České vysoké učení technické v Praze
Fakulta stavební

Algoritmy v digitální kartografii

Geometrické vyhledávání bodu

Bc. Petra Pasovská
Bc. David Zahradník

Obsah

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 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.

Hodnocení:

Krok	**Hodnocení**
Detekce polohy bodu rozlišující stavy uvnitř, vně na hranici polygonu.	10b
Ošetření singulárního případu u Winding Number Algorithm: bod leží na hraně polygonu.	*+2b*
Ošetření singulárního případu u obou algoritmů: bod je totožný s vrcholem jednoho či více polygonů.	*+2b*
Zvýraznění všech polygonů pro oba výše uvedené singulární případy.	*+2b*
Algoritmus pro automatické generování nekonvexních polygonů.	*+5b*
Max celkem:	**21b**

Čas zpracování: 2 týdny.

1.1 Údaje o bonusových úlohách

V úloze byly vypracovány bonusové části týkající se ošetření singulárního případu u Winding Number Algorithm, kdy bod leží na hraně polygonu. Dále byl ošetřen singulární případ u obou algoritmů, kdy bod leží ve vrcholu jednoho či více polygonů. Pro oba výše zmíněné singulární případy byly zvýrazněny polygony.

?? Nekonvexní polygon? když se povede???

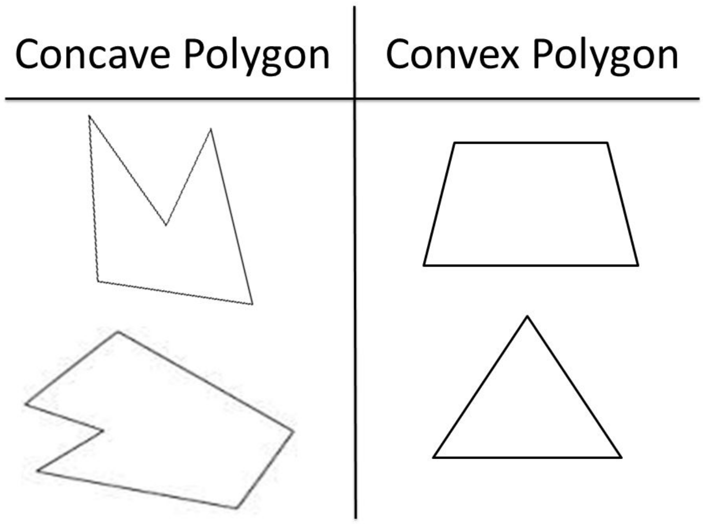
2 Popis a rozbor problému

Použít na důležité pojmy a objekty kurzívu/tučně, odřádkování apod ať to nějak vypadá!!!

Hlavním cílem této úlohy je tvorba aplikace, která uživateli určí pozici zvoleného bodu q. Termínem pozice je myšlen polygon, kterému zadaný bod přísluší.

V závislosti na tvaru polygonu rozlišujeme dva základní typy. Jedná se o konvexní a nekonvexní polygon. Polygon můžeme označit za konvexní právě tehdy, pokud jsou všechny vnitřní úhly konvexní, tedy v případě, že úhly jsou menší nebo rovny hodnotě 180 °. Zároveň pro takový polygon platí, že všechny přímky, jejichž oba krajní body leží uvnitř polygonu, mají s tímto polygonem všechny body společné. Takový polygon, který není konvexní, lze označit jako nekonvexní či konkávní.

Porovnání konvexního a nekonvexního objektu lze vidět na následujícím obrázku.



Obrázek 1: Porovnání konvexního a konkávního polygonu [zdroj: 1]

Bod q může mít vůči polygonu P jednu z těchto poloh:

1. Bod q leží uvnitř polygonu P.
2. Bod q leží vně polygonu P.
3. Bod q leží na hraně polygonu P.
4. Bod q je totožný s některým z vrcholu polygonu P.

Pro určení pozice bodu q vůči polygonu existuje několik metod. V této aplikaci jsou implementován metody Ray Crossing Algorithm (varianta s posunem těžiště polygonu) a metoda Winding Number Algorithm.

3 Popis použitých algoritmů

Opět projet formu zápisu a vzhled!!!

