

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

Název předmětu

ALGORITMY DIGITÁLNÍ KARTOGRAFIE A GIS

Úloha

1

Název úlohy

Geometrické vyhledávání bodu

Akademický rok

2018/2019

Semestr

3.

Studijní
skupina

60

Vypracoval

Janovský Michal
karving47@gmail.com

Datum

24. 10. 2018

Klasifikace

1 Obsah

1	Zadání úlohy	3
2	Údaje o bonusových úlohách	3
3	Popis a rozbor problému + vzorce.....	3
4	Popisy algoritmů formálním jazykem.	4
4.1	Winding Number	4
4.2	Ray Crossing	4
5	Problematické situace a jejich rozbor + ošetření těchto situací v kódu.....	4
5.1	Bod na hraně polygonu, bod je totožný s bodem polygonu.....	4
6	Vstupní data, formát vstupních dat, popis.....	5
7	Výstupní data, formát výstupních dat, popis.....	5
8	Printscreen vytvořené aplikace.	5
9	Dokumentaci: popis tříd, datových položek a jednotlivých metod.....	7
9.1	Třída Algorithms	7
9.1.1	Metody třídy Algorithms.....	7
9.2	Třída Draw	7
9.2.1	Datové položky třídy Draw	7
9.2.2	Metody třídy Draw	7
9.3	Třída Widget.....	8
9.3.1	Sloty třídy Widget	8
10	Závěr, možné či neřešené problémy, náměty na vylepšení.	8
10.1	Závěr	8
10.2	Náměty na vylepšení.....	9
10.3	Neřešené problémy	9
11	Zdroje	9

1 Zadání úlohy

Kopie originálního zadání úlohy:

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 následující algoritmy pro geometrické vyhledávání:

- 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 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
Ošetření singulárního případu u Winding Number Algorithm: bod leží na hraně polygonu.	+2b
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
Algoritmus pro automatické generování nekonvexních polygonů.	+5b
Max celkem:	21b

2 Údaje o bonusových úlohách

V rámci úlohy byly implementovány první, druhá a třetí bonusová úloha, tedy:

- Ošetření singulárního případu u Winding Number Algorithmu: bod leží na hraně polygonu
- Ošetření singulárního případu u obou algoritmů: bod je totožný s vrcholem jednoho či více polygonů
- Zvýraznění všech polygonů pro oba výše uvedené singulární případy

3 Popis a rozbor problému + vzorce

Na vstupu máme sadu uzavřených polygonů a jeden námi zvolený bod.

Řešený problém spočívá v nalezení těch polygonů, které obsahují námi zvolený bod (označený q). Dalším problémem je tvar polygonů, u kterých zjišťujeme, zda obsahují bod q , kde jejich tvar může být konvexní či nekonvexní, kde u nekonvexních polygonů dochází k dalším problémům, které se musejí vyřešit již v návrhu implementace řešení. Neposledním z problémů je ošetření singulárních případů umístění bodu q oproti testovaným polygonům. V této úloze se výše zmíněné problémy řeší dvěma metodami, a to metodami *Winding Number* a *Ray Crossing*. Popis těchto metod a použité vzorce jsou k nalezení v další kapitole.

4 Popisy algoritmů formálním jazykem.

Metody použité v této úloze, byly implementovány v programovacím jazyce C++ v prostředí Qt Creator. V TZ jsou popsány pouze použité metody, nicméně se nevylučuje existence mnoha dalších metod.

4.1 Winding Number

Na vstupu máme uzavřený polygon a bod q , jehož polohu vztahenou k polygonu se snažíme určit. Z bodu q pozorujeme proti směru hodinových ručiček postupně jdoucí body polygonu, přičemž zaznamenáváme směr otáčení a hodnotu úhlu ω , o kterou jsme se otočili. Takto získáme soubor úhlů o 2 směrech, kde úhly ve směru proti hodinovým ručičkám mají kladnou hodnotu a úhly v opačném směru mají hodnotu zápornou. Ze sumy těchto úhlů se dá vypočítat tzv. *winding number*, což je počet oběhů. Tato hodnota však počítána nebyla, jelikož si jsme schopni vystačit pouze s naměřenými úhly. Do výpočtu se zavádí hodnota ε , tuto hodnotu dále používáme jako max. odchylku od námi požadované hodnoty, a to z toho důvodu, že ve výpočtech úhlů se vyskytují chyby ze zaokrouhlení a strojové přesnosti počítače.

