



ALGORITMY DIGITÁLNÍ KARTOGRAFIE A GIS

Úloha č. 1: **Geometrické vyhledávání bodu**

Tatána Bláhová, Tomáš Krauz, Adéla Kučerová

19. října 2022

Obsah

1	Zadání	2
2	Bonusové úlohy	2
3	Popis a rozbor problému	3
4	Popis použitých algoritmů	3
4.1	Ray Crossing ALgorithm	3
4.2	Winding Number Algorithm	3
5	Problematické situace a jejich rozbor	4
5.1	Bod totožný s vrcholem mnohoúhelníku	4
5.2	Bod se nalézá na hranici mnohoúhelníku	5
6	Vzhled aplikace	5
7	Vstupní data	5
8	Výstupní data	5
9	Dokumentace	6
9.1	Třída Algorithms	6
9.2	Třída Draw	7
9.3	Třída CSV	8
9.4	Třída Mainform	8
10	Závěr	8

1 Zadání

Úloha č. 1: Geometrické vyhledávání bodu

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

Výstup: $P_i, q \in P_i$.

Nad polygonovou mapou implementujete Winding Number Algorithm pro geometrické vyhledání incidujícího polygonu obsahujícího zadaný bod q .

Nalezený polygon 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
Analýza polohy bodu (uvnitř/vně) metodou Ray Algorithm.	+5b
Ošetření singulárního případu u Ray Algorithm: bod leží na hraně polygonu.	+5b
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.	+3b
Max celkem:	25b

Čas zpracování: 1 týden.

Obrázek 1.1: Zadání úlohy

2 Bonusové úlohy

1. Analýza polohy bodu (uvnitř/vně) metodou Ray Algorithm.
2. Zvýraznění polygonů.

3 Popis a rozbor problému

Point location problem je jedno ze základních témat výpočetní geometrie. Lokalizace bodu je důležitá v oblastech geografických informačních systémů (GIS), v počítačové grafice i počítačem podporovaném kreslení (CAD).

Máme bod $q=[x_q, y_q]$ a množinu M ve které se nalézá m mnohoúhelníků $\{P_i\}$. Každý mnohoúhelník se skládá z několika vrcholů $\{p_i\}$. Zde se zabýváme, zda námi určený bod q leží uvnitř, vně nebo na hranici konvexních i nekonvexních mnohoúhelníku $\{P_i\}$. Pro řešení nekonvexních mnohoúhelníků jsou používány 2 algoritmy. Popis těchto algoritmů bude vysvětleno v následující kapitole.

4 Popis použitých algoritmů

4.1 Ray Crossing Algorithm

Máme mnohoúhelník $\{P_i\}$ a námi zkoumaný bod q . Do bodu q je umístěn počátek lokální souřadnicové soustavy (q, x', y') , který má osy rovnoběžné s hlavní souřadnicovou soustavou. Následně je určen počet průsečíků osy x' s mnohoúhelníkem $\{P_i\}$. Ze všech průsečíků jsou vybrány takový, kteří mají $x > 0$. Jestliže, je počet průsečíků lichý, pak je q uvnitř polygonu, pokud je sudý tak vně polygonu. Průsečík x'_m osy x' se stranou mnohoúhelníka se určí podle vzorce na stránkách [1].

$$x'_m = \frac{x'_i y'_{i-1} - x'_{i-1} y'_i}{y'_i - y'_{i-1}} \quad (1)$$

Algoritmus:

1. Inicializuj $k = 0$; kde k je počet průsečíků ϵ
2. Pro $\forall p_i$ opakuj
3. $x'_i = x_i - x_q; y'_i = y_i - y_q$
4. Jestliže $(y'_i > 0) \&\& (y'_{i-1} \leq 0) \parallel (y'_{i-1} > 0) \&\& (y'_i \leq 0)$
5. Vypočítej x'_m
6. if $x'_m > 0$, pak $k++$
7. if k je liché, pak $q \in P_i$
8. else $q \notin P_i$

4.2 Winding Number Algorithm

Máme mnohoúhelník $\{P_i\}$ a námi zkoumaný bod q . Je potřeba vypočítat sumu všech orientací nad všemi vrcholy mnohoúhelníku. Je potřeba spočítat sumu Ω všech rotací ω_i .

$$\Omega(q, P_i) = \sum_{i=1}^n \omega_j(p_i, q, p_{i+1}) \quad (2)$$

které musí průvodič opsat nad všemi body $p_i \in P$ a n je počet vrcholů mnohoúhelníku. Úhel ω_i se vypočítá podle vzorce

$$\cos(\omega_j) = \frac{\vec{u}_i * \vec{v}_i}{|\vec{u}_i| * |\vec{v}_i|} \quad (3)$$

kde $|\vec{u}_i| = (q, p_i)$, $|\vec{v}_i| = (q, p_{i+1})$.

