

155ADKG Geometrické vyhl'adávanie bodu

Martin Hulín, Lukáš Kettner

V Prahe 14.10.2019

1. Zadanie

Vytvorte aplikáciu s grafickým rozhraním, ktorá určí polohu bodu voči polygórovej mape.

Vstup do aplikácie : súvislá polygórová mapa n polygónov $\{P_1, \dots, P_n\}$, analyzovaný bod q .

Výstup aplikácie : $P_i, q \in P_i$

Nad polygórovou mapou implementujte nasledujúce algoritmy pre geometrické vyhľadávanie:

- Ray Crossing Algorithm (varianta s posunom ťažiska polygónu)
- Winding Number Algorithm

Nájdenny polygón obsahujúci zadaný bod q graficky zvýraznite vhodným spôsobom. Grafické rozhranie vytvorte s použitím frameworku Qt.

Pre generovanie nekonvexných polygónov môžete navrhnúť vlastný algoritmus alebo použiť existujúce geografické dáta.

Polygóny budú načítane z textového súboru vo Vami zvolenom formáte. Pre dátovú reprezentáciu jednotlivých modelov použite špagetový model.

1.1 Bonusové úlohy

V rámci úlohy sú vypracované tieto bonusové úlohy

- Ošetrovanie singulárneho prípadu pri Winding Number Algorithm
- Ošetrovanie singulárneho prípadu pri oboch algoritmoch
- Zvýraznenie všetkých polygónov pre oba uvedené singulárne prípady

2. Popis a rozbor problému

Máme v rovine daný bod q a polygóny P_j ktoré sú tvorené vrcholmi p_i a hranami σ_{p_i} .

Polygóny vytvárajú súvislú polygórovú mapu. Našou úlohou bolo určiť polohu bodu q voči polygórovej mape.

Úlohu sme riešili prevedením na riešenie vzájomnej polohy bodu a jedného polygónu.

Postupne sme určovali polohu bodu voči jednotlivým polygónom, ktoré tvorili polygórovú mapu. Pokiaľ sa bod q nachádzal vnútri polygónu ukončili sme analýzu. Následne bolo rozhodnuté o vzájomnej polohe v rámci celej mapy.

Varianty výsledkov analýzy :

- bod q leží vnútri polygónu
- bod q leží mimo polygónu
- bod q leží na hrane polygónu
- bod q je totožný s vrcholom polygónu

Pre učenie polohy bodu voči polygónom boli použité algoritmy Ray Crossing Algorithm a Winding Number Algorithm.

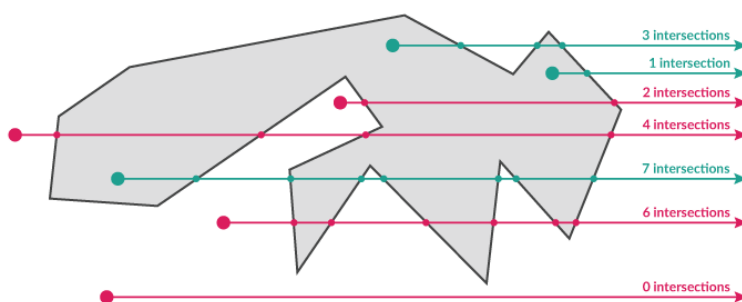
3. Popis použitých algoritmov

3.1 Ray Crossing Algorithm

Ray Crossing Algorithm je takzvaný lúčový algoritmus. Jeho hlavným účelom je určenie polohy bodu v konvexných mnohoúhľaníkoch. V prípade, že algoritmus zobecníme, je ho možné využiť aj pri nekonvexných mnohoúhľaníkoch. Algoritmus si môžeme predstaviť nasledovne. Z ľubovoľného bodu vedieme polpriamky a hodnotíme priesečníky priamky s hranami polygónu.

Určovaný bod označíme ako q . Z tohto bodu je vedený lúč r (ray). Priesečníky priamky r s hranami polygónu P označíme k . Potom platí :

1. Ak k je nepárne – bod q patrí polygónu P ($q \in P$).
2. Ak k je párne – bod q nepatrí polygónu P ($q \notin P$).



Obrázok 1: Ray Crossing Algorithm [2]

3.1.1 Problematické situácie

Pri použití Ray Crossing Algorithm môžu nastať problematické situácie – singularity. K singularite dochádza ak bod leží na hrane polygónu, prípadne ak je bod totožný s vrcholom polygónu. V takomto prípade je počet priesečníkov s hranou polygónu párny, aj napriek tomu, že bod q leží vnútri polygónu.