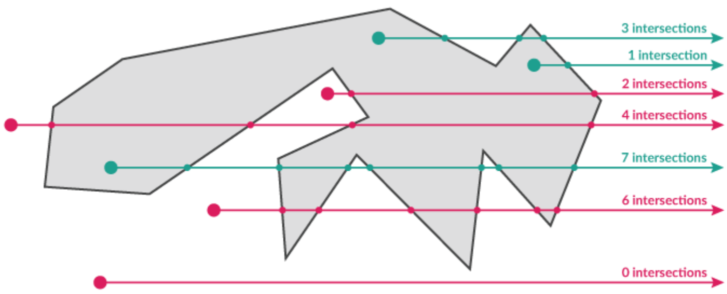
Existuje mnoho metod pro určení pozice bodu q vůči polygonu P. Při volbě metod je vždy potřeba zhodnotit několik důležitých bodů, například požadavky vstupních/výstupních dat, časová náročnost či zda typ problému nespadá mezi NP problémy. V aplikaci byly použity algoritmy Ray Crossing Algorithm a Winding Number Algorithm. Mezi další známé metody pro určení pozice bodů patří třeba Line Sweep Algorithm (Zametací přímka), Divide and Conquer (Rozděl a panuj) či lze pozice určit i metodou hrubé síly (Brute Force Algorithm).

3.1 Ray Crossing Algorithm

Ray Crossing Algorithm lze do češtiny přeložit jako paprskový algoritmus. Primárně slouží k určení polohy bodu v konvexních mnohoúhelnících. Lze jej však zobecnit i pro nekonvexní. Obecně si lze metodu představit tak, že z libovolného bodu vedeme polopřímky a hodnotíme průsečíky přímky s hranami polygonu.

Označme si určovaný bod q. Z tohoto bodu je veden paprsek r (ray). Pokud si průsečík přímky r s hranami polygonu P označíme jako k, pak platí:

1. Pokud je k liché: Bod náleží polygonu P. ($q \in P$)
2. Pokud je k sudé: Bod nenáleží polygonu P. ($q \notin P$)



Obrázek 2: Princip Ray Crossing Algorithm [zdroj: 2]

3.1.1 Problematické situace

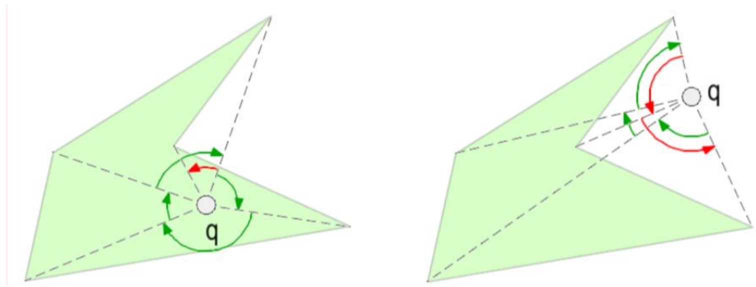
Při použití Ray Crossing Algorithm může nastat několik problematických situací, které nelze opomenout. Tímto problémem jsou singularity. K singularitě v této metodě může dojít tehdy, pokud bod leží na hraně polygonu či pokud je bod totožný s některým z vrcholů polygonu. Z tohoto důvodu se využívá upravená varianta Ray Crossing Algorithm, kdy je provedena redukce souřadnic bodů. V této úloze byla v části kódu, kde se počítá za pomoci paprskového algoritmu, použita již vytvořená funkce getPointLinePosition. Pokud bod náležel hraně či se nalézal ve vrcholu, byla návratová hodnota funkce rovna 1 a cyklus byl ukončen s vyhodnocením, že se bod v daném polygonu nachází.

3.1.2 Implementace metody

1. Nastavení počtu průsečíků rovno nule: $inters = 0$
2. Redukce souřadnic x všech bodů polygonu vůči x-ové souřadnici bodu q: $x'_i = x_i - x_q$
3. Redukce souřadnic y všech bodů polygonu vůči y-ové souřadnici bodu q: $y'_i = y_i - y_q$
4. Volba podmínky: $if(y'_i > 0)\&\&(y'_{i-1} <= 0) \| (y'_{i-1} > 0)\&\&(y'_i <= 0)$
5. Při splnění podmínky: $x'_m = (x'_i y'_{i-1} - x'_{i-1} y'_i)/(y'_i - y'_{i-1})$
6. Pokud $x'_m > 0$, zvýšení počtu průsečíků o jeden: $inters = inters + 1$
7. Určení zda počet průsečíků sudý či lichý: $if(inters\%2) = 0$, pak: $q \in P$ - počet průsečíků je sudý
8. V opačném případě: $q \notin P$

3.2 Winding Number Algorithm

Metoda ovíjení, či známá jako Winding Number Algorithm, je často používána pro určení pozice bodu vůči nekonvexnímu mnohoúhelníku. Algoritmus si lze představit tak, že se z určovaného bodu otáčíme postupně ke každému bodu polygonu a pokud se otáčíme po směru hodinových ručiček, úhel sčítáme, v opačném případě odčítáme. Pokud je výsledný úhel roven 2π, lze říci, že bod náleží polygonu. V opačném případě nenáleží.



