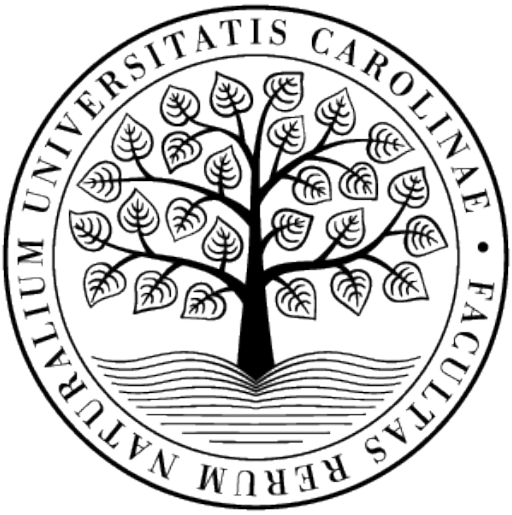
Algoritmy počítačové kartografie

Úloha 1
Geometrické vyhledání bodu

František Macek, Josef Zátka



Univerzita Karlova

Přírodovědecká fakulta

Katedra aplikované geoinformatiky a kartografie

Praha, 2025

1 Zadání

Vstup: Souvislá polygonová mapa n polygonů $\{P_1, ..., P_n\}$, 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.

2 Údaje o bonusových úlohách

Byly zprácovány následující bonusové úlohy s uvedeným bodovým ohodnocením:

Krok	**Hodnocení**
Detekce polohy bodu rozlišující stavy uvnitř, vně polygonu.	10b
Analýza polohy bodu (uvnitř/vně) metodou Winding Number Algorithm.	+10b
Ošetření singulárního případu u Winding Number Algorithm: bod leží na hraně polygonu.	+5b
Ošetření singulárního případu u Ray Crossing Algorithm: bod leží na hraně polygonu.	+5b
Ošetření singulárního případu u obou algoritmů: bod je totožný s vrcholem jednoho či více polygonů.	+2b
Zvýraznění úsech polygonů pro oba výše uvedené singulární případy.	+3b
Rychlé vyhledání potenciálních polygonů (bod uvnitř min-max boxu).	+10b
Načtení vstupních dat ze *.shp.	+10b
Celkem splněno:	**55b**

3 Úvod, popis a rozbor problematiky

Jedním z často řešených problémů v oblasti počítačové grafiky a digitální kartografie je určování vztahu mezi polohou daného bodu a polygonem. Tento úkol, známý jako

problém bodu v polygonu (Point Location Problem), slouží k určení, zda se bod nachází uvnitř, vně nebo na hranici daného polygonu (Bayer 2025).

Existují dvě hlavní techniky řešení problému bodu v polygonu (Bayer 2025):

1. **Převod problému na vztah bodu a mnohoúhelníku** – Opakovaně určujeme polohu bodu vzhledem k polygonu. Tato metoda je snadno implementovatelná, ale pomalejší.
2. **Planární dělení roviny** – Rovina je rozdělena na pásy nebo lichoběžníky, čímž vzniká trapezoidální mapa. Tato technika je výrazně rychlejší, ale složitější na implementaci, vyžaduje speciální datové struktury (např. binární stromy).

Řešení problému se dále liší v závislosti na tom, zda do problému vstupují pouze konvexní polygonu, nebo také polygony nekonvexní. Pro konvexní polygony (ty mají vnitřní úly o maximální velikosti 180°) je řešení výpočetně jednodušší. V geoinformatice a kartografii se ovšem běžně setkáváme s polygony nekonvexními, a tak je i v této úloze problém řešen pro nekonvexní polygony.

Mezi základí algoritmy pro řešení problému bodu v nekonvexním polygonu jsou řazeny Ray Crossing Algorithm a Winding Number Algorithm. Ty byly v této úloze implementovány a v dalších kapitolách jsou podrobněji popsány.

