

TECHNICKÁ ZPRÁVA

ÚLOHA Č. 1: GEOMETRICKÉ VYHLEDÁVÁNÍ BODU

ZADÁNÍ:

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

Výstup: $P_i, q \in P_i$

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.

SKUPINA: Kateřina Chromá, Štěpán Šedivý

ZPRACOVANÉ DOBROVOLNÉ ČÁSTI ÚKOLU:

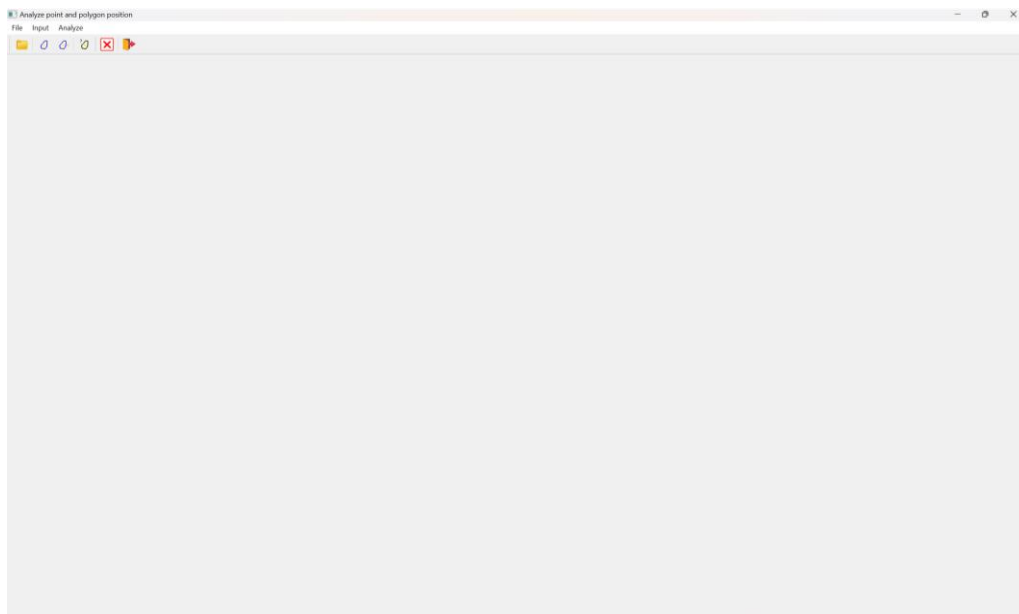
- *Analýza polohy bodu (vně/uvnitř) metodou Winding Number Algorithm*
- *Ošetření singulárního případu u Winding Number Algorithm: bod leží na hraně polygonu*
- *Ošetření singulárního případu u Ray Crossing Algorithm: bod leží na hraně polygonu*
- *Ošetření singulárního případu u obou algoritmů: bod je totožný s vrcholem jednoho či více polygonů.*

POSTUP ZPRACOVÁNÍ:

Hlavním cílem tohoto úkolu bylo vytvořit algoritmy určující polohu bodu vzhledem k jednomu či více polygonů. Také bylo vytvořené grafické rozhraní, kde tyto algoritmy můžeme implementovat.







Grafické rozhraní:

Grafické rozhraní bylo vytvořeno s využitím frameworku QT, kde bylo vytvořeno grafické rozložení (umístění tlačítek, ...). Toto rozhraní bylo zkonvertováno do kódu programovacího jazyka Python.