Obrázek 3: Princip Winding Number Algorithm [zdroj: 3]

Při této metodě je zapotřebí si implementovat Winding Number Ω. Pro Ω platí, že je rovna sumě všech rotací ω proti směru hodinových ručiček, které průvodič opíše nad všemi body: $\Omega = \frac{1}{2\pi} \sum_{i=1}^{n} \omega_i^2$
Orientace úhlů je dána:

1. Pokud je úhel $\sphericalangle p_i, q, p_{i+1}$ orientován ve směru hodinových ručiček, pak $\omega_i > 0$
2. Pokud je úhel $\sphericalangle p_i, q, p_{i+1}$ orientován proti směru hodinových ručiček, pak $\omega_i < 0$

V závislosti na výsledné hodnotě Ω lze vyvodit následující závěry:

1. Pokud je $\Omega = 1$, pak platí $q \in P$
2. Pokud je $\Omega = 0$, pak platí $q \notin P$

3.2.1 Problematické situace

Pro Winding Number Algorithm je snadnější řešení singulárních případů. K těm dochází pouze v případě, že $q \approx p_i$, tedy v případě že bod leží na hraně. Z toho důvodu je zapotřebí v kódu ošetřit blízkost obou hodnot. Pokud tomu nastane, vyhodnotí se funkce tak, že bod leží v obou polygonech.

3.2.2 Implementace metody

1. Nastavení výchozího úhlu ω rovno 0, volba tolerance $\epsilon : \omega = 0, \epsilon = 1e - 10$
2. Určení orientace o_i bodu q ke straně p_i, p_{i+1}
3. Určení úhlu: $\omega_i = \sphericalangle p_i, q, p_{i+1}$
4. Volba podmínky - pokud pod vlevo: $\omega = \omega + \omega_i$
5. V opačném případě: $\omega = \omega - \omega_i$
6. Volba podmínky - pokud rozdíl: $(\omega - 2\pi) < \epsilon$, pak platí: $q \in P$
7. V opačném případě: $q \notin P$

4 Vstupní data

Mezi vstupní data patří analyzovaný bod q a textový soubor s body jednotlivých polygonů. Bod q je vkládán interaktivně po spuštění aplikace v grafickém okně.

- Analyzovaný bod q
  Bod q je vkládán uživatelem interaktivně zmáčknutím levého tlačítka v grafickém okně aplikace.
- Souvislá mapa polygonů
  Vstupní soubor je ve formátu txt a obsahuje body jednotlivých polygonů.

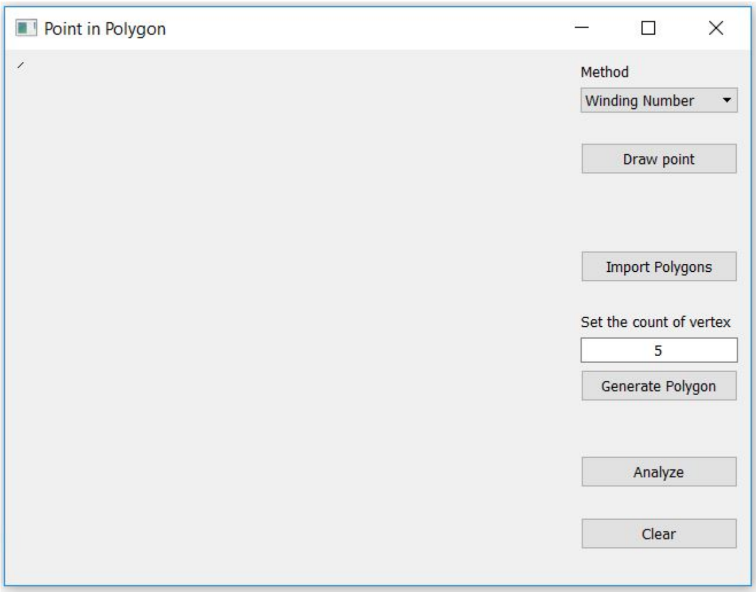
Struktura vstupních dat: [číslo bodu, souřadnice X, souřadnice Y]

Vstupní data musí být seřazená, tedy každý polygon v souboru musí začínat bodem jedna a končit n-tým bodem. Ve vstupních datech je nový polygon dán číslem bodu jedna. Jednotlivé body se sekvenčně ukládají do proměnné QPointF a následně polygonu do QPolygonF. Všechny polygony se sloučí do proměnné QPolygonF.