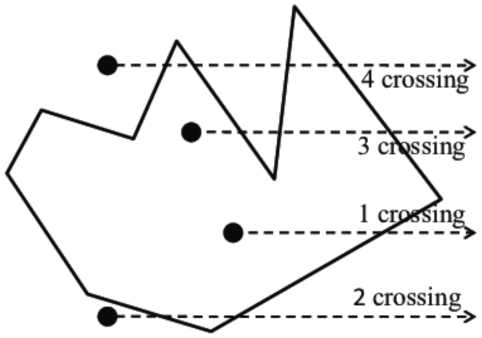
4 Ray Crossing Algorithm

4.1 Popis algoritmu

Základním principem *Ray Crossing Algorithm* je vysílání "paprsku"ze zvoleného bodu q do jednoho směru. Tento paprsek tvoří polopřímku, která k krát protne obálku polygonu P. Na základě počtu průsečíku lze určit, zda bod q leží uvnitř či vně polygonu P dle pravidla

$$k\%2 = \begin{cases} 1, & q \in P, \\ 0, & q \notin P. \end{cases}$$

Tedy pokud je počet průsečíků sudý, lze říci, že bod leží vně polygonu. Je-li počet průsečíků lichý, bod leží uvnitř polygonu. Obě varianty ilustruje obrázek 1.



Obrázek 1: Základní princip Ray Crossing Algorithm (Yan, Zhao a Ng 2012)

V praxi je jako paprsek volena polopřímka rovnoběžná s osou x, která míří od bodu q v kladném směru osy x (GeeksforGeeks 2024).

V algoritmu je polygon reprezentován jako uspořádaná posloupnost bodů, dvojice po sobě jdoucích bodů tedy tvoří hrany takového polygonu. Abychom zjistili, zda paprsek procházející bodem q protíná polygon, musíme pro každou hranu polygonu prověřit, zda protíná vytyčenou polopřímku.

Každá hrana je určena dvěma sousedními vrcholy polygonu

$$p_i = (x_i, y_i), \quad p_{i+1} = (x_{i+1}, y_{i+1}).$$

Pro každý úsek $[p_i, p_{i+1}]$ jsou vypočítány relativní souřadnice vůči bodu q jako

$$x'_i = x_i - x_q, \quad y'_i = y_i - y_q$$
$$x'_{i+1} = x_{i+1} - x_q, \quad y'_{i+1} = y_{i+1} - y_q$$

Tyto transformované souřadnice umožňují pracovat s bodem q jako novým počátkem souřadnicového systému. Díky tomu lze snadněji testovat, zda hrana protíná vodorovnou polopřímku v kladném směru osy x.

Aby hrana protnula paprsek, musí splňovat podmínku

$$(y'_i > 0 \wedge y'_{i+1} \leq 0) \quad \text{nebo} \quad (y'_i \leq 0 \wedge y'_{i+1} > 0)$$

Tato podmínka zajišťuje, že úsečka kříží osu x (tedy paprsek).

Pokud podmínka platí, pak je vypočtena souřadnice průsečíku hrany s osou x:

$$x_m = \frac{x'_{i+1}y'_i - x'_i y'_{i+1}}{y'_{i+1} - y'_i}$$

Pokud je $x_m > 0$, znamená to, že průsečík **leží napravo od bodu** q, a tedy **započítáme ho jako průsečík s paprskem**.

Po vyhodnocení každé hrany lze dle pravidla o sudosti celkového počtu průsečíků (viz výše) rozhodnout o tom, zda bod leží uvnitř nebo vně polygonu.

4.2 Singulární případy

4.2.1 Bod na hraně polygonu

Leží-li bod na hraně polygonu či na prodloužení vodorovné hrany, vyjde

$$x_m = 0,$$

tehdy proběhne (v naší implementaci) ověření, zda bod leží uvnitř minmax boxu tvořeného ověřovanou hranou. Pokud ano, bod leží na této hraně. Pokud nikoliv, bod leží vně polygonu. Běžnější je přístup, při kterém je tato singularita testována použitím dvou paprsků - pokud je výsledek (počet průsečíků s polygonem) pro jeden z nich lichý a pro druhý sudý, musí bod ležet na hraně polygonu.

4.2.2 Bod leží ve vrcholu polygonu

Další singularitou je situace, při které bod je vrcholem polygonu. V takovém případě by byl započítán jako průsečík každé hrany vedoucí z vrcholu. Tato singularita je řešena před samotným během programu tím, že je ověřeno zda bod není totožný s jedním z vrcholů polygonu.