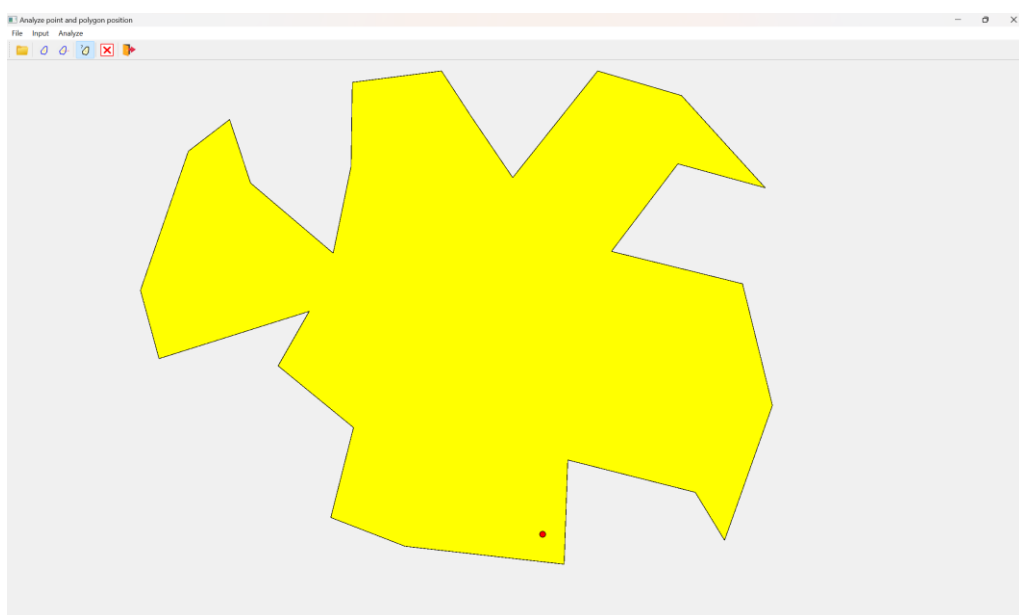
Obrázek 1: Grafické rozhraní

V grafickém rozhraní bylo vytvořeno pět tlačítek:

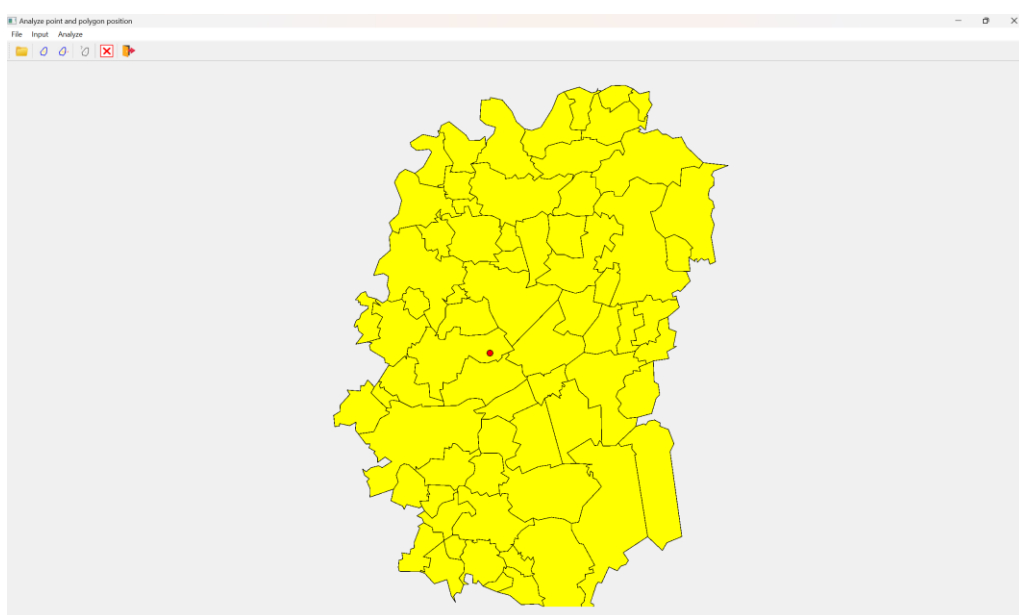
-  : otevření souboru (umožňuje pouze otevřít soubory formátu *shapefile*),
-  : spuštění metody *Winding Number Algorithm*,
-  : spuštění metody *Ray Crossing Algorithm*,
-  : přepínání mezi kreslením polygonu nebo bodu,
-  : odstranění všech prvků,
-  : vypnutí aplikace.

Všechny tyto možnosti jsou také spustitelné z MenuBar.

V grafickém rozhraní si uživatel může sám vytvořit polygon nebo si jej může nainportovat z formátu shapefile.



