PŘÍRODOVĚDECKÁ
FAKULTA
Univerzita Karlova

Algoritmy v počítačové kartografii

Úloha č. 1
Geometrické vyhledávání bodu

Čelonk Marek, Sýkora Matúš

1. Zadání

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

Výstup: P_i, $q \in P_i$.

Nad polygonovou mapou implementujete 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 geografické data (např. mapa evropských států).

Polygony budou načítány z textového souboru ve vámi zvoleném formátu. Pro datovou reprezentaci jednotlivých polygonů použijte špagetový model.

2. Popis a rozbor problému + vzorce

V úloze je řešen vztah polohy bodu vůči polygonu. Pokud je porovnáván jeden bod vůči jednomu polygonu může docházet ke čtyřem různým výsledkům:

- bod se nachází vně polygonu
- bod se nachází uvnitř polygonu
- bod se nachází na hraně polygonu
- bod je totožný s vrcholem polygonu

V prostředí GIS se jedná o základní dotazovací úlohu, při které se zjišťuje, zda vybraný bod splňuje podmínku, kdy náleží polygonu. S tímto problémem se lze setkat už při samotném výběru polygonů pomocí kurzoru, kdy k výběru dojde ve chvíli, když je kurzor uvnitř polygonu.

Důležitými aspekty algoritmů by měly být rychlé zpracování i pro dataset obsahující velké množství polygonů a funkčnost pro konvexní i nekonvexní polygony.

Nastíněný problém lze řešit s využitím různých algoritmů. Jedná se například o výše zadané algoritmy *Ray Crossing Algorithm* nebo *Winding Algorithm*. Další možnost nabízí metoda pásu (*Slabs method*) nebo metoda trapezoidních map.

3. Popisy algoritmů formálním jazykem

V úloze jsou využity dva méně složité algoritmy *Ray Crossing Algorithm* a *Winding Algorithm*. Pro oba algoritmy je nutné si předem vypočítat některé hodnoty pomocí dalších funkcí. První je výpočet součinu bodů, další je výpočet délky vektorů a poslední je výpočet úhlu dvou vektorů.

Jako návratové hodnoty funkcí jsou použity výčtové typy *enum* (*INSIDE, OUTSIDE, BOUNDARY, POINT*).

3.1. Ray Crossing Algorithm

Základní princip lze vyčíst i z názvu algoritmu. Hlavní myšlenkou je proložení daného bodu *q* přímkou *r* v horizontálním směru a vypočítat počet průsečíků s hranami polygonu. Lichý součet průsečíků platí pro bod nacházející se v polygonu, naopak sudý součet značí bod mimo polygon. Problém nastává v případě singularit, například pokud přímka prochází kolineární hranou nebo vrcholem polygonu. Algoritmus lze upravit, tak aby vyřešil problém i v případě singularit. Počátek os *x* a *y* se přesune do bodu *q* a dojde k vytvoření lokálního souřadnicového systému. Poté algoritmus počítá pouze v pravé polorovině.

Kroky výpočtu:

- výpočet redukovaných souřadnic
- test zda jsou souřadnice bodu totožné se souřadnicemi vrcholu polygonu
- redukované souřadnice následujícího vrcholu polygonu
- test průsečíku hrany a paprsku, počítané pouze pro pravou polorovinu, v případě splnění podmínek dochází k inkrementaci počtu průsečíků
- zjištění návratových hodnot (lichý počet průsečíků značí lokalizaci bodu v polygonu)

3.2. Winding Algorithm

Tento algoritmus vychází z jednoduché myšlenky, že pokud se nachází bot uvnitř polygonu, je možné projít všechny vrcholy polygonu a součet všech úhlů bude 360°. Pokud by bod byl mimo tento polygon výsledný úhle by byl menší. Výpočty úhlů k jednotlivým vrcholům se provádí postupně v jednom směru (CCL nebo CL). Tato podmínka je hlavně důležitá u nekonvexních

polygonů, kde dochází k výpočtům proti zvolenému směru. V tomto případě se úhel od celkového výsledku odečítá.