Táto situácia sa ošetrí zavedením miestneho súradnicového systému.

1. Počiatok sústavy je vložený do bodu q
2. Os x' je rovnobežná s osou x
3. Os y' je kolmá na os x'

Riešenie je obmedzené len na hrany polygónu, ktorých jeden bod leží nad osou x' a druhý pod touto osou.

Ošetrovanie prípadu kedy je bod q totožný s vrcholom polygónu alebo leží na hrane polygónu je vykonané pomocou určenia dĺžky hrany polygónu a následne súčtom vzdialeností medzi

bodom q a vrcholmi hrany. Pokiaľ sú tieto vzdialenosti identické, bod leží na hrane polygónu alebo je totožný s jeho vrcholom.

3.1.2 Implementácia metódy

1. *Inicializácia* $k = 0$
2. *Opakuj*: Pre $\forall p_i \in P$
3. *Redukcia súradníc* X a Y na bod q
4. $\text{if}(y_{i-1}^{''} > 0) \wedge (y_{i-1}^{''} \leq 0) \parallel (y_{i-1}^{''} > 0) \wedge (y_{i-1}^{''} \leq 0)$
 $x_m^{''} = (x_{i-1}^{''}y_{i-1}^{''} - x_{i-1}^{''}y_{i-1}^{''}) / (y_{i-1}^{''} - y_{i-1}^{''})$
 $\text{if}(x_m^{''} > 0), k = k + 1$
5. $\text{if}(k \% 2) \neq 0$ potom $q \in P$
6. *inak* $q \notin P$

3.2 Winding Number Algorithm

Metóda Winding Number je používaná pre určenie pozície bodu voči nekonvexným mnohouholníkom. Algoritmus si môžeme predstaviť ako nasledujúcu situáciu. Postavíme sa na určovaný bod a otáčame sa postupne ku každému vrcholu polygónu. Pokiaľ sa otáčame po smere hodinových ručičiek, uhol sčítame, v opačnom prípade odčítame. Pokiaľ je výsledný uhol rovný 2π , bod na ktorom stojíme leží vnútri polygónu. Pri tejto metóde je potrebné určiť Winding Number Ω . Ω je rovné sume rotácií ω proti smeru hodinových ručičiek, ktoré prievodič opíše nad všetkými bodmi $\Omega = \frac{1}{2\pi} * \sum_{i=1}^n \omega_i^2$

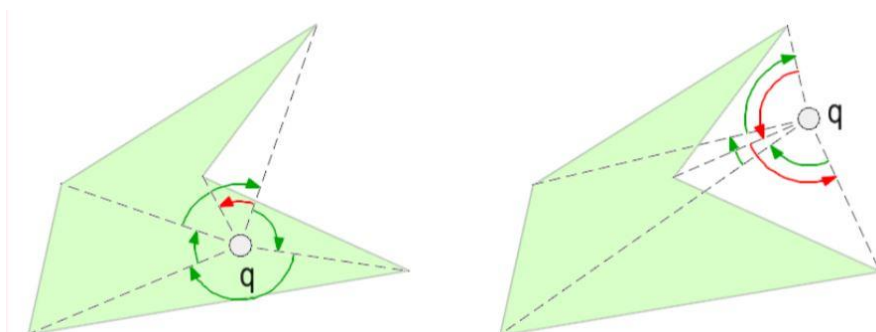
Určíme uhly $\omega_i(p_i, q, p_{i+1})$.

$$\cos \omega_i = \frac{\vec{u}_i * \vec{v}_i}{\|\vec{u}_i\| * \|\vec{v}_i\|}; \vec{u}_i = (q, p_i), \vec{v}_i = (q, p_{i+1})$$

Pokiaľ je uhol orientovaný v smere re hodinových ručičiek, nadobúda kladné znamienko. V protismere hodinových ručičiek záporné znamienko. Podľa sumy všetkých uhlov určíme polohu bodu q .

