OREZÁVANIE

Pod orezávaním rozumieme hľadanie časti objektu, ktorá je viditeľná v okne. Treba teda nájsť prienik objektu a okna. Štandardne má okno tvar axiálneho obdĺžnika, čiže jeho hrany sú rovnobežné so súradnicovými osami.

Samozrejme by sme mohli použiť postupy, ktoré sme si uviedli v časti o hľadaní prienikov. Orezávanie je špeciálny prípad počítania prienikov. Ide ale o veľmi často sa vyskytujúci špeciálny prípad, takže sa oplatí mať k dispozícii špeciálne algoritmy, ktoré túto prácu urýchlia.

Okno, pokiaľ nebude uvedené inak, bude mať tvar obdĺžnika, ktorého vrcholy majú súradnice

$$(x_{min}, y_{min}), \quad (x_{max}, y_{min}), \quad (x_{max}, y_{max}) \quad \text{a} \quad (x_{min}, y_{max}).$$

Bod so súradnicami (x, y) tak leží v okne, keď

$$x_{min} \leq x \leq x_{max} \quad \text{a} \quad y_{min} \leq y \leq y_{max}.$$

1. Orezávanie úsečky

1.1. Cohen–Sutherland.

Tento algoritmus je veľmi populárny, často sa implementuje.

Priamky, na ktorých ležia strany obdĺžnika predstavujúceho okno, rozdelia celú rovinu na 9 oblastí. Každej z týchto oblastí priradíme štvorbitový kód, kde každý bit nejakým spôsobom popisuje polohu oblasti vzhľadom na okno. Napríklad, prvý bit (menované sprava) hovorí, či sa oblasť nachádza vľavo od okna, druhý bit hovorí, či sa oblasť nachádza pravo od okna, tretí zas hovorí, či sa nachádza nahor a štvrtý nadol od okna, viď obrázok.

1001	1000	1010
0001	0000	0010
0101	0100	0110

Obr. 1. Priradenie štvorbitových kódov oblastiam.

Pre úsečku AB, ktorú chceme orezať, najprv spracujeme jej koncové body tak, že každému priradíme štvorbitový kód oblasti, v ktorej sa tento bod nachádza. Než začneme hľadať prienik úsečky a strán okna, skúsime pomocou kódov priradeným jej koncovým bodom najprv zistiť, či úsečku naozaj treba orezávať. Môžu nastať prípady (symbolom kod(P) označujeme štvorbitový kód priradený bodu P):

- kod(A) = 0000 a tiež kod(B) = 0000. Vtedy oba koncové body úsečky ležia vnútri okna a úsečku netreba orezávať.
- kod(A) & kod(B) $\neq$ 0000, kde & označuje logickú operáciu „a súčasne" po jednotlivých bitoch. V tomto prípade oba koncové body úsečky ležia na jednej strane od okna, a úsečka sa preto nachádza celá mimo okna. Znovu netreba hľadať prieniky.
- Ak nenastal ani jeden z dvoch predchádzajúcich prípadov, úsečku priamkou určenou niektorou hranou okna rozdelíme na dve podúsečky, a každú z týchto podúsečiek potom orezávame odznova – priradíme koncovým bodom kódy a testujeme.

Príklad 1.1. Koncovým bodom úsečky AB na obrázku priradíme kódy:

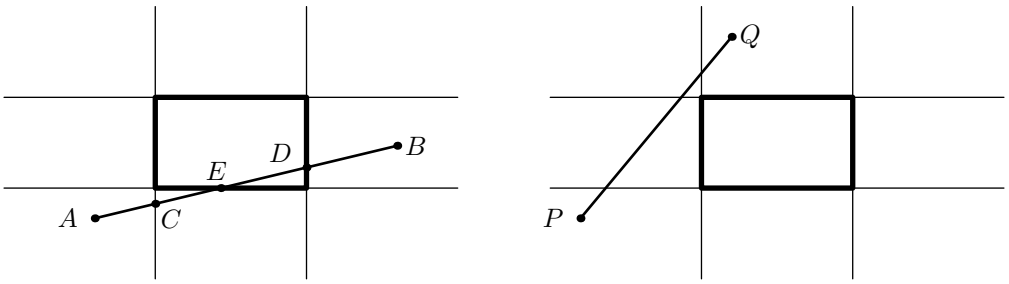
$$\text{kod}(A) = 0101, \quad \text{kod}(B) = 0010.$$