Kroky výpočtu:

- výběr tolerance
- test zda jsou souřadnice bodu totožné se souřadnicemi vrcholu polygonu
- výpočet úhlu a upravení *winding number*
- test, zda leží bod na hraně polygonu
- test, zda leží bod uvnitř polygonu (*winding number* je roven 2π, respektive rozdíl konečného *winding number* a 2π je menší než tolerance)

4. Problematické situace a jejich rozbor

Bylo důležité, aby alespoň jeden algoritmus zvládal zjistit, zda bod leží na hraně. Problém byl vyřešen u *winding algoritmu*, který je popsaný výše. Bod ležící v polygonu má *winding number* rovné 2π. Pokud se však nachází na hraně, nemůže dojít k součtu všech polygonů, ale pouze k půlce, tedy π. V kódu je tato možnost ošetřena podmínkou, kdy alespoň jeden vypočtený úhel je π.

Další problém nastal při správném vykreslování výsledků. Bylo nutné, aby každý polygon měl i informaci o prostorovém vztahu se zkoumaným bodem. Toho bylo docíleno tak, že byl použit list výsledných výběrových hodnot, tak aby jejich indexy pozice odpovídaly indexům polygonů v poli.

5. Vstupní data, formát vstupních dat, popis

Pro funkčnost aplikace je nutné mít minimálně dva vstupní prvky. Alespoň jeden polygon a bod, kterému bude testován prostorový vztah k polygonu. Bod je implementován po kliknutí do kreslícího panelu, na daném místě se vytvoří křížek pro lepší vizualizaci bodu. Pro práci s body je využívána knihovna *awt.Point*. Java nabízí také knihovnu pro práci s polygony, jedná se o *awt.Polygon*. Polygon se skládá z pole souřadnic *x* a z pole souřadnic *y.*

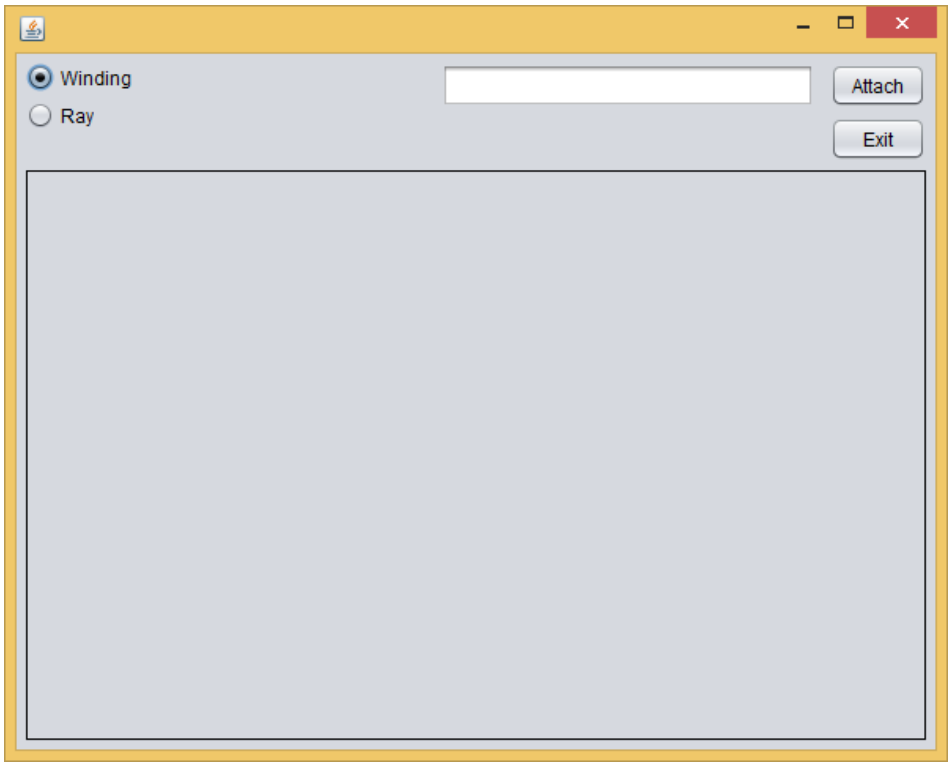
Do vykreslovacího okna jsou polygony nahrávány z textového souboru pomocí funkce *scanner*. Je důležité, aby bylo dodržené přesné zapsání polygonů v textovém souboru. Na prvním řádku je zaznamenán celkový počet polygonů v souboru. Další tři řádky slouží k vytvoření konkrétního polygonu. Nejdříve se zaznamená na jeden řádek počet vrcholů, na dalším řádku jsou zapsány souřadnice *x* a pod nimi jsou na dalším zapsány hodnoty *y* souřadnic. Další tři řádky tvoří nový polygon a opět je rozdělení informací do tří řádků totožné jako u prvního polygonu. Jako oddělovací znak mezi souřadnicemi se používá mezera. Správné zapsání vstupních polygonů v textovém souboru lze vidět v dokumentu *test.txt*.

