ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA STAVEBNÍ, OBOR GEODÉZIE A KARTOGRAFIE
KATEDRA GEOMATIKY

název předmětu

Algoritmy digitální kartografie a GIS

název úlohy

Geometrické vyhledávání bodu

školní rok	studijní skup.	číslo zadání	zpracoval, email	datum	klasifikace
2021/22	60	-	Jakub Šimek, Jan Kučera	18.10.21	

Obsah

1. Zadání

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 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.	*+2b*
Max celkem:	**24b**

Čas zpracování: 1 týden.

2. Bonusové úlohy

Ke standartnímu zadání byly zpracovány tyto bonusové úlohy:

1. Analýza polohy bodu (uvnitř/vně) metodou *Ray Algorithm.*
2. Ošetření singulárního případu u *Ray Algorithm*: bod leží na hraně polygonu.
3. Ošetření singulárního případu u obou algoritmů: bod je totožný s vrcholem jednoho či více polygonů.
4. Zvýraznění všech polygonů pro oba výše uvedené singulární případy.

3. Popis a rozbor problému

Mějme množinu $M\{P_i\}$, kde $i \in < 1, \dots, n >$ a n je celkový počet polygonů. Polygon P_i je reprezentován vrcholovými body $p_j[x_j, y_j]$ pro $j \in < 1, \dots, k >$, kde k je počet vrcholů polygonu. Hrany polygonu jsou reprezentovány hodnotou σ_j. Dalším vstupem je určovaný bod $q = [x_q, y_q]$.
Pak hledáme: $i; q \in P_i$.
Problém se anglicky nazývá *Point Location Problem.*
Prakticky se tento problém řeší tím, že procházíme celou množinu M a zjišťujeme vztah bodu q a polygonu P_i (způsob určení vzájemného vztahu je popsán v následující kapitole).

4. Popis algoritmů

Winding Number Algorithm

Na vstupu pro tento algoritmus je polygon P_i. Dále je na vstupu zkoumaný bod q.
Spočteme sumu všech rotací

$$\Omega(q, P_i) = \sum_{j=1}^{k} \omega_j(p_j,\ q, p_{j+1}),$$

které průvodič provede nad všemi body p_j.
Úhel ω_j je spočten ze vzorce

$$\cos \omega_j = \frac{\overrightarrow{u_j} * \overrightarrow{v_j}}{|\overrightarrow{u_j}| * |\overrightarrow{v_j}|}, \text{ kde } \overrightarrow{u_j} = (q, p_j),\ \overrightarrow{v_j} = (q, p_{j+1}).$$

Ω může nabývat hodnot:

a) $360° \gg q \in P_j$
b) $0° \gg q \notin P_j$
c) Jiné ≫ bod je totožný s hranou nebo vrcholem polygonu.

Jestli je bod vlevo, vpravo nebo na hraně, poznáme podle hodnoty diskriminantu

$$\det = \begin{vmatrix} u_x & u_y \\ v_x & v_y \end{vmatrix}.$$

Mohou nastat tři situace:

a) $\det < 0 \gg q\ je\ vlevo$
b) $\det > 0 \gg q\ je\ vpravo$
c) $\det = 0 \gg q\ je\ na\ hraně$

Implementace:

1. Inicializuj $\Omega = 0$, tolerance ε
2. Opakuj pro ∀ trojice $< p_j, q, p_{j+1} >$:
3. Urči polohu q vzhledem k $p = < p_j, p_{j+1} >$
4. Urči $\omega_j(p_j, q, p_{j+1})$
5. Jestliže q je vlevo od (p_j, p_{j+1}), pak:
6. $\Omega = \Omega - \omega$
7. Jestliže q je vpravo od (p_j, p_{j+1}), pak:
8. $\Omega = \Omega + \omega$
9. Jestliže $|\Omega - 2\pi| < \varepsilon$, pak:
10. Vrať: $q \in P_i$
11. Jinak:
12. Vrať: $q \notin P_i$

Ray Crossing Algorithm

Na vstupu tomuto algoritmu jsou stejné objekty jako v u algoritmu *Winding Number Algorithm*.
Do bodu q umístíme počátek lokální souřadnicové soustavy (q, x', y'), který má osy rovnoběžné s hlavní souřadnicovou soustavou. Následně určíme počet průsečíků osy x' s polygonem P_i. Ze všech určených průsečíku vybereme ty, které mají souřadnici $x > 0$. Jestliže je počet těchto průsečíku lichý, pak je q uvnitř polygonu. Je-li naopak sudý, pak je q vně polygonu.
Průsečík x'_m osy x' se stranou polygonu se určí podle vzorce:
$x'_m = \frac{x'_j y'_{j-1} - x'_{j-1} y'_j}{y'_j - y'_{j-1}}$.

