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

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 $q \epsilon \overline{\sigma}_l$, pak $\Omega = \Omega + \omega_i$.
   else $\Omega = \Omega - \omega_i$.
3. if $||\omega| - 2\pi| < \epsilon$, pak $q \epsilon P$.
4. else $q \notin p$.

4.2 Ray Crossing Algorithm

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

Princip algoritmu s redukcí:

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

5.Problematické situace

K těmto situacím řadíme řešení singularity – bod leží na linii nebo na vrcholu polygonu. Toto je ošetřeno ve funkci *getPointLinePosition*, kdy je porovnávána vzdálenost počátečního bodu úsečky *p1* a našeho bodu *q* (*p2* a *q*) se vzdáleností bodů *p1* a *p2*. Pokud jsou si tyto vzdálenosti blízké, je bod *q* kolineární.

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 souřanice). Tato data se do aplikace načítají pomocí tlačítka *Load Data*.

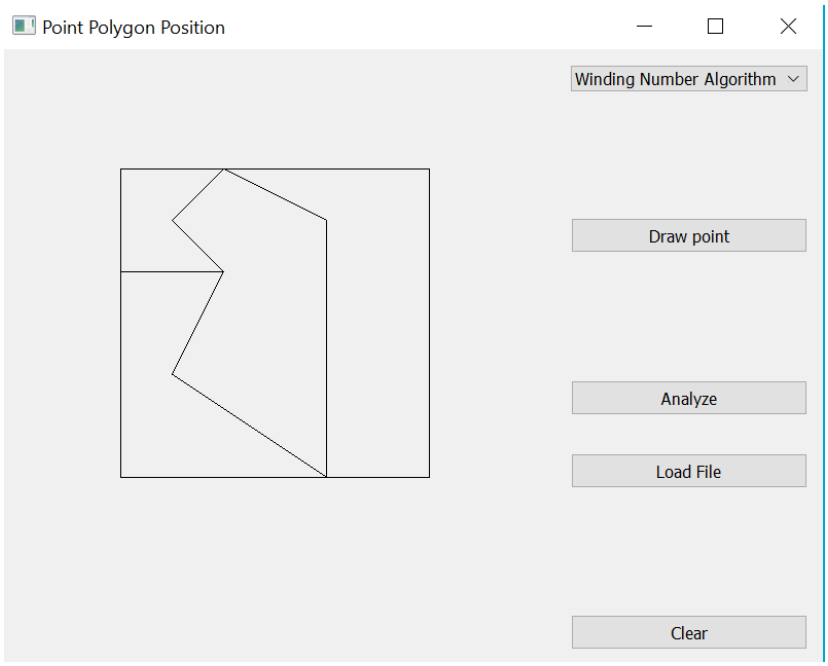
Vstupní data mají formát: *id* *x* *y*

Nový polygon vždy začíná číslem 1.

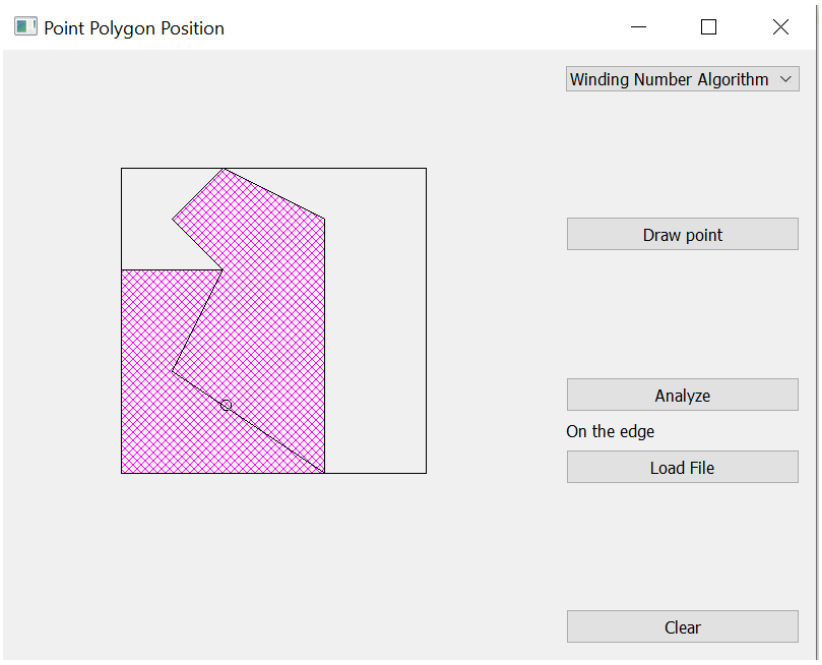
7.Výstupní data

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.

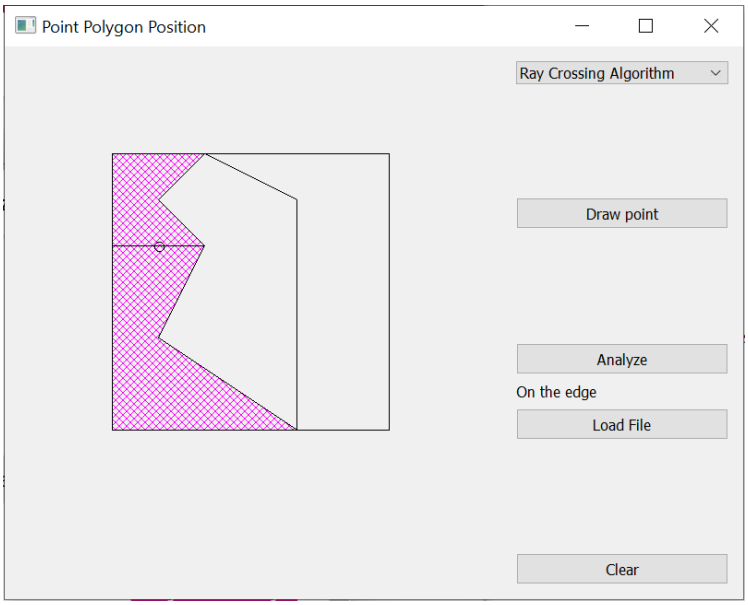
Ukázky aplikace:



Obrázek 1: Aplikace při spuštění



Obrázek 2: Singulární případ bodu na hraně - algoritmus Winding Number



Obrázek 3: 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.

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

Zdroje

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

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

Seznam obrázků