ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

FAKULTA STAVEBNÍ, OBOR GEODÉZIE A KARTOGRAFIE

KATEDRA GEOMATIKY

název předmětu

ALGORITMY DIGITÁLNÍ KARTOGRAFIE A GIS

číslo úlohy	název úlohy:			
1	**Geometrické vyhledávání bodu**			
školní rok:	semestr:	zpracovali:	datum:	klasifikace:
2019/20	zimní	**David Němec, Jan Šartner**	15. 10.	

Technická zpráva

Geometrické vyhledávání bodu

1. Zadání

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

Výstup: P_i, $q \subset P_i$.

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.

2. Doplňující úlohy

2.1 Ošetření případu, kdy bod leží na hranici dvou polygonů u algoritmu Winding Number
-řešeno

2.2 Ošetření případu, kdy je bod totožný s vrcholem jednoho nebo více polygonů u obou algoritmů
-řešeno

2.3 Zvýraznění všech dotčených polygonů pro případy 1) a 2)
-řešeno

2.4 Možnost automatického vygenerování nekonvexních polygonů
-neřešeno

3. Problematika vyhodnocování polohy bodu a polygonu

3.1 Algoritmus Winding Number

3.1.1 Potřebné vzorce

Výpočet úhlu P_1, Q, P_2 (P_1, P_2 – krajní body analyzované hrany polygonu; Q – analyzovaný bod):

$$\vec{u} = |P_1Q|$$

$$\vec{v} = |QP_2|$$

$$\omega = \left|\text{acos}\left(\frac{u * v}{|u| \cdot |v|}\right)\right|$$

Výpočet pozice bodu vzhledem k hraně polygonu:

$$\vec{u} = |P_1P_2|$$

$$\vec{v} = |P_1Q|$$

$$t = u \times v$$

-na základě znaménka hodnoty t lze rozhodnout, ve které polorovině, určené analyzovanou hranou, bod Q leží.

3.1.2 Princip

Tento algoritmus využívá orientace úhlu mezi krajními body hrany polygonu s vrcholem v analyzovaném bodě. Na prvním bodě polygonu lze na analyzovaný bod definovat pohled, který se přesouvá (otáčí) ve zvoleném směru přes všechny hrany polygonu. V případě, kdy pohled zamířil opět na první bod a prošel tedy jedním směrem přes hrany polygonu, musel se, v případě bodu ležícího uvnitř polygonu, otočit, oproti původnímu stavu, kolem tohoto bodu o úhel 2π. V případě bodu ležícího vně polygonu pohled oproti původnímu stavu nevykonal ani jednu otočku.

3.1.3 Výpočet Winding number

1. Inicializace $\Omega = 0$, ε = „zvolená hodnota tolerance“
2. Opakování pro všechny p_i, q, p_{i+1}
3. {
4. Zjištění polohy q vzhledem k hraně (p_i, p_{i+1})
5. Výpočet úhlu ω_i (p_i, q, p_{i+1})
6. Když q náleží levé polorovině
7. $\Omega = \Omega + \omega_i$
8. Jinak, když q náleží pravé polorovině
9. $\Omega = \Omega - \omega_i$
10. }
11. Když ($|\Omega - 2*pi| < \varepsilon$) nebo ($|\Omega + 2*pi| < \varepsilon$)
12. Bod leží uvnitř polygonu
13. Jinak
14. Bod neleží uvnitř polygonu

-v případě že je ošetřen kolineární případ, kdy bod leží na linii dané hranou polygonu, se pod podmínkou „jinak“ skrývá druhý a zároveň poslední disjunktní případ $\Omega = 0$
-při testování hodnoty Ω je zde disjunkcí přidán i případ se znaménkem + pro případ, že by se jednalo o CV orientaci

3.2 Algoritmus Ray Crossing

3.2.1 Potřebné vzorce

Redukce souřadnic polygonu k bodu Q:

$$X'_P = X_P - X_Q$$

$$Y'_P = Y_P - Y_Q$$

Výpočet průsečíku s paprskem y‘ = 0 (analyzovaná hrana P_1P_2):

$$X'_m = \frac{X'_{P1} \cdot Y'_{P2} - X'_{P2} \cdot Y'_{P1}}{Y_{P1}{}' - Y_{P2}{}'}$$