Implementace:

1. Inicializuj $k = 0; k \approx$ počet průsečíků
2. Pro $\forall\, p_j$ opakuj:
3. $x'_j = x_j - x_q;\ y'_j = y_j - y_q$
4. Jestliže $\left(y'_j > 0\right)$ && $\left(y'_{j-1} \leq 0\right)$ || $\left(y'_{j-1} > 0\right)$ && $\left(y'_j \leq 0\right)$, pak:
5. Spočti: x'_m
6. Jestliže $x'_m > 0$, pak:
7. $k++$
8. Jestliže k je liché, pak:
9. Vrať $q \in P_i$
10. Jinak:
11. Vrať: $q \notin P_i$

5. Problematické situace a jejich rozbor

První problematickou situací je detekce bodu, který se nachází na hranici polygonu.

Winding Number Algorithm – bod na hranici polygonu

Při výpočtu $\omega_j(p_j, q, p_{j+1})$ se při určení směru rotace určuje determinant det matice

$$\begin{pmatrix} u_x & u_y \\ v_x & v_y \end{pmatrix}, \text{ kde } \overrightarrow{u_j} = (q, p_j), \overrightarrow{v_j} = (q, p_{j+1}).$$

Jestliže $|det| < \varepsilon$, kde ε je stanovená tolerance numerické přesnosti výpočtu, pak můžeme prohlásit, že bod leží na přímce dané stranou polygonu. Za předpokladu, že bod leží v minimálním ohraničujícím obdélníku strany polygonu, který má strany rovnoběžné s osami souřadnicové soustavy, pak můžeme prohlásit, že bod náleží dané hraně polygonu.

Ray Crossing Algorithm – bod na hranici polygonu

Po transformaci polygonu do místní souřadnicové soustavy se spočtou průsečíky x'_m, y'_m hrany polygonu s osami. Při splnění podmínky

$$|x'_m| < \varepsilon \;\&\&\; |y'_m| < \varepsilon$$

je bod na hraně polygonu a algoritmus může být zastaven.

Další problematickou situací je bod, který je totožný s vrcholy polygonu. Tento problém je pro oba algoritmy řešen totožným způsobem.

Oba algoritmy – bod totožný s vrcholem polygonu

Pro každý vrchol je spočtena vzdálenost s od určovaného bodu. Za předpokladu, že všechny délky $s < \varepsilon$, lze algoritmus zastavit s tvrzením, že bod je totožný s vrcholem polygonu.

6. Vstupní data

Na úvodní obrazovce se nachází tlačítko *Load*, po jehož kliknutí se otevře průzkumník souborů, pomocí něhož najdeme požadovaný soubor.

Soubor s polygony je ve formátu csv. Jeden řádek vstupního souboru obsahuje jeden polygon. Hlavička souboru má tvar:

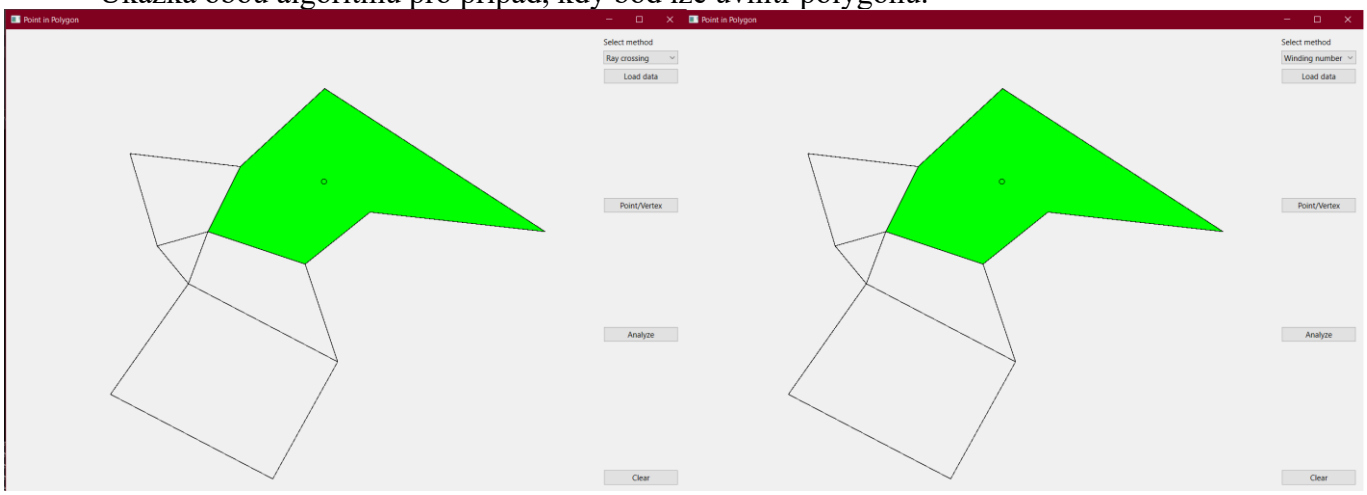
Object, ID, location

Formát řádku vypadá následovně:
Polygon, ID, “$x_1, y_1, \dots, x_k, y_k$*“*

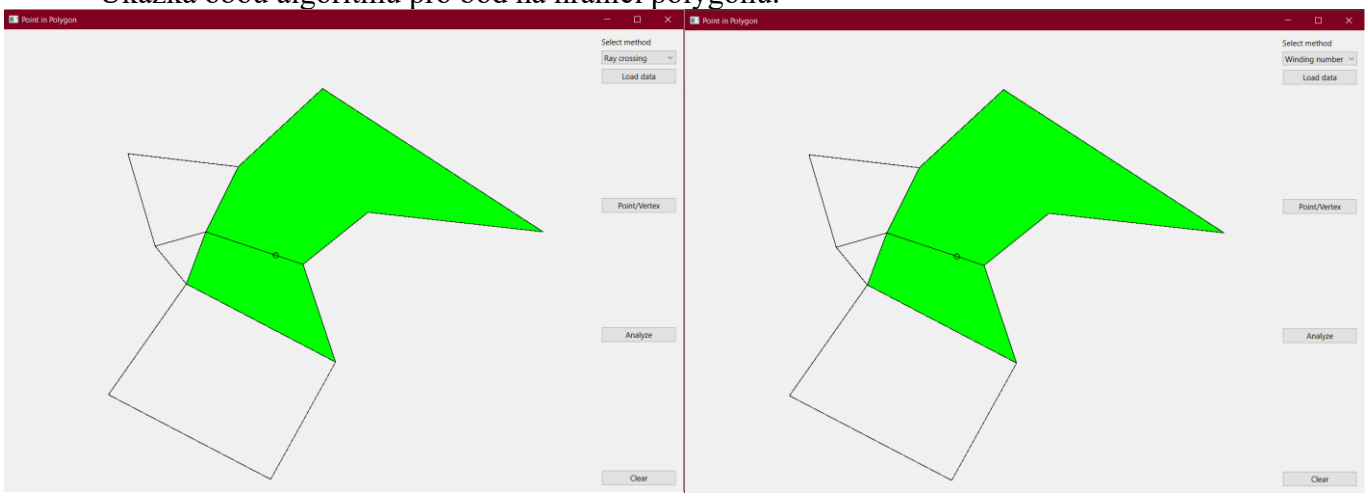
7. Výstupní data

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