Ak je $\sum \omega_i =$

1. 360° bod $q \in P_j$
2. 0° bod $q \notin P$



Obrázok 2: Winding Number Algorithm [3]

3.2.1 Problematické situácie

Pre metódu Winding Number Algorithm je jednoduchšie riešiť singulárne prípady. K tým dochádza v prípade ak $q \approx p_i$, to je prípad ak poloha bodu je takmer zhodná s niektorým z vrcholov polygónu. Tu je potreba ošetriť blízkosť oboch hodnôt.

$$\sum \omega \neq 360^\circ \wedge \omega \neq 0^\circ \rightarrow q \in \sigma_{p_i}$$

3.2.2 Implementácia metódy

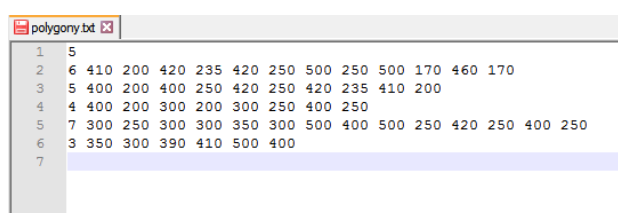
1. Inicializácia $\omega = 0$, tolerancia ε
2. Opakuj: Pre $\forall p_i, q, p_{i+1}$
3. Určenie orientácie o_i bodu q ku strane $p_i p_{i+1}$
4. Určenie uhlu $\omega_i = \angle p_i, p_{i+1}$
Podmienka if, pokiaľ bod vľavo $\omega = \omega + \omega_i$; else $\omega = \omega - \omega_i$
5. if $(|\omega - 2\pi| < \varepsilon)$ tak $q \in P$
6. else $q \notin P$

4. Vstupné dáta

4.1 Polygónová mapa

Polygónovú mapu do našej aplikácie importujeme pomocou textového súboru *.txt, ktorý obsahuje :

- počet polygónov
- počet bodov v prvom polygóne, súradnice x a y oddelené medzerou
- počet bodov v druhom polygóne, súradnice x a y oddelené medzerou
- počet bodov v treťom polygóne, súradnice x a y oddelené medzerou
- počet bodov v štvrtom polygóne, súradnice x a y oddelené medzerou
- počet bodov v piatom polygóne, súradnice x a y oddelené medzerou



```
polygony.txt
1 5
2 6 410 200 420 235 420 250 500 250 500 170 460 170
3 5 400 200 400 250 420 250 420 235 410 200
4 4 400 200 300 200 300 250 400 250
5 7 300 250 300 300 350 300 500 400 500 250 420 250 400 250
6 3 350 300 390 410 500 400
7
```

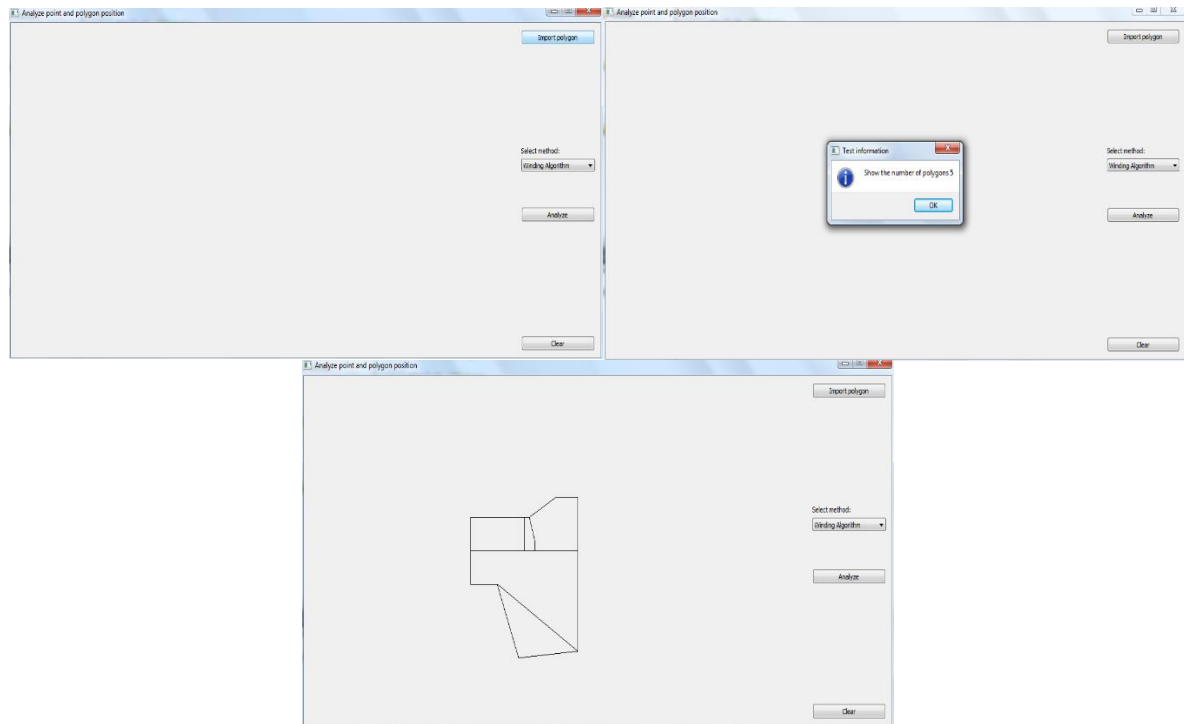
Obrázok 3: textový formát polygónovej mapy

