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 bodů				
školní rok	**studijní skup.**	**číslo zadání**	**zpracovali**	**datum**	**klasifikace**
2023/24	60	xxx	Koudelka Jan Müller Vojtěch	23.4. 2024	

Technická zpráva z úlohy 1 – Geometrické vyhledávání bodu

1. **Zadání:**

Vstup: Souvislá polygonová mapa n polygonů {P1, ..., Pn}, analyzovaný bod q.
Výstup: Pi, q ∈ Pi.

Nad polygonovou mapou implementujete Ray Crossing Algorithm pro geometrické vyhledání incidujícího polygonu obsahujícího zadaný bod q.

Nalezený polygon 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.

- **Detekce polohy bodu rozlišující stavy uvnitř, vně polygonu. (10 bodů)**
- Analýza polohy bodu (uvnitř/vně) metodou Winding Number Algorithm. (10 bodů)
- Ošetření singulárního případu u Winding Number Algorithm: bod leží na hraně polygonu. (5 bodů)
- Ošetření singulárního případu u Ray Crossing Algorithm: bod leží na hraně polygonu. (5 bodů)
- Ošetření singulárního případu u obou algoritmů: bod je totožný s vrcholem jednoho či více polygonů. (2 body)
- Zvýraznění všech polygonů pro oba výše uvedené singulární případy. (3 body)
- Rychlé vyhledávání potenciálních polygonů (bod uvnitř min-max boxu). (10 bodů)

2. **Popis:**

Geometrické vyhledání bodu je častá úloha řešena v prostředí GIS. I v praktickém životě je důležité znát polohu v závislosti na ostatních objektech. Je nutné vyhledávat dané mnohoúhelníky rychle i při velikém množství dat. [1]

3. **Postup:**

3.1. Vytvoření základního grafického rozhraní v prostředí QT:

V prostředí QT bylo navrženo grafické rozhraní výsledné aplikace. Získaný kód byl dále upravován ve skriptovacím prostředí Visual Studio Code.

3.2. Nahrání vytvořeného kódu do skriptovacího prostředí, propojení jednotlivých tříd, vytvoření kreslících funkcí:

V jazyce Python byl získaný kód v prostředí Visual Studio Code upraven do podoby, aby se aplikace dala spouštět. Ve třídě *draw* byly vytvořeny kreslící funkce, které umožňují ruční vykreslení polygonu a určení polohy bodu.

3.3. Ray Crossing algoritmus s ošetřením singulárních případů:

Ray Crossing algoritmus byl vytvořen ve třídě *algorithms*.

Algoritmus prochází všechny strany polygonu, na které vysílá myšlenou polopřímku směrem z bodu *q*. Úkolem algoritmu je rozhodnout, zda je bod *q* uvnitř či mimo analyzovaný polygon. Uvnitř je tehdy, když přímka s polygonem vytvoří lichý počet průsečíků. [1]

- Procházení polygonu bod po bodu, hledání počtu průsečíků přímky zpuštěné z bodu q na danou hranu polygonu s polygonem. Hledání průsečíků:

$$(y_i' > 0)\ \&\ (y_{i-1}' \leq 0)\ |\ (y_{i-1}' > 0)\ \&\ (y_i' \leq 0),$$

pokud platí podmínka tak:

$$x'_m = \frac{x'_i * y'_{i-1} - x'_{i-1} * y'_i}{y'_i - y'_{i-1}},$$

kde x_i‘ a y_i‘ jsou redukované souřadnice. Pokud je x_m‘ > 0, navyšuji počet průsečíků.

