

České vysoké učení technické v Praze

Fakulta stavební



Algoritmy v digitální kartografii

Úloha 1: Geometrické vyhledávání bodu

Skupina

Lucie Děkanová

Josef Pudil

Zimní semestr 2020/2021

Obsah

1. Zadání	3
2. Údaje o bonusových úlohách	3
3. Popis a rozbor problémů	3
4. Popis algoritmů	4
4.1 Winding Number Algorithm	4
4.2 Ray Crossing Algorithm	4
5. Problematické situace	6
6. Vstupní data	6
7. Výstupní data	7
8. Dokumentace	8
8.1 Třída Draw	8
8.2 Třída Algorithms	9
Závěr	9
Zdroje	10
Seznam obrázků	10

1. Zadání

Nad souvislou polygonovou mapou implementujete následující algoritmy pro geometrické vyhledávání bodu:

- 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 polygon 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.

2. Údaje o bonusových úlohách

V této úloze byla vyřešena bonusová úloha: ošetření singulárního případu u Winding Number Algorithm (i u Ray Crossing Algorithm), kdy bod leží na hraně polygonu. Pokud bod ležel na hraně více polygonů, byly zvýrazněny všechny polygony, na jejichž hraně bod ležel.

3. Popis a rozbor problémů

Aplikace byla vytvořena v SW Qt Creator v jazyce C++.

Byl dán bod q a 2D souřadnice polygonů. Hlavním cílem úlohy bylo zjistit polohu bodu vůči polygonu/polygonům - bod leží vně, uvnitř nebo na hraně. Polygony byly načítány z textového souboru a pro jejich grafické zobrazení bylo využito grafické okno v SW Qt Creator.

Poloha bodu vůči polygonům byla řešena dvěma různými algoritmy – Ray Crossing a Winding Number algoritmem.

Ke správnému fungování aplikace bylo vytvořeno nejdříve grafické okno obsahující prvky pro vykreslení/analýzu atd. Dále byly vytvořeny třídy Draw a Algorithms, ve kterých byly tvořeny kódy pro správné fungování aplikace. Definování funkcí v těchto dvou třídách bylo provedeno v jejich hlavičkových souborech.

4. Popis algoritmů

4.1 Winding Number Algorithm

U tohoto algoritmu jde o výpočet úhlů mezi jednotlivými průvodiči z bodu q do jednotlivých vrcholů polygonu. Přičítáním a odečítáním úhlů mezi průvodiči zjistíme, kde se bod vůči polygonu nachází. Podle poloroviny, ve které se úhel vůči spojnici bodu nachází zjistíme, jestli úhel mezi průvodiči do sumy úhlů přičteme nebo od ní odečteme. Když se úhly naházejí v levé polorovině, tak se odečítají, pokud jsou v pravé tak se přičítají. Podle výsledné sumy úhlů v polygonu můžeme zjistit polohu bodu. Pokud je výsledný úhel roven 2π , tak se daný bod nachází uvnitř polygonu, pokud je výsledný úhel není roven 2π , tak se bod nachází mimo polygon.

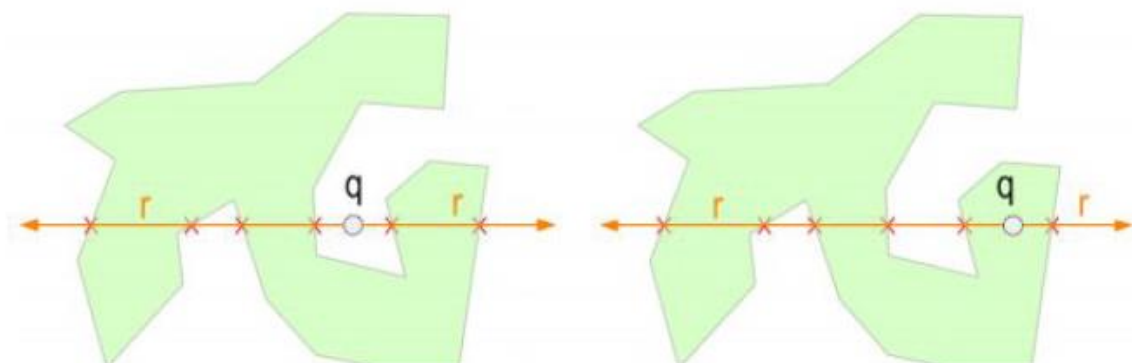
Princip algoritmu:

1. Inicializace $\Omega = 0$, tolerance ϵ
2. Opakuj pro \forall trojici (p_i, q, p_{i+1}) :
 Urči polohu q vzhledem k $e_i = (p_i, p_{i+1})$.
 Urči úhel $\omega_i = \angle p_i, q, p_{i+1}$.
 If $q \in \overline{p_i p_{i+1}}$, pak $\Omega = \Omega + \omega_i$.
 else $\Omega = \Omega - \omega_i$.
3. if $|\Omega - 2\pi| < \epsilon$, pak $q \in P$.
4. else $q \notin p$.