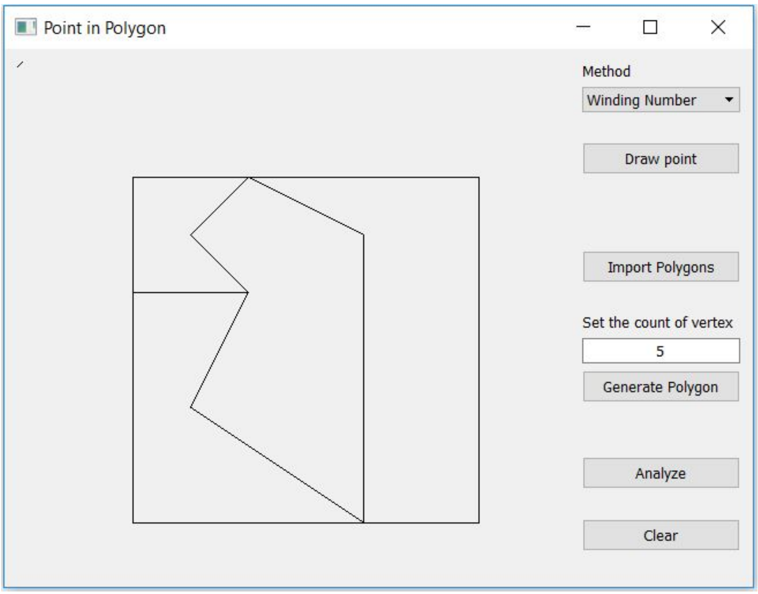
5 Výstupní data

Hlavním výstupem této úlohy je grafická aplikace, do níž je možné nahrát textový soubor s polygony a následně si body a mnohoúhelníky zobrazit. Zároveň je v aplikaci možné určit polohu bodu pomocí dvou metod. Dalším z výstupů jsou červeně zvýrazněné polygony v grafickém okně, které znázorňují polygon, kterému analyzovaný bod q náleží.
+ pokud se podaří tvorba nekonvexních polygonů

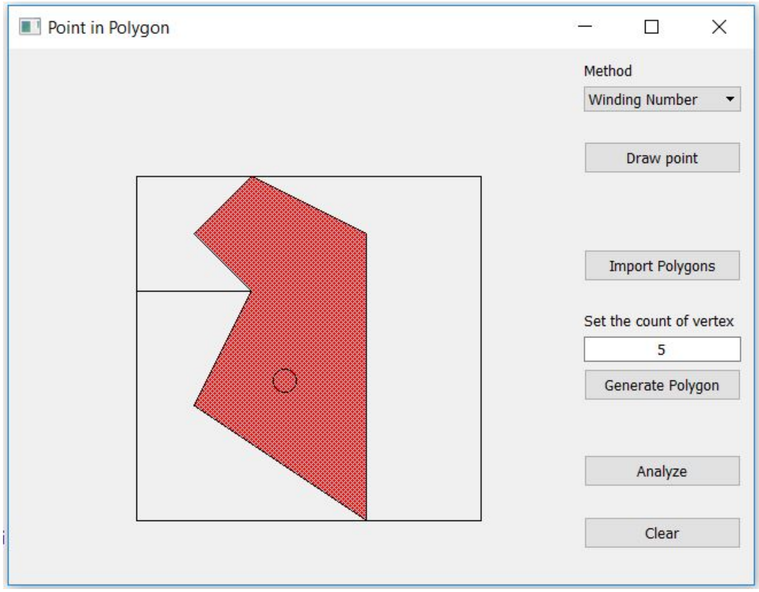
6 Aplikace



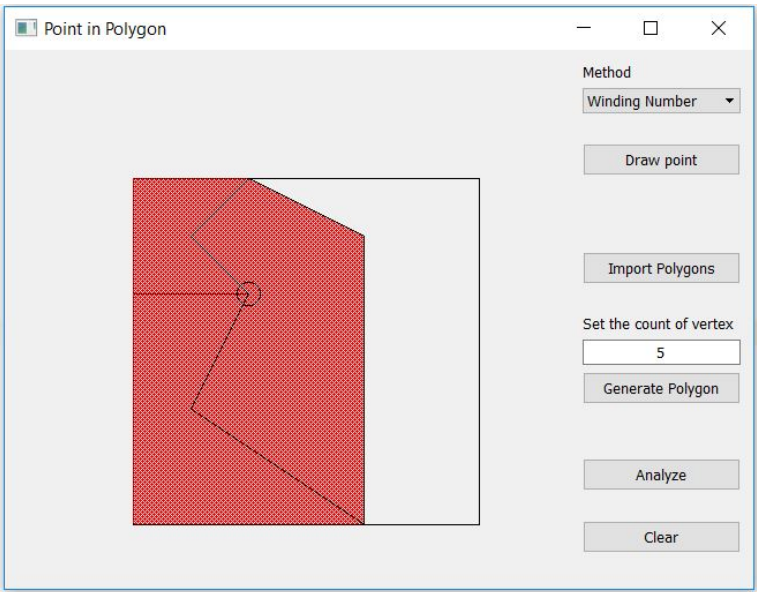
Obrázek 4: Okno aplikace po spuštění kódu



