ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

FAKULTA STAVEBNÍ

KATEDRA GEOMATIKY

název předmětu

ALGORITMY V DIGITÁLNÍ KARTOGRAFII

číslo úlohy	název úlohy				
1	Geometrické vyhledávání bodu				
školní rok		číslo zadání	zpracoval	datum	klasifikace
2019/2020		1	R. Pflug, T. Klemsa	15.10.2019	

Technická zpráva

Zadání úlohy

Vstup: Souvislá polygonová mapa n polygonů {*P1, ..., Pn*}, 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 gracky zvýrazněte vhodným způsobem (např. vyplněním, šrafováním, blikáním). Grackérozhraní vytvořte s využitím frameworku QT.

Pro generování nekonvexních polygonů můžete navrhnout vlastní algoritmus či použít existující geograckádata (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 a 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***

Pracovní postup

Základním zadáním úlohy bylo určit vzájemnou polohu bodu a polygonů. Problém byl řešen dvěma způsoby: algoritmem využívajícího *Winding Number* a algoritmem nazývaným *Ray Crossing*. Řešení problematiky je implementováno s využitím Widgedts aplikace napsané v jazyce C++. Aplikace byla tvořena na platformě Qt Creator v operačním systému Windows.

Prvním krokem před samotnou tvorbou aplikace byla tvorba kostry programu. Bylo nutné oddělit funkce provádějící přípravu dat a samotný výpočet algoritmů od funkcí zajišťujících chod grafického rozhraní, vstupu apod.

Pro aplikaci bylo nejprve navrženo základní grafické okno, které bylo postupně doplňováno o potřebné interaktivní prvky (především push buttony apod.). Pro zpracování grafické funkcionality

vstupu a výstupu dat byla vytvořena třída Draw. Pro funkce zpracovávající oba algoritmy byla vytvořena třída Algorithms. Načítání externích dat (polygonů) je prováděno ve třídě filereader.

Konkrétní řešení a vzorce použité při řešení problematiky jsou obsahem následujících kapitol.

Obecná formulace a řešení problému:
Ve 2D souřadnicích je dána množina *n* vrcholů, *m* polygonů a bod *q*.
Cílem řešení problému je určit takový polygon (mnohoúhelník), který bod *q* obsahuje. V případě řešení bonusových úloh je cílem nalézt takové polygony, jejichž hrana či vrchol obsahují bod *q*. Řešení je určeno pro nekonvexní polygony. Princip testu polohy bodu *q* vůči polygonu je za pomoci lokální procedury (tj. opakované určení polohy bodu *q* vzhledem k polygonu).

Winding Algorithm

Algoritmus vychází z výpočtu úhlů mezi jednotlivými průvodiči z analyzovaného bodu *q* do jednotlivých vrcholů p_i příslušného polygonu *P*. Postupným přičítáním a odčítáním úhlů (podle směru z předchozího p_i na následující vrchol p_{i+1} polygonu P) mezi průvodiči je získávána informace kde se bod *q* vůči polygonu *P* nachází. Zda se úhel mezi průvodiči do výsledné sumy úhlů přičte či odečte udává polorovina, ve které se úhel vůči spojnici z bodu *q* do vrcholu polygonu p_i nachází. Úhly nacházející se v pravé, takto definované polorovině, jsou orientovány kladně a proto se přičítají. Úhly v levé polorovině se odčítají. Na základě výsledné sumy všech úhlů pro daný polygon *P* mohu určit polohu bodu *q*. Je-li suma úhlů násobkem *2π* znamená to, že bod *q se nachází* uvnitř polygonu *P*, je-li úhel menší (důsledek odčítání záporně orientovaných úhlů), bod *q* leží vně polygonu *P*.
Winding number je udáváno jako násobek *2π* (v radiánové míře). Výsledná hodnota winding number závisí na orientaci směru pohybu (CW x CCW).

