

**České vysoké učení technické v Praze**  
Fakulta stavební



**155ADKG Algoritmy v digitální kartografii**

**Geometrické vyhľadávanie bodu**

Bc. Lukáš Kettner Bc. Martin Hulín  
23.10.2019

# Obsah

<b>1</b>	<b>Zadanie</b>	<b>2</b>
1.1	Bonusové úlohy . . . . .	2
<b>2</b>	<b>Popis a rozbor problému</b>	<b>2</b>
<b>3</b>	<b>Popis použitých algoritmov</b>	<b>3</b>
3.1	Ray Crossing Algorithm . . . . .	3
3.1.1	Problematické situácie . . . . .	3
3.1.2	Implementácia metódy . . . . .	4
3.2	Winding Number Algorithm . . . . .	5
3.2.1	Problematické situácie . . . . .	5
3.2.2	Implementácia metódy . . . . .	6
<b>4</b>	<b>Vstupné dáta</b>	<b>6</b>
4.1	Polygónová mapa . . . . .	6
4.2	Bod q . . . . .	7
<b>5</b>	<b>Ukážka vytvorenej aplikácie</b>	<b>7</b>
5.1	Generování bodů . . . . .	9
<b>6</b>	<b>Výsledok analýzy</b>	<b>9</b>
<b>7</b>	<b>Dokumentácia</b>	<b>11</b>
7.1	Trieda Algorithms . . . . .	11
7.1.1	Metódy . . . . .	11
7.2	Trieda Draw . . . . .	13
7.2.1	Členské premenné . . . . .	13
7.2.2	Metódy . . . . .	14
7.3	Trieda Widget . . . . .	15
7.3.1	Metódy . . . . .	15
7.4	Trieda SortByY . . . . .	15
7.5	Trieda SortByOmega . . . . .	15
<b>8</b>	<b>Záver</b>	<b>16</b>

# 1 Zadanie

Vytvorte aplikáciu s grafickým rozhraním, ktorá určí polohu bodu voči polygónovej mape. Vstup do aplikácie : súvislá polygónová mapa  $n$  polygónov  $\{P_1, \dots, P_n\}$  , analyzovaný bod  $q$ . Výstup aplikácie :  $P_i, q \in P_i$ .

Nad polygónovou 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

- Ošetrenie singulárneho prípadu pri Winding Number Algorithm. *+2b*
- Ošetrenie singulárneho prípadu pri oboch algoritmoch. *+2b*
- Zvýraznenie všetkých polygónov pre oba uvedené singulárne prípady. *+2b*
- Algoritmus pre automatické generovanie nekonvexných polygónov. *+5b*

# 2 Popis a rozbor problému

Problematikou úlohy je určenie vzájomného vzťahu polohy medzi bodom a danou oblasťou. Oblasť je tvorená polygónovou mapou. Pre bod sme zvolili označenie  $q$ . Následne analýzou pomocou dvoch typov algoritmov rozhodujeme o vzájomnej polohe bodu a polygónu. 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.

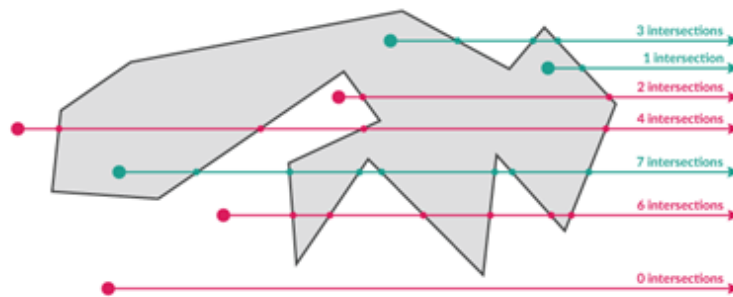
Pre učenie polohy bodu voči polygónom boli použité algoritmy Ray Crossing Algorithm a Winding Number Algorithm.

## 3 Popis použitých algoritmov

### 3.1 Ray Crossing Algorithm

Ray Crossing Algorithm je takzvaný lúčový algoritmus. Z bodu  $q$ , ktorého polohu určujeme, vedieme polopriamku a určíme počet priesečníkov s hranami polygónu. Priesečník priamky 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 ( $q \notin P$ ).



Obr. 1: Ray Crossing Algorithm [?]