- Zkoumání počtu průsečíků v jednotlivých průchodech, pokud je počet průsečíků vždy lichý, bod leží uvnitř polygonu:

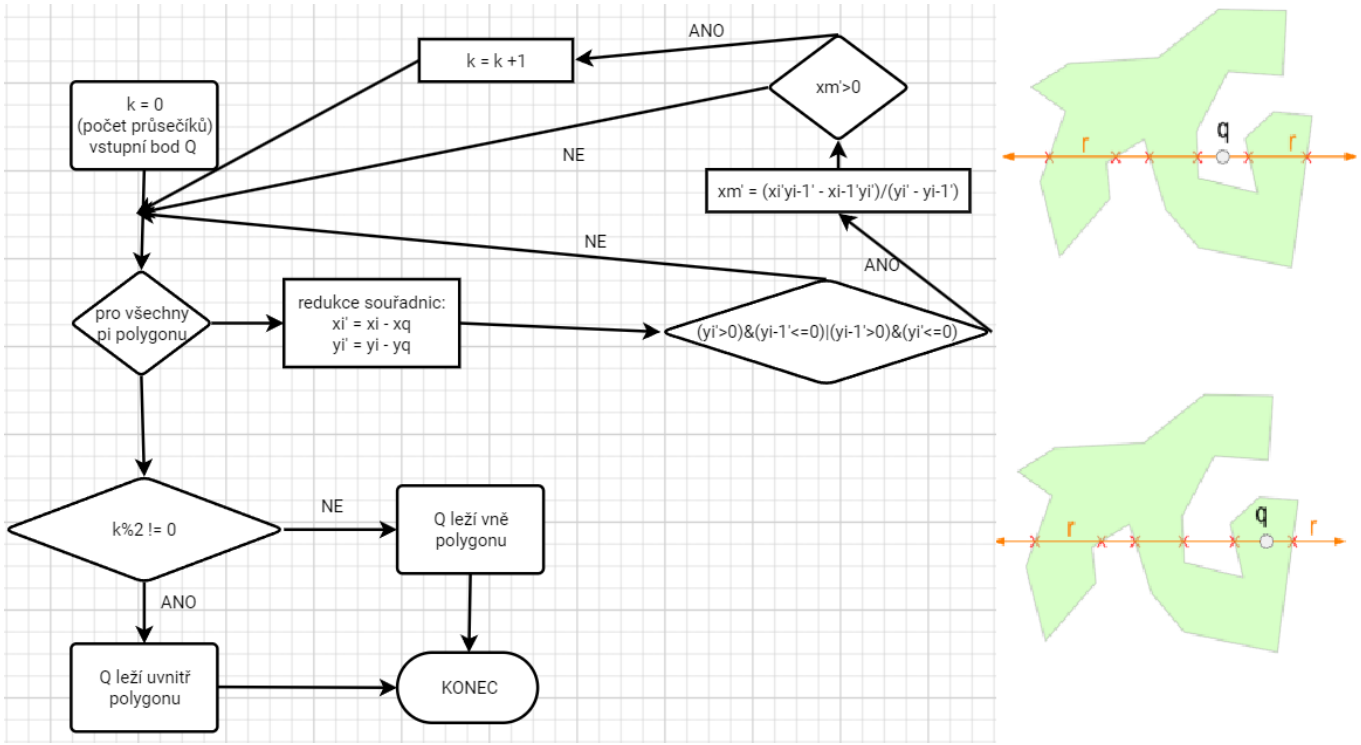
$$k\%2 \neq 0,$$

pak bod q leží uvnitř polygonu.

- Pokud neleží bod ani vlevo, ani vpravo, a zároveň leží uvnitř min-max boxu tvořeného těmito body, bod leží na hraně. Pokud se to stane vícekrát, leží ve vrcholu. Testovací kritérium:

$$t = \begin{vmatrix} x_{i+1} - x_i & y_{i+1} - y_i \\ x_q - x_i & y_q - x_i \end{vmatrix},$$

pokud t>0, bod leží vlevo. Pokud t<0, bod leží vpravo.



Obrázek 1 – Ray Crossing algoritmus [1]

3.4. Winding Number algoritmus s ošetřením singulárních případů:

Winding Number algoritmus byl vytvořen ve třídě *algorithms*.

Algoritmus prochází jednotlivé strany polygonu a vždy vytvoří trojúhelník s daným bodem *q*.

