Geometrické vyhledávání bodu
Algoritmy digitální kartografie a GIS

Jakub Šnopl, Dennis Dvořák, Jakub Felenda

31. prosince 2023

Obsah

1 Zadání **3**

2 Bonusové úlohy **4**

3 Použité algoritmy **4**

3.1 Načtení dat 4

3.2 Winding Number Algorithm 5

3.3 Ray - Crossing Algorithm 5

4 Problematické situace a jejich rozbor **6**

4.1 Bod totožný s vrcholem mnohoúhelníku 6

4.2 Bod leží na hraně mnohoúhelníku 6

4.2.1 Winding Number Algorithm 6

4.2.2 Ray - Crossing Algorithm 7

5 Vstupní data **7**

6 Vzhled aplikace **8**

7 Dokumentace **9**

7.1 Třída Algorithms 9

7.2 Třída Draw 9

7.3 Třída Data 10
7.4 Třída Mainform 10

8 Závěr **11**

Reference **12**

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 Ray Crossing 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 polygonu.	10b
Analýza polohy bodu (uvnitř/vně) metodou Winding Number Algorithm.	*+10b*
Ošetření singulárního případu u Winding Number Algorithm: bod leží na hraně polygonu.	*+5b*
Ošetření singulárního případu u Ray Crossing 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*
Rychlé vyhledání potenciálních polygonů (bod uvnitř min-max boxu).	*+10b*
Řešení pro polygony s dírami.	*+10b*
Max celkem:	**55b**

Čas zpracování: 1 týden.

2 Bonusové úlohy

- Analýza polohy bodu (vně/uvnitř) polygonu metodou Ray - Crossing Algorithm.
- Ošetření singulárního případu u Winding Number Algorithm: bod leží na hraně polygonu.
- Ošetření singulárního případu u Ray - Crossing Algorithm: bod leží na hraně polygonu.

3 Použité algoritmy

Mezi základní úlohy algoritmizace digitální kartografie patří geometrická lokalizace bodu (point location). Je základní úlohou v oblastech geografických informačních systémů (GIS), v počítačem podporovaném kreslení (CAD) či v počítačové grafice.

Je dán bod $q[x_q, y_q]$ se známými souřadnicemi a množinu M, ve které se nachází n mnohoúhelníků P_n. Mnohoúhelník se skládá z i počtu vrcholů. Základním principem lokalizace bodu je hledání, zdali bod q leží uvnitř, vně nebo na hranici konvexních, či nekonvexních mnohoúhelníků. Pro řešení se používá mnoho metod, pro náš případ jsme využili metody: *Winding Number Algorithm* a *Ray Crossing Algorithm*.

3.1 Načtení dat

Načítaná data jsou ve formátu popsaném viz kapitola 5. O načítání dat se stará třída s názvem *Data*, která obsahuje jednu metodu *readPolygonsFromFile* jejíž algoritmus je popsán níže.

Algoritmus načtení dat:

1. Otevření vybraného textového souboru s daty
2. Opakuj až na konec souboru (poslední řádek)
   (a) Vytvoření objektu polygonu
   (b) 0pakuj dokud není řádek prázdný
      i. Čti řádek
      ii. Rozděl řádek podle separátoru
      iii. Pokud je rozdělený řádek o velikosti větší jak 2
         A. Ulož souřadnice X a Y
   (c) Ulož polygon

3.2 Winding Number Algorithm

Na vstupu je dán bod q a mnohoúhelník P. Princip úlohy spočívá ve výpočtu sumy Ω všech rotací ω_i,

$$\Omega(q, P) = \sum_{i=1}^{n} \omega_i(p_i, q, p_{i+1}) \tag{1}$$

který průvodič opíše nad všemi vrcholy p_i mnohoúhelníku P, n je počet vrcholů a ω_i lze vypočítat následovně

$$cos(\omega_i) = \frac{\vec{u_i} * \vec{v_i}}{|\vec{u_i}| * |\vec{v_i}|}, \quad kde\ \vec{u_i} = (q, p_i)\ ,\ \vec{v_i} = (q, p_{i+1}). \tag{2}$$

Ω může nabývat hodnot:

- 2π - bod leží uvnitř polygonu
- 0 - bod leží vně polygonu

Algoritmus výpočtu:

1. Inicializuj $\Omega = 0$, tolerance ϵ
2. Opakuj pro trojice $\forall < p_i, q, p_{i+1} >$
   (a) Urči polohu q vzhledem k $p = (p_i, p_{i+1})$
   (b) Urči úhel $\omega_i = \angle p_i, q, p_{i+1}$
   (c) Pokud q je vlevo od (p_i, p_{i+1}), pak $\Omega = \Omega - \omega$
   (d) Jinak q je vpravo od (p_i, p_{i+1}), pak $\Omega = \Omega + \omega$
3. Pokud $|\Omega - 2\pi| < \epsilon$, pak $q \in P$
4. Jinak $q \notin P$

3.3 Ray - Crossing Algorithm

Stejně jako u předchozí metody je vstupem bod q a mnohoúhelník P. Metoda spočívá v tom, že se do bodu q vloží počátek lokální souřadnicové soustavy (q, x', y'). Osy lokální souřadnicové soustavy jsou rovnoběžné s hlavní souřadnicovou soustavou. Poté je určen počet průsečíků x'_m (3) osy x' s hranami mnohoúhelníku. Z vybraných průsečíků jsou vybrány

takové, které splňují kritérium $x > 0$. Pokud takovýchto průsečíků je lichý počet, pak je bod q uvnitř polygonu, v opačném případě je vně polygonu.

$$x'_m = \frac{x'_i y'_{i-1} - x'_{i-1} y'_i}{y_i -' y'_{i-1}} \tag{3}$$

Algoritmus výpočtu:

1. Inicializuj $k = 0$
2. Opakuj pro $\forall$ body $p_i \in P$
   (a) $x'_i = x_i - x_q$
   (b) $y'_i = y_i - y_q$
   (c) Pokud $(y'_i > 0)\&(y'_{i-1} \leq 0)||(y'_{i-1} > 0)\&(y'_i \leq 0)$
      i. $x'_m = (x'_i y'_{i-1} - x'_{i-1} y'_i)/(y'_i - y'_{i-1})$
      ii. Pokud $x'_m > 0$, pak $k = k + 1$
3. Pokud $(k\%2) \neq 0$, pak $q \in P$
4. Jinak $q \notin P$

4 Problematické situace a jejich rozbor

4.1 Bod totožný s vrcholem mnohoúhelníku

Tato situace může být podchycena u obou zmíněných algoritmů stejně, a to pouhým porovnáním souřadnic vrcholů p_i mnohoúhelníku P a bodu q. Pokud se souřadnice bodů v rámci zadané tolerance neliší, lze je prohlásit za totožné a ukončit výpočet.

4.2 Bod leží na hraně mnohoúhelníku

4.2.1 Winding Number Algorithm

Při řešení úlohy Winding Number algoritmem, je zapotřebí určit, zdali se bod nachází vlevo či vpravo od přímky danou body p_i, p_{i+1} a to pomocí výpočtu determinantu matice $A = \begin{pmatrix} u_x & u_y \\ v_y & v_x \end{pmatrix}$, pokud:

- $det(A) > 0$, bod q leží v pravé polorovině
- $det(A) < 0$, bod q leží v levé polorovině

- $det(A) = 0$, **bod q leží na přímce**

Další potřebnou podmínkou je, že bod q musí ležet v minimálním ohraničujícím obdélníku bodů p_i, p_{i+1}.

Dalším možným řešením je, přidání podmínky (4), pokud je splněna, můžeme prohlásit, že bod q leží na hraně mnohoúhelníku P.