#### 3.1.1 Problematické situácie

Problematickou situáciou je singularita. K singularite dôjde ak bod leží na hrane polygónu alebo je totožný s vrcholom polygónu. V takomto prípade je počet priesečníkov s hranou polygónu párny, aj napriek tomu, že bod  $q$  leží vnútri polygónu.

Ošetrenie singularity sa vykoná 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 sa v takomto prípade týka len takých hrán polygónu, ktorých jeden bod leží nad osou  $x'$  a druhý pod osou  $x'$ . Prípad kedy je bod  $q$  totožný s vrcholom polygónu alebo leží na hrane polygónu sa ošetrí 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. Určenie veľkosti polygónu  $\text{int } n = \text{pol.size}()$
3. Redukcia súradníc  $X$  a  $Y$  na bod  $q \rightarrow x', y'$ .
4. Určenie veľkosti polygónu  $\text{int } n = \text{pol.size}()$
5. Cyklus for pre všetky body polygónu
6. Podmienka v cykle pre započítanie priesečníkov

$$\text{if}(y'_i > 0) \& \& (y'_{i-1} \leq 0) \parallel \parallel (y'_{i-1} > 0) \& \& (y'_i \leq 0)$$

$$x'_m = (x'_i y'_{i-1} - x'_{i-1} y'_i) / (y'_i - y'_{i-1})$$

$$\text{if}(x'_m > 0), k = k + 1$$

7.  $\text{if}(k \% 2) \neq 0$  potom  $q \in P$ .
8. inak  $q \notin P$

## 3.2 Winding Number Algorithm

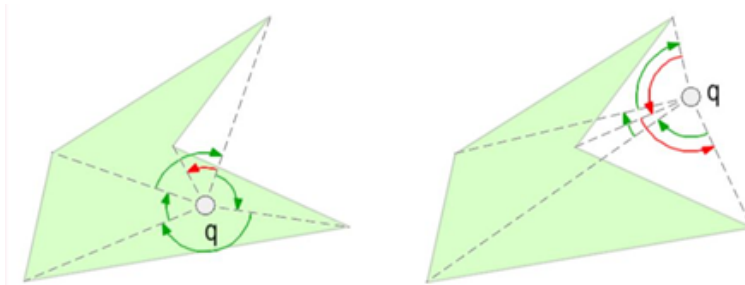
Metóda Winding Number je používaná pre určenie pozície bodu voči nekonvexným mnohouholníkom. Algoritmus si môžeme predstaviť ako nasledujúcu situáciu. Postavíme sa na určovaný bod a otáčame sa postupne ku každému vrcholu polygónu. Pokiaľ sa otáčame po smere hodinových ručičiek, uhol sčítame, v opačnom prípade odčítame. Pokiaľ je výsledný uhol rovný  $2\pi$ , bod na ktorom stojíme leží vnútri polygónu. Pri tejto metóde je potrebné určiť Winding Number  $\Omega$ .  $\Omega$  je rovné sume rotácií  $\omega$  proti smeru hodinových ručičiek, ktoré prievodič opíše nad všetkými bodmi  $\Omega = \sum_{i=1}^n * \omega_i$ . 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 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.  $360^\circ$  bod  $q \in P_j$
2.  $0^\circ$  bod  $q \notin P$
3. inak bod leží na hranici polygónu alebo vo vrchole polygónu



Obr. 2: Winding Number Algorithm [?]

### 3.2.1 Problematické situácie

K singulárnym prípadom dôjde pri Winding Algoritmu ak je poloha určovaného bodu zhodná s polohou vrcholu polygónu. Je potreba ošetriť blízkosť hodnôt.

$$\sum \omega \neq 360^\circ \omega \neq 0^\circ q \in \sigma_{pi}$$

### 3.2.2 Implementácia metódy

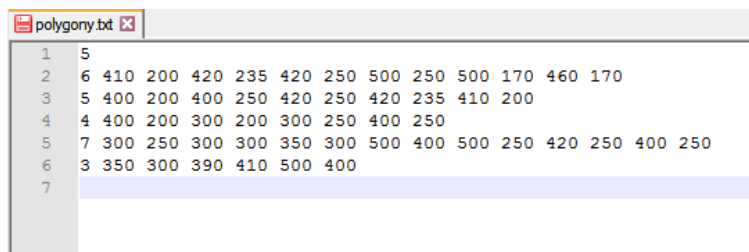
1. Inicializácia  $\omega = 0$ , tolerancia  $\epsilon$
2. Určenie veľkosti polygónu `int n = pol.size()`
3. Cyklus for - prechádzame všetky body polygónu
4. Určenie veľkosti uhlu  $\omega_i = \angle p_i, p_{i+1}$   
 Podmienka `if`, pokiaľ bod vľavo  $\omega = \omega + \omega_i$ ; `else`  $\omega = \omega - \omega_i$
5. `if` ( $|\omega - 2\pi| < \epsilon$ ) 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



```

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