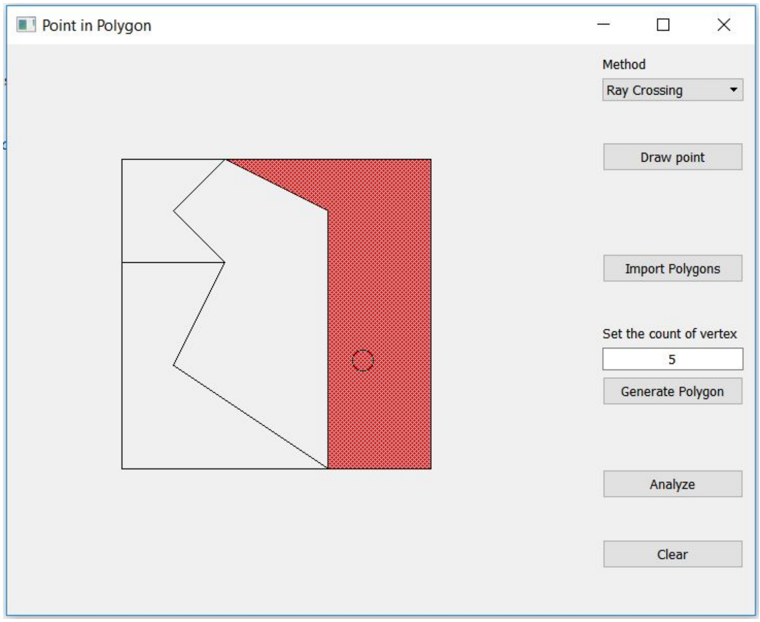
Obrázek 5: Naimportované souřadnice a jejich zobrazení v prostředí



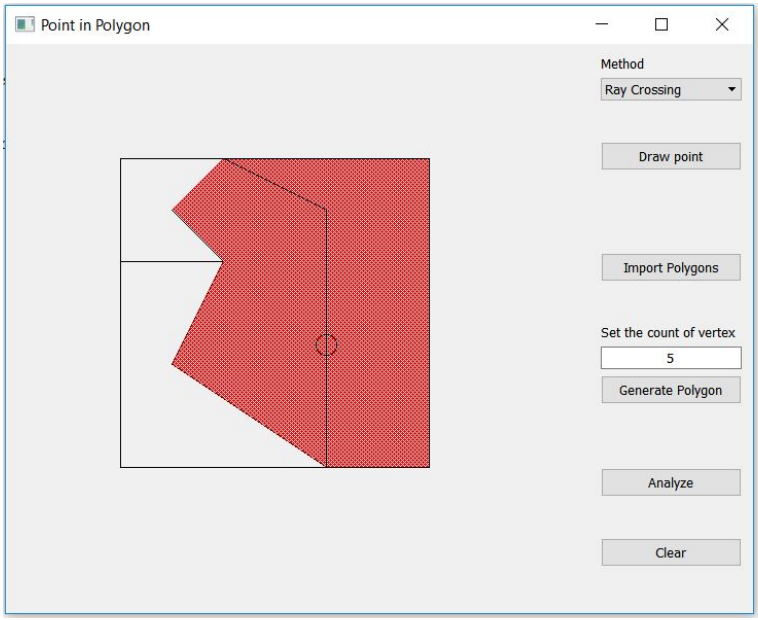
Obrázek 6: Analýza polohy bodu za pomoci Winding Number Algorithm



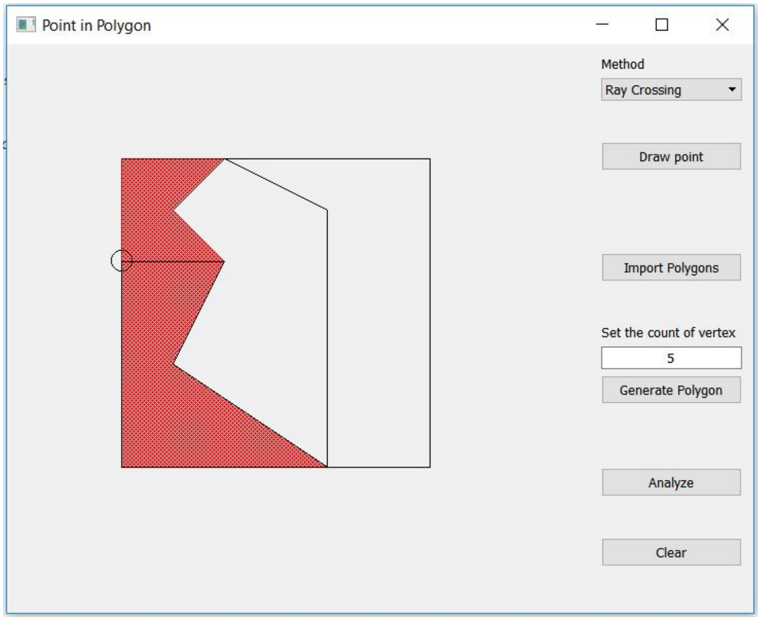
Obrázek 7: Vyhodnocení Winding Number Algorithm při bodu totožném s vrcholem