4.2 Bod q

Súradnice bodu q sa do aplikácie zadávajú kliknutím myšou.

5. Ukážka vytvorenej aplikácie

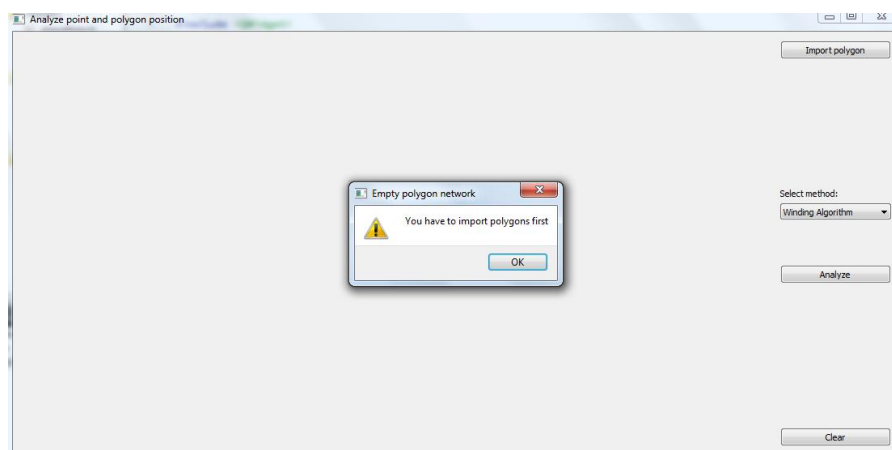
Načítanie polygónovej mapy sa uskutoční pomocou tlačidla Import polygon. Po zadaní *.txt súboru sa objaví dialógové okno, ktoré oznamuje počet nahratých polygónov. Po stlačení tlačidla „OK“ sa polygóny vykreslia.



Obrázok 4: načítanie polygónovej mapy

Následne kliknutím myšou do vykreslovacieho plátna zadáme polohu bodu q . Následne po stlačení tlačidla Analyze sa vykoná analýza polohy bodu q voči polygónovej mape.

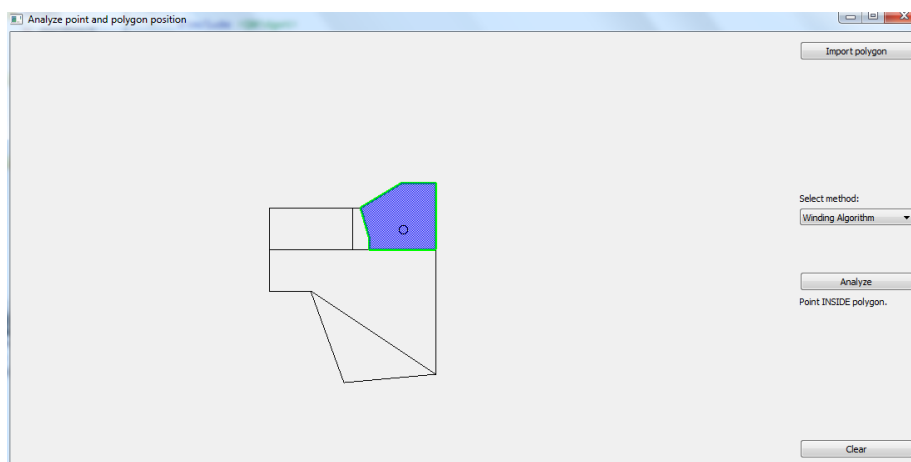
V prípade, že dôjde k neúspešnému nahratiu polygónovej mapy zobrazí sa varovné okno s varovnou hláškou.