V každém kroku počítá algoritmus úhel při vrcholu *q*. Tento úhel se nasčítává a na závěr se posuzuje velikost této sumy, která je v případě bodu uvnitř polygonu rovna 360°. [1]

- Procházení polygonu bod po bodu, výpočet úhlu sevřeného spojnicemi bodu *q* se dvěma vrcholy dané hrany polygonu.
- Pokud leží bod vlevo od hrany, úhel se přičte, pokud vpravo, úhel se odečte. Testovací kritérium:

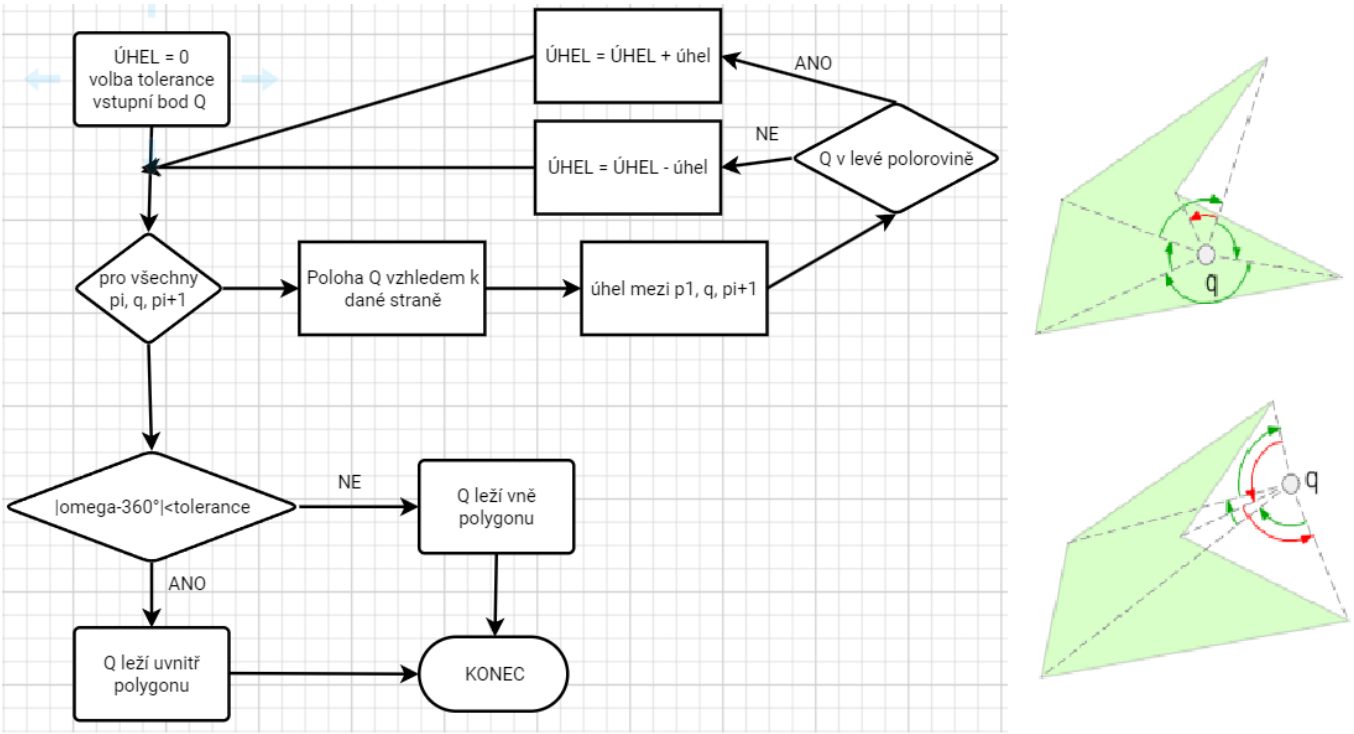
$$t = \begin{vmatrix} x_{i+1} - x_i & y_{i+1} - y_i \\ x_q - x_i & y_q - x_i \end{vmatrix},$$

pokud t>0, bod leží vlevo. Pokud t<0, bod leží vpravo.