6. Výstupní data, formát výstupních dat, popis

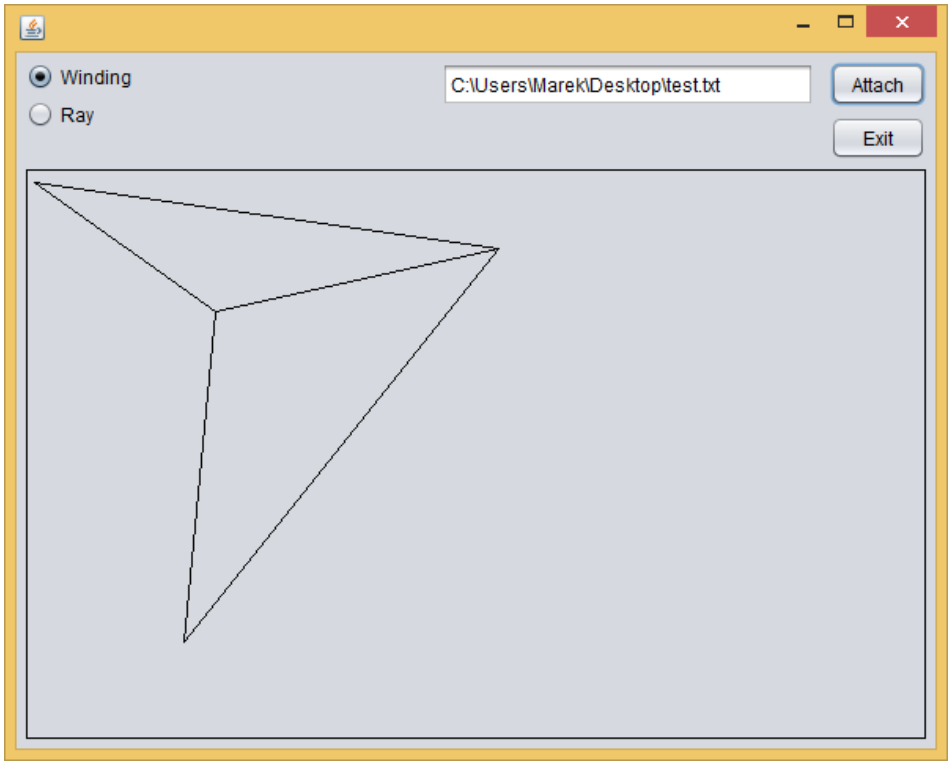
Aplikace nevytváří nový výstupový soubor. Lze říci, že hlavním výstupem je výsledný výčtový typ. V našem případě se jedná o výsledek *INSIDE, OUTSIDE, BOUNDARY, POINT*. Tyto hodnoty ovlivňují vzhled polygonů ve vykreslovacím okně. Pokud se bod nachází uvnitř polygonu, je tento polygon vybarven žlutě. V případě lokalizace bodu na hraně polygonu, jsou všechny hrany polygonu vykresleny zelenou barvou. Pokud hranu sdílí dva polygony, tak je barva změněna u obou (*winding algoritmus)*. Při situaci, kdy je bod totožný s vrcholem, dochází k přebarvení výběrového kříže z černého na červený. Pokud zkoumaný bod leží mimo jakýkoli polygon, tak nedochází k žádné změně ve vykreslovacím okně.