Ω může nabývat hodnot

- 2π , $q \in P$
- 2π , $q \notin P$
- Jiný úhel, bod je totožný s hranou a nebo s vrcholem mnohoúhelníku

Na jaké straně leží bod a nebo hraně je možno poznat podle hodnoty diskriminantu.

$$\det = \begin{vmatrix} u_x & u_y \\ v_x & v_y \end{vmatrix} \quad (4)$$

Poté mohou nastat tři scénáře:

- $\det > 0$, q se nachází na pravé straně
- $\det < 0$, q se nachází na levé straně
- $\det = 0$, q se nachází na hraně

Algoritmus:

1. Inicializuj $\Omega = 0$, tolerance ϵ
2. Opakuj pro trojice $\forall < p_i, q, p_{i+1} >$
3. Urči polohu q vzhledem k $p = (p_i, p_{i+1})$
4. Urči úhel $\omega_i = \angle p_i, q, p_{i+1}$
5. if q je vlevo od (p_i, p_{i+1}) , pak $\Omega = \Omega - \omega$
6. else q je v pravo od (p_i, p_{i+1}) , pak $\Omega = \Omega + \omega$
7. if $|\Omega - 2\pi| < \epsilon$, pak $q \in P$
8. else $q \notin P$

5 Problematické situace a jejich rozbor

5.1 Bod totožný s vrcholem mnohoúhelníku

Tento problém se řeší stejným způsobem pro oba algoritmy.

Pro každý vrchol je spočtena vzdálenost s od určovaného bodu. Za předpokladu, že všechny délky $s < \epsilon$, lze algoritmus zastavit a říct, že bod je totožný s vrcholem mnohoúhelníku.

5.2 Bod se nalézá na hranici mnohoúhelníku

5.2.1 Ray Crossing Algorithm

Když je mnohoúhelník transformován do místní souř. soustavy, se vypočítají průsečíky x'_m, y'_m s hrany polygonu s osami. Je potřeba, aby byla splněna podmínka $|x'_m| < \epsilon \&\& |y'_m| < \epsilon$. Což znamená, že bod je na hraně polygonu a algoritmus může být zastaven.

5.2.2 Winding Number Algorithm

Při výpočtu ω_i se určuje determinant det . Pokud $|det| < \epsilon$, kde epsilon je stanovena tolerance přesnosti výpočtu. Můžeme prohlásit, že bod leží na přímce dané stranou polygonu. Pokud předpokládáme, že bod leží v minimálním ohraničujícím obdelníku strany mnohoúhelníku, který má strany rovnoběžné s osami souř. soustavy, pak lze potvrdit, že bod leží na hraně polygonu.