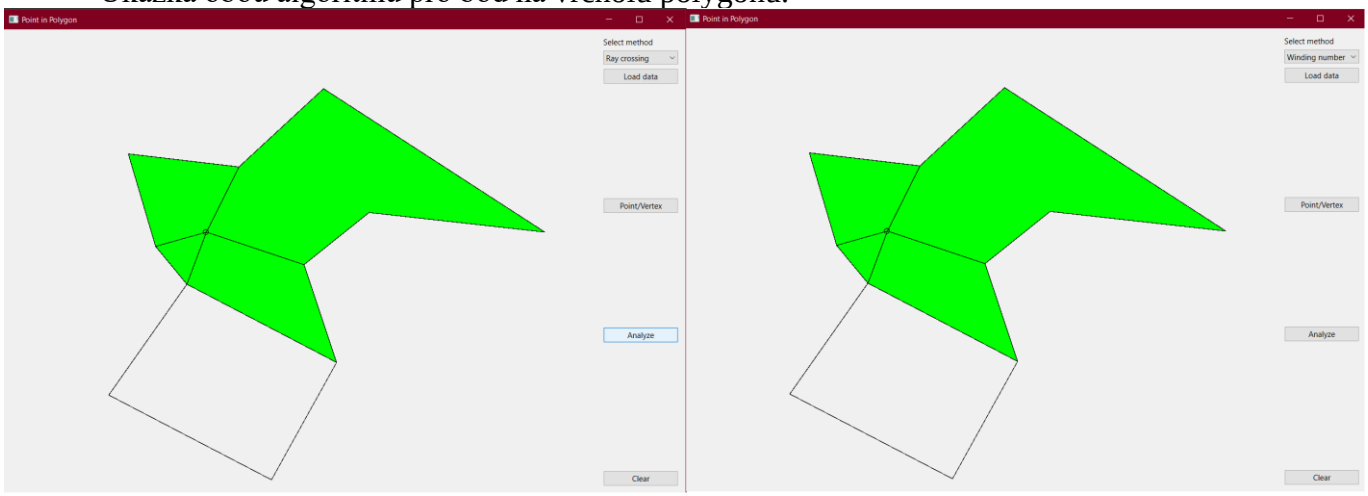
Ukázka obou algoritmů pro případ, kdy bod lže uvnitř polygonu.



Ukázka obou algoritmů pro bod na hranici polygonu.

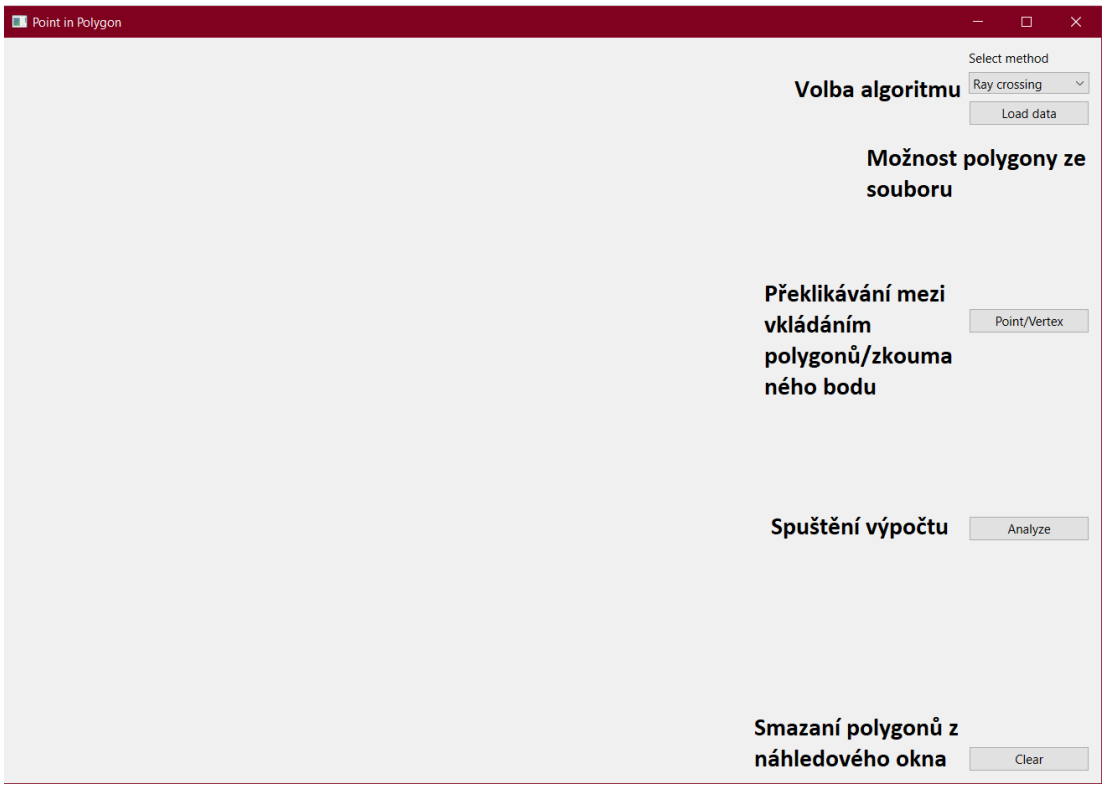


Ukázka obou algoritmů pro bod na vrcholu polygonu.



