

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

název předmětu

ALGORITMY DIGITÁLNÍ KARTOGRAFIE A GIS

číslo
úlohy

1

název úlohy:

Geometrické vyhledávání bodu

školní rok:

2019/20

semestr:

zimní

zpracovali:

David Němec, Jan Šartner

datum:

15. 10.

klasifikace:

Technická zpráva

Geometrické vyhledávání bodu

Zadání:

Cílem úlohy je vytvořit aplikaci v sw. Qt Creator využívající programovacího jazyku C++, která načte polygonovou síť a umožní uživateli analyzovat, zda jím zvolený bod leží uvnitř, vně nebo na hranici polygonu. K tomuto účelu bude možné využít algoritmy Winding Number a Ray Crossing. Polygon, ve kterém se analyzovaný bod nachází bude zvýrazněn.

Doplňující úlohy:

- 1) Ošetření případu, kdy bod leží na hranici dvou polygonů u algoritmu Winding Number
-řešeno
- 2) Ošetření případu, kdy je bod totožný s vrcholem jednoho nebo více polygonů u obou algoritmů
-řešeno
- 3) Zvýraznění všech dotčených polygonů pro případy 1) a 2)
-řešeno
- 4) Možnost automatického vygenerování nekonvexních polygonů
-neřešeno

Problematika vyhodnocování polohy bodu a polygonu - vzorce:

Algoritmus Winding Number

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

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

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

$$\omega = \left| \arccos \left(\frac{u \cdot v}{|u| \cdot |v|} \right) \right|$$

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.

Výpočet Winding Number:

$$WN = \text{sum}(\omega)$$

-o znaménku hodnoty ω rozhoduje znaménko hodnoty t

-na základě výsledné hodnoty WN lze určit, zda analyzovaný bod leží uvnitř/vně polygonu

$$WN = \pm 2\pi \rightarrow \text{uvnitř polygonu}$$

$$WN = 0 \rightarrow \text{vně polygonu}$$

Algoritmus Ray Crossing

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):

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

Počet průsečíků:

$$k = \text{počet průsečíků } X'_m \text{ s jednou poloosou paprsku } y' = 0$$

$$\text{liché } k \rightarrow \text{uvnitř polygonu}$$

$$\text{liché } k \rightarrow \text{vně polygonu}$$

Problematika vyhodnocování polohy bodu a polygonu - popis:

Algoritmus Winding Number

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.

Algoritmus Ray Crossing

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.

Problematické situace (+ řešení doplňkových úloh):

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 porovnání souřadnic krajních bodů hrany a analyzovaného bodu.

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.

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.

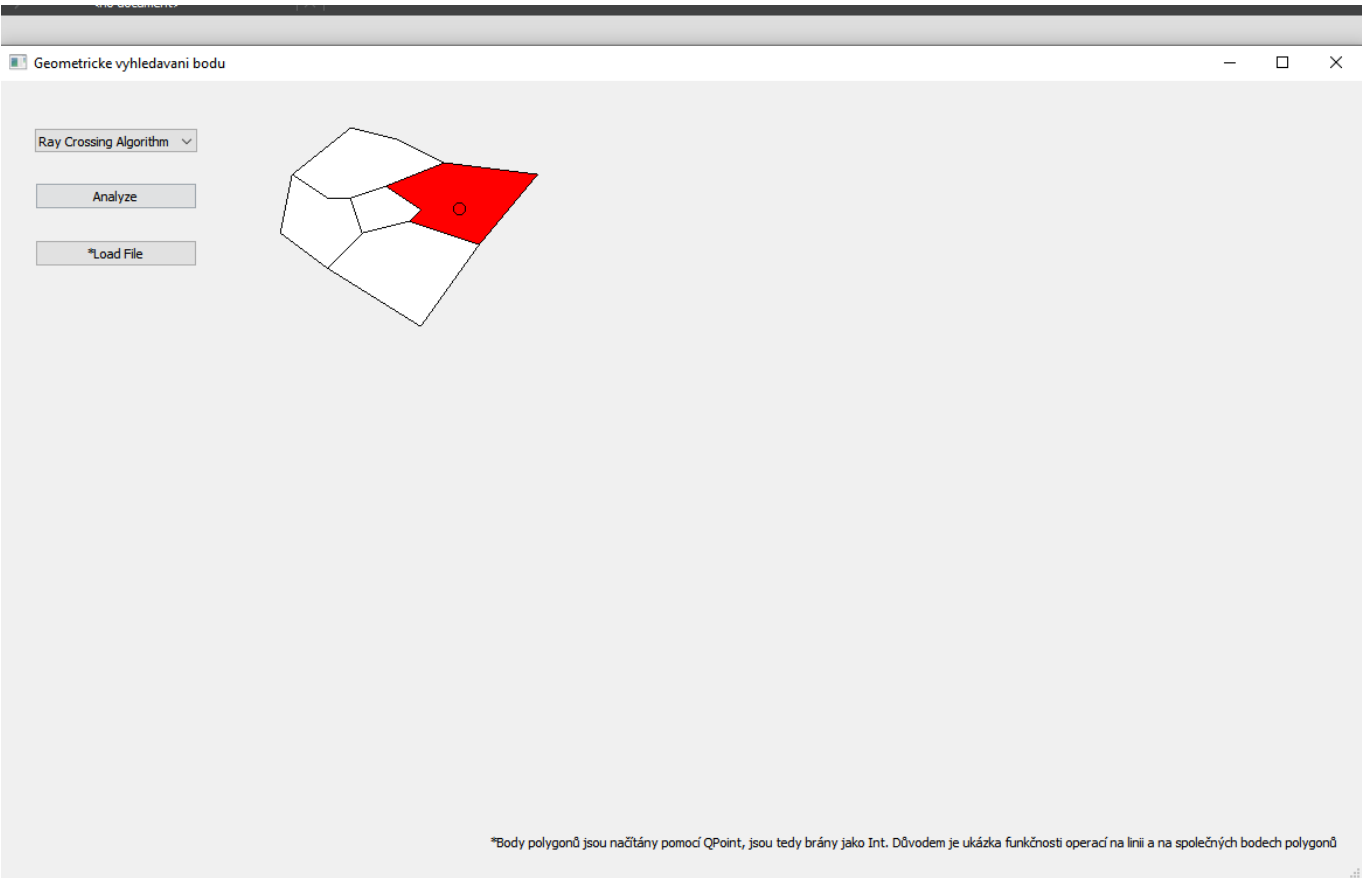
Vstupní data:

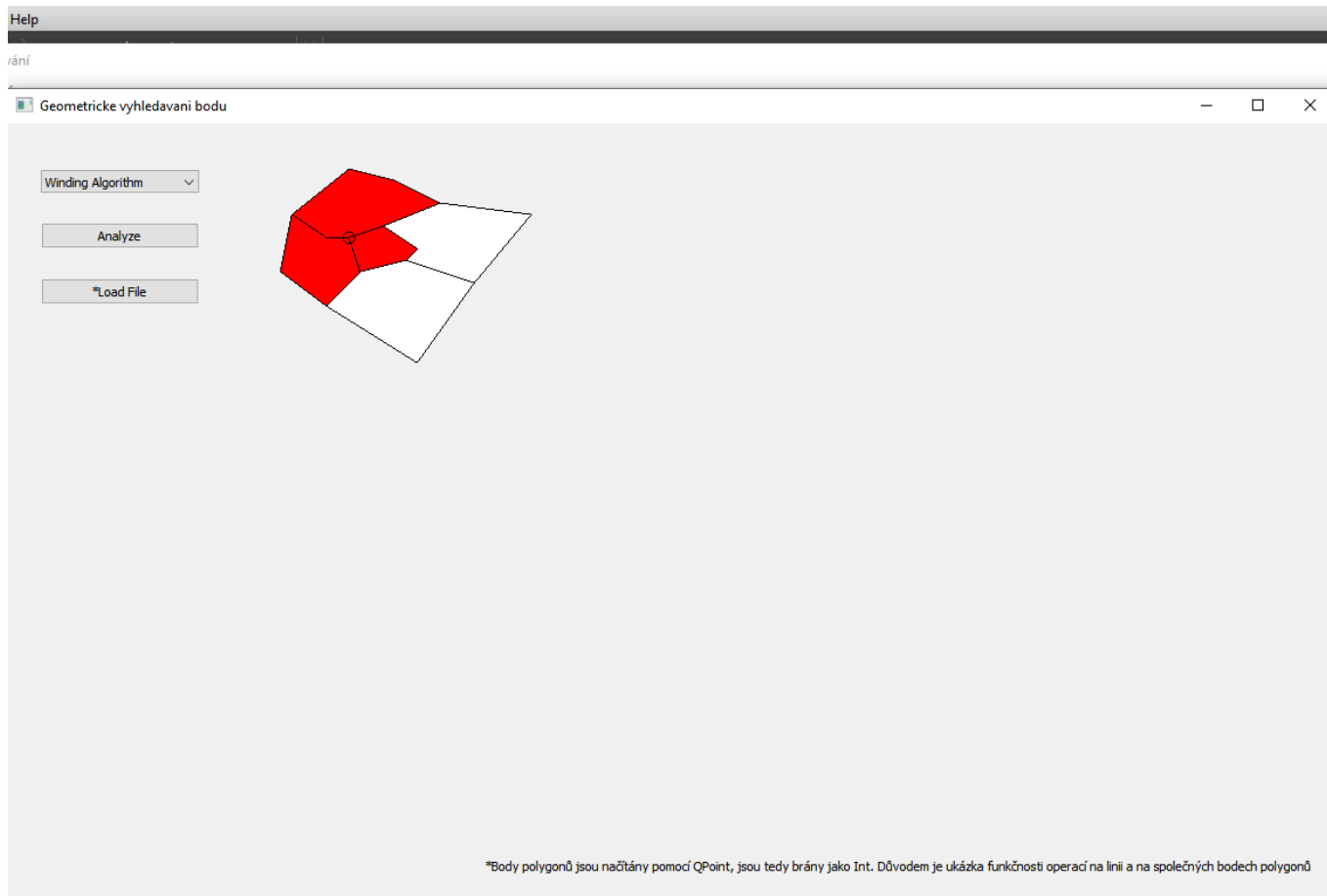
Seznam polygonů v textovém souboru (`polygons.txt`), který obsahuje vždy id polygonu, počet bodů a souřadnice jednotlivých bodů.

Výstupní data:

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

Vzhled aplikace:





Dokumentace:

Třídy:

algoritmus

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 přímce)*.

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 vrácen 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> colorResult;

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 vykresluje bod **q**, který je určen kliknutím myši.

QPoint getPoint() {return 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.

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.

Přílohy:

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