5 Winding Number Algorithm

5.1 Popis algoritmu

Winding Number Algorithm je druhou implementovanou technikou pro určení, zda bod q leží uvnitř nebo vně polygonu P. Tento algoritmus vyhodnocuje, jak moc se polygon

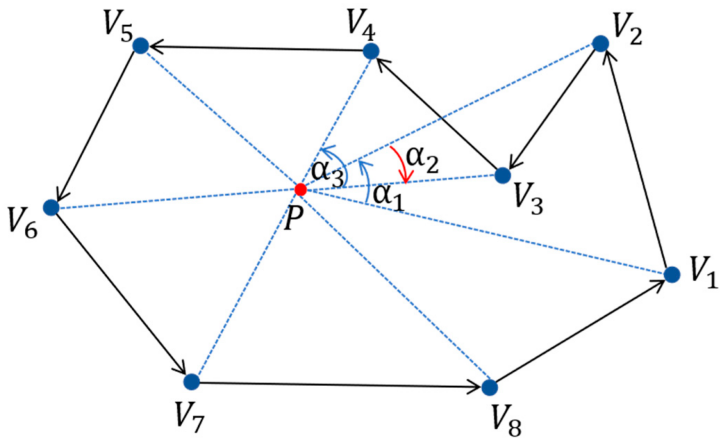
"obtáčí"kolem daného bodu pomocí výpočtu součtu úhlových změn mezi sousedními vrcholy polygonu vůči bodu q.

Princip algoritmu spočívá ve výpočtu celkového úhlového přírůstku *Omega*, který vzniká postupným měřením úhlů mezi sousedními hranami polygonu a testovaným bodem. Po sečtení těchto úhlů platí následující pravidlo:

$$|\Omega| \approx 2\pi \Rightarrow q \in P$$

$$|\Omega| < 2\pi \Rightarrow q \notin P$$

Tedy pokud je absolutní hodnota součtu úhlů přibližně 2π, bod se nachází uvnitř polygonu. Pokud je menší, bod leží vně polygonu, nebo na jeho hraně (princip ošetření takového případu je popsán v kapitole 5.2 Singulární případy). Princip fungování algoritmu je ilustrován na obrázku 2.



Obrázek 2: Základní princip Winding Number Algorithm (Liu et al. 2023)

Polygon je opět reprezentován jako uspořádaná posloupnost bodů, kde dvojice po sobě jdoucích bodů tvoří hrany polygonu:

$$p_i = (x_i, y_i), \quad p_{i+1} = (x_{i+1}, y_{i+1}).$$

Pro každý úsek $[p_i, p_{i+1}]$ jsou nejprve vypočteny vektory z bodu q k oběma vrcholům hrany:

$$\mathbf{v}_1 = (x_i - x_q, y_i - y_q), \quad \mathbf{v}_2 = (x_{i+1} - x_q, y_{i+1} - y_q).$$

Následně je spočítán skalární součin těchto dvou vektorů:

$$\mathbf{v}_1 \cdot \mathbf{v}_2 = (x_i - x_q)(x_{i+1} - x_q) + (y_i - y_q)(y_{i+1} - y_q).$$

Velikosti vektorů jsou dány vztahem:

$$\|\mathbf{v}_1\| = \sqrt{(x_i - x_q)^2 + (y_i - y_q)^2}, \quad \|\mathbf{v}_2\| = \sqrt{(x_{i+1} - x_q)^2 + (y_{i+1} - y_q)^2}.$$

Úhel θ_i mezi vektory je pak získán pomocí vztahu:

$$\cos\theta_i = \frac{\mathbf{v}_1 \cdot \mathbf{v}_2}{\|\mathbf{v}_1\|\|\mathbf{v}_2\|}$$

$$\theta_i = \arccos\left(\frac{\mathbf{v}_1 \cdot \mathbf{v}_2}{\|\mathbf{v}_1\|\|\mathbf{v}_2\|}\right).$$

Pro správné započítání úhlu je třeba určit, zda bod q leží na levé nebo pravé straně hrany $[p_i, p_{i+1}]$. Toho lze dosáhnout pomocí determinantového testu (označovaného v anglické terminologii jako *cross product*):

$$D = (x_{i+1} - x_i)(y_q - y_i) - (y_{i+1} - y_i)(x_q - x_i).$$

Pokud $D > 0$, bod q leží **vlevo** od hrany a úhel θ_i je **přičten** k Ω.