3.2.2 Princip

Algoritmus využívá paprsku (přímky) procházejícího analyzovaným bodem. Rozhodující hodnotou je počet průsečíků jedné poloosy tohoto paprsku. Např. v případě bodu ležícího uvnitř paprsku se po průchodu prvním průsečíkem změní stav paprsku (vzhledem k polygonu) na „vně“. Při průchodu 2. průsečíkem se stav změní na „uvnitř“ atd. Jestliže se jedná o nekonečnou přímku, musí konečný stav paprsku být „vně“. V případě bodu uvnitř polygonu tento stav nastane při lichém průchodu průsečíkem a v případě bodu vně polygonu v sudém.

3.2.3 Výpočet

1. Inicializace k = 0
2. Opakování pro všechny body p_i, které náleží polygonu
3. {
4. $x_i‘ = x_i - x_q$
5. $y_i‘ = y_i - y_q$
6. Když (($y_i‘ > 0$) a zároveň ($y_{i-1}‘ <= 0$)) nebo (($y_i‘ <= 0$) a zároveň ($y_{i-1}‘ > 0$))
7. $x_m‘ = (x_i‘ * y_{i-1}‘ - x_{i-1}‘ * y_i‘) / (y_i‘ - y_{i-1}‘)$
8. Když ($x_m‘ > 0$)
9. k = k + 1
10. }
11. Když (k je liché)
12. Bod leží uvnitř polygonu
13. Jinak
14. Bod neleží uvnitř polygonu

4. Problematické situace (+ řešení doplňkových úloh)

4.1 Analyzovaný bod leží na přímce dané hranou polygonu

V případě Winding Number bude hodnota vektorového součinu t (viz. vzorce) rovna 0. V takovém případě nelze dále pracovat se sumou úhlů, nýbrž rozhodnout, zda bod leží na hraně polygonu či nikoliv.

Ve zdrojovém kódu je tento případ řešen pomocí vzorce odvozeného z parametrického vyjádření přímky z krajních bodů hrany. Do tohoto vzorce jsou dosazeny souřadnice analyzovaného bodu a je vypočteno kolikanásobek je vektor P_1Q vektoru P_1P_2. Pohybuje-li se tato hodnota v intervalu <0 ; 1>, bod leží na hraně polygonu, v opačném případě leží vně polygonu.

1. „Násobek vektoru P1P2“ m = $(x_Q - x_{P1}) / (x_{P2} - x_{P1})$
2. Když (t se blíží nule a zároveň m náleží intervalu <0; 1>)
3. Vrať informaci, že se polygon má vykreslit
4. Když (t se blíží nule a zároveň m nenáleží intervalu <0; 1>)
5. Vrať informaci, že se polygon nemá vykreslit

V případě Ray Crossing byly ve zdrojovém kódu deklarovány proměnné pro počet průsečíků v pravé i levé polorovině paprsku. Paprsek z bodu, který leží na hraně, musí v polorovinách obsahovat kombinaci počtu průsečíků suchá/lichá nebo naopak.

1. Když (zb. poč. průs. v levé polorovině / 2 a zb. poč. průs. v pravé polorovině / 2 se nerovnají)
2. Vrať informaci, že se polygon má vykreslit

4.2 Analyzovaný bod je totožný s vrcholem jednoho či více polygonů

U Winding Number je tento případ ošetřen zahrnutím hodnot 0 a 1 do intervalu v předchozím odstavci. Takto algoritmus označí bod jako součást hrany i v případě že by analyzovaný bod byl totožný s jedním z krajních bodů hrany.

U Ray Crossing je tento případ ošetřen funkcí count, která porovná polohy vrcholů polygonu a analyzovaného bodu.

1. Když (count (body polygonu, analyzovaný bod))
2. Vykresli polygon
3. Jinak
4. Nevykresluj polygon

5. Vstupní data

Seznam polygonů v textovém souboru (polygons.txt), který obsahuje vždy id polygonu, počet bodů a souřadnice jednotlivých bodů.