Postup výpočtu:

- 1) Vstup $\omega = 0, \varepsilon = 10^{-x}$, kde x je námi zvolená hodnota
- 2) Orientace z bodu q na bod p_i a následující body p_{i+1}
- 3) Určení úhlu ω_i mezi p_i, p_{i+1}
- 4) $\omega = \omega + \omega_i$ proti směru hodin; $\omega = \omega - \omega_i$ po směru hodin

Výsledky výpočtu:

- a) pokud $(|\omega - 2\pi| < \varepsilon)$, pak $q \in P$
- b) pokud $(|\omega - 2\pi| > \varepsilon)$, pak $q \notin P$

4.2 Ray Crossing

Na vstupu máme uzavřený polygon a bod q , jehož polohu vztahenou k polygonu se snažíme určit. Z bodu q vedeme paprsek v horizontální poloze, který protíná polygon v několika bodech. Podle počtu průniků se určí, zda bod leží uvnitř či vně polygonu.

Postup výpočtu:

- 1) Vstup počet průsečíků = 0
- 2) Redukce souřadnic X a Y bodů polygonu k bodu q
- 3) Pro všechny body:
 - a) if $(y'_i > 0) \iff (y'_{i-1} \leq 0) \parallel (y'_{i-1} > 0) \iff (y'_i \leq 0)$
 - b) $x'_m = \frac{x'_i y'_{i-1} - x'_{i-1} y'_i}{y'_i - y'_{i-1}}$
 - c) if $(x'_m > 0)$, přičti 1 k průsečíkům

Výsledek výpočtu:

- a) pokud je počet průsečíků lichý, pak $q \in P$
- b) pokud je počet průsečíků sudý, pak $q \notin P$

5 Problematické situace a jejich rozbor + ošetření těchto situací v kódu.

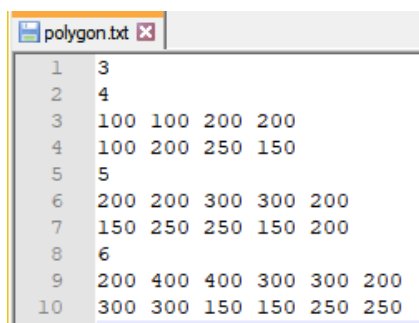
5.1 Bod na hraně polygonu, bod je totožný s bodem polygonu

Bod na hraně polygonu je třeba ošetřovat u metody Ray Crossing, zde se provádí posun, resp. redukce vrcholů polygonů směrem k poloze bodu q . Tento postup v implementaci však nebyl použit a bylo využito alternativního řešení popsaného níže.

Pro vyřešení tohoto problému bylo zvoleno jednoduché řešení. Máme bod q a koncové body hrany polygonu AB . Vypočteme vzdálenosti $\|Aq\|, \|Bq\|$ a porovnáme jejich součet se vzdáleností $\|AB\|$. Pokud se obě vzdálenosti shodují ($\pm \varepsilon$), pak bod q leží na hraně AB . Pokud se bod q shoduje s bodem A , nebo B , pak se shodují i vzdálenosti, a je bráno že bod leží na hraně a tudíž náleží oběma polygonům, a to pro všechny hrany které mají za koncový bod tento bod q , tudíž by měli být označeny všechny polygony s tímto bodem.

6 Vstupní data, formát vstupních dat, popis.

Vstupem do programu jsou souřadnice bodu q, které se získají „naklikáním“ přímo v okně programu. Dalším vstupem je textový soubor obsahující souřadnice rohových bodů polygonů.