Pokud $D \leq 0$, bod q leží **vpravo** od hrany a úhel θ_i je **odečten** od Ω.

Po spočtení celkového součtu úhlových přírůstků Ω se bod klasifikuje podle pravidla:

$$|\Omega - 2\pi| < \varepsilon \Rightarrow q \in P$$

$$|\Omega - 2\pi| \geq \varepsilon \Rightarrow q \notin P,$$

kde ε představuje zvolenou toleranci pro zaokrouhlovací chyby.

5.2 Singulární případy

5.2.1 Bod na hraně polygonu

Leží-li bod na hraně polygonu či na prodloužení vodorovné hrany, vyjde cross product (determinantový test) = 0. Tehdy je opět, stejně jako v případě Ray Crossing algoritmu testováno, zda bod leží uvnitř minmax boxu ověřované hrany tehdy proběhne (v naší implementaci) ověření, zda bod leží uvnitř minmax boxu tvořeného ověřovanou hranou. Pokud ano, bod leží na této hraně. Pokud nikoliv, bod leží vně polygonu.

5.2.2 Bod leží ve vrcholu polygonu

Další singularitou je opět situace, při které je ověřovaný bod vrcholem polygonu. V takovém případě by byl započítán jako průsečík každé hrany vedoucí z vrcholu. Tato singularita je řešena před samotným během programu tím, že je ověřeno zda bod není totožný s jedním z vrcholů polygonu.