4.2 Ray Crossing Algorithm

Máme danou přímku r , která prochází naším bodem q . Poloha bodu a polygonu P je určována na základě počtu průsečíků přímky k a hran polygonu p_i . Pokud je počet průsečíků sudý, tak je bod q vně polygonu P , pokud lichý, tak je bod uvnitř polygonu. Singularita je ošetřena tím, že pro určení výsledku je uvažována jen jedna polorovina vzhledem k r , která prochází naším bodem q – upravený model. Další úprava redukuje souřadnice vrcholů polygonu k bodu. Pro model jsou hledány jen průsečíky M , které leží v pravé polorovině od osy x , který po redukci prochází naším bodem q . Výsledný algoritmus uvažuje jen průsečíky ležící v 1 kvadrantu lokální souř. soustavy vzhledem k našemu bodu.

Princip Ray Crossing algoritmu:



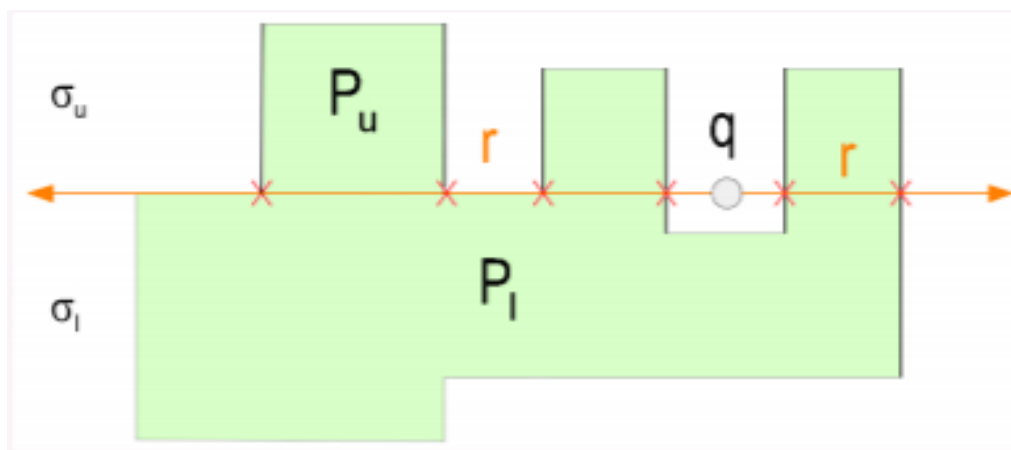
Obrázek 1: Princip Ray Crossing algoritmu

Upravená varianta Ray Crossing algoritmu:

Eliminace singularit: $r(q)$ prochází vrcholem P nebo hranou.

Přímka definovaná $r(q)$ dělí σ na σ_u, σ_l

$$\sigma_u' = \sigma_u - \{r(q)\}, \sigma_l' = \sigma_l - \{r(q)\}$$



Obrázek 2: Upravená varianta Ray Crossing

σ_u' obsahuje body $p_i = [x_i, y_i]$, kde $y_i > y_q$

σ_l' obsahuje body $p_i = [x_i, y_i]$, kde $y_i < y_q$

Inkrementace $k(q, P)$:

- jeden z bodů hrany nad $r(q)$ ($\nu \sigma_u'$) a druhý z bodů na/pod $r(q)$ ($\nu \sigma_l'$)

Princip algoritmu s redukcí:

1. Inicializuj $k = 0$
2. Opakuj pro \forall body $p_i \in P$:
$$x'_i = x_i - x_q$$
$$y'_i = y_i - y_q$$
$$\text{if } (y'_i > 0) \ \&\& \ (y'_{i-1} \leq 0) \ || \ (y'_i \leq 0) \ \&\& \ (y'_{i-1} > 0)$$
$$x'_m = (x'_i y'_{i-1} - x'_{i-1} y'_i) / (y'_i - y'_{i-1})$$
$$\text{if } (x'_m > 0) \text{ pak } k = k + 1$$
3. if $(k \bmod 2) \neq 0$ pak $q \in P$
4. else $q \notin p$.

5. Problematické situace

K těmto situacím řadíme řešení singularity – bod leží na linii nebo na vrcholu polygonu.

Pokud chceme zjistit, jestli bod leží na vrcholu, souřadnice q porovnáme se souřadnicemi bodů polygonu. Pokud se souřadnice rovnají, je bod q totožný s jedním z vrcholů.