Definice: *Winding number [Ω] je suma všech rotací [ω_i], které musí průvodič [q, p_i] opsat nad všemi body [p_i] polygonu [P].* [1]

$$\Omega(q,P)=\frac{1}{2\pi}\Sigma_{i=1}^{n}\omega(q,e_i) \qquad e_i=(p_i,p_i+1)$$
$$\omega(q,e_i): p_{i+1}\, vpravo\, od(q,p_i) \qquad -\omega(q,e_i): p_{i+1}\, vlevo\, od(q,p_i)$$
$$\Omega(q,P)=1, q\epsilon P \qquad \Omega(q,P)=0, q\notin P$$

Ray crossing algorithm

Základem Ray crossing algoritmu je přímka *r* procházející analyzovaným bodem *q*. Poloha bodu *q* vůči polygonu *P* je určována na základě počtu průsečíků přímky *k* a hran polygonu p_i. Obecně platí pro bod *q* vně polygonu *P*, že počet průsečíků *k* je sudý, lichý počet *k* platí pro bod *q* uvnitř polygonu *P*. Problém singularit pro takto daný model algoritmu je řešen převedením na upravený model. V upraveném modelu algoritmu je pro určení výsledku uvažována pouze jedna polorovina vzhledem k paprsku *r* procházejícím bodem *q*. Další modifikace redukuje souřadnice vrcholů p_i polygonu *P* k analyzovanému bodu *q*. Pro model s takto vytvořenou lokální souřadnicovou soustavou jsou hledány pouze průsečíky *M* ležící v pravé polorovině od osy *x*, která po redukci prochází bodem *q*. Výsledná varianta algoritmu tedy uvažuje pouze průsečíky *M* ležící v jednou kvadrantu (v popisovaném případě se jedná o první kvadrant) lokální souřadnicové soustavy

vzhledem k bodu *q*. Takto upravený algoritmus řeší i případné singularity (např. Identickou hranu p_i polygonu *P* s polopřímkou *r* z bodu *q* apod.).

Definice: *Bodem q je veden paprsek r, nechť k představuje počet průsečíků přímky r s hranami P. Pokud je k beze zbytku dělitelné dvěma, pak bod q neleží v polygonu P. Pokud k není dělitelné beze zbytku dvěma, bod q leží v polygonu q.*[1]

Pro upravenou variantu platí: *Průsečík M hrany e = (p_{i-1}, p_i) a r(q) započítáváme pokud:* $(y_i < y_q) \wedge (y_{i-1} \geq y_q) \vee (y_{i-1} < y_q) \wedge (y_i \geq y_q)$ [1]
platí pro horní polorovinu vzhledem k ose y ($y > y_q$).

Redukce souřadnic k bodu *q*: $x_i' = x_i - x_q$; $y_i' = y_i - y_q$

Dalším krokem je test existence průsečíku hrany polygonu a osy x′. Průsečík M, se souřadnicemi $y_m = 0$ *a* $x_m > 0$ (první kvadrant): $x_m' = \frac{x_i' y_{i-1}' - x_{i-1}' y_i'}{y_i' - y_{i-1}'}$

Test existence průsečíku *M*: $(y_i < 0) \wedge (y_{i-1} \geq 0) \vee (y_{i-1} < 0) \wedge (y_i \geq 0)$; *($y_q = 0$)*

Počet průsečíků *k*: $k(q,P) = k(q,P) + 1; x_m' > 0$ $\quad k(q,P) = k(q,P) + 0; x_m' \leq 0$

Simplexy

Problémy u řešení polohy bodu vůči polygonu nastávají obecně u poloh mezně se blížících k hraně či vrcholu polygonu. Tento problém bylo třeba ošetřit u obou použitých algoritmů.
Winding number algorithm: Singulární případ pro bod na hraně polygonu zde byl ošetřen ve funkci „*getPointLinePosition*“, která vrací hodnotu 1 pro bod v levé polorovině vůči hraně, hodnotu 0 pro bod v pravé polorovině vůči hraně a hodnotu -1 pro bod na hraně polygonu. Z důvodu nepřesností výpočtu hardwaru byla definována mezní tolerance, která zde nahrazuje hodnotu 0 pro testovací kritéria. Při nahrazení hodnotou tolerance v podmínce určující polohu bodu vůči přímce, funkce vyhodnotí body nacházející se pod touto hranicí jako body hraniční (bod na hraně). Návratová hodnota funkce se pak rovná -1.