```

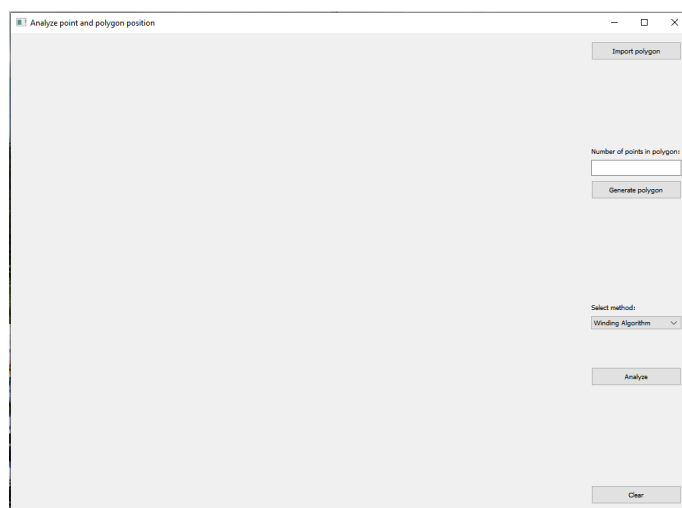
Obr. 3: Textový formát polygónovej mapy

## 4.2 Bod $q$

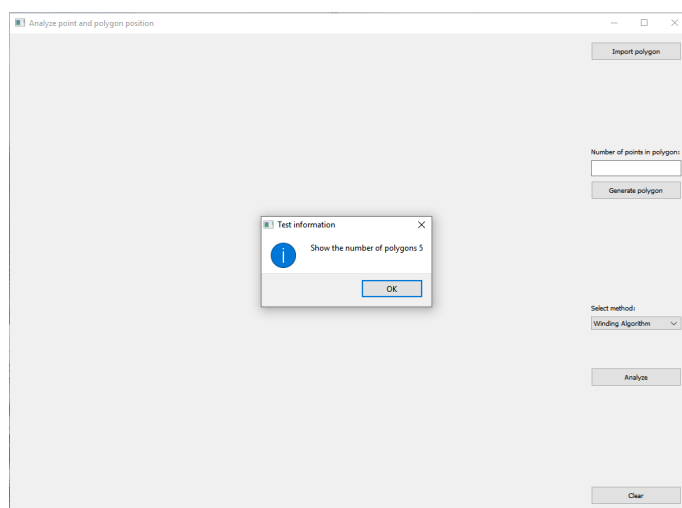
Súradnice bodu  $q$  sa do aplikácie zadávajú kliknutím myšou.