Pokud bod leží na linii (Winding Number Algorithm), využíváme orientaci bodu vůči přímce, kterou lze určit pomocí funkce *getPointLinePosition*. Je nutné zjistit, zda se bod nachází na hraně polygonu, tzn. velikost jeho x -ové souřadnice je v intervalu daným x -ovými souřadnicemi koncových bodů linie. Je tedy nutné zkontrolovat podmínku:

$$((X_q \geq X_{pi}) \wedge (X_q \leq X_{pi+1})) \vee ((X_q \leq X_{pi}) \wedge (X_q \geq X_{pi+1}))$$

Další problematické situace nebyly v úloze řešeny.

6. Vstupní data

Vstupní data byla ve formátu *txt*. V souboru byly dány vrcholy polygonů (x a y souřadnice). Tato data se do aplikace načítají pomocí tlačítka *Load Data*.

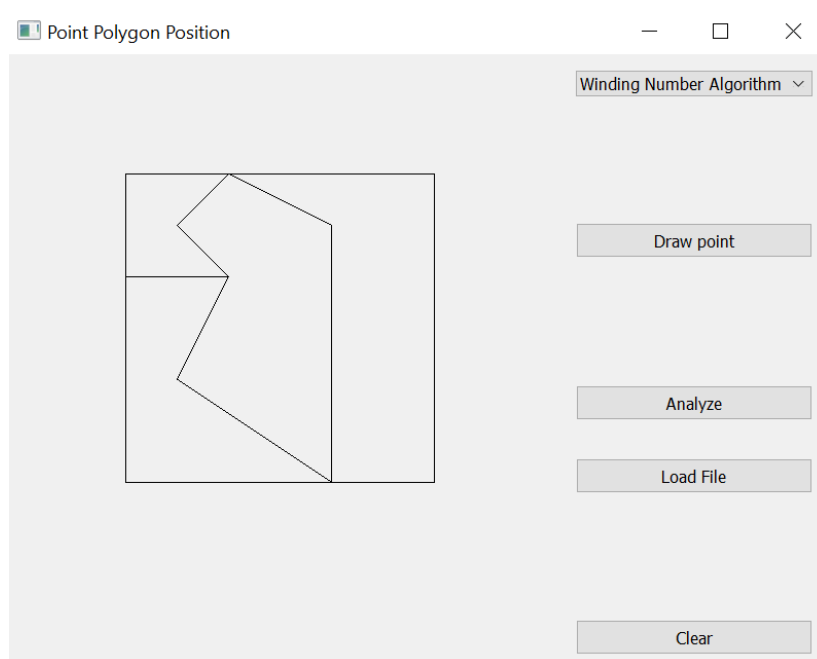
Vstupní data mají formát: id x y

Nový polygon vždy začíná číslem 1.

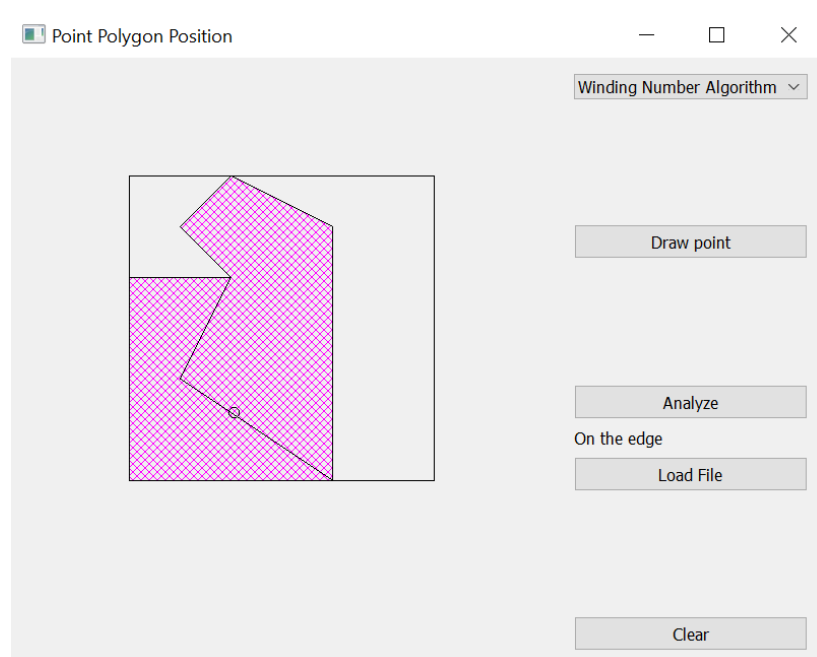
7. Výstupní data

Výstupem úlohy je grafické okno. Okno obsahuje několik tlačítek; vyskakovací okno pro výběr algoritmu, analýza bodu a polygonu – zvýrazní polygon fialovou mřížkou, vypíše, kde se nachází, a tlačítko *Clear*, které vymaže obsah.