Ray crossing algorithm: Singulární případ, kdy analyzovaný bod leží krajně blízko hraně polygonu byl v algoritmu ošetřen přímo v těle funkce „*positionPolygonRayCrossing*“. Byla zde opět definována tolerance, která určuje od jaké hodnoty již spadá bod na hranu polygonu. Ošetření singulárního případu pak bylo provedeno u porovnání hodnoty souřadnice *x* bodu průsečíku *M*. Pokud se souřadnice *x* bodu *q* krajně blížila hodnotě *x* průsečíku *M*, pak je bod *q* na hranici polygonu.

Vstupní data

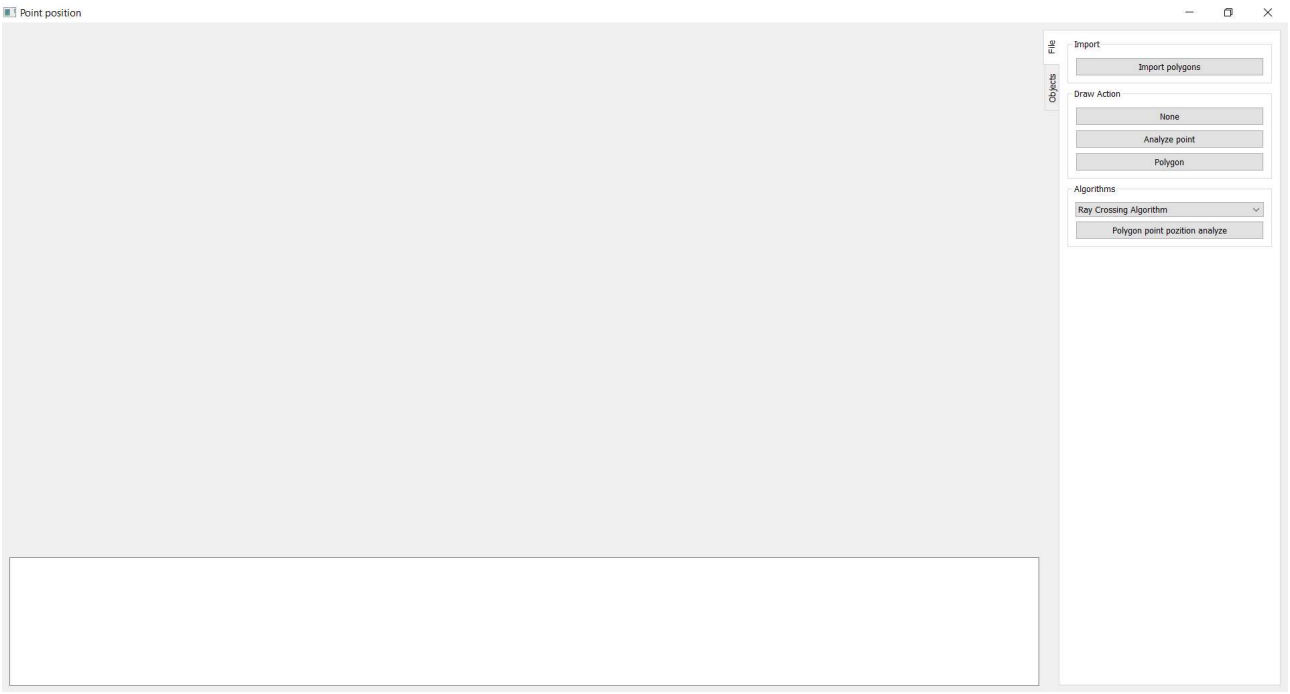
Vstup dat lze provádět dvojím způsobem:
Grafickým vstupem, kdy jsou snímány souřadnice bodů kurzorem myši v grafickém rozhraní aplikace.