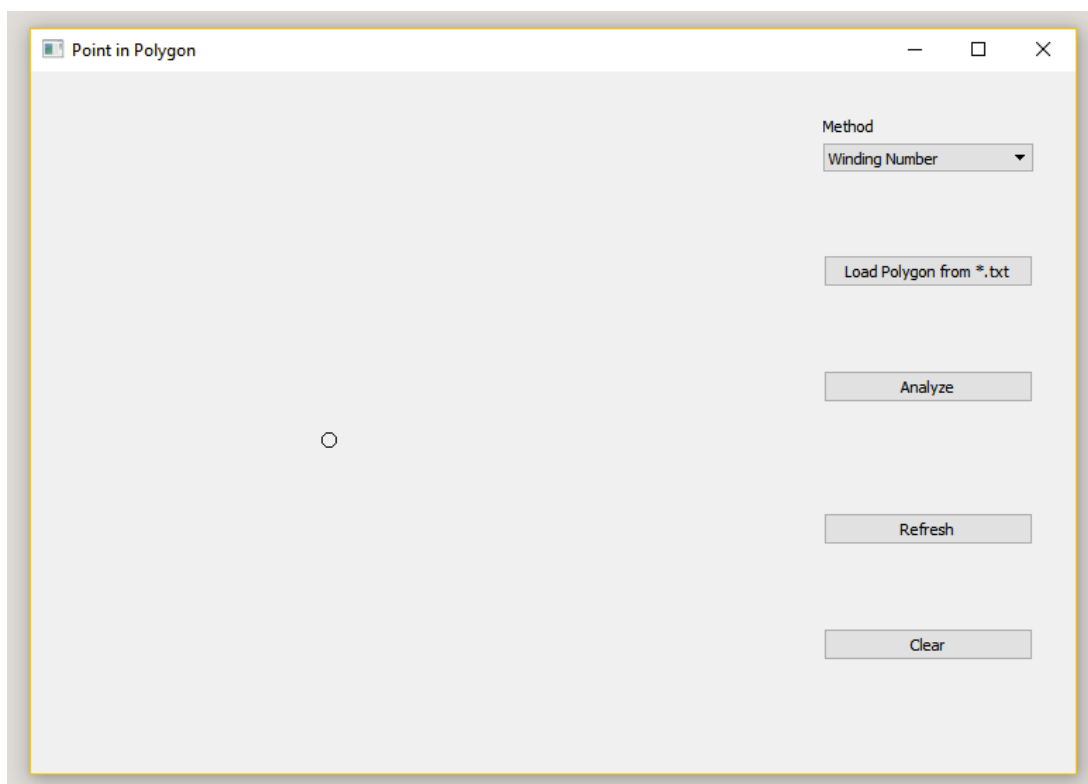


```
1 3
2 4
3 100 100 200 200
4 100 200 250 150
5 5
6 200 200 300 300 200
7 150 250 250 150 200
8 6
9 200 400 400 300 300 200
10 300 300 150 150 250 250
```

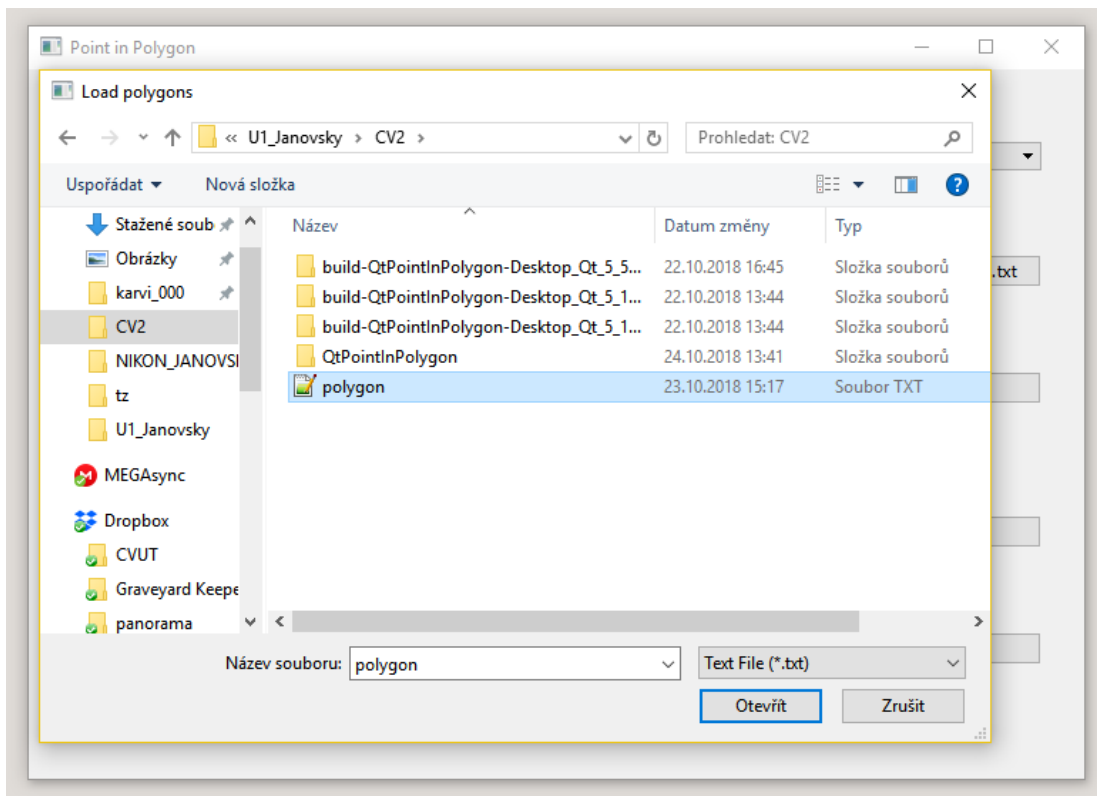
7 Výstupní data, formát výstupních dat, popis.

