155ADKG - Algoritmy digitální kartografie a GIS: Konvexní obálka

Dne 17. prosince 2016

Ing. Tomáš Bayer, Ph.D.

Bc. Petr Bezděka, Bc. Šimon Gajzler, Bc. Lukáš Středa

Obsah

Zadání

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.

Rozbor problematiky

Problematika určování vzájemné polohy vektorových prvků je v GIS systémech jedna ze základních úloh. V přednáškách a cvičeních jsme se seznámili s algoritmy využívanými pro určování polohy bodového prvku vůči polygonovému. V následující části si tyto algoritmy popíšeme.

Paprskový algoritmus

Paprskový algoritmus (Ray Crossing Algorithm) je metoda založená na principu, kdy z určovaného bodu q vedeme polopřímku r. Následně je sledovám počet průsečíků k s polygonem. V případě, že bod leží uvnitř polygonu P nabývá k lichých hodnot, v případě bodu ležícího vně polygonu nabývá k sudých hodnot. U konvexních polygonů se pak $k = 2$ právě v případě bodu vně a $k = 1$ v případě bodu uvnitř polygonu. U singulárních případů je třeba ošetřit situace, kdy bod leží na hraně polygonu (počet průsečíků je roven dvěma) nebo v jeho vrcholu (počet průsečíků je roven jedné nebo dvěma).

Na obrázku **??** jsou znázorněny dva příklady bodů uvnitř a vně polygonu. Hodnoty *cn* vyjadřují počet průsečíků vedené polopřímky s hranami polygonu.

Metoda ovíjení

Metoda ovíjení (Winding Algorithm) je založena na principu sčítání úhlů na určovaném bodě utvořených z vrcholů polygonů. Sčítání úhlů se řídí pravidlem, kdy se v orientovaném polygonu vždy přičte či odečte úhel mezi vrcholy jedné hrany, podle toho zda je hrana orientována z pohledu určovaného bodu doprava nebo doleva. Součty úhlů mohou nabývat hodnot $\pm 2\pi$ pro bod uvnitř polygonu, nula pro bod ležící vně polygonu a jichých hodnot pro bod na hraně či vrcholu.

Problematické situace

Paprskový algortimus

V paprskovém algoritmu bylo třeba ošetřit singulární případy. Jedná se o situaci, kdy určovaný bod je na hraně či vrcholu polygonu. Nejprve jsme ošetřili situaci, kdy bod leží na hraně, která není rovnoběžná s

osou x. To bylo vyřešeno ukončením algoritmu s návratovou hodnotou -1, pokud proměnná, která určuje jestli hranu protíná pravá nebo levá polopřímka, navrací hodnotu blízkou nule. Kvůli případu, kdy testovaná hrana je rovnoběžná s osou x a tedy i s testovací přímkou, bylo nutno tento test provádět i pro směr v ose y.

Metoda ovíjení

Ošetření případů, kdy je bod na hraně polygonu bylo vyřešeno přidáním podmínky při rozhodování o poloze bodu. Podmínka řešila případy kdy suma úhlů nabývala hodnot různých od nuly a násobků 2π.

Generování polygonů

Generování nekonvexních polygonů je prováděno generováním náhodných bodů. Ze seznamu bodů byla vytvořena počáteční hrana. Následně byly určovány úhly z počátečního bodu na následující bod v seznamu ve vztahu k výchozí hraně. Vypočtené úhly byly vzestupně seřazeny a tím i určeno pořadí následného spojování bodů. Tento krok vykreslí všechny body vzájemně spojené v jeden polygon. Tento polygon je pak dělen na menší celky. Rozdělování bylo ošetřeno tak, aby s třetinovou pravděpodobností pokaždé došlo k rozdělení na menší polygony nebo aby byl polygon rozdělen na dvě poloviny.

Ukázka vytvořené aplikace

Výsledná aplikace obsahuje jedno tlačítku *Generate*, kterým uživatel spustí vygenerování náhodných polygonů. Následně si vybere algoritmus z nabýdky comboboxu a kliknutím do mapy spustí výpočet. Pokud je zkoumaný bod nachází v polygonu barevně se zvýrazní.

