Algoritmy v digitální kartografii

Geometrické vyhledávání bodu

Tereza Kulovaná
Markéta Pecenová

Obsah

1 Zadání

Vstup: Souvislá polygonová mapa n polygonů $\{P_1, ..., P_n\}$, *analyzovaný bod* q.

Výstup: P_i, $q \in P_i$.

Nad polygonovou mapou implementujete následující algoritmy pro geometrické vyhledávání:

- Ray Crossing Algorithm (varianta s posunem těžiště polygonu).
- Winding Number Algorithm.

Nalezený polygon obsahující zadaný bod q graficky zvýrazněte vhodným způsobem (např. vyplněním, šrafováním, blikáním). Grafické rozhraní vytvořte s využitím frameworku QT.

Pro generování nekonvexních polygonů můžete navrhnout vlastní algoritmus či použít existující geografická data (např. mapa evropských států).

Polygony budou načítány z textového souboru ve Vámi zvoleném formátu. Pro datovou reprezentaci jednotlivých polygonů použijte špagetový model.

Hodnocení:

Krok	**Hodnocení**
Detekce polohy bodu rozlišující stavy uvnitř, vně na hranici polygonu.	10b
Ošetření singulárního případu u Winding Number Algorithm: bod leží na hraně polygonu.	*+2b*
Ošetření singulárního případu u obou algoritmů: bod je totožný s vrcholem jednoho či více polygonů.	*+2b*
Zvýraznění všech polygonů pro oba výše uvedené singulární případy.	*+2b*
Algoritmus pro automatické generování nekonvexních polygonů.	*+5b*
Max celkem:	**21b**

Čas zpracování: 2 týdny.

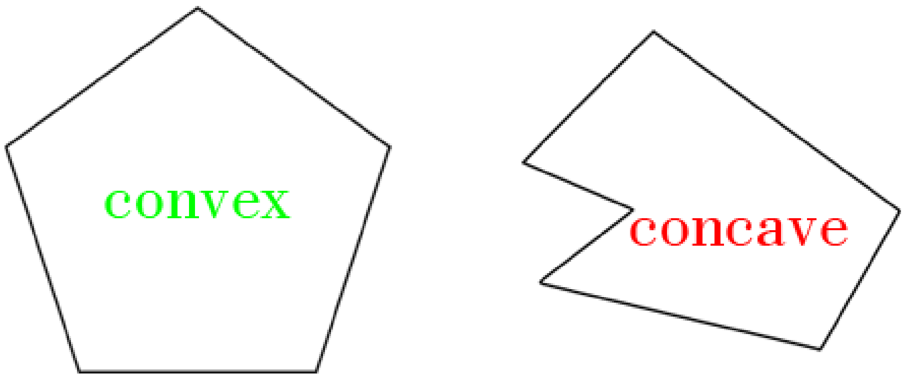
1.1 Řešené bonusové úlohy

1.

2 Popis a rozbor problému

Úloha *Geometrické vyhledávání bodu* se zabývá vytvořením aplikace, která umožní uživateli zjistit polohu jím zvoleného bodu q vzhledem k příslušnému mnohoúhelníku. Jako vhodné řešení bylo vzhledem k náročnosti problému zvoleno opakované určovnání polohy bodu q a mnohoúhelníku.

Existují dva základní druhy mnohoúhelníků (pro účely této úlohy je nazývejme polygony), konvexní a nekonvexní. Konvexní polygon je takový polygon, jehož všechny diagonály leží uvnitř polygonu a žádná z nich neprotíná jeho hranu. Konkávní polygon tuto podmínku nesplňuje. Pro představu je níže přiložen obrázek obou druhů polygonů.



Obrázek 1: Ukázka konvexního (vlevo) a konkávního polygonu (vpravo) (*zdroj*)

Z výše uvedeného vyplývá, že bod q může vůči polygonu P zaujímat 4 pozice:

1. Bod q se nalézá uvnitř polygonu P.
2. Bod q se nalézá vně polygonu P.
3. Bod q se nalézá na hraně polygonu P.
4. Bod q se nalézá ve vrcholu polygonu P.