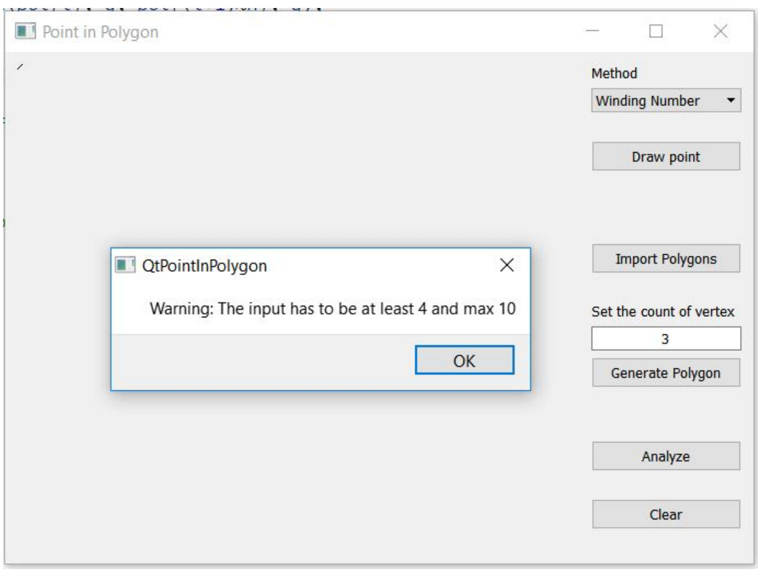
Obrázek 8: Analýza polohy bodu za pomoci Ray Crossing Algorithm



Obrázek 9: Analýza polohy bodu na hraně za pomoci Ray Crossing Algorithm



Obrázek 10: Analýza polohy bodu ve vrcholu za pomoci Ray Crossing Algorithm



Obrázek 11: Upozornění při vkládání nedefinované hodnoty

7 Dokumentace

7.1 Třídy

7.1.1 Algorithms

Třída Algorithms obsahuje celkem 6 metod. Metody jsou určeny pro výpočty použitých algoritmů.

getPositionRay
Jedná se o funkci, která vypočítá geometrickou polohu bodu vůči polygonu za pomoci paprskového algoritmu (Ray Crossing Algorithm). Do funkce vstupuje určovaný bod q a polygon, vůči kterému je poloha zjišťována. V případě, že výsledná hodnota funkce je rovna *1*, znamená to, že bod leží v polygonu, případně na hraně či vrcholu polygonu. V ostatních případech leží bod mimo polygon.

getPositionWinding
Tato funkce určí polohu bodu vůči polygonu metodou ovíjení (Winding Number Algorithm). Vstupem je určovaný bod q a polygon, vůči kterému je poloha určována. Výstupem může být některá z následujících hodnot. Pokud je výstupem celé číslo rovno *1*, znamená to, že bod leží buďto ve vrcholu polygonu či na hraně polygonu. V případě, že výstupem je číslo *0*, bod leží mimo polygon. Pokud nenastane žádná z uvedených možností, výstupem je hodnota *-1*. Ve funkci je mimo jiné ošetřena singularita, která může nastat v případě, že bod leží na hraně či ve vrcholu.

getPointLinePosition
Tato funkce má za úkol určit pozici bodu vůči linii. Do funkce vstupuje určovaný bod q a dva body přímky, ve funkci značené a a b. Ze znalosti dvou bodů přímky jsme schopni určit determinant, jehož hodnotu následně porovnáváme se zvolenou minimální hodnotou, v tomto kódu označenou jako *eps*. Pokud je hodnota determinantu větší než *eps*, funkce vrací hodnotu *1* a znamená to, že bod leží v levé polorovině. Pokud je hodnota determinantu menší než *eps*, funkce vrací hodnotu *0* a znamená to, že bod leží v pravé polorovině. Pokud nenastane ani jeden z výše uvedených závěrů, znamená to, že bod leží na hraně a výstupem je hodnota *-1*.

get2LinesAngle
V této funkci je počítán úhel mezi dvěma hranami. Vstupem jsou 4 body typu QPointF. Nejprve je vypočten skalární součin vektorů a velikost obou vektorů. Úhel je poté vypočten jako arcus cosinus poměru skalárního součinu a součinu obou velikostí. Defaultně se v prostředí počítá v radiánech, je tedy zapotřebí na toto brát ohled, z toho důvodu máme veškeré vypočtené hodnoty úhlů převáděny na stupně.