7. Printscreen vytvořené aplikace

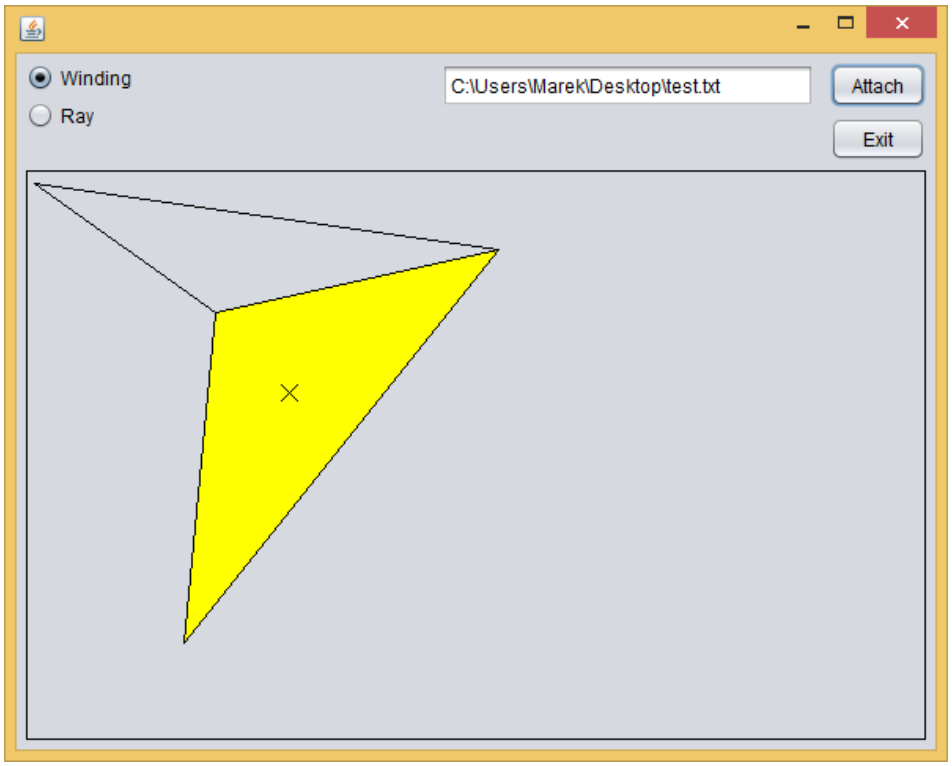
Na obrázku č. 1 je zobrazena aplikace po spuštění. Pomocí tlačítka *Attach*, lze načíst vybraný textový soubor s polygony, které se zobrazí ve vykreslovacím okně (viz obr. č. 2). Poté lze pomocí kurzoru testovat lokalizaci bodu. Možné výsledky jsou zobrazeny na obrázcích 3 až 5.