$$\omega_i(p_i, q, p_{i+1}) = \pi \pm \epsilon \tag{4}$$

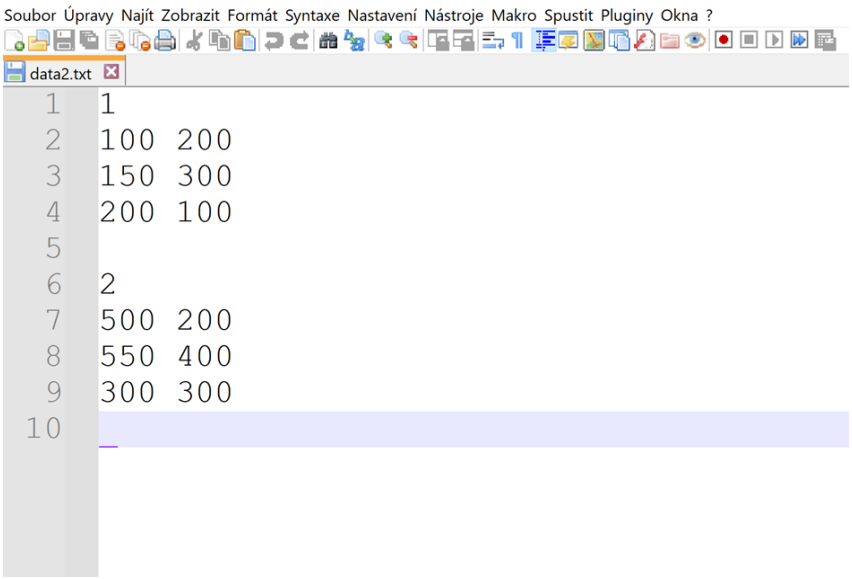
4.2.2 Ray - Crossing Algorithm

Pokud je splněna podmínka, že vypočtené průsečíky $|x'_m| < \epsilon \ \wedge \ |y'_m| < \epsilon$. Bod q leží na hraně mnohoúhelníku P.

5 Vstupní data

Aplikace podporuje načítání dat ve formátu *txt*. Polygony jsou reprezentovány špagetovým modelem viz obr. č. 1. Po řádku, na kterém se nachází jedno číslo (*ID* polygonu) vždy následují jednotlivé vrcholy $[x, y]$, polygon je ukončen prázdným řádkem.

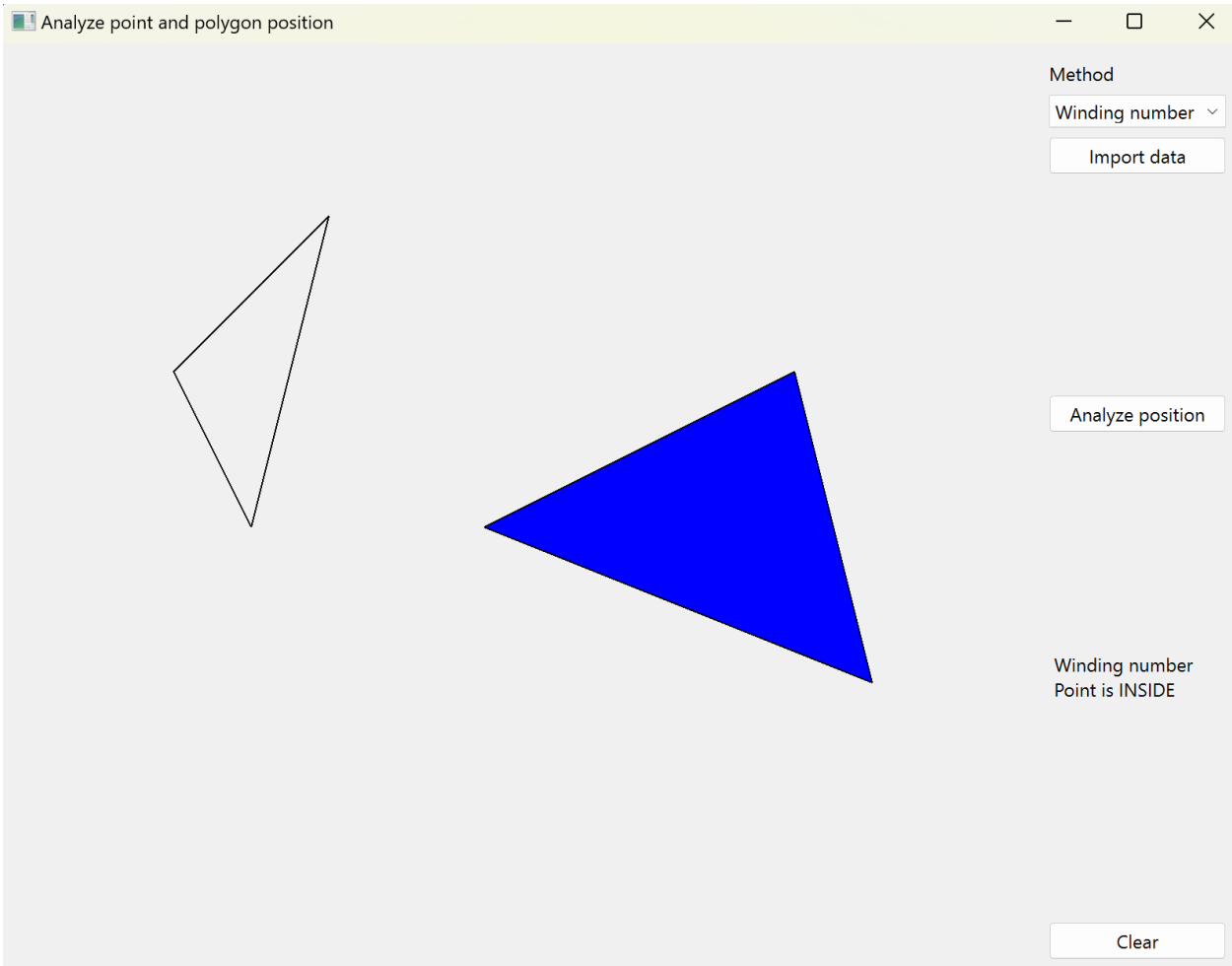
Po kliknutí na tlačítko „Import data“ se otevře průzkumník souborů, kde lze textový soubor importovat do programu.