Kombinovaným vsupem, kde jsou souřadnice polygonů načítány z textového souboru v předepsaném formátu a souřadnice analyzovaného bodu jsou snímány kurzorem myši v grafickém rozhraní aplikace.

Grafický vstup: Grafický vstup vrcholů polygonu je aktivován tlačítkem *Polygon* v sekci *Draw Action*. Pro ukončení snímání aktuálního polygonu je třeba aktivovat tlačítko *None*. Poté lze opět po aktivací tlačítka *Polygon* snímat a vykreslovat nový polygon nebo aktivací tlačítka *Analyze point* snímat souřadnice analyzovaného bodu.
Kombinovaný vstup: Pro načtení polygonů z textového souboru je určena sekce *Import* s tlačítkem *Import polygons*. Po aktivaci tlačítka lze v dialogovém okně vybrat příslušný soubor a nahrát polygony hromadně do aplikace.

Formát souboru pro import dat: Každý polygon začíná číslem vrcholu od čísla 1 po odsazením tabulátorem následuje souřadnice x, po dalším odsazením souřadnice y. Jednotlivé vrcholy jsou odřádkovány. Nový polygon začíná opět číslem vrcholu 1.
Příklad:

```
//Č.V.    X       Y
   1      100     500     // 1. vrchol prvního polygonu
   2      120     550     // 2. vrchol prvního polygonu
   .
   .
   .
   N      XXX     YYY     // N. Vrchol prvního polygonu
   1      120     550     // 1. vrchol druhého polygonu
```



Výstupní data

Výstupní data jsou prezentována grafickou cestou. Po zadání souřadnic polygonů a analyzovaného bodu jsou vstupní data vykreslena v grafickém okně. Po aktivaci tlačítka *Polygon point pozition analyze* je graficky zvýrazněn červenou barvou polygon, ve kterém se bod nachází. Nachází-li se bod na hraně či vrcholu polygonu, je polygon zvýrazněn zelenou barvou.

Dokumentace

Algorithms: Třída obsahující metody potřebné pro výpočet polohy bodu vzhledem k polygonu.

Metody: getPointLinePosition – Určení polohy bodu vůči přímce. Návratová hodnota *1* pro bod v levé polorovině, *0* pro bod v pravé polorovině, -1 pro bod na přímce.*(intrger)*

getAngle2Vectors – Vrací úhel mezi dvěma vektory.*(double)*

positionPointPolygonWinding – Obsahuje winding number algoritmus pro výpočet polohy bodu vůči polygonu. Návratová hodnota 1 pro body v polygonu, 0 pro bod mimo polygon a -1 pro bod na hraně polygonu.*(integer)*

positionPointRayCrossing – Obsahuje ray crossing algoritmus pro výpočet polohy bodu vůči polygonu. Návratová hodnota 1 pro body v polygonu, 0 pro bod mimo polygon a -1 pro bod na hraně polygonu.*(integer)*

createRandomPolygon – Metoda generující náhodný polygon o 4 až 20 vrcholech.(vector<QPoints>)

Draw: Třída s metodami zajišťující snímání a ukládání dat grafického vstupu. Metody také zajišťují grafické vykreslení dat. Třída také umožňuje přepínání metod grafického vstupu

FileReader: Třída obsahující funkce zajišťující import polygonů v podobě textového souboru.

Widget: Třída vytvářející grafické rozhraní aplikace.

Závěr

Pro předpřipravená data aplikace funguje bez problému, problematické situace algoritmu podle zadání byly ošetřeny a jsou blíže rozebrány v kapitole **Simplexy.** Aplikacke by byla o mnoho všestranější, pokud by při importu neupravených polygonů z textového souboru, automaticky redukovala souřadnice vrcholů tak, aby je bylo možné zobrazit v grafickém okně. Tento problém aplikace neřeší a proto je nutné před importem souřadnice upravit jiným způsobem.

Seznam literatury:

[1] Tomáš Bayer: Geometrické vyhledávání bodu, web.natur.cuni.cz/~bayertom/images/courses/Adk/adk3.pdf

Robin Pflug, Tomáš Klemsa V Praze 15.10.2019