Výstupem z programu je vizuální zvýraznění všech polygonů, do kterých bod q spadá a zobrazení jejich počtu.

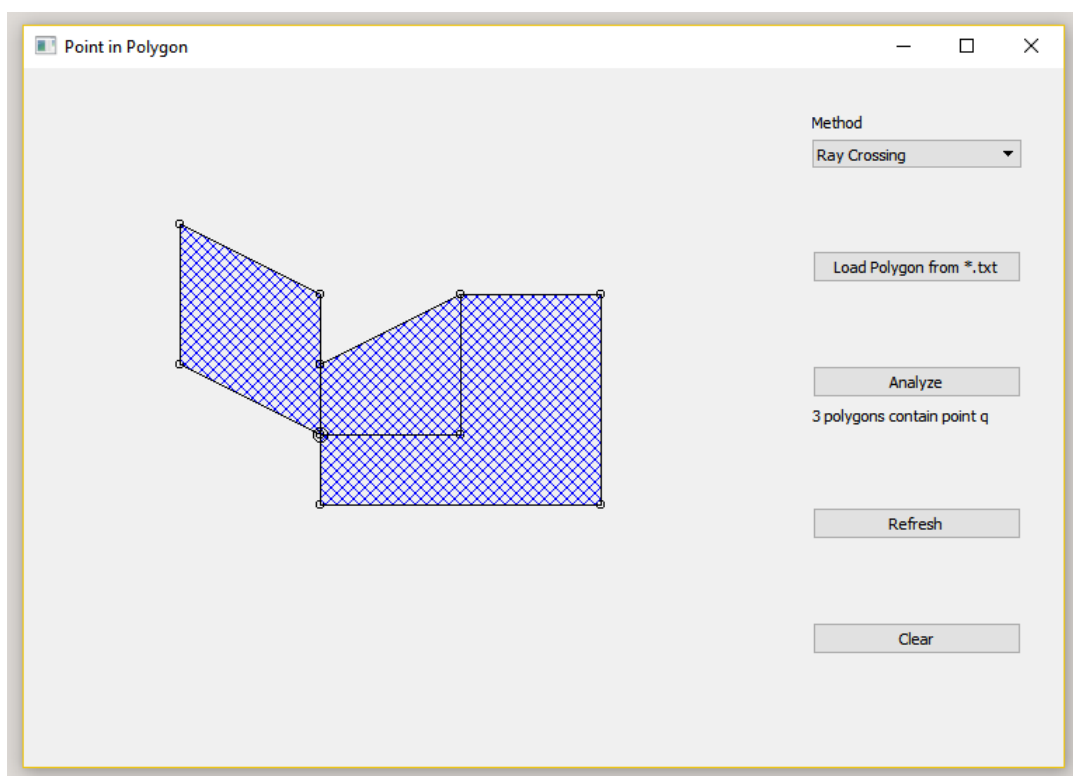
8 Printscreen vytvořené aplikace.



Obrázek 1 Program po spuštění



Obrázek 2 Výběr souboru s polygony



Obrázek 3 Testování na polygonech

9 Dokumentaci: popis tříd, datových položek a jednotlivých metod.

9.1 Třída Algorithms

Třída algoritmus zajišťuje veškeré výpočty pro určení vzájemné polohy bodu q a polygonů.

9.1.1 Metody třídy Algorithms

static int getPositionRay(QPoint q, vector<QPoint> pol)

metodou Ray Crossing určí zda bod je uvnitř nebo vně polygonu:

vstup: určovaný bod q, vektor se všemi polygony

výstup: 1 = je uvnitř, 0 = je venku

static int getPositionWinding(QPoint q, vector<QPoint> pol)

metodou Winding Number určí zda bod je uvnitř nebo vně polygonu:

vstup: určovaný bod q, vektor se všemi polygony

výstup: 1 = je uvnitř, 0 = je venku

static int getPointLinePosition(QPoint &q, QPoint &a, QPoint &b)

určí, zda je bod q vlevo, vpravo nebo na linii

vstup: bod q, počáteční a koncový bod linie

výstup: 0 = vpravo, 1 = vlevo, 2 = na linii

static double get2LinesAngle(QPoint &p1, QPoint &p2, QPoint &p3, QPoint &p4)