Pro účely této aplikace byly zvoleny výpočetní algoritmy *Ray Crossing* a *Winding Number*, jejichž princip je vysvětlen v následující kapitole.

3 Algoritmy

Tato kapitola se zabývá popisem algoritmů, které byly v aplikaci implementovány.

3.1 Ray Crossing Algorithm

Prvním zvoleným algoritmem je tzv. *Ray Crossing Algorithm* neboli *Paprskový algoritmus.* Svůj název získal po metodě, jež využívá pro nalezení řešení polohy bodu vůči polygonu. Tento algoritmus je primárně využíván pro konvexní polygony, lze ho však zobecnit a využít ho i pro nekonvexní polygony.

Mějme polygon P a daný bod q. Z bodu q veďme libovolný počet polopřímek (paprsků). Princip algoritmu je založen na vyhodnocení počtu průsečíků k, které vzniknou protnutím paprsků vedených z bodu q s hranami polygonu P. Pro k mohou nastat dvě situace:

1. Hodnota k je rovna lichému číslu $\rightarrow$ bod q se nachází uvnitř polygonu P.

2. Hodnota k je rovna sudému číslu $\rightarrow$ bod q se nachází vně polygonu P.

Základní varianta algoritmu neošetřuje problémové situace, které mohou během výpočtu nastat. Konkrétně se jedná o situace, kdy je bod q totožný s jedním z vrcholů polygonu P nebo pokud bod q leží na jedné z hran polygonu P. Pro eliminaci těchto tzv. singularit je vhodné použít modifikovanou variantu algoritmu, která redukuje souřadnice bodů polygonu k bodu q.

Zjednodušený zápis takto modifikovaného algoritmu lze zapsat způsobem uvedeným níže:

1. Načtení bodů polygonu p_i, počet průsečíků $k = 0$

2. Postupně pro všechny p_i opakuj kroky 3-6

3. Redukce souřadnic bodu p_i k bodu q:
   $x'_i = x_i - x_q$
   $y'_i = y_i - y_q$

4. Podmínka $(y'_i > 0)\&\&(y'_{i-1} <= 0)\|(y'_{i-1} > 0)\&\&(y'_i <= 0)$

5. Je-li podmínka splněna: $x'_m = \frac{x'_i y'_{i-1} - x'_{i-1} y'_i}{y'_i - y'_{i-1}}$

6. Splněno $(x'_m > 0) \rightarrow k = k + 1$

7. Výpočet $k\%2$

8. Vyhodnocení k (liché k: q náleží P, sudé k: q nenáleží P)

3.2 Winding Number Algorithm

Druhý algoritmus použitý v aplikaci je tzv. *Winding Number Algorithm* neboli *Metoda ovíjení*, který je vhodný pro nekonvexní polygony. Princip tohoto algoritmu je založen na součtovém úhlu ω.

Mějme polygon P a bod q, na kterém stojí pozorovatel. Nachází-li se q uvnitř P, pak pozorovatel, který by si přál postupně vidět všechny vrcholy polygonu, se musí otočit celkem o 2π. Výsledkem algoritmu je pak tzv. Winding number ω, které říká, o kolik otáček se pozorovatel otočil:

$$\Omega = \frac{1}{2\pi} \sum_{i=1}^{n} \omega_i$$

Zde se hodí zdůraznit, že záleží na zvoleném směru otáčení. Otáčí-li se pozorovatel ve směru chodu hodinových ručiček, uhly se sčítají. V opačném směru se od sebe úhly odečítají a ω by vyšlo záporné. Během výpočtů je také nutno zavést určitou toleranci ϵ, která pokrývá chyby způsobené zaokrouhlováním. Z výše uvedených vztahů vyplývá:

1. $w = 2\pi \rightarrow q$ se nachází uvnitř P
2. $w < 2\pi \rightarrow q$ se nachází vně P

Mezi singularity pro tento algoritmus patří pouze případ, je-li $q \approx p_i$.