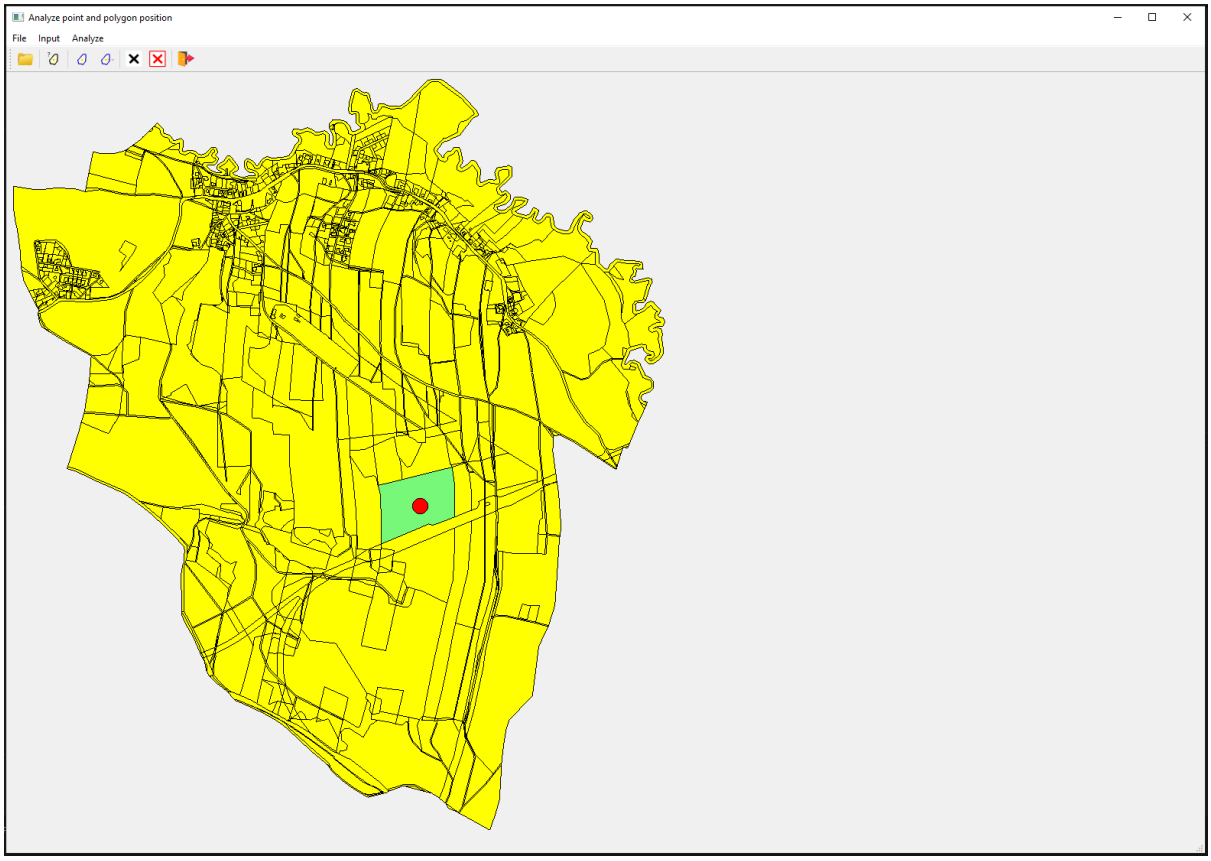
6 Vlastní implementace

Součástí zadání bylo nejen implementovat algoritmy pro analýzu polohy bodu v prostoru, ale také navrhnout intuitivní a uživatelsky přívětivé rozhraní ve frameworku Qt. Toto rozhraní umožňuje snadnou demonstraci funkčnosti obou zmíněných algoritmů nad vybranou vrstvou polygonů, čímž uživateli poskytuje přehledný a interaktivní způsob vizualizace výsledků.

6.1 Vstupní a výstupní data

Aplikace umožňuje importovat polygony ze souboru formátu *.shp. Pro správné vykreslování polygonů v okně aplikace je předpokladem, že geometrie je v shapefilu uložena v souřadnicovém systému WebMercator (EPSG:3857). U něj s rostoucí zeměpisnou délkou roste hodnota souřadnice X (směrem doprava) a s klesající zeměpisnou šířkou roste hodnota soouřadnice Y (ve směru dolů). Tato vlastnost umožňuje neinvertovat souřadnici Y a rovnou ji použít na vykreslení v knihovně Qt, protože v ní je počátkem souřadnic (0,0) levý horní bod vykreslovací plochy.

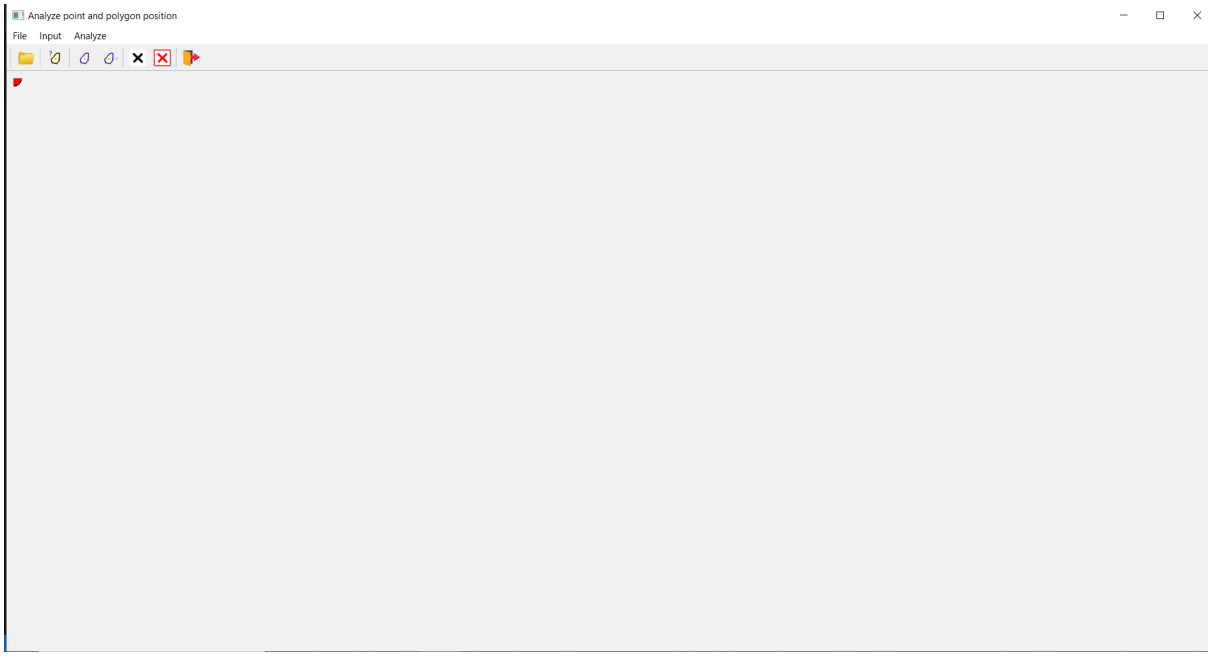
Výstupem aplikace je podbarvení polygonu uvnitř kterého leží zvolený bod (obrázek 3).