## 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.

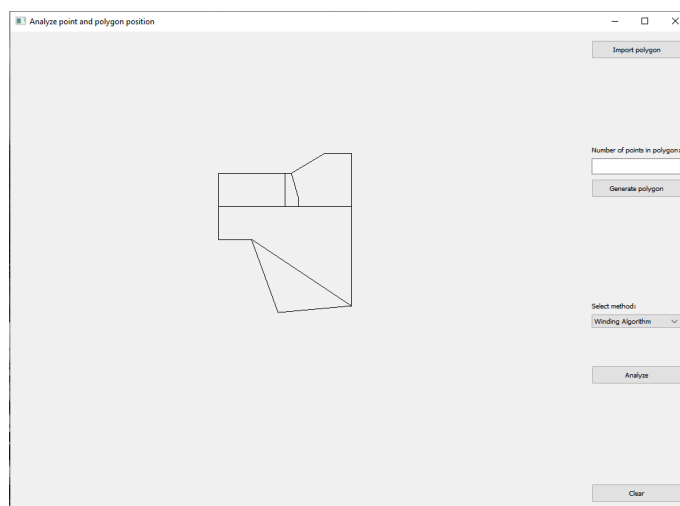


Obr. 4: Uvodné okno





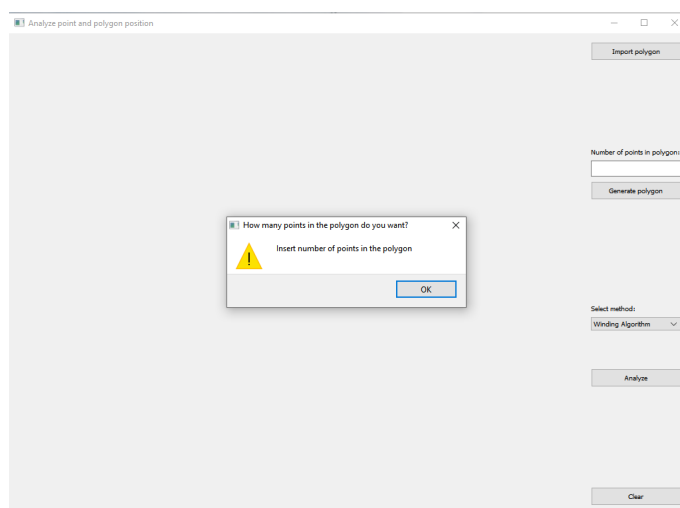
Obr. 5: Informace o počtě načítaných polygónov



Obr. 6: Zobrazenie 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.

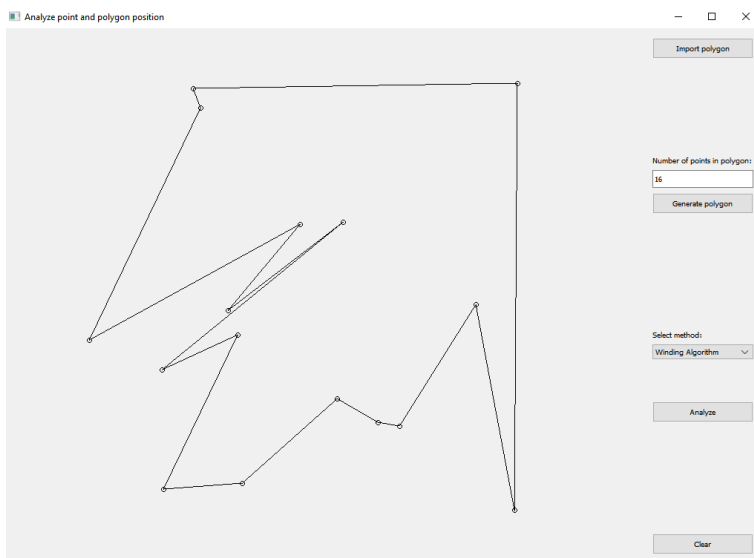


Obr. 7: Neúspešné načítanie polygónovej mapy - varovné okno

## 5.1 Generování bodů

Aplikácia umožňuje generovanie nekonvexného polygónu. Vstupom pre toto generovanie je požadovaný počet bodov v generovanom polygóne.

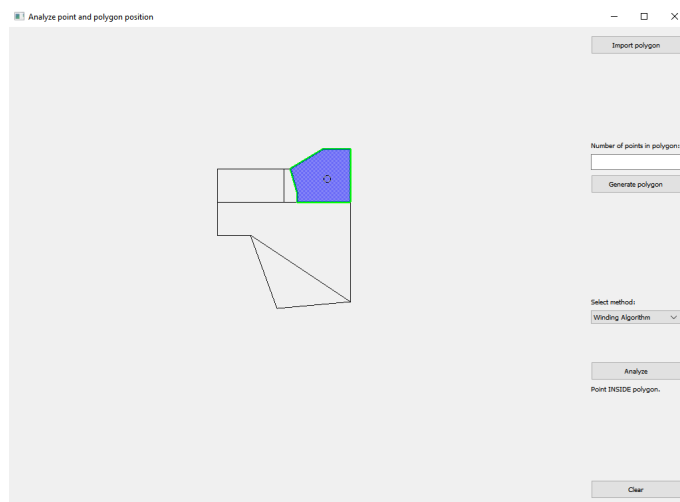
Generovanie je prevádzané pomocou `std :: random_device`. Použili sme generátor `std :: mt19937` (A Mersenne Twister pseudo-random generator of 32-bit numbers). Pre súradnicu x boli hodnoty generované z rozsahu 100 až 700 a pre súradnicu y v rozsahu 50 až 650 pomocou `std :: uniform_int_distribution <>`.



