
$$H_2 = \langle R_0, \boldsymbol{e_y} = (0,1)\rangle \qquad H_4 = \langle R_1, -\boldsymbol{e_y} = (0,-1)\rangle$$

2. Inicializácia $\alpha_0 = 0$ a $\alpha_1 = 1$.

3. Pre každú polrovinu $H = \langle R, \boldsymbol{n}\rangle$;

   (a) Vypočítaj $d_l = (B - A).\mathbf{n}$ a $d_r = (R - A).\mathbf{n}$

   (b) Ak $d_l > 0$, tak $\alpha \geq \frac{d_r}{d_l}$ a $\alpha_0 = max(\alpha_0, \frac{d_r}{d_l})$.

   (c) Ak $d_l < 0$, tak $\alpha \leq \frac{d_r}{d_l}$ a $\alpha_1 = min(\alpha_1, \frac{d_r}{d_l})$.

   (d) Ak $d_l = 0$, tak α nie je definované. Ak $(A-R).\mathbf{n} < 0$, a úsečka je mimo polroviny, a preto prienik s mnohouholníkom je prázdny. Inak pokračujeme v orezávaní ďalšími polrovinami.

   (e) Ak je $\alpha_0 > \alpha_1$ ako výsledok dostaneme, že orezávaná úsečka je prázdna a algoritmus končí

4. Krajné body orezanej úsečky sú:

   - $A = P(\alpha_0) = (1 - \alpha_0)A + \alpha_0 B$
   - $B = P(\alpha_1) = (1 - \alpha_1)A + \alpha_1 B$

Príklad

Na ilustráciu algoritmu Liang-Barsky si ukážeme orezanie úsečky s krajnými bodmi $A = (x_A, y_A) = (0, 8)$ a $B = (x_B, y_B) = (16, 0)$.

Je dané okno, kde $x \in < 4, 10 >$ a $y \in < 2, 9 >$. Určenie polrovín si vyžaduje voľbu minimálne dvoch bodov, $R_0 = (4, 2)$ a $R_1 = (10, 9)$ a dvoch vektorov $\boldsymbol{e_x} = (1, 0)$ a $\boldsymbol{e_y} = (0, 1)$.

1. Vytvoríme nasledujúce štyri polroviny:

$$H_1 = \langle R_0 = (4,2,), \boldsymbol{e_x} = (1,0)\rangle; \qquad H_3 = \langle R_1 = (10,9), -\boldsymbol{e_x} = (-1,0)\rangle$$
$$H_2 = \langle R_0 = (4,2,), \boldsymbol{e_y} = (0,1)\rangle \qquad H_4 = \langle R_1 = (10,9), -\boldsymbol{e_y} = (0,-1)\rangle$$

2. Inicializácia $\alpha_0 = 0$ a $\alpha_1 = 1$.

3. $H_1 = \langle R_0, \boldsymbol{e_x}\rangle$:
   $d_l = (B - A).\boldsymbol{e_x} = (16, -8).(1, 0) = 16 > 0$
   $d_r = (R_0 - A)\boldsymbol{e_x} = (4, -6).(1, 0) = 4 \Rightarrow \alpha \geq \frac{1}{4}$
   $d_l > 0 : \alpha_0 = max(0, \frac{1}{4}) = \frac{1}{4}$
   Čiastkový výsledok: $(\alpha_0 = \frac{1}{4}, \alpha_1 = 1)$

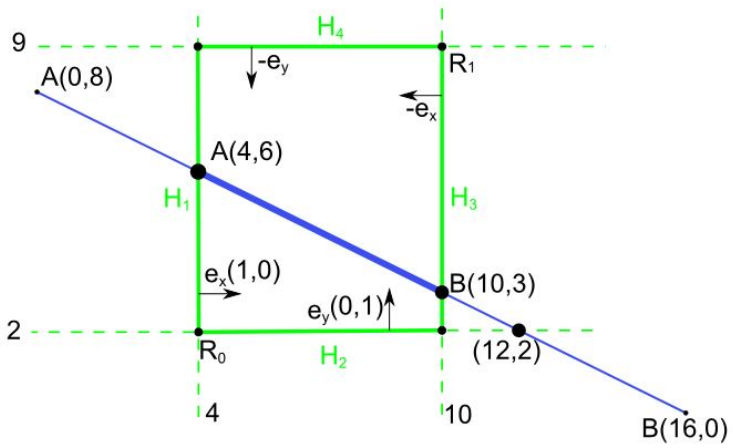
4. $H_2 = \langle R_0, \boldsymbol{e_y}\rangle$:
   $d_l = (B - A).\boldsymbol{e_y} = (16, -8).(0, 1) = -8 < 0$
   $d_r = (R_0 - A)\boldsymbol{e_y} = (4, -6).(0, 1) = -6 \Rightarrow \alpha \leq \frac{3}{4}$
   $d_l < 0 : \alpha_1 = min(1, \frac{3}{4}) = \frac{3}{4}$
   Čiastkový výsledok: $(\alpha_0 = \frac{1}{4}, \alpha_1 = \frac{3}{4})$