Obrázek 1: Ukázka vstupních dat

6 Vzhled aplikace

Hlavní část uživatelského rozhraní zaujímá prvek třídy QWidget, tzv. plátno, na kterém se vykreslují analyzované polygony a bod. V pravé části se nacházejí ovládací prvky aplikace, využívající tříd QComboBox, QPushButton a QLabel. Jednotlivá tlačítka umožňují výběr metody, import dat, spuštění výpočtu a vyčištění plátna.



Obrázek 2: Uživatelské rozhraní

7 Dokumentace

7.1 Třída Algorithms

- int getPointAndLinePosition(QPoint &a, QPoint &p1, QPoint &p2)
  Poloha bodu vůči přímce
- double get2LinesAngle(QPoint &p1, QPoint &p2, QPoint &p3, QPoint &p4)
  Úhel svírající dvě přímky
- int getPointAndPolygonPositionWinding(QPoint &q, QPolygon &pol)
  Winding Number Algorithm
- int getPointAndPolygonPositionRayCrossing(QPoint &q,QPolygon &pol)
  Ray - Crossing Algorithm
- int processAll(vector QPolygon &polygons, QPoint &point, int &algorithm_index)
  Hromadné zpracování všech dat

7.2 Třída Draw

- explicit Draw(QWidget *parent = nullptr)
- void mousePressEvent(QMouseEvent *event)
  Souřadnice kurzoru po kliknutí na plátno
- void paintEvent(QPaintEvent *event)
  Vykreslení všech prvků na plátno
- void clear()
  Vyčistí plátno
- QPoint getQ()
  Souřadnice bodu na plátně
- QPolygon getPol()
  Souřadnice polygonu
- void drawPolygons(vector QPolygon &polygons)
  Vykreslení polygonů na plátno
- vector QPolygon getPolData()
  Navrátí načtená data polygonů

- void highlitePolygon(QPolygon &polygon_highlite)
  Zvýraznění polygonu
- void reset_brush()
  Uvedení barvy polygonu do původního nastavení

7.3 Třída Data

- vector QPolygon readPolygonsFromFile(QString &filePath)
  Načtení dat ze souboru

7.4 Třída Mainform

- void on_pushButton_3_clicked()
  Vyčištění plátna
- void on_pushButton_2_clicked()
  Spustí analýzu
- void on_pushButton_data_clicked()
  Import dat

8 Závěr

V rámci první úlohy předmětu Algoritmy digitální kartografie a GIS byla vytvořena aplikace nesoucí název PointAndPolygonPosition. Aplikace byla napsána v jazyce C++ ve vývojovém prostředí Qt Creator.

Aplikace umožňuje načítání textových souborů ve výše zmíněném formátu a následné určení polohy bodu vůči importovaným polygonům. Na výběr je celkem ze dvou metod výpočtu (Winding number a Ray - Crossing). Uživatel je o poloze bodu obeznámen výpisem v textovém okně a případným vybarvením polygonu.

Výsledná aplikace je k nalezení na této webové adrese: `https://github.com/jaksno/adk_2023_24`.

Reference

[1] BAYER Tomáš, *Point location problem*, [online]. [cit. 2023-10-29]. Dostupné z: `https://web.natur.cuni.cz/~bayertom/images/courses/Adk/adk3_new.pdf`