7.1.2 Draw

Třída Draw obsahuje několik metod.

paintEvent

Tato metoda slouží k vykreslení bodu q. Zároveň jsou díky této metodě vykresleny naimportované polygony. V metodě jsou také implementovány způsoby vykreslení polygonu, pokud bod leží uvnitř, případně na hraně či ve vrcholu.

mousePressEvent

V této funkci je po kliknutí v okně vytvořen testovaný bod q.

setDrawPoint

Po spuštění aplikace je nutné stisknout tlačítko Draw Point pro vykreslení bodu. V případě, že se stiskne tlačítko opětovně, je přivolána tato funkce a není možné bod vykreslit. V kódu, který byl psán na cvičení, sloužilo překlikávání k ruční tvorbě polygonu.

clearCanvas

Touto metodou je překleseno okno aplikace.

importPolygons

Tato funkce slouží k importování textového souboru, ve kterém se nachází body polygonů.

generatePolygon

V této metodě je hlavním cílem vytvoření nekonvexního polygonu. Do tvorby se může zapojit i uživatel volbou počtu vrcholů polygonu. V případě, že není dodržen limit vstupních hodnot, zobrazí se Message Window, které upozorňuje na chybné vložení dat.

7.1.3 Widget

on_pushButton_3_clicked

Díky této funkci se vyčistí okno aplikace. Tlačítko lze považovat za užitečné, při zkoumání většího množství dat by mohl vznikat zmatek. V aplikaci je funkce ukryta pod tlačítkem Clear.

on_pushButton_clicked

Touto funkcí se volá metoda setDrawPoint, která je implementována v Draw. V aplikace se funkce nalézá pod tlačítkem Draw Points.

on_pushButton_2_clicked

Tato metoda se nachází pod tlačítkem Analyze. Zobrazí nám, zda se bod nachází v polygonu, případně na hraně či vrcholu.

on_importPolygons_clicked

Funkce se ukrývá pod tlačítkem Import Polygons. V těle funkce je implementován způsob načtení dat.

on_generatePolygon_clicked

Při stisknutí tlačítka Generate Polygon je touto funkcí přivolána metoda generatePolygon, která vykreslí dle zadaných parametrů polygon.

7.2 Popis bonusových úloh

7.2.1 Ošetření singulárního případu u Winding Number Algorithm, kdy bod leží na hraně polygonu

Uvnitř funkce je porovnávána velikost úhlu ϕ. V případě, že rozdíl hodnoty ϕ a přímého úhlu je menší než zvolená mez (eps), znamená to, že bod leží na hranici a návratová hodnota je rovna *1*.
musíme ošetřit tu vzdálenost!!

7.2.2 Ošetření singulárního případu u obou metod, kdy je bod totožný s vrcholem dvou a více polygonů

Ve funkci getPositionWinding je na začátku kódu ošetřeno, zda vyhledávaný bod q není totožný s některým z vrcholů. V případě, že ano, návratová hodnota je *-1* a úhel ani umístění bodu vůči linii není počítáno. Následně je volena podmínka, pokud je návratová hodnota rovna *-1*, funkce vrací číslo *1* a bod se nachází ve vrcholu některých z polygonů. Ve funkci getPositionRay je využita hotová funkce pro určení pozice bodu vůči linii getPointLinePosition.

7.2.3 Zvýraznění všech polygonů pro oba výše uvedené singulární případy

Pokud je v kódu vyhodnoceno, že bod leží v některém z polygonů, návratová hodnota je rovna *1*. Stejnou návratovou hodnotu mají i případy, kdy bod leží na hraně či ve vrcholu. Stačilo tedy jen napsat for-cyklus, který ověří všechny polygony a zvýrazní ty, pro něž je návratová hodnota rovna *1*.

7.2.4 Tvorba nekonvexního polygonu

Pro tvorbu polygonu je nutné zvolit počet vrcholů. Defaultně je hodnota nastavena na číslo 5. Ve funkci pro generování polygonů je na začátku uvedena podmínka, že vkládané číslo nesmí být menší než 4, neboť pokud bychom vložili tři body, vznikne trojúhelník a trojúhelník je vždy konvexní. V případě, že uživatel zadá číslo 3 nebo text, zobrazí se okno, které upozorňuje na neplatný vstup. Při generování souřadnic byla původně použita knihovna RandomGenerator, ovšem tato knihovna je podporována pouze vyššími verzemi prostředí. Z toho důvodu je použita existující funkce rand a modulo 500, abychom byli schopni určitým způsobem regulovat oblast, ve které se bod zobrazí.