Obrázek 3: Podbaarvení polygonu uvnitř kterého leží vybraný bod

6.2 Aplikace

Aplikace byla vytvořena prostřednictvím knihoven Qt v programovacím jazyce Python. Uživatel může otevřít soubor formátu shapefile z vlastního adresáře. Následně lze v aplikaci levým tlačítkem myši zvolit bod pro analýzu jeho polohy vůči nahraným polygonům. Volby načtení souboru, analýzy jeho polohy metodou Ray Crossing a Winding Number a další operace lze spustit pomocí ikon umístěných v horní části okna aplikace (obrázek 4). Nalezený polygon, uvnitř kterého leží zvolený bod je zeleně podbarven.



Obrázek 4: Okno aplikace

Uživatel může také nahraná data vymazat ikonou červeného křížku a ukončit běh celého programu k tomu určenou ikonou.

7 Třídy a metody tříd

Pro zdárný běh celé aplikace jsou nezbytné soubory `MainForm.py`, `draw.py`, `algorithms.py` a `read_shp.py`.

7.1 Třída MainForm

Třída zajišťuje vykreslení okna aplikace, včetně všech ikon a tlačítek. Také zajišťuje jejich propojení s metodami, které vyvolávají další akce. Metody třídy MainForm a jejich funkce jsou následující:

- **setupUi a retranslateUi** – Nastavují grafické uživatelské rozhraní aplikace, včetně rozložení, menu, toolbaru a jejich propojení s funkcemi.
- **analyzeRay** – Určuje, zda bod leží uvnitř polygonu pomocí Ray-Crossing algoritmu volaného ze třídy Algorithms.
- **inside**$_{b}$*ounding*–*Analyzuje, zdabodležíuvnitřminmaxboxutvořenéhodvěmajinýmibody*.**analyze**

- **switchClick** – Přepíná vstup mezi bodem a polygonem.
- **openFileDialog** – Otevírá dialogové okno pro výběr souboru `.shp` a načítá data do vykreslovacího okna.
- **pointInside** – Zobrazí dialogové okno s informací, že bod je uvnitř polygonu, a zvýrazní ho.
- **pointOutside** – Zobrazí dialogové okno s informací, že bod je mimo polygon.
- **pointEdge** – Zobrazí dialogové okno s informací, že bod se nachází na hraně polygonu, a zvýrazní odpovídající polygon/y ve vykreslovacím okně.
- **pointVertex** – Zobrazí dialogové okno s informací, že bod se shoduje s vrcholem jednoho nebo více polygonů, a vizuálně je zvýrazní.

7.2 Třída Draw

Třída zajišťuje vykreslování bodů a polygonů v aplikaci a umožňuje interakci uživatele s grafickou plochou prostřednictvím stlačení tlačítka myši. Metody třídy Draw a jejich funkce jsou následující:

- **mousePressEvent** – Zaznamenává kliknutí myší, ukládá souřadnice bodu nebo přidává vrcholy k polygonu.
- **paintEvent** – Zajišťuje vykreslení bodů a polygonů na plátno včetně zvýraznění označeného polygonu.
- **paintInputEvent** – Aktualizuje seznam vykreslovaných polygonů po nahrání vstupního souboru formátu .shp.
- **switchInput** – Přepíná mezi režimem zadávání bodu a režimem přidávání vrcholů do polygonu.
- **getQ** – Vrací souřadnice označeného bodu.
- **getPol** – Vrací aktuální polygon.
- **getPolygons** – Vrací seznam všech polygonů.
- **highlightPolygon** – Zvýrazňuje vybraný polygon a zajišťuje překleslení plátna.
- **unhighlightPolygon** – Odbarví zvýrazněné polygony.
- **clearData** – Odstraňuje všechna data z vykreslovací plochy.