5. $H_3 = \langle R_1, -\boldsymbol{e_x} \rangle$:
   $d_l = (B - A).-\boldsymbol{e_x} = (16, -8).(-1, 0) = -16 < 0$
   $d_r = (R_1 - A).-\boldsymbol{e_x} = (10, 1).(-1, 0) = -10 \Rightarrow \alpha \leq \frac{5}{8}$
   $d_l < 0 : \alpha_1 = min(\frac{3}{4}, \frac{5}{8}) = \frac{5}{8}$
   Čiastkový výsledok: $(\alpha_0 = \frac{1}{4}, \alpha_1 = \frac{5}{8})$

6. $H_4 = \langle R_1, -\boldsymbol{e_y} \rangle$:
   $d_l = (B - A).-\boldsymbol{e_y} = (16, -8).(0, -1) = 8 > 0$
   $d_r = (R_1 - A).-\boldsymbol{e_y} = (10, 1).(0, -1) = -1 \Rightarrow \alpha \geq -\frac{1}{8}$
   $d_l > 0 : \alpha_0 = max(\frac{1}{4}, -\frac{1}{8}) = \frac{1}{4}$

7. Výsledok je $(\alpha_0 = \frac{1}{4}, \alpha_1 = \frac{5}{8})$. Teda krajné body orezanej úsečky sú:
   - Pre $\alpha_0 = \frac{1}{4} : \frac{3}{4}(0, 8) + \frac{1}{4}(16, 0) = (4, 6) = A$
   - Pre $\alpha_1 = \frac{5}{8} : \frac{3}{8}(0, 8) + \frac{5}{8}(16, 0) = (10, 3) = B$

8. Výslednú orezanú úsečku môžete vidieť na Obr 8.



Obr. 8: Výsledná orezaná úsečka algoritmom Cohen-Sutherland

Literatúra

Algoritmus Liang-Barsky vieme nájsť podrobne spracovaný v anglickej literatúre [3] od strany 230. Zo slovenskej a českej literatúry sa nenachádza v [1] ani v [2].

1.3.2 Algoritmus Cyrus-Beck

Princíp algoritmu

Staršia verzia Liang-Barského algoritmu je známa ako algoritmus Cyrus-Beck. Je to rýchlejšia metóda ako Algoritmus Cohen-Sutherland.

Je to orezávanie do konkrétneho okna, no nie nutne axiálneho.

Je založený na hľadaní tzv. vstupného bodu úsečky AB do okna, ktorý je určený maximálnym vstupným parametrom t_e a výstupného bodu úsečky AB z okna, ktorý je určený zasa minimálnym výstupným parametrom t_o v intervale $< 0, 1 >$. Predpokladá sa, že úsečka AB je určená parametrickým vyjadrením:

$$P(t) = A + dt, \tag{1.4}$$

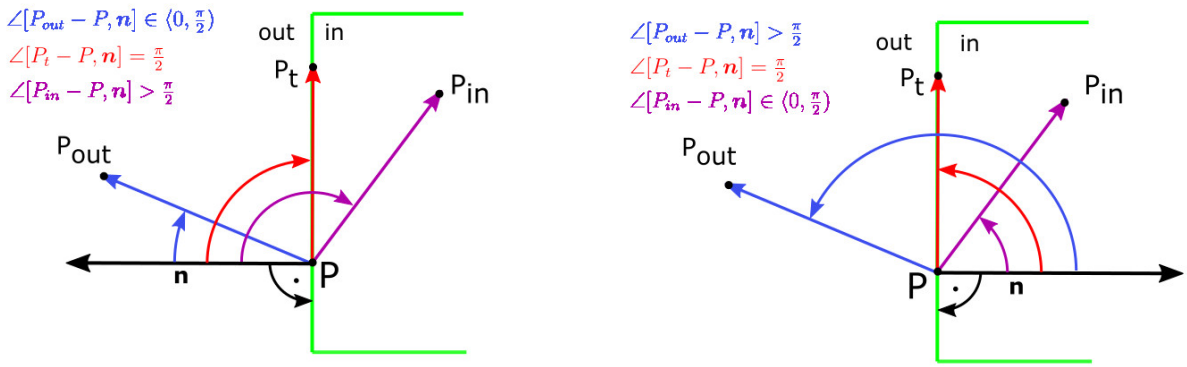
kde vektor $s = B - A$ a $t \in < 0, 1 >$ a chceme určiť jej prienik so stranou E (edge) konvexného mnohouholníka určenou bodom P a vonkajším normálovým vektorom **n**.

Vo všeobecnosti pre uhol vektora a vonkajší normálový vektor strany mnohouholníka platí:

- $\angle[P_{out} - P, \mathbf{n}] \in \langle 0, \frac{\pi}{2})$
- $\angle[P_{in} - P, \mathbf{n}] > \frac{\pi}{2}$
- $\angle[P_t - P, \mathbf{n}] = \frac{\pi}{2}$

a pre uhol vektora a vnútorný normálový vektor strany mnohouholníka platí:

- $\angle[P_{out} - P, \mathbf{n}] > \frac{\pi}{2}$
- $\angle[P_{in} - P, \mathbf{n}] \in \langle 0, \frac{\pi}{2})$
- $\angle[P_t - P, \mathbf{n}] = \frac{\pi}{2}$



(a) Vonkajší normálový vektor

(b) Vnútorný normálový vektor

Obr. 9: Orezávanie s použitím normálového vektora

Platí:

1. $P(t)$ je vonkajším bodom, ak $\angle[P(t) - P, \boldsymbol{n}] \in \langle 0, \frac{\pi}{2}) \Leftrightarrow [P(t) - P].\boldsymbol{n} > 0$
2. $P(t)$ je vnútorným bodom, ak $\angle[P(t) - P, \boldsymbol{n}] > \frac{\pi}{2} \Leftrightarrow [P(t) - P].\boldsymbol{n} < 0$

3. $P(t)$ leží na strane E, ak $\angle[P(t)-P,\boldsymbol{n}]=\frac{\pi}{2} \Leftrightarrow [P(t)-P].\boldsymbol{n}=0 \Leftrightarrow (A-P+t.\boldsymbol{d}).\boldsymbol{n}=0 \Leftrightarrow (A-P).\boldsymbol{n}+t.(\boldsymbol{d.n})=0 \Leftrightarrow \left[(\boldsymbol{d.n})\neq 0 \wedge t=-\frac{(A-P).\boldsymbol{n}}{(\boldsymbol{d.n})}\right] \vee \left[(\boldsymbol{d.n})=0 \wedge (A-P).\boldsymbol{n}=0\right] \vee \left[(\boldsymbol{d.n})=0 \wedge (A-P).\boldsymbol{n}\neq 0\right]$. Teda $t=-\frac{(A-P).\boldsymbol{n}}{(\boldsymbol{d.n})}$, kde $(\boldsymbol{d.n})\neq 0$.

Pokiaľ priamka AB obsahuje stranu mnohouholníka t.j. prienikom priamky a mnohouholníka je táto strana, čiže strana je orezaním priamky AB mnohouholníkom – algoritmus končí.

Ak priamka AB a strana E nemajú spoločný ani jeden bod, čiže priamka je so stranou E rovnobežná, pričom môže, ale nemusí pretínať ďalšie strany – prejdeme k ďalšej strane.

Môžu nastať dve situácie:

1. Ak $\boldsymbol{d.n} < 0$, nazveme ho potenciálne vstupným parametrom a označíme t_e. Bod $P(t_e)$ budeme nazývať potenciálne vstupným bodom úsečky AB do okna.