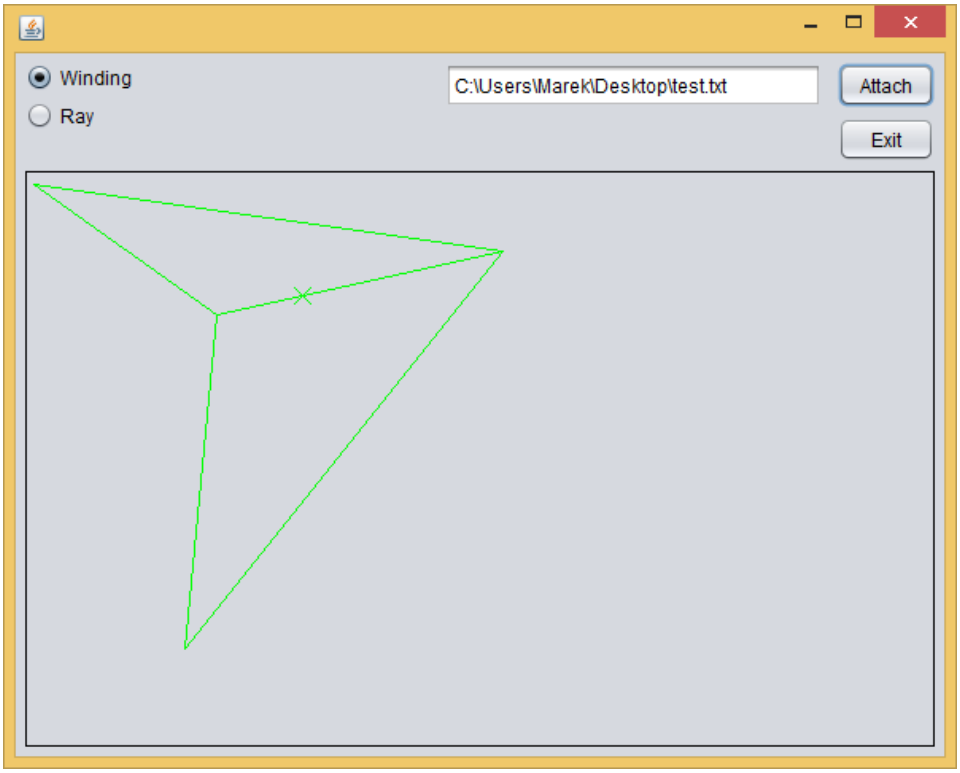
obr. č. 1: Aplikace po otevření



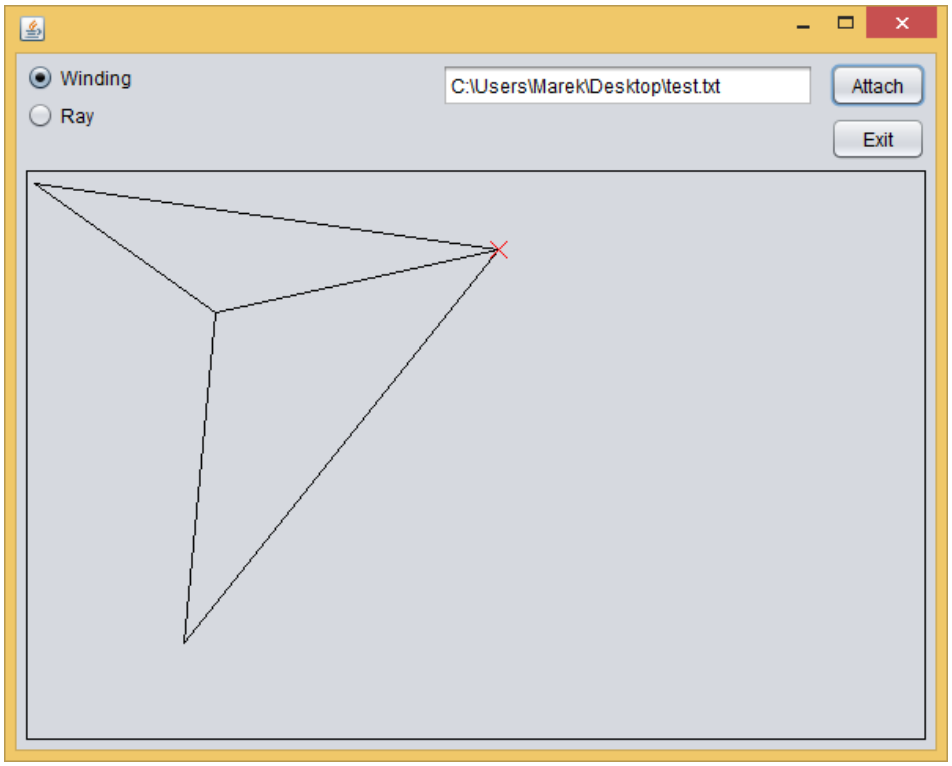
obr. č. 2: Načtení polygonů z textového souboru