- Pokud neleží bod ani vlevo, ani vpravo, a zároveň leží uvnitř min-max boxu tvořeného těmito body, bod leží na hraně. Pokud se to stane vícekrát, leží ve vrcholu.
- Po projetí všech bodů se získaný výsledný úhel porovná s hodnotou 360° (vložena hodnota tolerance).
- Pokud je absolutní hodnota (řeší možnost opačné definice polygonu) výsledného úhlu roven 360°, leží bod uvnitř polygonu. Kontrola:

$$\left|\left|\Omega\right| - 2 * \pi\right|,$$

pokud je výraz menší než tolerance, bod q leží uvnitř polygonu.



Obrázek 2 – Winding Number algoritmus [1]

3.5. Vytvoření třídy pro automatické nahrávání polygonových vrstev ze souborů a aplikace vytvořených algoritmů na tuto situaci:

Pro testování aplikace byl vytvořen polygon obsahující několik vybraných ORP v okolí Prahy. Aplikace umožňuje načítání souborů ve formátu *shapefile*. Uživatel může tyto soubory nahrát v prostředí samotné aplikace.

3.6. Zvýraznění vyhledaných polygonů:

Polygon obsahující zadaný bod se po spuštění přebarví. Je tomu po ošetření tak, i když bod leží na hraně polygonu, či v jeho vrcholu. V těchto případech se obarví všechny polygony sdílící tuto hranu či vrchol.

3.7. Rychlé vyhledávání pomocí min-max boxu:

Min-max box dokáže rychleji detekovat body nacházející se ve větší vzdálenosti od vložených polygonů. S množstvím dat, která máme k dispozici není ovšem časové zjednodušení pozorovatelné. Podmínka:

$$(x_q > x_{min}) \,\&\, (y_q > y_{min}) \,\&\, (x_q < x_{max}) \,\&\, (y_q < y_{max}),$$

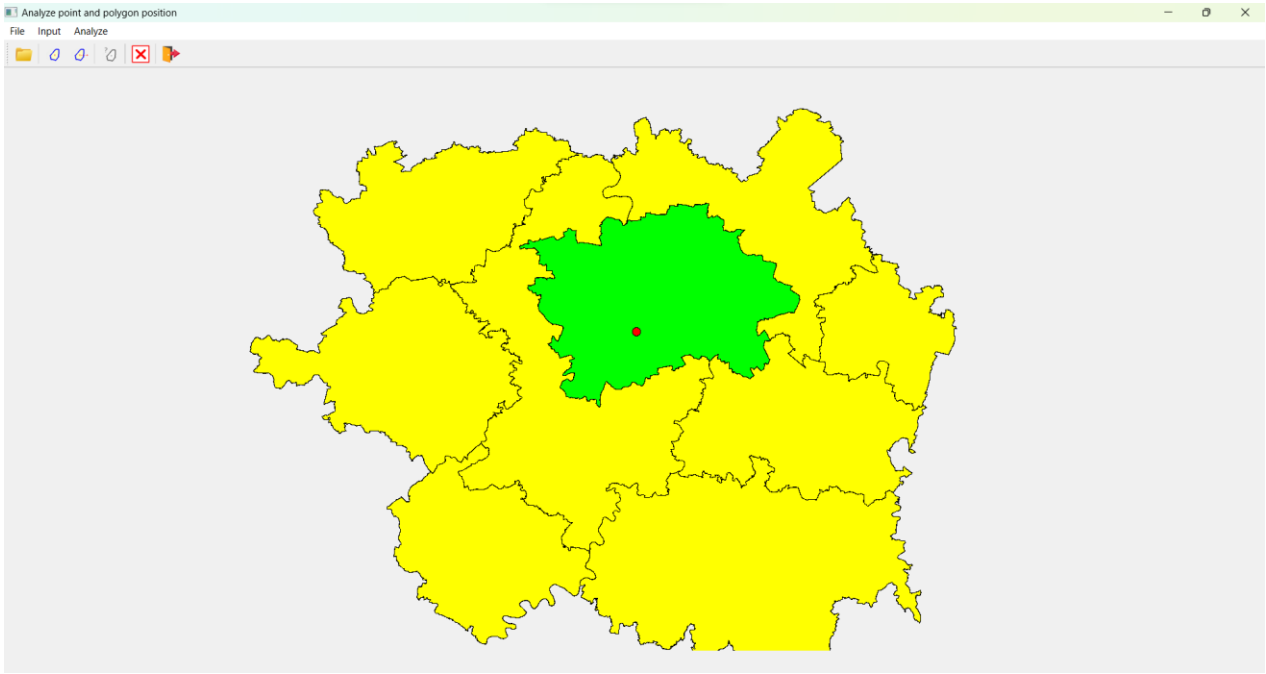
pokud je podmínka splněna, bod q leží uvnitř mim-max boxu daného polygonu.