2. Ak $\boldsymbol{d.n} > 0$, nazveme ho potenciálne výstupným parametrom a označíme t_o. Bod $P(t_o)$ budeme nazývať potenciálne výstupným bodom úsečky AB z okna.

Nevýhodou tohto algoritmu je, že nevieme ošetriť prípad, keď sa úsečka nachádza celá mimo mnohouholníka.

Postup

Na ilustráciu algoritmu Cyrus-Beck si ukážeme orezanie úsečky s krajnými bodmi $A=(x_A,y_A)$ a $B=(x_B,y_B)$ do mnohouholníka určeného vrcholmi A_i.

1. Zvolíme si jednu z dvoch orientácií mnohouholníka, nech je kladná, t.j. proti smeru hodinových ručičiek. Uvažujeme vnútorné normály strán mnohouholníka.

2. Inicializácia $t_e=0, t_o=1$, urči smerový vektor orezávanej úsečky $\boldsymbol{s}=B-A$.

3. Pre každý vektor hrany $\boldsymbol{e_i}$ a jemu prislúchajúci normálový vektor $\boldsymbol{n_i}$ vykonáme:

   (a) Urči vektor $\boldsymbol{e_i}$ a normálový vektor danej hrany $\boldsymbol{n_i}$ a jeden bod ležiaci na hrane, nech je to bod A_i

   (b) Urči skalárny súčin $u=\langle \boldsymbol{s},\boldsymbol{n_i}\rangle$

   (c) Urči $t=\frac{\langle P_i-A,\boldsymbol{n_i}\rangle}{u}$

   (d) Ak $u<0$ platí $t_o=min(t_o,t)$. Ak $u>0$ platí $t_e=max(t_e,t)$.

4. Výsledná orezaná úsečka má krajné body:

   - Pre t_e: $P(t_e)=A+t_e(B-A)$
   - Pre t_o: $P(t_o)=A+t_o(B-A)$

Príklad

Na ilustráciu algoritmu Cyrus-Beck si ukážeme orezanie úsečky s krajnými bodmi $A=(x_A,y_A)=(-2,2)$ a $B=(x_B,y_B)=(8,3)$ do mnohouholníka určeného vrcholmi A_i, kde $A_1=(2,1)$, $A_2=(5,2)$, $A_3=(4,5)$ a $A_4=(2,3)$.

1. Zvolíme si jednu z dvoch orientácií mnohouholníka, nech je kladná, t.j. proti smeru hodinových ručičiek. Uvažujeme vnútorné normály strán mnohouholníka.

2. $t_e=0, t_o=1$ a $\boldsymbol{s}=(8,3)-(-2,2)=(10,1)$.

3. $i = 1:$ (a) $\boldsymbol{e_1} = A_2 - A_1 = (5,2) - (2,1) = (3,1)$, $\boldsymbol{n_1} = (-1,3)$, $P_1 = (3,1)$.
   (b) $u = \langle \boldsymbol{s}, \boldsymbol{n_1} \rangle = \langle (10,1), (-1,3) \rangle = -7$
   (c) $t = \frac{\langle P_1 - A, \boldsymbol{n_1} \rangle}{u} = \frac{\langle (5,-1),(-1,3) \rangle}{-7} = \frac{8}{7}$
   (d) $u < 0$, teda $t_o = min(1, \frac{8}{7}) = 1$.

