ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

ALGORITMY DIGITÁLNÍ KARTOGRAFIE A GIS

KATEDRA GEOMATIKY

Úloha č. 1: Geometrické vyhledávání bodu

Monika KŘÍŽOVÁ
Marek HOFFMANN

18.10.2021

Obsah

1 Zadání

Navrhněte aplikaci s grafickým rozhraním, která vyhledá a vyznačí polygon obsahující zadaný bod.

Využijte existující geografická data nebo si vytvořte vlastní soubor s nekonvexními polygony. Data načítejte z textového souboru, ve kterém jsou polygony zaznamenány ve formě špagetového modelu.

2 Popis problému

Geometrické vyhledávání bodu představuje jednu ze základních úloh digitální kartografie. Princip metody spočívá v určování prostorového vztahu mezi zadaným bodem q a množinou n bodů p_i tvořících vrcholy m mnohoúhelníků P_j v rovině.

Úkol vyžaduje nejdříve pomocí aplikace Qt Creator vytvořit grafické prostředí aplikace a připravit si data, která jsou následně aplikací zpracována. Výstupem má být interaktivní prostředí, které zvýrazňuje polygon, v němž se nachází uživatelem zadaný bod.

Úloha je řešena převedením problému na opakované určování polohy bodu q vzhledem k mnohoúhelníku P_j. Pro řešení úlohy pro nekonvexní mnohoúhelníky se nejčastěji využívají algoritmy *Winding Number Algorithm* a *Ray Crossing Algorithm*.

3 Popisy algoritmů

3.1 Winding Number algoritmus

Winding Number algoritmus je založen na výpočtu celkového úhlu Ω, který musí průvodič mezi zadaným bodem q a vrcholy m mnohoúhelníku P_j opsat nad všemi lomovými body p_i polygonu.

$$\Omega(q, P) = \frac{1}{2\pi} \sum_{i=1}^{n} \omega\left(p_i, q, p_{i+1}\right) \tag{1}$$

Vrcholový úhel $\omega\left(p_i, q, p_{i+1}\right)$ se vypočte vztahem 2.

$$\cos \omega_i = \frac{\vec{u}_i * \vec{v}_i}{||\vec{u}_i|| * \mid \vec{v}_i||}; \vec{u}_i = \left(q, p_i\right), \vec{v}_i = \left(q, p_{i+1}\right) \tag{2}$$

Následně se na základě vektorového součinu 3 určí topologická poloha bodu a přímky procházející body p_i, p_{i+1}.

$$t = \vec{u}_i \times \vec{v}_i \tag{3}$$

Nachází-li se bod vlevo od q vlevo od přímky (p_i, p_{i+1}), pak má hodnota ω_i kladnou hodnotu. Pokud se bod q nachází vpravo od přímky (p_i, p_{i+1}), pak má hodnota ω_i zápornou hodnotu.

Pokud se rovnice (1) rovná 1, nachází se bod q v polygonu P_j. Pokud je výsledek nulový, pozice bodu se nachází mimo polygon P_j. Uvedené vztahy popisuje rovnice (4).

$$\Omega(q, P) = \begin{cases} 1, q \in P \\ 0, q \notin P \end{cases} \tag{4}$$

3.2 Ray Crossing algoritmus

Ray Crossing algoritmus je založen na docela jiném principu.

Máme-li bod q a polygon P_j tvořený body p_i a vedeme-li polopřímku r z bodu q libovolným směrem, určuje se poloha bodu q na základě počtu průsečíků q polopřímky r s oblastí P_j.

$$k(r, P)\%2 = \begin{cases} 1, q \in P \\ 0, q \notin P \end{cases} \tag{5}$$

V tomto tvaru je sice algoritmus rychlejší než Winding Number algoritmus, nicméně je nutné zabývat se problémem singularity, kdy bod leží na hraně polygonu či přímo inciduje s vrcholem ,nebo polopřímka r prochází vrcholem p_i polygonu P_j.

Problém singularity bude více rozebrán v následující kapitole.

4 Problematické situace

4.1 Winding Number algoritmus

Případ singularity u Winding Number algoritmu, kdy bod q leží na hraně h polygonu P_j byl ošetřen následovně; Pokud se bod na hraně nachází, tak $\Omega(q, P)$ neodpovídá ani jednomu výsledku z 4, tj.

$$\Omega(q, P) \neq 1 \wedge \Omega(q, P) \neq 0, \quad q \in h \tag{6}$$

4.2 Ray Crossing algoritmus

Prochází-li polopřímka r jedním z vrcholů p_i polygonu P_j či jeho hranou h, dochází k tzv. singularitě.

Pro odstranění této singularity se zavádí varianta Ray Crossing algoritmu s redukcí ke q. V této metodě se vytváří lokální souřadnicový systém (q, x', y') s posunutím bodů p_i o hodnotu q.