4. Vstupní data (součást odevzdaných souborů, včetně dokumentace):

- Soubory aplikace:
  - MainForm.py
  - algorithms.py
  - draw.py
  - pio.py
- Soubor obsahující polygonovou vrstvu pro nahrání do aplikace: ORP.shp

5. Výsledky:

Výsledkem této úlohy je grafická aplikace pro rychlé vyhledávání bodu, respektive určování polygonu, ve kterém se bod nachází. Práce aplikace lze vidět na obrázku 3. Více viz dokumentace.



Obrázek 3 – ukázka aplikace

6. Dokumentace

Grafické rozhraní pro danou aplikaci bylo vytvořeno užitím frameworku QT, ve kterém byla nahraná veškerá tlačítka a byly jim přiřazeny základní funkce. Tlačítka viz. Obrázek 3:

- Otevření souboru – umožňuje nahrát soubor obsahující polygony ve formátu shapefile
- Winding Number algoritmus – lokace bodu pomocí Winding Number algoritmu
- Ray Crossing algoritmus – lokace bodu pomocí Winding Number algoritmu
- Přepínač – kreslení bodu / polygonu
- Smazat – smaže dosavadní práci
- Exit – ukončení aplikace

Po nahrání / vytvoření polygonu uživatel přepnutím tlačítka pro umístění bodu umístí bod do uživatelského prostředí, zvolí metodu nalezení bodu a s její pomocí se dozví, zda je umístěný bod vně polygonu, uvnitř, či zda leží na hraně či vrcholu. V případě bodu vně, upozorní uživatele vyskakovací okno. V ostatních případech se vybarví dotčené polygony (viz. Obrázek 3).

Winding Number algoritmus

Tato metoda lokace bodu spočívá ve sledování úhlu, který je nutno z daného bodu vykroužit, aby byly spatřeny popořadě všechny vrcholy polygonu. V případě, že se bod nachází uvnitř polygonu, je úhel otočení roven plnému úhlu 360°. V opačném případě je úhel nižší. Pracováno je vždy s jistou mírou tolerance.

Ray Crossing algoritmus

Druhá metoda určení polohy bodu pracuje s výpočtem počtu průsečíků polopřímky vedené z analyzovaného bodu s polygonem. V případě lichého počtu průsečíku se jedná o bod uvnitř polygonu.

7. Závěr:

Byla vytvořena grafická aplikace umožňující určit aktuální polohu bodu vzhledem k okolním polygonům. Tato funkce může probíhat na základě dvou různých algoritmů – Ray Crossing, Winding Number. Oba algoritmy jsou také ošetřeny o singulární případy, kdy se bod nachází buď na hraně polygonu nebo na jeho vrcholu. Aplikace umožňuje ruční zadání polygonu, dokáže ovšem také nahrát vlastní polygonovou vrstvu ve formátu *shapefile* a analyzovat bod v reálném území. Hledané polygonu jsou po spuštění intuitivně obarveny.

Poslední bonusová úloha, řešení pro polygony s dírami, nebyla nakonec řešena. Hlavním důvodem byl nedostatek času.

8. Seznam literatury:

[1] BAYER, Tomáš. Point Location Problem [online]. Praha [cit. 2024-03-13]. Dostupné z: https://web.natur.cuni.cz/~bayertom/images/courses/Adk/adk3_new.pdf. Katedra aplikované geoinformatiky a kartografie. Přírodovedecká fakulta UK.

Dne: 23.4.2024 **Město:** Praha

Jméno: Jan Koudelka, Vojtěch Müller