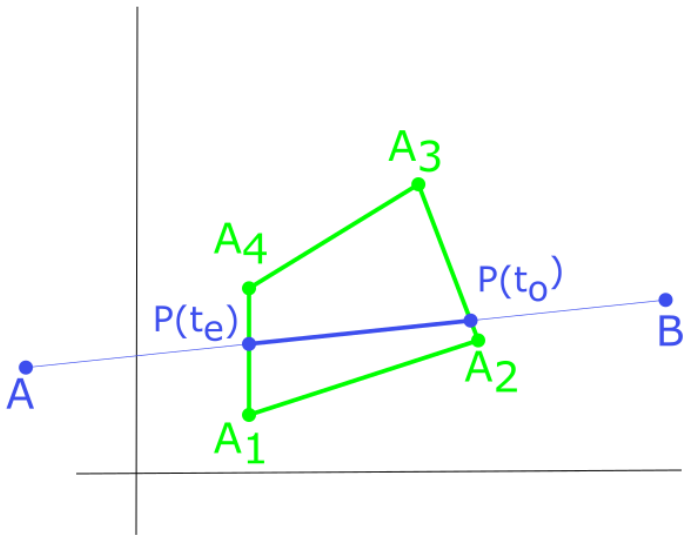
   $i = 2:$ (a) $\boldsymbol{e_2} = A_3 - A_2 = (4,5) - (5,2) = (-1,3)$, $\boldsymbol{n_2} = (-3,-1)$, $P_2 = (5,2)$.
   (b) $u = \langle \boldsymbol{s}, \boldsymbol{n_2} \rangle = \langle (10,1), (-3,-1) \rangle = -31$
   (c) $t = \frac{\langle P_2 - A, \boldsymbol{n_2} \rangle}{u} = \frac{\langle (7,0),(-3,-1) \rangle}{-31} = \frac{21}{31}$
   (d) $u < 0$, teda $t_o = min(1, \frac{21}{31}) = \frac{21}{31}$.

   $i = 3:$ (a) $\boldsymbol{e_3} = A_4 - A_3 = (2,3) - (4,5) = (-2,2)$, $\boldsymbol{n_3} = (2,-2)$, $P_3 = (4,5)$.
   (b) $u = \langle \boldsymbol{s}, \boldsymbol{n_3} \rangle = \langle (10,1), (2,-2) \rangle = 18$
   (c) $t = \frac{\langle P_3 - A, \boldsymbol{n_3} \rangle}{u} = \frac{\langle (6,3),(-3,-1) \rangle}{18} = \frac{1}{3}$
   (d) $u < 0$, teda $t_o = min(\frac{21}{31}, \frac{1}{3}) = \frac{21}{31}$.

   $i = 4:$ (a) $\boldsymbol{e_4} = A_1 - A_4 = (2,1) - (2,3) = (0,-2)$, $\boldsymbol{n_4} = (2,0)$, $P_4 = (2,3)$.
   (b) $u = \langle \boldsymbol{s}, \boldsymbol{n_4} \rangle = \langle (10,1), (2,0) \rangle = 20$
   (c) $t = \frac{\langle P_4 - A, \boldsymbol{n_4} \rangle}{u} = \frac{\langle (4,1),(2,0) \rangle}{20} = \frac{2}{5}$
   (d) $u > 0$, teda $t_e = max(0, \frac{2}{5}) = \frac{2}{5}$.

4. Krajné body orezanej úsečky:
   - Pre t_e: $P(t_e) = (-2,2) + \frac{21}{31}(10,1) = (\frac{148}{31}, \frac{83}{31})$
   - Pre t_o: $P(t_o) = (-2,2) + \frac{2}{5}(10,1) = (2, \frac{12}{5})$

5. Výslednú orezanú úsečku môžete vidieť na Obr. 10.



Obr. 10: Výsledná orezaná úsečka algoritmom Cyrus-Beck

Pseudokód

Implementácia algoritmu Cyrus-Beck je zhrnutá v nasledujúcom pseudokóde. Na vstupe sú dva krajné body úsečky A a B.

```
Cyrus-Beck (int x_A, int y_A, int x_B, int y_B)
{

  floor t_e, t_o, t;
  if (A = B) then orez-bod; {úsečka = bod, orež-bod}
  else {
    t_e = 0;
    t_o = 1;
    d = B − A;
    {pre každú hranu okna a jej vonkajší normálový vektor}
    for(i = 1; i ≤ 4; i++){
      if (n_i.d_i ≠ 0) t = (−n_i.(A−P_i)) / (n_i.d_i);
      if (n_i.d_i < 0 ∧ t ≤ 1) t_e = max(t_e, t);
      if (n_i.d_i > 0 ∧ t ≥ 0) t_o = max(t_o, t);
    }
    if (t_e > t_o) return; {úsečka mimo okna}
    else Vykresli-usecku(P(t_e), P(t_o));
}
```

Literatúra

Môžeme sa s ním stretnúť v [1] od strany 28 pod názvom parametrické orezávanie a v českej literatúre [2] môžeme nájsť vysvetlenie tohto algorimtmu od strany 107. V anglickej literatúre [3] sa tento algoritmus neuvádza.

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