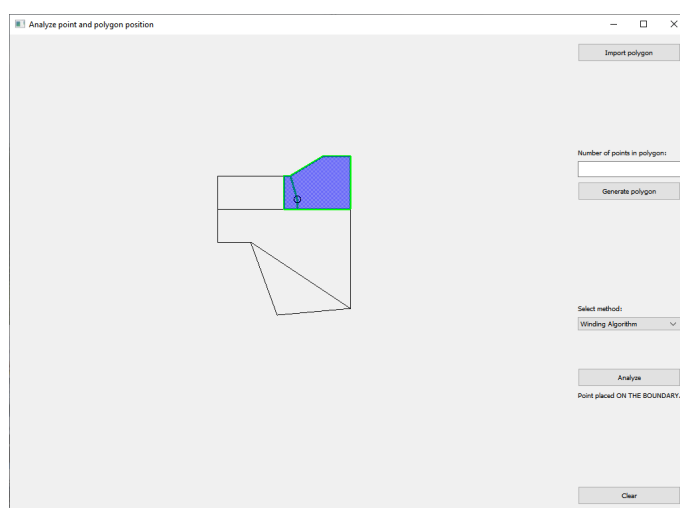
Obr. 8: Ukážka vygenerovaného polygonu s 16 bodmi

## 6 Výsledok analýzy

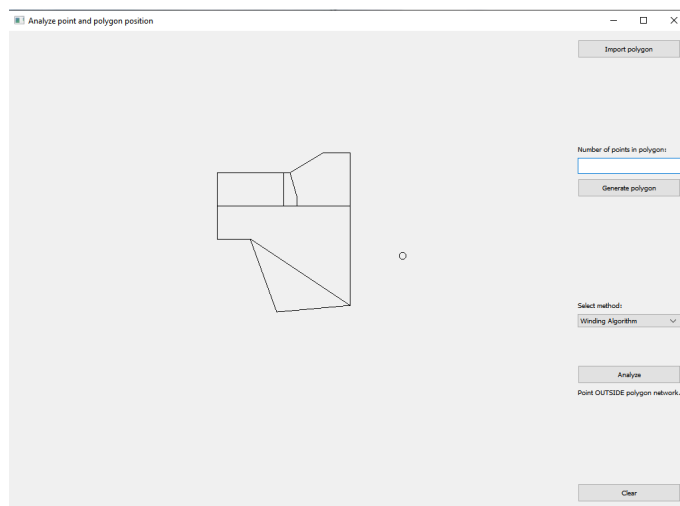
V grafickom rozhraní aplikácie sa po vykonaní analýzy zobrazí výsledok analýzy. Pokiaľ bod leží v polygóne, vysvieti a vyšráfuje sa polygón obsahujúci bod q. Pokiaľ bod leží v spoločnom vrchole viacerých polygónov vysvietia a vyšráfujú sa všetky polygóny pre ktoré je daný vrchol spoločný. Pokiaľ bod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



Obr. 9: Bod  $q$  vnútri polygónu



Obr. 10: Bod  $q$  na hrane



Obr. 11: Bod q mimo polygónu

## 7 Dokumentácia

### 7.1 Trieda Algorithms

Triedu Algorithms sme použili pre deklarovanie funkcií pre výpočtové algoritmy určenia polohy bodu voči polygónu a pre deklaráciu ich pomocných metód.

#### 7.1.1 Metódy

##### positionPointPolygonRayCrossing

- slúži k určeniu polohy bodu prostredníctvom Ray Crossing algoritmu. Návrátový 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

### 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

### getAngle2Vectors

- je pomocnou metódou pre metódu `positionPointPolygonWinding`. Slúži k určeniu uhlu medzi dvoma priamkami. Jej návratovou hodnotou je `double`
- na vstupe má : súradnice bodov  $p_1, p_2, p_3, p_4$  určujúcich prvú a druhú priamku
- výstupom je hodnota uhlu medzi priamkami

### 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  $p_1 p_2$
- na výstupe hodnoty :
  - 1 – bod sa nachádza na priamke
  - 0 – bod sa nachádza vpravo od priamky
  - 1 – bod sa nachádza vľavo od priamky

### polGen

- náhodne generuje polygón s daným počtom bodov. V súčasnom stave len jeden polygón.