Documentace

Třída Algorithms

Třída *Algorithms* obsahuje funkce:

- jarvis
- qhull
- incr
- graham
- getPointLinePosition
- getTwoVectorsOrientation
- getPointLineDistance

jarvis

Do této funkce vstupuje jedna proměnná typu *std::vector<QPoint>*. Reprezentuje zkoumanou množinu bodů. Výstupem funkce je opět množina bodů uspořádaná do *std::vector<QPoint>*. Množina bodů obsahuje právě ty body, které tvoří konvexní obálku. Tato funkce ja následně volána ve třídě *Draw*.

qhull

Do této funkce vstupuje jedna proměnná typu *std::vector<QPoint>*. Reprezentuje zkoumanou množinu bodů.
Výstupem funkce je množina bodů uspořádaná do *std::vector<QPoint>*, reprezentující konvexní obálku. Tento vektor je následně volán ze třídy *Draw*.

incr

Do této funkce vstupuje jedna proměnná typu *std::vector<QPoint>*. Reprezentuje zkoumanou množinu bodů.
Výstupem funkce je množina bodů uspořádaná do *std::vector<QPoint>*, reprezentující konvexní obálku. Tento vektor je následně volán ze třídy *Draw*.

graham

Do této funkce vstupuje jedna proměnná typu *std::vector<QPoint>*. Reprezentuje zkoumanou množinu bodů.
Výstupem funkce je množina bodů uspořádaná do *std::vector<QPoint>*, reprezentující konvexní obálku. Tento vektor je následně volán ze třídy *Draw*.

getPointLinePosition

Do této funkce vstupují tři proměnné typu *QPointF*.
Výstupem funkce je hodnota typu integer, která reprezentuje polohu zkoumaného bodu (první proměnná) vůči vektoru daného druhou a třetí proměnnou. Výstup nabývá hodnoty 1 pokud je zkoumaný bod v levé polorovině od vektoru, 0 pokud je v pravé polorovině od vektoru a -1 pokud na přímce procházející vektorem.

getTwoVectorsOrientation

Do této funkce vstupují čtyři proměnné typu *QPointF*.
Výstupem funkce je hodnota typu double, která reprezentuje úhel ve stupních mezi vektory, které jsou tvořeny první a druhou proměnnou a třetí a čtvrtou proměnnou.

getPointLineDistance

Do této funkce vstupují tři proměnné typu *QPointF*.
Výstupem funkce je hodnota typu double, která reprezentuje vzdálenost zkoumaného bodu (první proměnná) vůči vektoru daného druhou a třetí proměnnou.

Třída Draw

Třída Draw obsahuje privátní proměnné:

- *Qvector<QPolygonF>* pols
- *QPointF* cursor
- *bool* ignoreDrawPols
- *Qvector<bool>* results
- *bool* draw_what

Třída Draw obsahuje dále veřejnou proměnnou:

- *int* typeAlgorithms

Třída Draw obsahuje dále funkce:

- mousePressEvent
- paintEvent
- generatePoint
- generatePolygons
- setDrawWhat
- getCursor

mousePressEvent

Funkce se aktivuje kliknutím myší do oblasti *QWidgetu* Canvas, který je propojený s třídou *Draw*. Zavolá zvolený algoritmus (*rayAlgorithm*, *windingAlgorithm*) a naplní proměnnou *results*. Následně zavolá funkci *paintEvent* pomocí příkazu *repaint()*.

paintEvent

Funkce je volána funkcí *mousePressEvent* nebo po kliknutí na tlačítko *Generate*. Pomocí privátní proměnné *draw_what* se zavolá funkce *generatePolygons*, která naplní proměnnou *pols* nebo se zvýrazní polygon v němž leží zkoumaný bod.

generatePoint

Funkce generující jeden náhodný bod typu *QPoint*.

generatePolygons

Funkce naplní proměnnou *pols* náhodným počtem náhodně generovaných polygonů. Vstupní hodnota označuje počet generovaných bodů (volání funkce *generatePoint*), z kterých se polygony vytváří.
Funkce používá dvě funkce z třídy *Algorithms* (*getPointLinePosition*, *getTwoVectorsOrientation*) a třídu *sortByXAsc*.

setDrawWhat

Funkce slouží k přepínání uživatelem zvoleného algoritmu.

getCursor

Funkce slouží k získání souřadnic zkoumaného bodu z polohy myši.

Třída MainForm

Třída *MainForm* obsahuje funkce:

- on_pushGenerate_clicked
- on_comboBox_currentIndexChanged

on_pushGenerate_clicked

Funkce se spouští kliknutím uživatele na tlačítko *Generate* a příkazem *repaint()* spustí funkci *paintEvent* ve třídě *Draw*.

on_comboBox_currentIndexChanged

Funkce se spouští změnou *QComboBoxu* a předá informaci o uživatelem zvoleném algoritmu do třídy *Draw*.

sortByXAsc

Třída *sortByXAsc* slouží funkci *generatePolygons* z třídy *Draw* k vzestupnému setřídění úhlů.

Závěr

Aplikace byla vytvořena dle zadání v QtCreator. Pro určení polohy bodu vůči polygonu byly vytvořeny dva algoritmy, mezi kterými si uživatel volí výběrem z comboBoxu. Generování polygonů probíhá na základě námi vytvořeného algoritmu po stisknutí tlačítka Generate. Uživatel klikem do kreslící oblasti určí skoumaný bod a polygony v nichž se bod nachází se barevně zvýrazní.