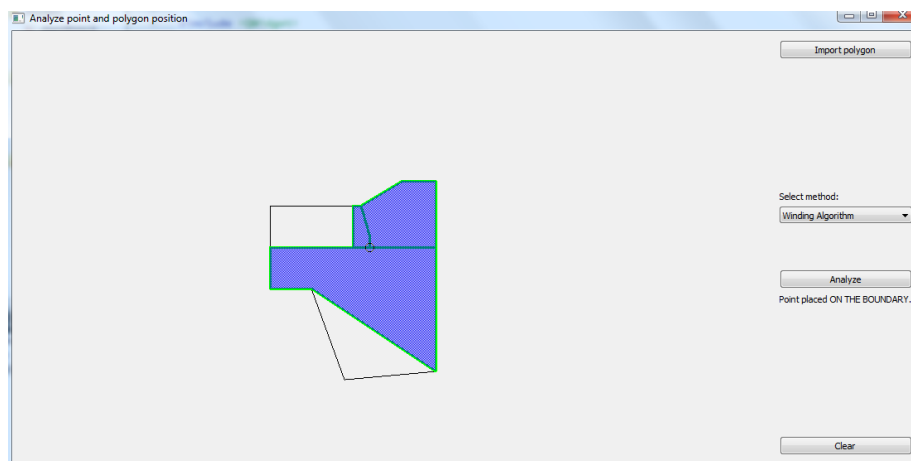
Obrázok 5: neúspešné načítanie polygónovej mapy - varovné okno

6. Výsledok analýzy

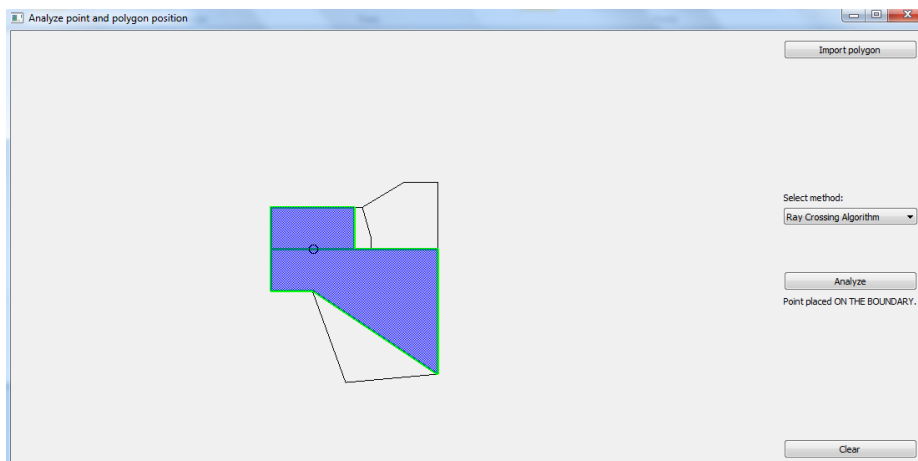
V grafickom rozhraní aplikácie sa po vykonaní analýzy zobrazí výsledok analýzy. Pokiaľ bod leží v polygóne, vysvieti a vyšráfuje sa polygón obsahujúci bod q . Pokiaľ bod leží v spoločnom vrchole viacerých polygónov vysvietia a vyšráfujú sa všetky polygóny pre ktoré je daný vrchol spoločný. Pokiaľ bod leží mimo polygónu v grafickom rozhraní aplikácie nenastane žiadna zmena. O polohe bodu takisto informuje hláška vypísaná v label pod tlačidlom Analyze.