8 Závěr

Autoři splnili povinné zadání a některé bonusové úkoly. Vyhotovený program načte soubor polygonů a podle umístění značky stisknutím tlačítka myši spočítá, zda značka leží v polygonu či nikoli. Autoři hodnotí úlohu velmi pozitivně. Za velký přínos považují práci v programovacím jazyce C++. Zároveň mají možnost naučit se spolupracovat v prostředí gitHub, se kterým ani jeden nemá příliš mnoho zkušeností. V neposlední řadě je psaní dokumentace v prostředí LaTeX, díky čemuž je vzhled práce na poměrně dobré úrovni. Úlohu považují autoři za poměrně náročnou vzhledem k jejich zkušenostem. ?? PSÁT TAM TOHLE ČI NE?

Ve funkci getPositionWinding jsou porovnány různé výsledky. Pokud není splněna při porovnání úhlu žádná z podmínek, návratová hodnota je rovna -1 a tato část je nadepsána komentářem "something else?". Po zamyšlení, kdy by taková situace mohla nastat, nás napadla akorát taková možnost, kdy si definujeme prostor, ve kterém se polygony vykreslují a analyzovaný bod q je umístěn mimo tento prostor (Out of range). V praxi by se mohlo jednat o takový případ, kdy budeme mít na vstupu zeměpisnou mapu Evropy a bod q by vyjadřoval místo, které chceme navštívit a zvýrazněný polygon by znázornil zemi, v níž se bod nalézá. V případě, že bychom zvolili místo v oceánu, nebyla by funkce pro tuto oblast definována a výstupem by byla hodnota -1.
(LZE TO TAKTO NAPSAT? NEJSEM SI JISTÁ JESTLI TO ÚPLNĚ DOBŘE CHÁPU, ALE U KOMENTÁŘE JSOU OTAZNÍKY, TAK ASI NENÍ ÚPLNĚ ŠPATNÝ SE NAD TÍM ZAMYSLET :-)

8.1 Náměty na vylepšení

8.1.1 Zobrazení dat

Jelikož program má předdefinovanou velikost vykreslovacího okna, načtená data mimo toho okno se nezobrazí. Tento problém by se dal řešit transformací velikosti okna.

8.1.2 Souřadný systém

Vykreslovací okno vývojového prostředí Qt má souřadný systém s definovaný počátkem v levém horním rohu, kladnou osou X vpravo a kladnou osou Y vzhůru. Souřadný systém v Qt se liší od používaných souřadných systémů (matematických i kartografických), proto se importovaná data nezobrazí, jak by uživatel očekával. Řešením by byla transformace bodů.

8.1.3 Určovaný bod q

Při určování pozice bodu q se vždy řeší jeho pozice ve středu. Při testování jednotlivých funkcí a podmínek bylo poměrně náročně trefit vrchol či hranu. Vhodné by bylo, aby byla porovnávána i pozice bodů ležících no hlavních poloosách. Nejideálnější řešení by bylo, pokud bychom byli schopni porovnávat celkovou plochu, kterou bod zaujímá, i pro uživatele by to bylo jednodušší pro představu. Přesto by zde mohl být v některých případech problém s nižší přesností. Zároveň bychom si značně ulehčili práci v případě analýzy, zda se bod nachází na hraně či ve vrcholu. V tom případě by totiž byly analyzovány i krajní body a ve většině případů by problém singularity byl opomenut.

8.1.4 Datový typ funkcí ve třídě Algorithm

Při vyhodnocování jednotlivých funkcí je pro nás podstatné, zda je návratová hodnota rovna jedné. Ostatní případy nás nezajímají. Po zamyšlení by zřejmě bylo vhodné, pokud bychom datový typ zvolili bool místo int. Přesto pokud bychom chtěli brát v úvahu i rozšířené situace, kdy by se bod dostal mimo dosah, je vhodnější použít datový typ int a v případě, že by tak nastalo, přes funkci QMessageBox zobrazit zprávu, že bod se nachází Out of Range.

9 Reference

1. Presentation about convex and concave polygons [online][cit. 21.10.2018].
   Dostupné z: https://slideplayer.com/slide/6161031/

2. Introducing Wherewolf - A serverless boundary service from WNYC [online][cit. 21.10.2018].
   Dostupné z: https://source.opennews.org/articles/introducing-wherewolf/

3. BAYER, Tomáš. Geometrické vyhledávání [online][cit. 21.10.2018].
   Dostupné z: https://web.natur.cuni.cz/ bayertom/images/courses/Adk/adk3.pdf