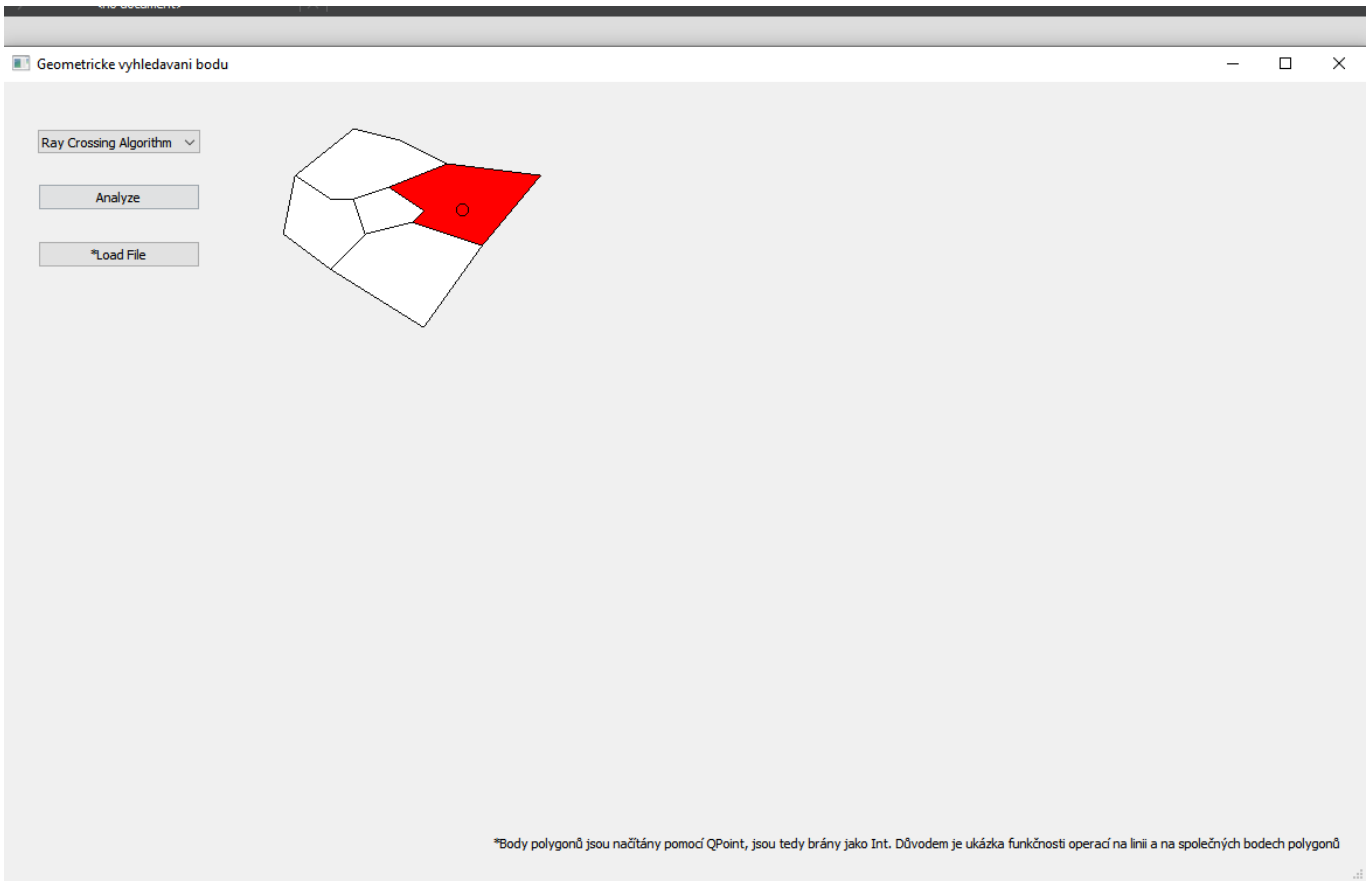
Formát vstupních dat:

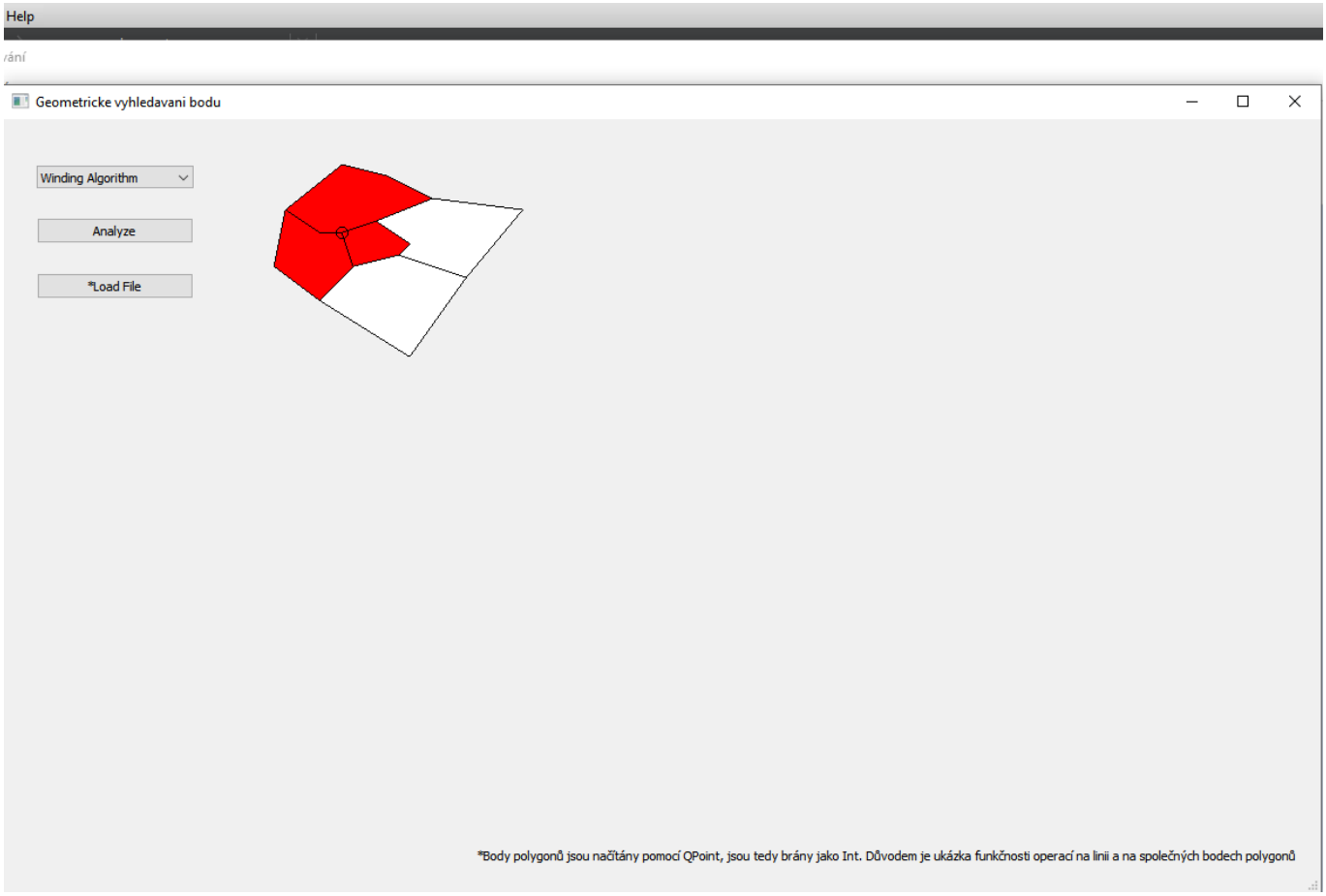
id_polygonu počet_bodů_v_polygonu
číslo_bodu souradnice_bodu_x souradnice_bodu_y
číslo_bodu souradnice_bodu_x souradnice_bodu_y
.
.
.
ko nec

6. Výstupní data

Aplikace neposkytuje žádná výstupní data. Výsledkem je označení polygonu, jehož je analyzovaný bod součástí přímo v rozhraní aplikace.

