

České vysoké učení technické v Praze  
Fakulta stavební



Algoritmy v digitální kartografii

Geometrické vyhledávání bodu

Bc. Robin Pflug  
Bc. Tomáš Klemsa

# Obsah

<b>1</b>	<b>Zadání úlohy</b>	<b>2</b>
<b>2</b>	<b>Obecná formulace a řešení problému</b>	<b>3</b>
<b>3</b>	<b>Aplikované algoritmy</b>	<b>3</b>
3.1	Winding Algorithm . . . . .	3
3.1.1	Algoritmus winding number . . . . .	4
3.2	Ray crossing . . . . .	4
3.2.1	Algoritmus Ray crossing s redukcí . . . . .	5
<b>4</b>	<b>Vstup dat do aplikace</b>	<b>5</b>
4.1	Grafický vstup . . . . .	6
4.2	Kombinovaný vstup . . . . .	6
4.3	Formát souboru pro import dat . . . . .	6
<b>5</b>	<b>Výstup aplikace</b>	<b>6</b>
<b>6</b>	<b>Dokumentace</b>	<b>9</b>
6.1	Třídy . . . . .	9
6.1.1	Algorithms . . . . .	9
6.1.2	Draw . . . . .	9
6.1.3	FileReader . . . . .	9
6.1.4	Widget . . . . .	9
<b>7</b>	<b>Řešení bonusových úloh</b>	<b>10</b>
7.0.1	Ošetření singulárního případu u Winding Number Algorithm: bod leží na hraně polygonu . . . . .	10
7.0.2	Algoritmus pro automatické generování nekonvexních polygonů . . . . .	10
<b>8</b>	<b>Závěr</b>	<b>10</b>
<b>9</b>	<b>Náměty pro vylepšení</b>	<b>10</b>
<b>10</b>	<b>Reference</b>	<b>10</b>

# 1 Zadání úlohy

**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 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 geogracká data (např. mapa evropských států).

Polygony budou načítány z textového souboru ve Vámi zvoleném formátu. Pro datovou reprezentaci jednotlivých polygonů použijte špagetový model.

## **Hodnocení:**

<b>Krok</b>	<b>Hodnocení</b>
Detekce polohy bodu rozlišující stavy uvnitř, vně a 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
<b>Max celkem:</b>	<b>21b</b>

Obrázek 1: Bodové hodnocení úlohy [zdroj: 1]

## 2 Obecná formulace a řešení problému

Ve 2D souřadnicích je dána množina  $n$  vrcholů,  $m$  polygonů a bod  $q$ . Cílem řešení problému je určit takový polygon (mnohoúhelník), který bod  $q$  obsahuje. V případě řešení bonusových úloh je cílem nalézt takové polygony, jejichž hrana či vrchol obsahuje bod  $q$ . Řešení je určeno pro nekonvexní polygony. Testu polohy bodu  $q$  vůči polygonu je za realizován pomocí lokální procedury (tj. opakované určení polohy bodu  $q$  vzhledem k polygonu).

Základním zadáním úlohy bylo určit vzájemnou polohu bodu a polygonů. Problém byl řešen dvěma způsoby: algoritmem využívajícího Winding Number a algoritmem nazývaným Ray Crossing. Řešení problematiky je implementováno s využitím Widgets aplikace napsané v jazyce C++. Aplikace byla tvořena na platformě Qt Creator v operačním systému Windows.

Prvním krokem před samotnou tvorbou aplikace byla tvorba kostry programu. Bylo nutné oddělit funkce provádějící přípravu dat a samotný výpočet algoritmů od funkcí zajišťujících chod grafického rozhraní, vstupu apod.

Pro aplikaci bylo nejprve navrženo základní grafické okno, které bylo postupně doplňováno o potřebné interaktivní prvky (především push buttony apod.). Pro zpracování grafické funkcionality vstupu a výstupu dat byla vytvořena třída Draw. Pro funkce zpracovávající oba algoritmy byla vytvořena třída Algorithms. Načítání externích dat (polygonů) je prováděno ve třídě filereader.

Konkrétní řešení a vzorce použité při řešení problematiky jsou obsahem následujících kapitol.

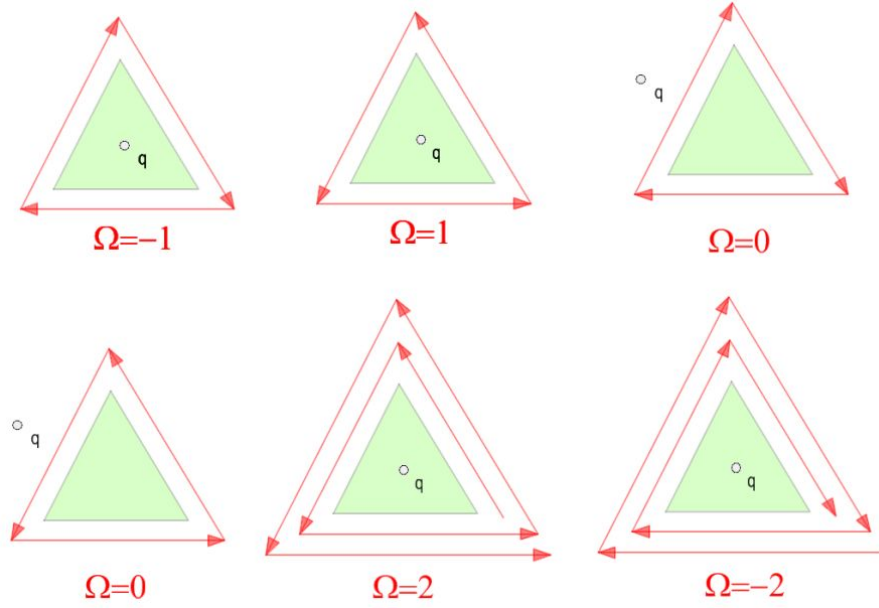
## 3 Aplikované algoritmy

Algoritmy využívané pro geometrické vyhledávání bodu.