Kódy sú nenulové, teda koncové body úsečky ležia mimo okna. Tiež platí, že

$$\text{kod}(A) \ \& \ \text{kod}(B) = 0000,$$

takže úsečku budeme deliť na časti. Keďže bod A leží vľavo od okna (najpravejší bit je 1) a bod B nie, rozdelíme úsečku priamkou ľavej hrany na úsečky AC a CB. Podľa kódov koncových bodov nových úsečiek zistíme, že úsečka AC sa celá nachádza pod oknom, preto ju z ďalšieho uvažovania vylúčime. Úsečku BC však musíme orezávať ďalej. Bod B sa na rozdiel od bodu C nachádza vpravo od okna, preto príslušnou priamkou rozdelíme úsečku na CD a DB. Úsečku DB vylúčime, úsečku CD rozdelíme ešte na dve časti. Výsledkom je orezaná úsečka ED.

Príklad 1.2. Úsečka PQ sa celá nachádza mimo okna. Kým to však pomocou Cohen–Sutherlandovho algoritmu zistíme, určite úsečku aspoň raz rozdelíme na dve časti.



Obr. 2. Orezávanie úsečky.

Tento algoritmus je možné modifikovať tak, že keď úsečku musíme deliť na dve časti, tak ju nerozdelíme v bode, kde pretína niektorú z priamok obsahujúcich hrany okna, ale ju rozdelíme na polovicu. Takto delíme, až kým o každej časti nevieme rozhodnúť, že celá leží vnútri alebo zvonka okna, alebo kým nedosiahneme rozlíšenie výstupného zariadenia. Výhodou je, že sa vyhneme počítaniu prieniku úsečky a priamky. Nevýhodou je, že úsečku budeme takto deliť viackrát, než pri pôvodnom prístupe.

Tiež si všimnime, že algoritmus sa dá veľmi ľahko modifikovať do 3-rozmerného priestoru, kde budeme úsečku orezávať na axiálny kváder. Stačí, keď 4-bitové kódy nahradíme 6-bitovými.

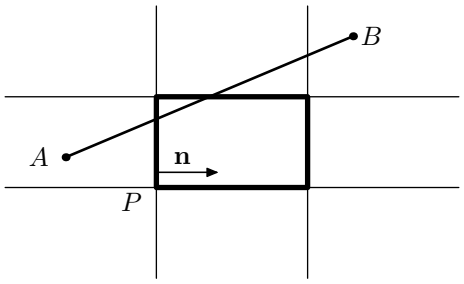
1.2. **Cyrus–Beck (parametrické orezávanie).** Tento algoritmus je priamou aplikáciou hľadania prieniku úsečky a konvexného mnohouholníka, keď za mnohouholník zoberieme okno. Úsečku si pre tento algoritmus parametrizujeme: Ak $A = (x_1, y_1)$ a $B = (x_2, y_2)$ sú koncové body úsečky, tak

$$\begin{aligned} x &= x_1 + \Delta x \cdot t \\ y &= y_1 + \Delta y \cdot t, \quad s \in \langle 0, 1 \rangle \end{aligned}$$

kde $\Delta x = x_2 - x_1$, $\Delta y = y_2 - y_1$, je jej parametrizácia. Pre každú priamku obsahujúcu niektorú hranu okna vypočítame parameter t_i, ktorý zodpovedá prieniku úsečky s touto priamkou, čiže bod $(x_1, y_1)+(\Delta x, \Delta y)t_i$ je priesečník. Navyše o každom tomto priesečníku rozhodneme, či úsečka týmto bodom potenciálne vstupuje do okna, alebo z neho vystupuje. Orezaná úsečka potom leží medzi posledným bodom, ktorým pôvodná úsečka vstupuje, a prvým, ktorým vystupuje. Po rozpísaní detailov tak máme nasledovný výpočet:

Nech $\mathbf{n}_i$ označuje normálu i-tej strany okna tak, že táto smeruje dovnútra okna. Úsečka AB do okna vstupuje, ak pre skalárny súčin platí

$$(B - A) \cdot \mathbf{n}_i > 0.$$



Obr. 3. Parametrické orezávanie – algoritmus Cyrus–Becka.