- Body polygónu se generované v rozsahu 100 - 700 pre x-ovú súradnicu, 50 - 650 pre y-onovú súradnicu.
- Vstupom je zadaný počet bodov od užívateľa prostredníctvom widget lineEdit.

### GrahamScan

- opravuje poradie vstupných bodov podľa veľkosti uhlu omega, tak aby bolo možné zostaviť nekonvený polygón s využitím Graham Scan algoritmu pre tvorbu nekonvexnej obálky.

## **7.2 Trieda Draw**

Trieda Draw slúži ku grafickému vykresleniu bodu q a polygónovej mapy.

### **7.2.1 Členské premenné**

#### QPoint q

- súradnice bodu, ktorého polohu zisťujeme. Východiskové hodnoty sú nastavené v konštruktore. Hodnoty súradníc určovaného bodu sa menia stlačením tlačidla myši na vykreslovacom plátne..

#### std::vector<stdvector<QPoint>> polygons

- vektor obsahujúci body jednotlivých polygónov

#### std::vector<int> analyze\_results\_by\_polygons

- vektor v ktorom sú uložené výsledky analýzy

#### std::vector<QPoint> generated\_points

- vektor uloženými vygenerovanými body pro tvorbu generovaného polygonu

### 7.2.2 Metódy

#### paintEvent

- slúži k vykresleniu zisťovaného bodu, vykresleniu importu polygónovej mapy a k vyfarbeniu polygónov ktorým náleží určený bod q. Návrátovým typom je void.

#### void mousePressEvent

- slúži k vykresleniu bodu q stlačením tlačidla myši, v okamihu stlačenia tlačidla na myši sa uložia súradnice 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. etóda 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.

#### void generatePoints

- pristupuje k členskej premennej generated\_points a vkladá do nej vygenerované a Graham Scan algoritmom spracované (zoradené) body.

## 7.3 Trieda Widget

Trieda Widget obashuje metódy ktoré sú odkazom na sloty umožňujúce vykonávať príkazy z grafického rozhrania aplikácie. Nemajú žiadne vstupné hodnoty, návratovým typom je void.

### 7.3.1 Metódy

#### on\_clearButton\_clicked

- tlačidlo Clear, vyčistí grafické okno Canvasu

#### on\_analyzeButton\_clicked

- tlačidlo Analyze. Po jeho stlačení sa vykoná analýza polohy bodu q voči polygónovej mape.

#### on\_importPolygonButton\_clicked

- tlačidlo Import polygon. Zobrazí sa dialógové okno v ktorom je možné vybrať textový súbor obsahujúci polygónovú mapu. V prípade úspešného nahratia polygónovej mapy sa zobrazí dialógové okno s počtom nahratých polygónov. V prípade nesprávneho importu sa zobrazí varovné okno.

#### showResultOfAnalysis

- pomocou tejto metódy vypisujeme výsledkok analýzy na label v grafickom okne. Vstupom je výsledok analýzy pre jeden polygón a premenná show result, ktorá je typu bool.

## 7.4 Trieda SortByY

- Triedi body vo vektore podľa veľkosti y-onovej súradnice každého bodu.

## 7.5 Trieda SortByOmega

- Triedi body vo vektore podľa veľkosti uhlu omega. Uhol omega je uhol od rovnobežky daným bodom, najčastejšie pivotom s minimálnou alebo maximálnou súradnicou. V našom prípade sa jedná o bod s najmenšou y-onovou súradnicou.



## 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 nahrania \*.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. Pokúsili sme sa pridať možnosť generovania jedného polygónu. Generovanie je funkčné pre 16 bodov. Pri 17tom bode nastáva nevyriešený problém, znehodnocujúci ďalšie generovanie.

Bohužiaľ po implementovaní generovania nekonvexného polygónu nastal problém pri analýze pomocou Ray Crossing algoritmu. Túto chybu se nám bohužiaľ aj napriek veľkej snahe a maximálnemu nasadeniu nepodarilo doposiaľ odstrániť a zostáva jedným z cieľov vylepšenia do budúcnosti.

## Literatúra

- [1] Cplusplus. cplusplus.
- [2] Qt documentation archives. qt documentation archives.
- [3] Tomáš Bayer. Geometrické vyhledání.
- [4] Tomáš Bayer. *Algoritmy v digitální kartografii*. Karolinum, 2008.
- [5] Introducing Wherewolf. A serverless boundary service from wny.