8. Vzhled aplikace

Na úvodní obrazovce aplikace se nachází v levé části náhledové okno *Canvas* třídy *Draw*.
Na pravé straně je pak panel s obslužnými elementy. Prvním je *comboBox* třídy *QComboBox*, který slouží k výběru algoritmu. Dalšími prvky jsou pak tlačítka třídy *QPushButton*, a to: *pushButton*, *pushButtonAnalyze*, *pushButtonClear* a *pushButtonLoad*.



9. Dokumentace

1. Třída *Algorithms*
   a) *bool ifCloseToPoint (QPoint &q, vector<QPoint> &pol);*
      - rozhodne, zda *q* leží blízko nějakého polygonového bodu
   b) *int getPointLinePosition(QPoint &a,QPoint &p1,QPoint &p2)*
      - analyzuje vzájemný vztah bodu a přímky
   c) *double get2LinesAngle(QPoint &p1, QPoint &p2, QPoint &p3, QPoint &p4)*
      - spočte úhel mezi dvěma vektory
   d) *int getPositionWinding(QPoint &q, vector<QPoint> &pol)*
      - analyzuje vztah bodu a polygonu pomocí *Winding Number Algorithm*
   e) *vector<QPoint> getLocalCoords(QPoint &q, vector<QPoint> &pol)*
      - transformuje souřadnice polygonu do místního souřadnicového systému s počátkem v bodě *q*
   f) *int getPositionRay(QPoint &q, vector<QPoint> &pol)*
      - analyzuje vztah bodu a polygonu pomocí *Ray Crossing Algorithm*
   g) *int processPolygons(QPoint &q, vector<QPolygon> &pols, QString &Alg, vector<int> &results)*
      - analyzuje vztah všech polygonů s bodem *q*

2. Třída *CSV*
   a) *vector<QPolygon> read_csv(string &filename)*
      - čte vstupní *csv* soubor a ukládá polygon do struktury *vector<QPolygon>*

3. Třída *Draw*
   a) *void paintEvent(QPaintEvent *event)*
      - vykreslí polygony na *Canvas*
   b) *void mousePressEvent(QMouseEvent *event)*
      - vrátí souřadnice kurzoru po kliknutí na *Canvas*
   c) *void clear()*
      - vyčistí *Canvas*
   d) *void drawPolygons(vector<QPolygon> &pols)*
      - vykreslí polygony na *Canvas*
   e) *void addResults(vector<int> &results)*
      - označuje polygony, které mají být zvýrazněné

4. Třída *Widget*
   a) *void on_pushButtonClear_clicked()*
      - vyčistí *Canvas*
   b) *void on_pushButton_clicked()*
      - mění způsob zadávání dat mezi kreslením polygonu/požadovaného bodu
   c) *void on_pushButtonAnalyze_clicked()*
      - provede požadovanou analýzu
   d) *void on_pushButtonLoad_clicked()*
      - otevře průzkumníka souborů za účelem ukázání na soubor s polygony

10. Závěr

Výsledkem je aplikace *Point in Polygon* s grafickým uživatelským rozhraním, která je uvolněna pod licencí GNU GPL. Aplikace je psána v jazyce C++. Pro grafické uživatelské rozhraní byl použit framework QT.

Aplikace umožňuje nahrát polygony z textového souboru, případně si naklikat vlastní polygon. Pro analýzu vzájemného vztahu bodu a polygonu byly zvoleny dva algoritmy, a to: *Winding Number Algorithm* a *Ray Crossing Algorithm*.

Pro oba výše uvedené polygony byla implementována řešení pro případy, kdy bod

a) leží uvnitř polygonu,
b) leží vně polygonu,
c) bod leží na hraně polygonu,
d) bod leží ve vrcholu polygonu.

Jako možná cesta vylepšení aplikace se jeví funkcionalita na generování náhodných nekonvexních polygonů.

11. Zdroje

[1] Geometrické vyhledávání [online]. [cit. 2021-10-18]. Dostupné z: https://web.natur.cuni.cz/~bayertom/images/courses/Adk/adk3.pdf