Keď je tento skalárny súčin záporným tak úsečka z okna vystupuje. Pre priamku, na ktorej leží i-ta strana okna, zoberieme jej všeobecnú rovnicu v tvare

$$(X - P) \cdot \mathbf{n}_i = 0,$$

kde P je ľubovoľný bod na priamke (napríklad niektorý z rohov okna). Bod $X = A + (B - A)t$ úsečky AB leží na tejto priamke, keď spĺňa jej rovnicu, teda keď platí

$$((A + (B - A)t) - P) \cdot \mathbf{n}_i = 0.$$

Odtiaľ už ľahko dostávame podmienku pre t:

$$t = \frac{(P - A) \cdot \mathbf{n}_i}{(B - A) \cdot \mathbf{n}_i}.$$

Tak isto, ako pri hľadaní prieniku úsečky s konvexným mnohouholníkom, nájdeme

$$t_{in} = \max(\{0\} \cup \{t_i \mid (B - A) \cdot \mathbf{n}_i > 0\}),$$
$$t_{out} = \min(\{1\} \cup \{t_i \mid (B - A) \cdot \mathbf{n}_i < 0\}).$$

Ak $t_{in} > t_{out}$, tak celá úsečka leží mimo okna. V opačnom prípade dopočítame súradnice koncových bodov $A + (B - A)t_{in}$ a $A + (B - A)t_{out}$ orezanej úsečky.

Výhodou parametrického orezávania je, počítame iba skutočné koncové body orezanej úsečky. Pre ostatné body priebežne počítame len hodnotu parametra, čo je iba jedno číslo na rozdiel od dvoch čísel, keď rátame obe súradnice priesečníka ako v Cohen–Sutherlandovom algoritme.

1.3. **Liang–Barsky.** Myšlienka tohto algoritmu je presne tá istá ako v predchádzajúcom prípade. Tento algoritmus je vlastne len dotiahnutím parametrického orezávania s využitím faktu, že strany okna sú rovnobežné so súradnicovými osami.

Úsečka bude parametrizovaná presne ako v predchádzajúcom prípade. Hraničné hodnoty okna sú $x_{min}, x_{max}, y_{min}, y_{max}$. Bod úsečky $X = A + (B - A)t$ teda leží v okne práve vtedy, keď

$$x_{min} \le x_1 + \Delta x \cdot t \le x_{max} \quad \text{a} \quad y_{min} \le y_1 + \Delta y \cdot t \le y_{max}.$$

Máme tak štyri nerovnice, ktoré všetky upravíme na spoločný tvar $p_i t \le q_i$:

$$\begin{aligned} -\Delta x \cdot t &\le x_1 - x_{min} \\ \Delta x \cdot t &\le x_{max} - x_1 \\ -\Delta y \cdot t &\le y_1 - y_{min} \\ \Delta y \cdot t &\le y_{max} - y_1 \end{aligned}$$

Upresníme teraz postupy z parametrického orezávania:

- Ak pre niektoré i platí $p_i = 0$, tak úsečka je s príslušnou stranou okna rovnobežná – ani nevchádza dnu ani nevychádza von z okna. Navyše ak príslušné $q_i < 0$, tak hneď usúdime, že celá úsečka sa nachádza mimo okna.
- Ak $p_i < 0$, tak úsečka príslušnou stranou potenciálne vstupuje do okna. Ak $p_i > 0$, tak úsečka z okna vystupuje.

- Keď $p_i \neq 0$, vypočítame parameter zodpovedajúci priesečníku: $t_i = q_i/p_i$.