Lokální souřadnicová soustava je popsána následovně:

- Počátek v bodě q
- osa x' rovnoběžná s osou x
- osa y' rovnoběžná s osou y

Redukce popisují následující rovnice.

$$\begin{aligned} x'_i &= x_i - x_q \\ y'_i &= y_i - y_q \end{aligned} \tag{7}$$

$$\begin{aligned} x'_{i-1} &= x_{i-1} - x_q \\ y'_{i-1} &= y_{i-1} - y_q \end{aligned} \tag{8}$$

Zdali existuje průsečík přímky p a paprsku r zjistíme následující podmínkou

$$\left(y'_i > 0\right) \wedge \left(y'_{i-1} <= 0\right) \vee \left(y'_{i-1} > 0\right) \wedge \left(y'_i <= 0\right) \tag{9}$$

Pokud existuje, vypočítáme jeho x-ovou souřadnici podle 10.

$$x'_m = \frac{x'_i y'_{i-1} - x'_{i-1} y'_i}{y'_i - y'_{i-1}} \tag{10}$$

Následně pak hledáme pouze ty průsečíky M, které se nachází v pravé polorovině σ_r.

$$k(r, P) = \begin{cases} k(r, P) + 1, x'_m > 0 \\ k(r, P), x'_m <= 0 \end{cases} \tag{11}$$

Zda bod leží v polygonu zjistíme dle rovnice rovnice 5.

Případ singulárního případu, kdy bod leží na hraně polygonu nebyl v naší aplikaci ošetřen.

5 Vstupní data

Polygonová mapa se do projektu nahrává z textového souboru, který musí splňovat specifický formát, bez něhož nebude zajištěno správné načítání dat.

Vrcholy polygonů jsou načítány postupně řádek po řádku, přičemž musí v nahrávaném souboru platit následující pravidla:

- každý vrchol polygonu je definován na jednotlivém řádku,
- na řádku je pořadí proměnných id » x » y , hodnoty jsou od sebe odděleny jednou mezerou,
- id je identifikátor jednotlivých vrcholů polygonu, x a y jsou souřadnice vrcholů polygonu,
- nový polygon má vždy hodnotu id prvního vrcholu rovnu 1, od této hodnoty se postupně načítají body, a to až do okamžiku, kdy program dojde k dalšímu identifikátoru s označením 1.

```
1 300 300
2 300 400
3 400 400
4 400 300
1 400 400
2 300 400
3 400 500
```

Obrázek 1: Špagetový model vstupního souboru

Souřadnice bodu q jsou získávány interaktivně odečtením souřadnic kurzoru myši.

6 Výstupní data

Po načtení souboru s polygonovou mapou, přidání bodu q a stisknutí tlačítka Analyze se v místě vyznačeném na Obr. XX vytiskne výsledek analýzy polohy bodu.

obrazek
Výsledek dotazu může mít 3 různé hodnoty:

- je-li bod mimo polygonovou mapu, vypíše se zpráva „Outside“,
- je-li bod uvnitř polygonu, vypíše se zpráva „Inside“ a polygon obsahující zadaný bod se graficky zvýrazní,
- je-li bod na hranici polygonu, vypíše se „Point is on the line“ a graficky se zvýrazní polygon, na jehož linii bod leží.

obrazek

7 Printscreen vytvořené aplikace

obrazky

8 Dokumentace

Kód zahrnuje 3 třídy – Draw, Algorithms a Widget, které budou následně detailněji popsány.

8.1 Třída Algorithms

Třída Algorithms obsahuje 4 funkce :

- int getPointLinePosition(QPoint &a, QPoint &p1, QPoint &p2);
- double get2LinesAngle(QPoint &p1, QPoint &p2, QPoint &p3, QPoint &p4);
- int getPositionWinding(QPoint &q, std::vector<QPoint> &pol);
- int getPositionRayCrossing(QPoint &q, std::vector<QPoint> &pol);

int getPointLinePosition(QPoint &a, QPoint &p1, QPoint &p2);
Analyzuje vzájemnou polohu mezi bodem a linií polygonu, resp. v jaké polorovině vůči linii se bod nachází. Vstupními argumenty funkce jsou souřadnice určovaného bodu (jako QPoint) a souřadnice 2 bodů určujících polohu linie (vrcholy polygonu taktéž jako QPoint). Funkce vrací vždy buď hodnotu 1, 0 nebo -1 dle následujících pravidel:

- 0 v případě, že bod leží v pravé polorovině,
- 1 v případě, že bod leží v levé polorovině,
- -1 v případě, že bod leží na linii.

double get2LinesAngle(QPoint &p1, QPoint &p2, QPoint &p3, QPoint &p4);
Počítá úhel mezi dvěma liniemi pomocí vztahu 2, vstupními argumenty jsou body určující linie - tedy vrcholy polygonu). Návratovou hodnotou funkce je double – desetinné číslo s velikostí úhlu mezi těmito přímkami.

int getPositionWinding(QPoint &q, std::vector<QPoint> &pol);
Funkce analyzuje polohu bodu metodou Winding number, jež byla podrobně vysvětlena v kapitole 3. Vstupními argumenty jsou souřadnice bodu Q (jako QPoint) a vektor vrcholů polygonu (vektor naplněný prvky QPoint). Funkce vrací integer, jež může nabývat 3 hodnot:

- 1, leží-li analyzovaný bod uvnitř polygonu
- -1 leží-li na linii polygonu
- a 0, leží-li mimo kontrolovaný polygon.

int getPositionRayCrossing(QPoint &q, std::vector<QPoint> &pol);
Funkce analyzuje polohu bodu metodou Ray Crossing, jež byla podrobně vysvětlena v kapitole 3. vstupními argumenty jsou souřadnice bodu Q (jako QPoint) a vektor vrcholů polygonu (vektor naplněný prvky QPoint). Funkce vrací, stejně jako funkce předchozí, celé číslo, jež může nabývat následujících hodnot:

- 1, pokud leží analyzovaný bod uvnitř polygonu
- a 0, leží-li mimo kontrolovaný polygon.