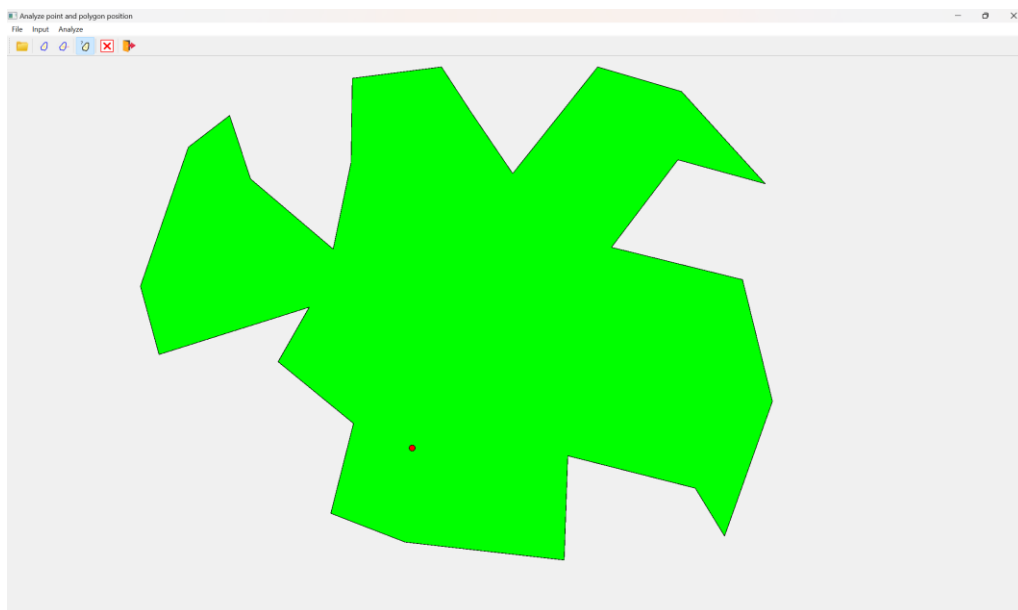
Obrázek 2: Grafické rozhraní - polygon nakreslený uživatelem



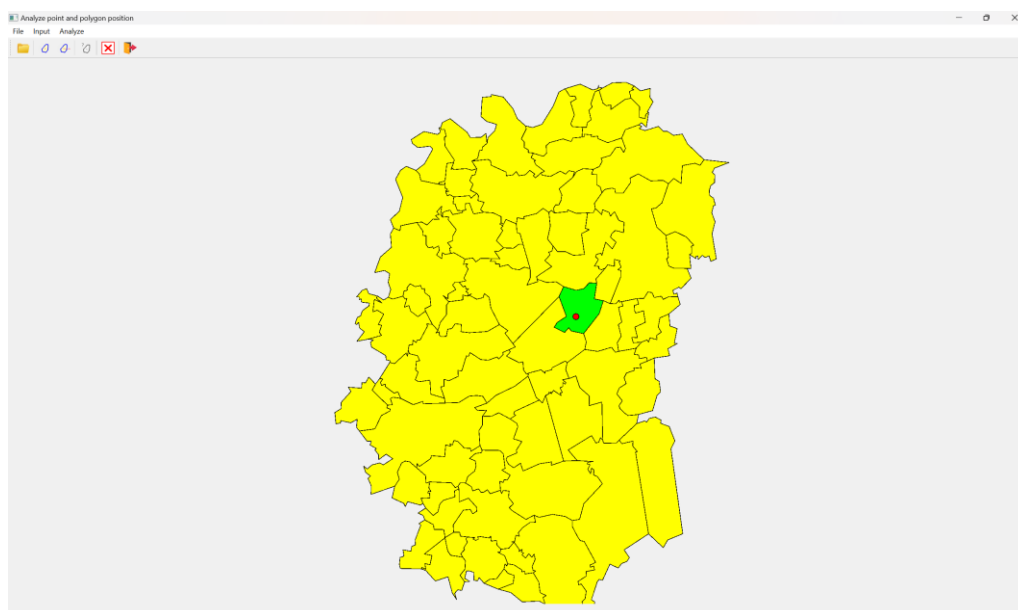
Obrázek 3: Grafické rozhraní – import polygonů

Pokud uživatel zvolí jeden ze dvou algoritmů na zjištění polohy bodu nastane jeden ze dvou možných případů:

1. Bod leží uvnitř polygonu – polygon, kterému bod náleží se vybarví

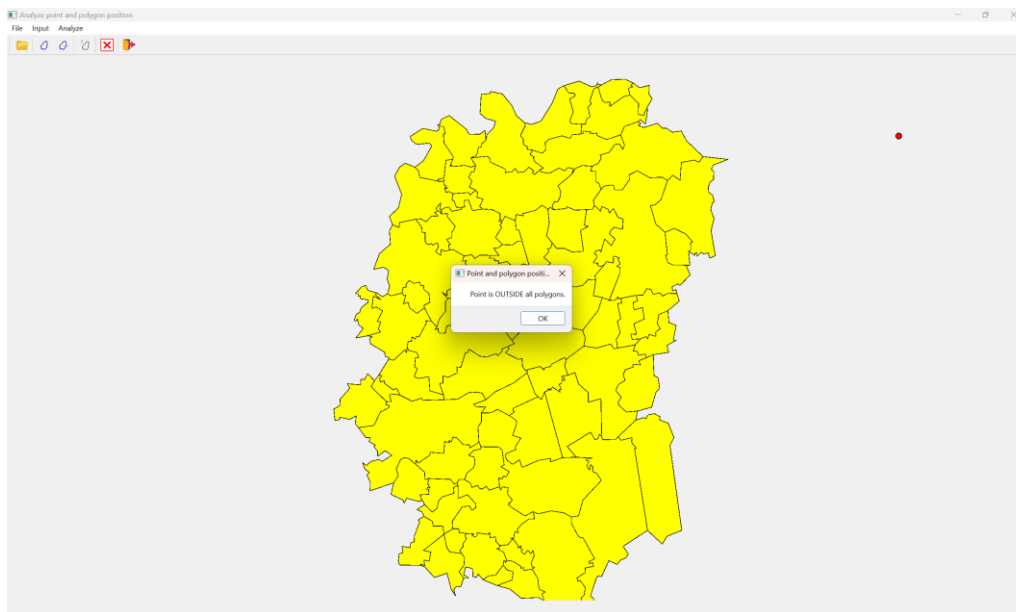


Obrázek 4: Grafické rozhraní – bod leží uvnitř nakresleného polygonu



Obrázek 5: Grafické rozhraní - bod leží uvnitř importovaného polygonu

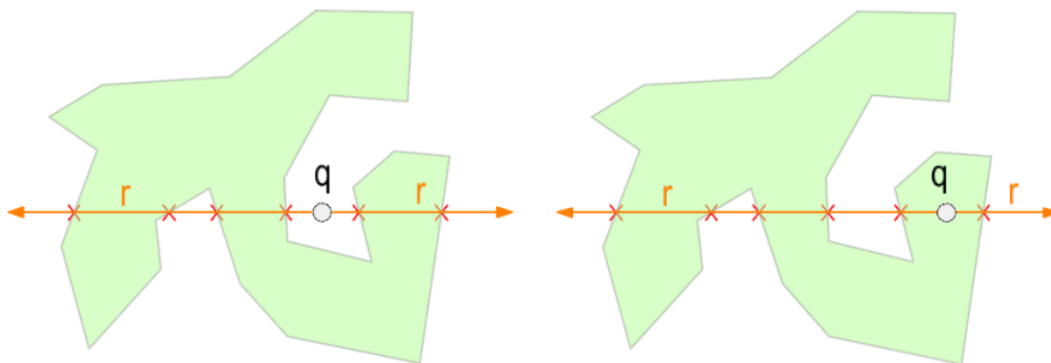
2. Bod leží vně polygonu – zobrazí se okno oznamující, že bod leží vně polygonu



Obrázek 6: Grafické rozhraní - bod leží vně polygonu

Metoda Ray Crossing Algorithm:

Při této metodě jsou počítány průsečíky polopřímky vedené z bodu q a polygonem P . Pokud je průsečíků sudý počet, bod leží vně polygonu. Jestliže je počet průsečíků lichý, bod náleží polygonu.



Obrázek 7: Metoda Ray Crossing Algorithm

Inicializuj $k = 0$

Opakuj pro všechny body $p_i \in P$:

$$x_i' = x_i - x_q$$

$$y_i' = y_i - y_q$$

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

$$x_m' = (x_i' y_{i-1}' - x_{i-1}' y_i') / (y_i' - y_{i-1}')$$

if $(x_m' > 0)$ pak $k = k + 1$

if $(k \% 2) \neq 0$ pak $q \in P$

else $q \notin P$

počet průsečíků

redukce uzlů polygonu podle bodu q

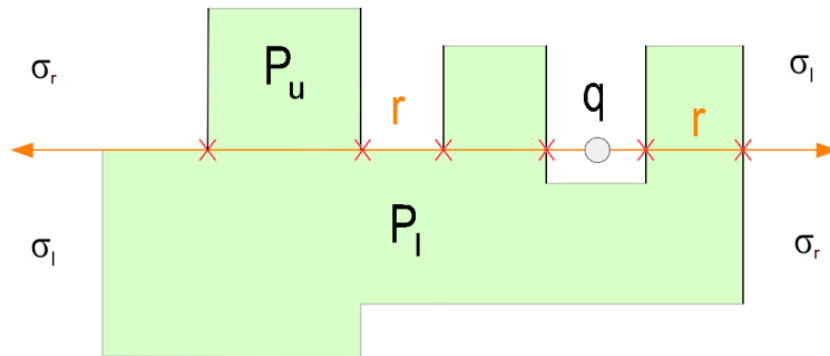
vhodný segment

vhodný průsečík

Metoda Ray Crossing Algorithm – singulární případy:

U této metody nastává problém při rozeznání polohy bodu, když leží na hraně nebo na uzlu polygonu.