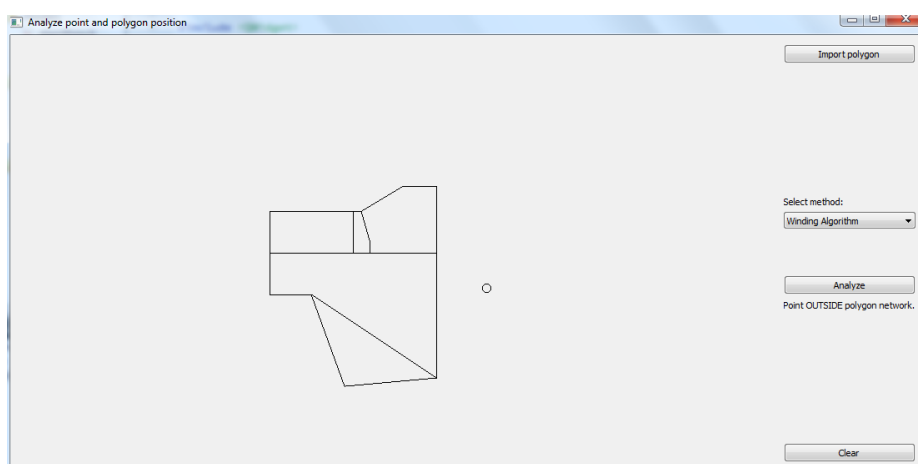
Obrázok 6: bod q vnútri polygónu



Obrázok 7: bod q v spoločnom vrchole viacerých polygónov



Obrázok 8: bod q na hrane



Obrázok 9: bod q mimo polygónu

7. Dokumentácia

1. Trieda Algorithms

V triede Algorithms sú staticky implementované výpočtové algoritmy pre určenie polohy bodu voči polygónu a taktiež ich pomocné metódy.

- metóda **positionPointPolygonRayCrossing**
 - slúži k určeniu polohy bodu prostredníctvom Ray Crossing algoritmu. V metóde je zakomponované rozhodovacie pravidlo, či sa bod nachádza na hrane polygónu alebo v niektorom z vrcholov. Jej návratový typ je integer.
 - na vstupe má : `QPoint q` – bod ktorého polohu určujeme, `std::vector<QPoint> pol` – polygon, voči ktorému určujeme polohu bodu q
 - výstupom je hodnota :
 - 1 – bod sa nachádza na hranici alebo vo vrchole polygónu
 - 0 – bod sa nachádza mimo polygón
 - 1 – bod sa nachádza vnútri polygónu

- metóda **positionPointPolygonWinding**
 - slúži k určeniu polohy bodu prostredníctvom Winding Number algoritmu. Jej návratový typ je integer.
 - na vstupe má : `QPoint q` – bod ktorého polohu určujeme,
`std::vector<QPoint> pol` – polygon, voči ktorému určujeme polohu bodu `q`
 - výstupom je hodnota :
-1 – bod sa nachádza na hranici alebo vo vrchole polygónu
0 – bod sa nachádza mimo polygón
1 – bod sa nachádza vnútri polygónu
- metóda **getAngle2Vectors**
 - je pomocnou metódou pre metódu **positionPointPolygonWinding**. Slúži k určeniu uhlu medzi dvoma priamkami. Jej návratovou hodnotou je double.
 - na vstupe má : súradnice bodov `p1`, `p2`, `p3`, `p4` určujúcich prvú a druhú priamku
 - výstupom je hodnota uhlu medzi priamkami
- metóda **getPointLinePosition**
 - je pomocnou metódou pre metódu **positionPointPolygonWinding**. Slúži na určenie polohy bodu voči priamke. Jej návratovou hodnotou je integer.
 - na vstupe má : súradnice určovaného bodu `q` , súradnice bodov priamky `p1 p2`
 - na výstupe hodnoty : -1 – bod sa nachádza na priamke
0 – bod sa nachádza vpravo od priamky
1 – bod sa nachádza vľavo od priamky

2. Trieda Draw

Trieda Draw slúži ku grafickému vykresleniu bodu `q`, polygónovej mapy, k výplni polygónov, ktorým bod `q` náleží. Trieda Draw obsahuje nasledujúce členské premenné a metódy.