Zjednodušený zápis algoritmu:

1. Načtení bodů polygonu, úhel $\omega = 0$, tolerance $\epsilon = 1e - 10$
2. Postupně pro všechny orientované trojice p_i, q, p_{i+1} opakuj kroky 3-5
3. Výpočet úhlu $\omega_i = \sphericalangle p_i, q, p_{i+1}$
4. Podmínka (q je vlevo) $\rightarrow \omega = \omega + \omega_i$
5. Jinak $\omega = \omega - \omega_i$
6. Podmínka $(\mid \omega - 2\pi \mid) < \epsilon \rightarrow q$ náleží P
7. Jinak q nenáleží P

3.3 •

4 Vstupní data

Aplikace požaduje dva druhy vstupních dat:

1. soubor daných polygonů
2. daný bod q

Seznam bodů jednotlivých polygonů je uložen v textovém souboru polygon.txt. Pro vykreslení jednotlivých polygonů v aplikaci je nutno tento soubor do aplikace nahrát pomocí tlačítka *Load*. K vygenerování bodů byla použita online aplikace ze stránek mobilefish.com. Struktura souboru s polygony je následující:

První řádek: počet polygonů v souboru
Sloupec 1: číslo polygonu, jehož součástí daný bod je
Sloupec 2: souřadnice X daného bodu polygonu
Sloupec 3: souřadnice Y daného bodu polygonu

Bod q není součástí textového souboru, do aplikace vstupuje na základě ručního zadání uživatelem. Pro zadání bodu je nutné v aplikaci kliknout na tlačítko *Draw point* a levým tlačítkem myši kliknout do oblasti s polygony.

5 Výstupní data

Výstup z aplikace je

6 Aplikace

V následují kapitole je představen vizuální vzhled vytvořené aplikace tak, jak ji vidí prostý uživatel.

7 Dokumentace

Tato kapitola obsahuje dokumentaci k jednotlivým třídám.

7.1 Algorithms

Třída *Algorithms* obsahuje metody, které určují polohu zvoleného bodu q vzhledem k polygonu. Dále obsahuje pomocné metody pro výpočet úhlu mezi třemi body a na zjištění polohy bodu vzhledem k přímce.

7.1.1 getPositionRay

Metoda **getPositionRay** určuje polohu bodu q vzhledem k polygonu za použití algoritmu *Ray Crossing*. Na vstupu je bod třídy QPoint a vektor bodů polygonu. Návratová hodnota je typu *int*.

Input:

- QPoint q
- $std :: vector < QPoint > pol$

Output:

- 0 $\rightarrow$ bod se nachází vně polygonu
- 1 $\rightarrow$ bod se nachází uvnitř nebo na hraně polygonu

7.1.2 getPositionWinding

Metoda **getPositionWinding** určuje polohu bodu q vzhledem k polygonu za použití algoritmu *Winding Number*. Na vstupu je bod třídy QPoint a vektor bodů polygonu. Návratová hodnota je typu *int*.

Input:

- QPoint q
- $std :: vector < QPoint > pol$

Output:

- 0 $\rightarrow$ bod se nachází vně polygonu
- 1 $\rightarrow$ bod se nachází uvnitř polygonu
- -1 $\rightarrow$ bod se nachází na hraně polygonu

Metoda **getPointLinePosition** určuje polohu bodu q vzhledem k přímce tvořené dvěma body. Na vstupu jsou souřadnice X a Y pro 3 body typu *double*, návratová hodnota je typu *int* podle výsledku.

Input:

- double xq
- double yq
- double $x1$
- double $y1$
- double $x1$
- double $y1$

Output:

- 0 $\rightarrow$ bod se nachází vpravo od přímky
- 1 $\rightarrow$ bod se nachází vlevo od přímky
- -1 $\rightarrow$ bod se nachází na přímce

Metoda **getTwoVectorsAngle** počítá úhel

Input:

-

Output:

-

7.2 draw

8 Závěr

9 Literatura

1. *BAYER, Tomáš. Geometrické vyhledávání bodů* [online][cit. 24. 10. 2018]. Dostupné z: https://web.natur.cuni.cz

2. *What is concave & convex polygon?* [online][cit. 23. 10. 2018]. Dostupné z: https://www.nextgurukul.in

3. *Record XY mouse coordinates* [online][cit. 23. 10. 2018]. Dostupné z: https://mobilefish.com/