Této chybě se dá ovšem vyvarovat pomocí determinantu této matice $\begin{pmatrix} x_{i+1} - x_i & y_{i+1} - y_i \\ x_q - x_i & y_q - y_i \end{pmatrix}$.



Obrázek 8: Metoda Ray Crossing Algorithm – singulární případ

Inicializuj $V = 0$

Opakuj pro všechny body $p_i \in P$:

 Spočti determinant mezi uzly p_i, p_{i+1} a bodem q

 if determinant == 0:

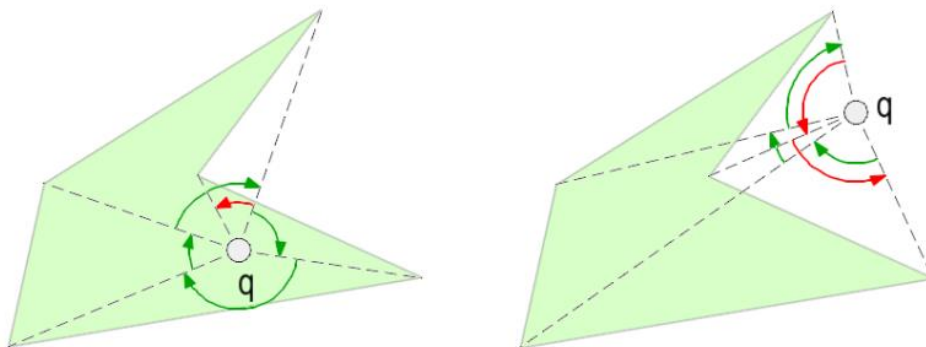
$V = V + 1$

if $V == 1$ pak q leží na hraně

if $V > 1$ pak q leží na uzlu

Metoda Winding Number Algorithm:

Tato metoda zakládá na principu pozorovatele vůči objektu. Jestliže se pozorovatel musí otočit o úhel 2π při pozorování všech rohů objektu, je pozorovatel uvnitř objektu. Ale pokud stojí pozorovatel vně objektu, úhel otočení při pozorování všech rohů objektu je menší než 2π . Počítá se tedy suma všech rotací, které musí průvodič opsat nad všemi body polygonu. Jestliže bod leží v levé polorovině, jeho rotace se přičítá. Pokud ovšem leží v pravé polorovině, jeho rotace se odečítá.



Obrázek 9: Metoda Winding Number Algorithm

Inicializuj $\Omega = 0$, tolerance ε

Opakuj pro všechny trojice (p_i, q, p_{i+1}) :

 Urči polohu q vzhledem k p

 Urči úhel $\omega_i = \angle p_i, q, p_{i+1}$

 if $q \in \sigma_L$, pak $\Omega = \Omega + \omega_i$

 else $\Omega = \Omega - \omega_i$

if $||\Omega| - 2\pi| < \varepsilon$, pak $q \in P$

else $q \notin P$

Metoda Winding Number Algorithm – singulární případy:

U této metody, tak jako u předchozí také nastává problém při rozeznání polohy bodu, když leží na hraně nebo uzlu polygonu. Vyvarovat se jí dá stejně – pomocí determinantu této matice

$$\begin{pmatrix} x_{i+1} - x_i & y_{i+1} - y_i \\ x_q - x_i & y_q - y_i \end{pmatrix}.$$

ZÁVĚR:

Bylo vytvořené grafické rozhraní, kde si uživatel může sám nakreslit bod a polygon nebo si polygony naimportovat ze souboru shapefile. Do tohoto rozhraní byly zaimplementována metoda Winding Number Algorithm a metoda Ray Crossing Algorithm, která určuje, zda bod leží vně nebo uvnitř polygonu. Výsledek těchto algoritmů je v grafickém rozhraní uveden.

ZDROJE:

Obrázky [7], [8], [9] a informace o průběhu výpočtu:

Point Location Problem. Online. BAYER, Tomáš. Point Location Problem. 2024. Dostupné z: https://web.natur.cuni.cz/~bayertom/images/courses/Adk/adk3_new.pdf. [cit. 2024-03-31].

V Plzni dne 31.3.2024

Kateřina Chromá, Štěpán Šedivý