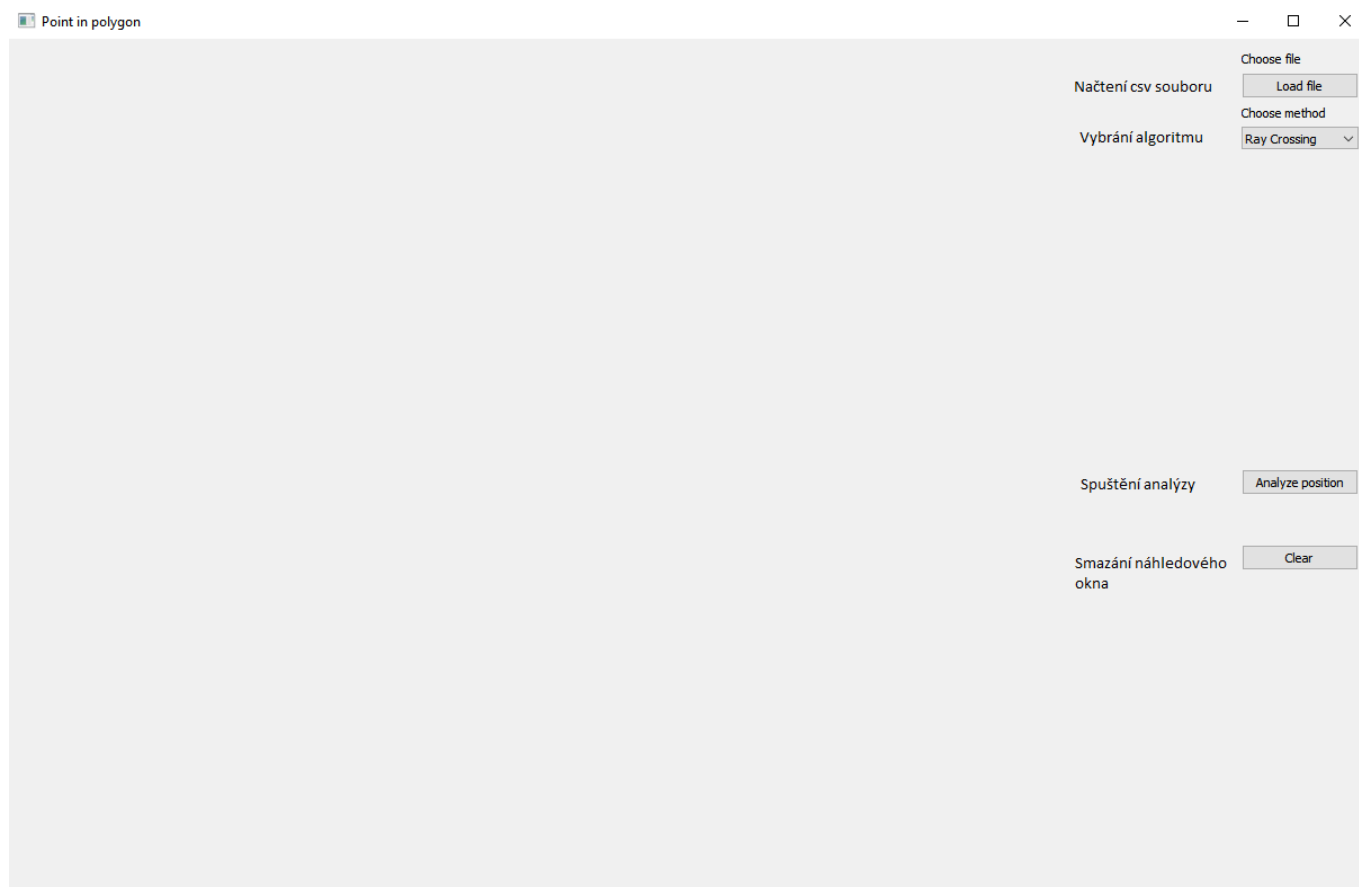
6 Vzhled aplikace

Na úvodní obrazovce aplikace se nachází v levé části náhledové okno Canvas třídy Draw. Na pravé straně je pak panel s obslužnými tlačítky. Prvním je comboBox třídy QComboBox, který slouží k výběru algoritmu. Dále jsou další tlačítka třídy QPushButton. Tyto tlačítka slouží k analýze, k vyčištění plochy a nebo pro načtení csv souboru.

7 Vstupní data

Na úvodní obrazovce se nachází tlačítko Load file, po jehož kliknutí se otevře průzkumník souborů, pomocí něhož najdeme požadovaný soubor. Soubor s polygony musí být ve formátu csv.