7.3 Třída Algorithms

Třída obsahuje algoritmy pro analýzu polohy bodu vzhledem k polygonu a výběr potenciálně relevantních polygonů. Metody třídy Algorithms a jejich funkce jsou následující:

- **ray_crossing** – Určuje, zda bod leží uvnitř polygonu pomocí Ray-Crossing algoritmu na základě počtu průsečíků s hranicemi polygonu.
- **winding_number** – Testuje polohu bodu vůči polygonu pomocí Winding Number algoritmu na základě součtu úhlů.
- **minmaxbox** – Vytváří ohraničující obdélník (min-max box) pro daný polygon pomocí minimálních a maximálních souřadnic jeho vrcholů.
- **point_inside_minmaxbox** – Určuje, zda bod leží uvnitř ohraničujícího obdélníku polygonu, což slouží k předběžné eliminaci nerelevantních polygonů.
- **select_suspicious_polygons** – Vyhledává a vrací seznam polygonů, jejichž min-max box obsahuje zadaný bod.
- **point_vertex** – Vrací informaci o tom, zda je analyzovaný bod shodný s některým z vrcholů polygonu.

7.4 Soubor read_shapefiles.py

Soubor obsahuje jedinou funkci load_shapefile, která načítá geometrie polygonů ze souboru ve formátu `.shp` pomocí knihovny GeoPandas a následně upravuje jejich souřadnice pro vykreslení v aplikaci. Nejprve se soubor načte a získají se jeho minimální a maximální souřadnice, které určují rozsah dat. Poté se vypočítá měřítko pro přizpůsobení souřadnic velikosti okna, přičemž se zachovává poměr stran. Funkce následně prochází jednotlivé polygony, převádí jejich vrcholy na odpovídající souřadnice v okně a ukládá je jako objekty `QPolygonF`. Tímto způsobem se zajistí, že načtené polygony budou správně zobrazeny v aplikaci. Na závěr funkce vrací seznam těchto polygonů ve formátu `QPolygonF`, které lze použít pro vykreslení.

8 Závěr

Prostřednictvím funkční aplikace byly implementovány algoritmy Ray Crossing a Winding Number pro určení polohy bodu a polygonu. Součástí řešení je i ošetření singulárních

případů, kdy bod leží na hraně polygonu či v jednom z jeho vrcholů. Program v současné verzi nedokáže zpracovávat polygony s dírami, v této oblasti se nabízí prostor pro budoucí vylepšení. Také možnost importu vstupních polygonů je omezena pouze na soubory formátu shapefile. Vylepšením předkládaného programu by mohla být možnost importovat data také z jiných geoprostorových formátů (např. GeoJSON). Současná verze validně vykresluje pouze shapefily se souřadnicovým systémem WebMercator, podpora jiných souřadnicových systémů. Vhodným vylepšením by také byla možnost posouvání a přibližování//oddalování plochy na které jsou vykresleny polygony. Vytvořená aplikace je uživatelům dostupná na adrese .

Zdroje

Bayer, Tomáš (2025). *Přednáška z algoritmů počítačové kartografie, Point Location Problem.* Slidy dostupné online, navštíveno 15. března 2025. URL: https://web.natur.cuni.cz/~bayertom/images/courses/Adk/adk3_new.pdf.

GeeksforGeeks (2024). *Point in Polygon in C.* URL: https://www.geeksforgeeks.org/point-in-polygon-in-c/ (cit. 16. 03. 2025).

Liu, Lei et al. (2023). "Efficient Construction of Voxel Models for Ore Bodies Using an Improved Winding Number Algorithm and CUDA Parallel Computing". In: *ISPRS International Journal of Geo-Information* 12.12. ISSN: 2220-9964. DOI: 10.3390/ijgi12120473. URL: https://www.mdpi.com/2220-9964/12/12/473.

Yan, Da, Zhou Zhao a Wee Keong Ng (2012). *Monochromatic and Bichromatic Reverse Nearest Neighbor Queries on Land Surfaces.* Conference paper. DOI: 10.1145/2396761.2396880.