určí úhel mezi 2 liniemi

vstup: počáteční a koncový bod 2 linií

výstup: úhel mezi 2 liniemi

static vector<int> itWinding(QPoint &q, vector<vector<QPoint>> pol_list)

iterace getPositionWinding přes všechny polygony

vstup: bod q, vektor vektorů obsahující jednotlivé polygony

výstup: index polygonů, které obsahují q

static vector<int> itRay(QPoint &q, vector<vector<QPoint>> pol_list)

iterace getPositionRay přes všechny polygony

vstup: bod q, vektor vektorů obsahující jednotlivé polygony

výstup: index polygonů, které obsahují q

9.2 Třída Draw

Třída draw slouží k vykreslení GUI programu

9.2.1 Datové položky třídy Draw

bool draw_point

slouží k zapnutí nebo vypnutí kreslení bodu q

vstup: nic

výstup: true/false

QPoint q

Bod q obsahující souřadnice XY bodu q

vector<vector<QPoint>> poly_list

vektor obsahující vektory obsahující souřadnice XY lomových bodů polygonů

vector<int> result_polygons

vektor obsahující indexy polygonů, které mají být vabarveny

9.2.2 Metody třídy Draw

void paintEvent(QPaintEvent *e)

slouží k vykreslení bodu q, lomových bodů polygonů, šrafování a vybarvení výsledných polygonů

vstup: volání vykreslení

výstup: nic

void mousePressEvent(**QMouseEvent** *e)

slouží k nastavení (kliknutím v GUI) souřadnic XY bodu q

vstup: kliknutí v Canvas

výstup: nic

void clearCanvas()

slouží ke smazání obsahu Canvasu, tedy bodu q a polygonů

vstup: nic

výstup: nic

QPoint getPoint()

Předá souřadnice bodu q

Vstup: nic

Výstup: souřadnice bodu q

void import(const char* path, std::ifstream &file)

slouží k importu polygonů do programu

vstup: cesta k souboru, soubor ke čtení

výstup: nic

vector<vector<QPoint>> getList()

předá souřadnice polygonů

Vstup: nic

Výstup: vektor polygonů

void setResultPolygons(**vector<int>** res)

předá indexy polygonů obsahující bod q

Vstup: vektor indexů polygonů obsahující q pro zvolenou metodu

Výstup: vektor indexů polygonů obsahující q pro zvolenou metodu

9.3 Třída Widget

Třída Widget slouží k práci s widgety, tedy s tlačítky pro ovládání programu.

9.3.1 Sloty třídy Widget

void on_clear_clicked()

provede vyčištění GUI

Vstup: nic

Výstup: nic

void on_anal_clicked()

provede analýzu polohy bodu q k načteným polygonům

Vstup: nic

Výstup: nic

void on_Import_clicked()

vyvolá okno pro výběr *.txt souboru s polygony a načte jej

Vstup: nic

Výstup: nic

void on_repaint_clicked()

překreslí okno GUI

Vstup: nic

Výstup: nic

10 Závěr, možné či neřešené problémy, náměty na vylepšení.

10.1 Závěr

Dle zadání byla splněna povinná část programu a $\frac{3}{4}$ bonusových úloh. U bonusové úlohy „q zároveň vrcholem“ nebyl použit konvenční způsob řešení, který je uveden v přednáškách, ale i přes to dosahuje očekávaných výsledků.

10.2 Náměty na vylepšení

Dle slovního zadání by měla existovat samostatná třída pro vstup/výstup dat. Při pokusu provádět načítání dat se však nepodařilo umístit metodu Import do takovéto třídy, jelikož je provázána s widgetem a třídou Draw. Bylo by zajímavé pokusit se o obejití této vazby např. předáváním parametrů, či najít jiné řešení provedení. Dále by se při načítání dat jistě našla elegantnější cesta, jak načítat data ze souboru než za pomoci dočasných objektů.

10.3 Neřešené problémy

Mezi neřešené (ne však problémy) se dá uvést poslední bonusová úloha, která z časových důvodů nebyla implementována.

11 Zdroje

<http://www.cplusplus.com/reference/fstream/ifstream/>

<http://doc.qt.io/archives/qt-4.8/qbrush.html>

<https://en.cppreference.com/w/cpp/language/types>

<http://doc.qt.io/qt-5/qfiledialog.html>

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

načítání ze souboru

zvýraznění/vybarvení polygonu

signed/unsigned, řešení warningů

otevírání souborů (widgety)

vzorce pro použité metody