obr. č. 3: Vizualizace pokud se bod nachází v polygonu



obr. č. 4: Vizualizace pokud se bod nachází na hraně jednoho, či dvou polygonů



obr. č. 5: Vizualizace je bod totožný s vrcholem polygonu

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

V aplikaci se používají celkem 4 třídy. Třída *Algorithm* obsahuje veškeré výpočty a algoritmy pro vyhledávání bodu. V této třídě se používá již výše vícekrát zmíněná proměnná *enum* označená jako *PositionEnum*. Třída obsahuje několik krátkých výpočtových funkcí. Mezi tyto funkce patří *dotProd*, které počítá skalární součin. Dále funkce *len*, která počítá velikost vektorů. Třetí pomocná funkce *angle* počítá úhel mezi vektory. Dvě hlavní funkce v této třídě jsou *pointPolygonWinding* a *pointPolygonRay*. V obou probíhá geometrické vyhledávání bodu dle stejnojmenného algoritmu. Návratovou hodnotou je vždy výběr z proměnné *enum*.

Druhou třídou je *DrawPanel*. Tato třída zajišťuje funkčnost vykreslovacího okna. Obsahuje několik proměnných. Mezi ně patří například *point* datového typu *Point*, reprezentující bod vzniknutý po kliknutí do vykreslovacího okna. Dále pole zkoumaných polygonů datového typu *Polygon*. Proměnná *res* datového typu *enum* a dále *List* těchto výsledků. Jednou ze dvou metod v této třídě je *formMouseClicked*. Tato metoda po kliknutí zjistí *x* a *y* souřadnice kurzoru a dle zvolené metody vypočítá prostorový vztah k polygonům. Výstupem je naplněný *List* výsledků datového typu *enum*, který je dále použit pro vykreslování polygonů v druhé metodě *paintComponent*. Tato metoda při prvním volání vykreslí nahrané polygony. Avšak po kliknutí a využití metody *formMouseClicked* slouží také ke správné prezentaci výsledků (viz 6. Výstupní data, formát výstupních dat, popis).

Třetí třídou je *JFrame* v našem případě představuje GUI aplikace. V této třídě jsou zapsány metody jednotlivých komponentů, s kterými pracuje uživatel. Důležitou metodou je *attachActionPerformed*, které je spojená s tlačítkem A*ttach*. Po stisku tohoto tlačítka a tudíž i aktivování metody dochází k několika akcím. Nejdříve se spustí výběrové okno, kde uživatel vybere textový soubor s daty o polygonech. Poté se spustí metoda, které přečte celý dokument a vytvoří z něj pole polygonů. Nakonec dojde k vykreslení načtených polygonů ve vykreslovacím okně. Metody *WindingActionPerformed* a *RayActionPerformed* zastupují výběrová tlačítka, která dovolují uživateli zvolit mezi možnými algoritmy.

Poslední třída *reader_array* zastupuje jednu metodu *retArray*. Tato metoda se volá v třídě *JFrame* a slouží k převedení vstupních informací z textového souboru do pole datového typu Polygon. Pro čtení textového souboru je využita metoda *Scanner*, která dovoluje číst soubor po řádcích.

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

V aplikaci lze otestovat funkčnost obou vybraných algoritmů pro vlastní polygony. Možný problém k vyřešení je u *Rey* algoritmu zjištění výsledku pokud se bod nachází na hraně polygonu. Aplikaci lze vylepšit možností generování náhodných testovacích polygonů. Dalším vylepšením by bylo nahrávání polygonů z různých typů vstupních souborů, převážně by bylo užitečné využívat *shapefile* soubory.

10. Seznam literatury

BAYER, T. 2018. Geometrické vyhledávání: Ray algoritmus. Winding algoritmus. Lichoběžníkové (trapezoidální) mapy [online]. Praha. Dostupné z:

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