8 Výstupní data



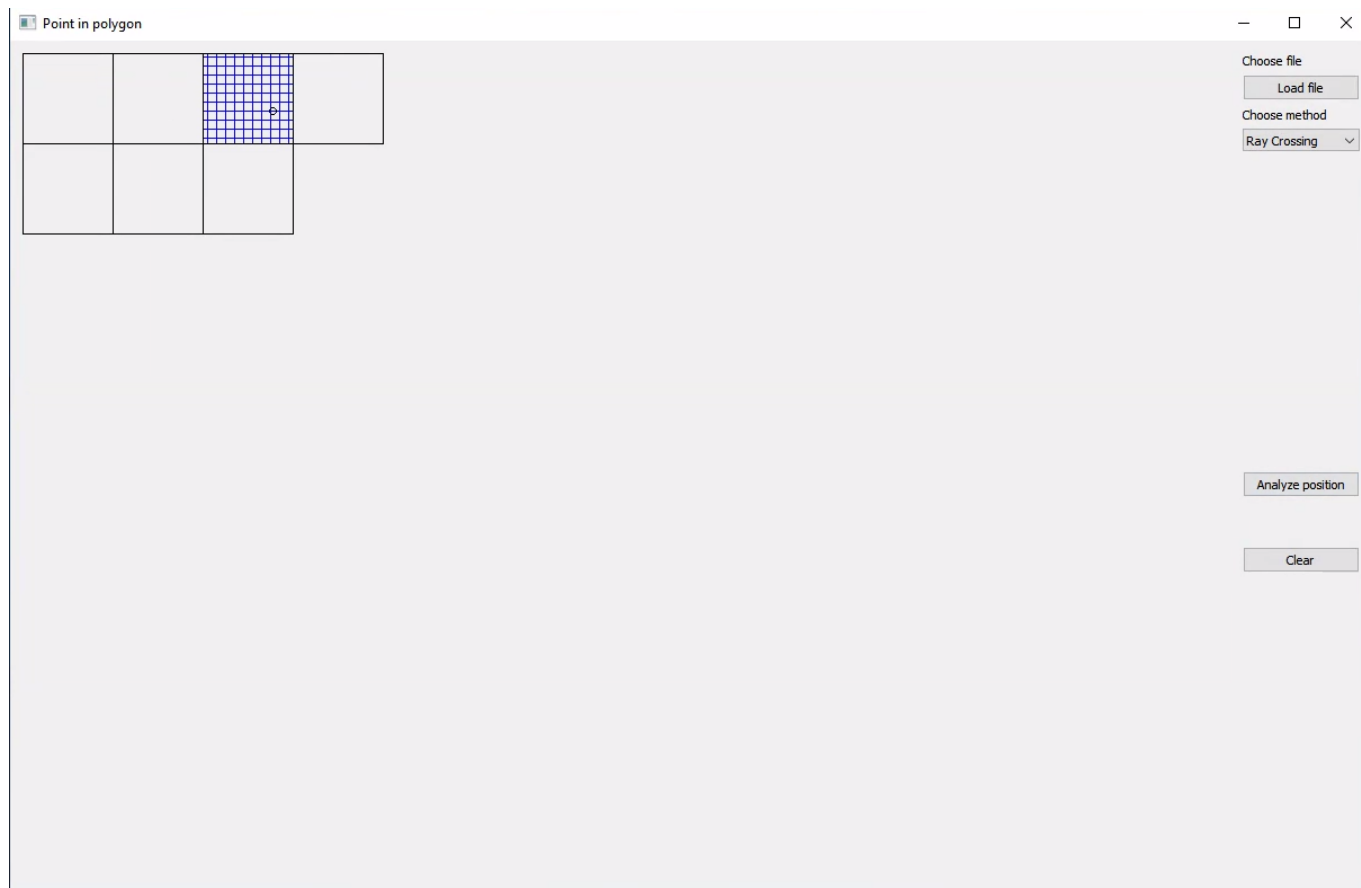
Obrázek 6.1: Vzhled aplikace

Výstupem aplikace je grafické znázornění dotčeného/dotčených polygonů. Polygon je zvýrazněn souvislou výplní zelené barvy.

9 Dokumentace

9.1 Třída Algorithms

- *int getPointLinePosition(QPointF &p1, QPointF &p2, QPointF &q)*
– analyzuje vzájemný vztah bodu a přímky
- *double getTwoLinesAngle(QPointF &p1, QPointF &p2, QPointF &p3, QPointF &p4)*
– vypočítá úhel mezi dvěma vektory
- *int getPointAndPolygonPosition(QPointF &q, vector < QPointF > &pol)*
– analyzuje vztah bodu a polygonu pomocí Ray Crossing Algorithm
- *int getPosWinding(QPointF &q, vector < QPointF > &pol)*



Obrázek 8.1: Výstupní data

- analyzuje vztah bodu a polygonu pomocí Winding Number Algorithm
- *vector < QPoint > getLocalCoordinates(QPointF &q, vector < QPointF > &pol)*
 - transformuje souřadnice do místního souřadnicového systému
- *int processPols(QPointF &q, vector < QPolygon > &pols, QString &alg, vector < int > &res)*
 - analyzuje vztah všech polygonů s bodem q

9.2 Třída Draw

- *void mousePressEvent(QMouseEvent * event)*
 - vrátí souřadnice kurzoru po kliknutí na Canvas
- *void paintEvent(QPaintEvent * event)*
 - vykreslí polygony na Canvas
- *void clearScreen()*
 - vyčistí Canvas

9.3 Třída CSV

- `vector < QPolygon > read_Csv(string filename)`
 - načte vstupní csv soubor

9.4 Třída Mainform

- `void on_pushButton_Position_clicked()`
 - provede analýzu
- `void on_pushButton_File_clicked()`
 - otevře průzkumníka souborů a je možno načíst požadovaný soubor
- `void on_pushButton_Clear_clicked()`
 - vyčistí Canvas

10 Závěr

Byla vytvořena aplikace Point in Polygon (nvm jestli jsme si ji my pojmenovali) s grafickým rozhráním. Aplikace byla napsána v programovacím jazyce C++. Aplikace umožňuje nahrání souboru .csv s polygony. K analýze polohy bodu bylo využito dvou metod Ray Crossing Algorithm a Winding Number Algorithm. Analýza je provedena pokud

- a) leží uvnitř polygonu
- b) leží vně polygonu

Literatura

- [1] Tomáš Bayer, *Point location problem*, https://agony.natur.cuni.cz/~bayertom/images/courses/Adk/adk3_new.pdf, [14.10.2022].