Ošetření případu, kdy bod leží na linii nebylo do kódu implementováno, funkce tedy nevrací hodnotu -1, leží-li bod na linii.

8.2 Třída Draw

Třída Draw obsahuje následující funkce, které budou dále podrobně popsány:

- void paintEvent(QPaintEvent *event);
- void mousePressEvent(QMouseEvent *event);
- void clear();
- QPoint getPoint()return q;
- std::vector<QPolygon> getPolygon(){return polygons;}
- void fillPolygon(int result);
- void loadData(QString &file_name);

a následující proměnné:

- std::vector<QPolygon> polygons;
- QPoint q; - analyzovaný bod
- int highlighted_polygon=-99; - id polygonu, který obsahuje analyzovaný bod

• **void paintEvent(QPaintEvent *event);**
Funkce nastavuje grafické atributy vykreslovaných objektů a kreslí na plátno vyhodnocovaný bod a zadané polygony a znázorňuje polygon incidující s bodem.

void mousePressEvent(QMouseEvent *event);
Metoda získává souřadnice myši a ukládá je do proměnné QPoint q, tedy jako souřadnice analyzovaného bodu Q.

void clear();
Funkce smaže veškeré objekty, jež jsou vykresleny na canvasu, funkce nemá vstupní argumenty.

QPoint getPoint(){return q;}
Funkce vrací souřadnice bodu Q, návratový typ funkce je QPoint, funkce nemá vstupní argumenty.

std::vector<QPolygon> getPolygon(){return polygons;}
Funkce vrací vektor polygonů načtených z textového souboru – návratový typ je std::vector<QPolygon>, funkce nemá vstupní argumenty.

void fillPolygon(int result);
Funkce ukládá identifikátor polygonu obsahujícího analyzovaný bod, pomocí identifikátoru se tento polygon následně ve funkci paintEvent() zvýrazní. Vstupním argumentem je int result, jež stanovuje polohu bodu vůči linii (má hodnoty 0,1,-1,dle definice výše).

void loadData(QString &file_name);
Funkce načítá data z textového souboru a ukládá je do vektoru QPolygon. Vstupní argumentem je cesta k souboru, jež chceme načíst do aplikace.

8.3 Třída Widget

Třída Widget obsahuje 3 metody:

- void on_pushButtonClear_clicked();
- void on_pushButtonAnalyze_clicked();
- void on_pushButtonLoad_clicked();

void on_pushButtonClear_clicked();
Funkce se volá při stisknutí tlačítka Clear, volá funkci clear(), jež je definovaná v třídě Draw.

void on_pushButtonAnalyze_clicked();
Funkce se volá při stisknutí tlačítka Analyze. Ve funkci nejprve dojde k uložení jednotlivých polygonů do vektoru vrcholů polygonu, následně je zavolána funkce getPositionRayCrossing ze třídy Algorithms nebo getPositionWinding dle výběru v comboboxu.

void on_pushButtonLoad_clicked();
Funkce slouží k inicializaci načítání souboru, spustí se po stisknutí tlačítka Load, čímž zobrazí dialog pro výběr souboru obsahujícího data, jež mají být načtena do aplikace. Po výběru souboru se zavolá funkce loadData ze třídy Draw, která přečte data ze souboru a uloží je do proměnné.

9 Závěr

Vytvořená aplikace vizualizuje polygonovou mapu uloženou ve výše popsaném formátu, po přidání bodu taktéž analyzuje polohu tohoto bodu vůči jednotlivým polygonům mapy. Po analýze bodu se vypíše výsledek – tedy zda bod leží uvnitř nebo vně polygonové mapy, v případě umístění bodu uvnitř polygonu se polygon obsahující tento bod graficky zvýrazní.

V aplikaci jsou pro vyhodnocení polohy bodu implementovány 2 algoritmy – Winding number a Ray Crossing, mezi nimiž je možné přepínat pomocí comboboxu umístěném v pravé části okna. Princip fungování obou algoritmů je popsán v kapitole 3.

Možné zlepšení kódu programu by bylo v řešení singularit polohy bodu q, tedy případu, kdy bod q leží na vrcholu jednoho či více polygonů a dále přidání analýzy případu, kdy bod q leží na linii polygonu pomocí Ray Crossing algoritmu. Dalším možným krokem pro zlepšení funkčnosti by mohlo být přidání bodu q pomocí vepsání souřadnic, nikoliv pouze odečtením polohy myši.