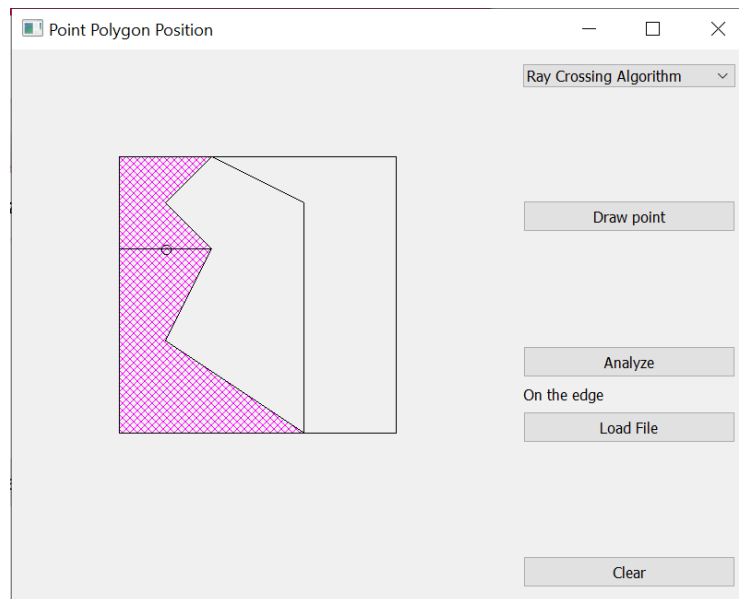
Ukázky aplikace:



Obrázek 3: Aplikace při spuštění



Obrázek 4: Singulární případ bodu na hraně - algoritmus Winding Number



Obrázek 5: Analýza polohy- algoritmus Ray Crossing

8. Dokumentace

8.1 Třída Draw

private:

`boolean draw_mode;` -vykresluje bod

`QPointF q;` -definice proměnné pro souřadnice

`vector<int> result;` -definice vektoru pro obarvené polygony

`vector<QPolygonF> polygons;` -proměnné pro načtení polygonů

public:

`void mousePressEvent(QMouseEvent *e);` -získání souřadnic bodu

`void paintEvent(QPaintEvent *e);` -vykreslení polygonu, bodu, obarvení

`void changeMode(){draw_mode = !draw_mode;}` -vykreslení bodu

`QPointF &getPoint(){return q;}` -získání souřadnic bodu

`void erasePoints();` -mazání okna

`void txtFile(string &path);` -funkce pro načtení polygonů

`void setResult(vector<int> res){result=res;}` -výsledky ve vektoru + vybarvení polygonu

`vector<QPolygonF> getPolygons(){return polygons;}` -získání souřadnic polygonů

8.2 Třída Algorithms

int **getPointLinePosition**(QPointF &q, QPointF &p1, QPointF &p2);

- orientace bodů a přímky, zjišťuje, jestli se bod nachází v pravé/levé polorovině / přímce
- vstup: souřadnice q, souřadnice hran polygonů

double **getAngle**(QPointF &q1, QPointF &q2, QPointF &q3, QPointF &q4);

- počítá úhel mezi dvěma vektory
- vstup: souřadnice bodů vektorů

int **getPositionWinding**(QPointF &q, std::vector<QPointF> &pol);

- výpočet polohy bodu vůči polygonům
- vstup: souřadnice bodu a polygonů

int **getPositionRay**(QPointF &q, std::vector<QPointF> &pol);

- výpočet polohy bodu a polygonu
- vstup: jako *getPositionWinding*

Závěr

Z bonusových úloh byla řešena pouze 1, ale úspěšně.

Aplikace umí načíst textový soubor, vykreslit polygony do vykreslovacího okna a analyzovat vzájemnou polohu vykreslených polygonů a námi zakresleného bodu.

V úloze by mohlo být řešeno více bonusových úloh. Z časových důvodů a programovacích zkušeností zpracovatele byla vyřešena pouze jedna.

Vylepšit by se dalo načítání dat mimo předdefinované vykreslovací okno. Problém by bylo možné vyřešit transformací velikosti okna.

Dále by bylo dobré zajistit, aby při importu souřadnic polygonů byla ošetřena chyba, pokud by struktura vstupního souboru nebyla správná. Toto by bylo možné kontrolovat funkcí.

Zdroje

1. BAYER, Tomáš. [online] [cit. 23.10.2020].

Dostupné na: <https://web.natur.cuni.cz/~bayertom/images/courses/Adk/adk3.pdf>

Seznam obrázků

Obrázek 1: Princip Ray Crossing algoritmu	5
Obrázek 2: Upravená varianta Ray Crossing	5
Obrázek 3: Aplikace při spuštění	7
Obrázek 4: Singulární případ bodu na hraně - algoritmus Winding Number	7
Obrázek 5: Analýza polohy- algoritmus Ray Crossing.....	8