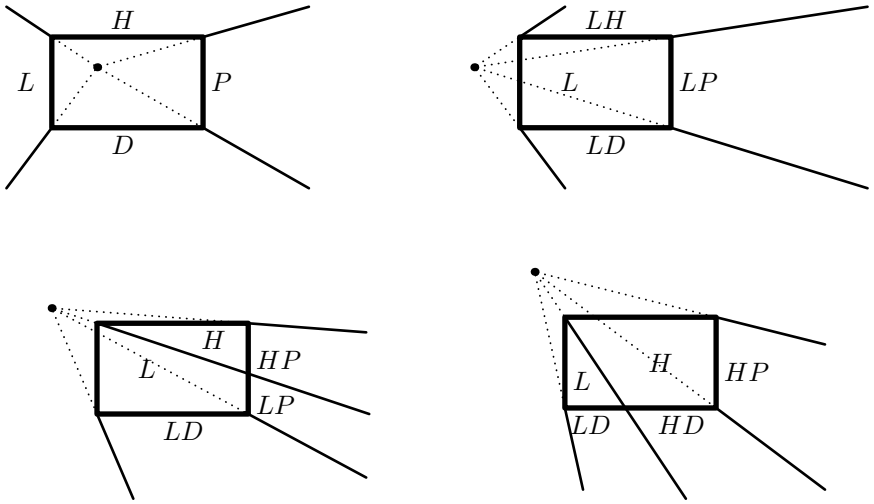
Na záver vypočítané parametre vyhodnotíme:

$$t_{in} = \max(\{0\} \cup \{t_i \mid (B - A) \cdot \mathbf{n}_i > 0\}),$$
$$t_{out} = \min(\{1\} \cup \{t_i \mid (B - A) \cdot \mathbf{n}_i < 0\}).$$

Podobne ako algoritmus Cohen–Sutherlanda sa dajú aj algoritmy Cyrus–Becka a Liang–Barskeho bez väčších komplikácií aplikovať na orezávanie úsečky v trojrozmernom priestore.

1.4. **Nicholl–Lee–Nichol.** Tento algoritmus orezávania úsečky vykonáva v porovnaní s ostatnými uvedenými najmenej výpočtov hodnôt. Namiesto toho sa najprv viacerými testami urobí dôkladnejšia analýza situácie, aby sa potom spočítali len tie priesečníky, ktoré sú naozaj potrebné. Jeho nevýhodou zase je, že nie je ľahké ho bez väčších problémov rozšíriť do trojrozmerného priestoru.

Pri tomto prístupe najprv zistíme, v ktorej časti roviny vzhľadom na okno sa nachádza prvý koncový bod úsečky. Rovina je rozdelená na časti podobne ako pri algoritme Cohen–Sutherlanda. Potom podľa vzájomnej polohy tohto bodu a okna rozdelíme rovinu stranami okna a polpriamkami z bodu cez rohy okna na niekoľko častí. Tieto sú na obrázku označené každá jedným alebo dvoma písmenami, pričom tieto písmená napovedajú, ktorými stranami okna treba úsečku orezať (L – ľavou, P – pravou, H – hornou, D – dolnou), keď sa druhý koncový bod nachádza v popísanej oblasti. Na záver zrátame potrebné priesečníky. Polohy druhého koncového bodu v



Obr. 4. Príklady delenia roviny v algoritme Nicholl–Lee–Nichol.

rovine zistíme porovnaním sklonov polpriamok a úsečky. Napríklad v situácii na obrázku vpravo hore za druhý koncový bod (x_2, y_2) nachádza v oblasti označenej LH, ak platí, že $x_2 > x_{max}$ a

$$\frac{y_{max} - y_1}{x_{max} - x_1} < \frac{y_2 - y_1}{x_2 - x_1} < \frac{y_{max} - y_1}{x_{min} - x_1}.$$

Všetky hodnoty vypočítané počas zaraďovania druhého koncového bodu do oblasti sú uchovávané, aby mohli byť neskôr znova použité.

2. Orezávanie mnohouholníka

Úlohou je pre mnohouholník zadaný zoznamom svojich vrcholov na hranici nájsť jeho prienik s oknom, pričom tento by mal byť tiež vyjadrený zoznamom vrcholov.

2.1. **Sutherland–Hodgeman.**

2.2. **Weiler–Atherton.** KAGDM FMFI UK Bratislava
Email address: jana.pilnikova@fmph.uniba.sk