7. Vzhled aplikace





8. Dokumentace

Třídy:

Algorithms

Třída obsahuje 4 funkce, které se používají pro analýzu zadaných polygonů:

static int getPointLinePosition(QPoint &q, QPoint &p1, QPoint &p2);

Funkce zjišťuje, zda bod **q** leží vlevo či vpravo od vybrané úsečky spojující body p1 a p2. Také určuje, zda neleží přímo na úsečce. Funkce vrací hodnoty *int(1 pokud je bod vlevo, 0 pokud je bod vpravo, -1 pokud bod leží na úsečce, -2 pokud leží na přímce, ale neleží na úsečce).*

static double getAngle2Vectors(QPoint &p1, QPoint &p2, QPoint &p3, QPoint &p4);

Funkce vrací velikost úhlu mezi dvěma vektory, které jsou určeny body p1,p2 a p3,p4. Úhel je vracen v radiánech.

static int positionPointPolygonWinding(QPoint &q, vector<QPoint> &pol);

Funkce zjišťuje zda bod leží na hraně, uvnitř či vně polygonu. Používá přitom metodu Winding, která je popsána výše. Vrací tři možné hodnoty: *int(1 pokud bod leží v polygonu, 0 pokud bod leží vně polygonu, -1 pokud bod leží na hraně).* Vstupní hodnoty jsou: určený bod **q** a vektor bodů, které prozkoumávaný polygon obsahuje: **pol**.

static int positionPointPolygonRayCrossing(QPoint &q, vector<QPoint> &pol);

Funkce pracuje stejně jako v případě Winding metody, pouze pro zjišťování polohy bodu používá metodu Ray Crossing. Vstupní i výstupní hodnoty jsou totožné.

Draw

Třída obsahuje 4 privátní proměnné a 7 funkcí. Třída se používá pro načtení polygonů z textového souboru, k vykreslení polygonů a získání souřadnic určovaného bodu na obrazovce.

Proměnné v třídě draw:
Qpoint q;

Bod q, který je určen kliknutím na obrazovce. Počáteční hodnota nastavena na -100, -100, aby při spuštění aplikace nebyl vidět.

vector<QPolygon> polygons;

Vektor polygonů načtených z textového souboru.

vector<bool> color_result;

Vektor bool, který určuje, jaké polygony se mají zvýraznit barvou. True => vybarvit, False => nebarvit.

string path;

Obsahuje cestu k textovému souboru s polygony.

Funkce v třídě draw:
*void mousePressEvent(QMouseEvent *e);*

Funkce snímá pozici myši při kliknutí.

*void paintEvent(QPaintEvent *e);*

Funkce vykresluje polygony dle předdefinovaných stylů (vybarvené, nevybarvené). Také vykresluji bod **q**, který je určen kliknutím myši.

QPoint getPoint() {retunr q;}

Funkce vrací určený bod **q**.

vector<QPolygon> getPolygon() {return polygons;}

Funkce vrací polygon načtený z textového souboru.

void loadTxtFile(vector<QPolygon> &polygons, string &path);

Funkce načítá informace z textového souboru s polygony a třídí je. Výsledkem je vektor jednotlivých polygonů.

void setColorResult(vector<bool> colRes) {colorResult=colRes;}

Funkce vrací bool vektor, který určuje, které polygony se budou vybarvovat.

void setTxtPath(string path_x) {path=path_x;
loadTxtFile{polygons, path};}

Funkce otevírá okno pro vyhledávání v souborech, kde si určíme, který txt soubor s polygony budeme používat.

Widgets

Třída obsahuje 2 funkce, které rozhodují o následných procesech po stisknutí daného tlačítka.

void on_pushButton_analyze_clicked();

Funkce se spustí po stisknutí tlačítka „Analyze“. Zobrazí, zda se bod nachází v polygonu, mimo polygony, na hraně dvou polygonů či na společném bodě více polygonů.

void on_pushButton_loadTxtFile_clicked();

Funkce se spustí po stisknutí tlačítka „Load File“. Otevře okno pro vyhledávání v souborech, kde si určíme, který txt soubor s polygony budeme používat.

9. Závěr

Byla vytvořena aplikace podle zadání. Řešeny byly kromě úlohy automatického generování nekonvexních polygonů všechny ostatní doplňkové úlohy.

Kromě generování polygonů by bylo u metody Ray Crossing ještě možné vyřešit případ, kdy paprsek prochází hranou polygonu.

10. Přílohy

1) Zdrojový kód aplikace (algorithms.h / .cpp; draw.h / .cpp; main.cpp; u1_ADK_Nemec_Sartner.pro; widgets.h / .cpp / .ui)
2) Vstupní soubor s polygony (polygons.txt)

15. 10. 2019
David Němec, Jan Šartner

oprava TZ: 25.11.2019

2. oprava TZ: 1.12.2019