155ADKG: Geometrické vyhledávání bodu

Datum odevzdání: 13.11.2017

Petra Millarová, Bc. Oleksiy Maybrodskyy

Contents

1 Zadání

Následuje kopie oficiálního zadání úlohy. Autoři z nepovinných bodů zadání implementovali všechny kromě algoritmu pro automatcké generování nekonvexních polygonů.

Úloha č. 1: Geometrické vyhledávání bodu

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:

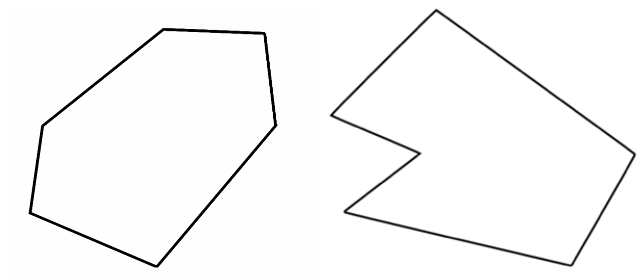
Krok	**Hodnocení**
Detekce polohy bodu rozlišující stavy uvnitř, vně na hranici polygonu.	10b
Ošetření singulárního případu u Winding Number Algorithm: bod leží na hraně polygonu.	*+2b*
Ošetření singulárního případu u obou algoritmů: bod je totožný s vrcholem jednoho či více polygonů.	*+2b*
Zvýraznění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

2 Popis a rozbor problému

Tato úloha se věnuje řešení praktického problému určování pozice uživatelem zadaného bodu q vůči polygonům načteným ze souboru. Jako implementaci si lze zjednodušeně představit zjišťování polohy konkrétního bodu kliknutím na digitální mapě.

*Nechť existuje pole ve dvojrozměrné kartézské soustavě s **n** body. Uzavřením tohoto pole vznikne polygon. Polygon může nabývat jak konvexní, tak nekonvexní tvar.*

Polygon je konvexní právě tehdy, když poloha všech bodů je vůči jakékoliv přímce procházející vedle polygonu, vždy na stejné straně.
Rozdíl mezi *k*onvexním a *n*ekonvexním polygonem je možné názorně vidět na obrázcích níže.



obr 1.:*konvexní polygon (vlevo) a nekonvexní polygon (vpravo)*

Pokud následně polygon rozdělíme na menší polygonové útvary, pak polohu zvoleného bodu q můžeme popsat následovně:

1. Bod q se nachází uvnitř polygonu $q \in P_i$
2. Bod q se nachází vně všech polygonů $q \notin P_i$
3. Bod se nachází na hraně jednoho $q \notin P_i$ nebo dvou polygonů $q \in P_{i,i+1}$
4. Bod je totožný s vrcholem jednoho polygonu nebo více polygonů $q \in P_{i,i+1,\dots,i+n}$

Výpočet se bude provádět na základě metod **Ray Crossing algorithm** a **Winding Number algorithm**. Jejich výpočet je popsán v následujících kapitolách.

3 Popisy algoritmů

V dané úloze jsou použity následující algoritmy, avšak existují i další možnosti, jak polohu bodu určit (metoda pásů, Line Sweep algorithm aj.)

3.1 Ray crossing algorithm

*Nechť existuje uzavřený polygon ve dvojrozměrné kartézské soustavě, tvořený **n** body. Nechť následně existuje bod q, kteréhož polohu se snažime určit. Proložíme-li bodem q nekonečný počet paprsků směrem k polygonu, pak pro jednotlivý paprsek nastane jedna z následujících situací:*

1. počet průsečíků paprsku k je roven sudému počtu, pak se bod q nachází vně polygonu $q \notin P_i$
2. počet průsečíků paprsku k je roven lichému počtu, pak se bod q nachází uvnitř polygonu $q \in P_i$

Zároveň mohou nastat singularity, respektive jisté situace, kdy algoritmus "nefunguje" a nedokáže přímo nalézt správný výsledek. V algoritmu ray crossing se konkrétně jedná o tyto případy:

1. Bod se nachází na hraně jednoho $q \in P_i$ nebo dvou polygonů $q \in P_{i,i+1}$
2. Bod je totožný s vrcholem jednoho polygonu nebo více polygonů $q \in P_{i,i+1,\ldots,i+n}$

Řešením je posun, respektive redukce vrcholů polygonů směrem k poloze bodu q.
Hledaný algoritmus je možné popsat následovně:

1. Inicializace bodů polygonu p_i, počet průsečíku $= 0$;
2. Redukce souřadnic x bodů polygonu k bodu q, respektivě k paprskovému segmentu, $x_i^{'} = x_i - x_q$.
3. Redukce souřadnic y bodů polygonu k bodu q, respektivě k paprskovému segmentu, $y_i^{'} = y_i - y_q$.
4. Znovu pro ostatní body daného polygonu $p_i/$
5. if $(y_i^{'} <= 0) \&\& (y_i + 1^{'} > 0) \| (y_i^{'} > 0) \&\& (y_i + 1^{'} <= 0)$.
6. $x_m^{'} = (x_i + 1^{'} y_i^{'} - x_i^{'} y_i + 1^{'})/(y_{i+1}^{'} - y_i^{'})$.
7. Sčítaní počtu redukováných bodů, pro $x_i^{'} > 0$
8. Pokud je počet průsečíku sudý, pak $q \in P$, pokud není, pak $q \notin P$

3.2 Winding Number Algorithm

*Nechť existuje uzavřený polygon ve dvojrozměrné kartézské soustavě, tvořený pomocí **n** bodů. Nechť následně existuje bod q, polohu kteréhož se snažíme určit. Z pohledu bodu q provedeme orientaci směru, ze které se pak následně určí součet všech úhlů na jednotlivé body uvedeného polygonu.*

Součtový úhel bude dále značen jako w. Výpočet je lepší provádět proti směru hodinových ručiček, jelikož v případě tohoto směru hodnota počítaných oběhů Winding Number Ω nabývá kladných hodnot. Je třeba také pamatovat, že hodnota Ω je uváděna v počtech oběhů a je záporná při oběhu po směru hodinových ručiček a kladná ve směru opačném. Do výpočtu také vstupuje tolerance ϵ, která zahrnuje chyby vzniklé zaokrouhlováním a strojovou přesností. Dle uvedených matematických podmínek mohou nastat následující případy:

1. w=**2R**, pak $q \in P_i$
2. $w <$ **2R**, pak $q \notin P_i$

Níže je uveden algoritmus výpočtu:

1. Vstup $\omega = 0$, tolerance ϵ
2. Orientace z bodu q káždého následujícího bodu p_{i+1} od orientaci na bod p_i
3. Určení úhlu $\omega_i = \angle p_i, q, p_{i+1}$
4. $\omega = \omega + \omega_i$, pro bod vprávo od orientace na bod p_i, pokud je bod vlevo od orientace na bod p_i, pak $\omega = \omega - \omega_i$
5. Pokud platí podmínka $(|\omega - 2\Pi| < \epsilon)$, pak $q \in P$
6. Pokud neplatí podmínka $(|\omega - 2\Pi| < \epsilon)$, pak $q \notin P$

4 Problematické situace a singularity

5 Vstupní data

5.1 Formát vstupních dat

5.2 Popis dat

6 Výstupní data

6.1 Formát výstupních dat

6.2 Popis dat

7 Ukázka aplikace

8 Závěr

Autoři splnili většinu bodů zadání a vznikl program, který načítá soubor polygonů, následně uživatele nechá umístit bod a po stisknutí tlačítka urči, zda a ve kterých polygonech bod leží.

8.1 Náměty na vylepšení

Aplikace, ač funkční a splňující daný účel, má spoustu nedostatků, které by bylo dobré v budoucnu odstranit. Autoři zde uvádí pár těch nejzjevnějších.

Vykreslování dat: Aplikace bez problému vykreslí body, které se vejdou do jejího okna 665x605px. Problém nastává až tehdy, když jsou souřadnice větší než tato hodnota. Tato chyba jde odstranit vhodnou transformací okna (nebo souřadnic), která by ale probíhala na základě načtených dat.

Souřadnicové osy: Vykreslovací okno má v Qt, stejně jako ve většině podobných nástrojů, počátek souřadnic v levém horním rohu, kladnou osu x vpravo a kladnou osu y směrem dolů. Tento model se však neshoduje ani s geodetickými souřadnicemi používanými na našem území (kladná y doleva, kladná x dolů), ani s klasickým označením os (kladná x doprava, kladná y nahoru). Proto se body v současné verzi zobrazují jinak, než by možná uživatel očekával. Vhodným řešením by byla opět transformace.

Přesnější souřadnice: V současném stavu aplikace sice načítá data ve formátu `double`, avšak datová struktura `std::vector<std::vector<QPoint»` body načítá bez desetinných míst jako typ `int`. V Qt knihovnách existuje i datový typ `QPointF`, který ukládá body jako typ `float`. Změně však bude potřeba přizpůsobit porovnávání čísel v algoritmech.