Členské premenné

- **QPoint q** – súradnice bodu, ktorého polohu zisťujeme. Východiskové hodnoty sú nastavené v konštruktoze. Hodnoty sa menia stlačením tlačidla myši na vykreslovacom plátne.
- **std::vector<std::vector<QPoint>> polygons** – vektor ktorom sa ukladajú vektory obsahujúce body jednotlivých polygónov
- **std::vector<int> analyze_results_by_polygons** – vektor v ktorom sú uložené výsledky analýzy polohy bodu voči jednotlivým polygónom

Metódy

- **paintEvent**
 - slúži k vykresleniu zisťovaného bodu, nahraných polygónov a k vyfarbeniu polygónov ktorým náleží určovaný bod `q`. Návratovým typom je void.

- **void mousePressEvent**
 - slúži k vykresleniu bodu stlačením tlačidla myši, nastaveniu súradníc bodu q. Návrátovým typom je void.
 - **void clearCanvas**
 - slúži k vymazaniu obsahu vykreslovacieho plátna. Návrátovým typom je void.
 - **void importPolygon**
 - metóda slúži k importu polygónovej mapy z textového súboru *.txt, ich uloženiu do zoznamu polygónov polygons. Vstupom je cesta k súboru. Výstupom je správa obsahujúca počet nahratých polygónov.
 - **int getNumberOfPolygons**
 - slúži k určeniu počtu polygónov v Canvase. Návrátový typ je integer.
3. Trieda Widget
- Trieda Widget slúži pre komunikáciu s GUI. Všetky metódy ktoré obsahuje sú sloty umožňujúce vykonávať príkazy z grafického rozhrania aplikácie. Nemajú žiadne vstupné hodnoty, návrátovým typom je void.

Metódy

- **on_clearButton_clicked**
 - metóda reaguje na stlačenie tlačidla Clear. Volá metódu clearCanvas() z triedy Draw.
- **on_analyzeButton_clicked**
 - metóda reaguje na stlačenie tlačidla Analyze. V závislosti na zvolenom algoritme zavolá danú metódu z triedy Algorithms.
- **on_importPolygonButton_clicked**
 - metóda reaguje na stlačenie tlačidla Import polygon. Otvára dialógové okno a adresárom a ukladá cestu zvoleného súboru v *.txt formáte. Túto cestu posiela do metódy importPolygon triedy Draw.
- **showResultOfAnalysis**
 - pomocou tejto metódy vypisujeme výsledkok analýzy na label grafického okna. Vstupom je výsledok analýzy pre jeden polygón a premenná show_result, ktorá je typu bool. Je predávaná referenciou, informuje či chceme text labelu zmeniť alebo to nie je potrebné, pretože výsledok už bol zobrazený.

8. Záver

V priebehu vypracovania úlohy sme narazili na množstvo problémov, ktoré sme úspešne dokázali vyriešiť a tým sme obohatili naše vedomosti v oblasti programovania. Veľké problémy nám spôsobilo správne kódovanie textového súboru. Obsah súboru nebolo možné dlho načítať, s riešením sme strávili niekoľko hodín. Avšak podarilo sa nám tento problém vyriešiť, vykonali sme zmenu kódovania na UTF-8 s BOM (byte order mark). Výsledkom je funkčná aplikácia, ktorá umožňuje analyzovať polohu bodu voči polygónu z polygónovej mapy. V prípade, vylepšenia aplikácie by bolo zaujímavé implementovať možnosť nahratia *.shp súboru. V takej situácii by bolo potrebné vyriešiť vhodný vstupný formát súboru *.shp. Ďalším možným vylepšením je implementácia algoritmu pre automatické generovanie nekonvexných polygónov. Túto bonusovú úlohu sme z časových dôvodov do našej aplikácie nezakomponovali.

9. Zoznam použitej literatúry

[1] Qt Documentation Archives. *Qt Documentation Archives* [online]. 2019 [cit. 2019-10-13]. Dostupné z: <https://doc.qt.io/archives/>

[2] Introducing Wherewolf - A serverless boundary service from WNYC [online][cit. 2019-10-13]. Dostupné z: <https://source.opennews.org/articles/introducing-wherewolf/>

[3] BAYER, Tomáš. *Geometrické vyhledání* [online]. [cit. 2019-10-13]. Dostupné z: <https://web.natur.cuni.cz/~bayertom/images/courses/Adk/adk3.pdf>

[4] Cplusplus. *Cplusplus* [online]. [cit. 2019-10-13]. Dostupné z: <http://www.cplusplus.com>