### 3.1 Winding Algorithm

Algoritmus vychází z výpočtu úhlů mezi jednotlivými průvodiči z analyzovaného bodu  $q$  do jednotlivých vrcholů  $p_i$  příslušného polygonu  $P$ . Postupným přičítáním a odčítáním úhlů (podle směru z předchozího  $p_i$  na následující vrchol  $p_{i+1}$  polygonu  $P$ ) mezi průvodiči je získávána informace, kde se bod  $q$  vůči polygonu  $P$  nachází. Zda se úhel mezi průvodiči do výsledné sumy úhlů přičte či odečte udává polorovina, ve které se úhel vůči spojnici z bodu  $q$  do vrcholu polygonu  $p_i$  nachází. Úhly nacházející se v pravé, takto definované polorovině, jsou orientovány kladně a proto se přičítají. Úhly v levé polorovině se odčítají. Na základě výsledné sumy všech úhlů pro daný polygon  $P$  mohou určit polohu bodu  $q$ . Je-li suma úhlů násobkem  $2\pi$  znamená to, že bod  $q$  se nachází uvnitř polygonu  $P$ , je-li úhel menší (důsledek odčítání záporně orientovaných úhlů), bod  $q$  leží vně polygonu  $P$ .

Winding number je udáváno jako násobek  $2\pi$  (v radiánové míře). Výsledná hodnota winding number závisí na orientaci směru pohybu (CW x CCW).



Obrázek 2: Princip výpočtu winding number [zdroj: 1]

### 3.1.1 Algoritmus winding number

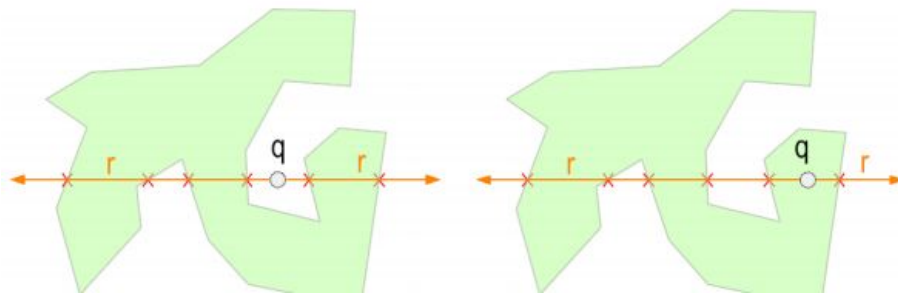
1. Inicializace  $\Omega = 0$ , tolerance  $\epsilon$
2. Opakuj pro  $\forall$  trojici  $(p_i, q, p_{i+1})$  :  
 Urči polohu  $q$  vzhledem k  $e_i = (p_i, p_{i+1})$ .  
 Urči úhel  $\omega_i = \angle p_i, q, p_{i+1}$ .  
 If  $q \in \overline{\sigma}_l$ , pak  $\Omega = \Omega + \omega_i$ .  
 else  $\Omega = \Omega - \omega_i$ .
3. if  $|\omega| - 2\pi| < \epsilon$ , pak  $q \in P$ .
4. else  $q \notin p$ .

[zdroj: 1]

## 3.2 Ray crossing

Základem Ray crossing algoritmu je přímka  $r$  procházející analyzovaným bodem  $q$ . Poloha bodu  $q$  vůči polygonu  $P$  je určována na základě počtu průsečíků přímky  $k$  a hran polygonu  $p_i$ . Obecně platí pro bod  $q$  vně polygonu  $P$ , že počet průsečíků  $k$  je sudý, lichý počet  $k$  platí pro bod  $q$  uvnitř polygonu  $P$ . Problém singularit pro takto daný model algoritmu je řešen převedením na upravený model. V upraveném modelu algoritmu je pro určení výsledku uvažována pouze jedna polovina vzhledem k paprsku  $r$  procházejícím bodem  $q$ . Další modifikace redukuje souřadnice vrcholů  $p_i$  polygonu  $P$  k analyzovanému bodu  $q$ . Pro model s takto vytvořenou lokální souřadnicovou soustavou jsou hledány pouze průsečíky  $M$  ležící v pravé polovině od osy  $x$ , která po redukcí prochází bodem  $q$ . Výsledná varianta algoritmu tedy uvažuje pouze průsečíky  $M$  ležící v jednu kvadrantu (v

popisovaném případě se jedná o první kvadrant) lokální souřadnicové soustavy vzhledem k bodu q. Takto upravený algoritmus



Obrázek 3: Princip Ray crossing algoritmu [zdroj: 1]

### 3.2.1 Algoritmus Ray crossing s redukcí

1. Inicializuj  $k = 0$

2. Opakuj pro  $\forall$  body  $p_i \in P$  :

$$x'_i = x_i - x_q$$

$$y'_i = y_i - y_q$$

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

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

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

3. if  $(k \bmod 2) \neq 0$  pak  $q \in P$

4. else  $q \notin p$ .

[zdroj: 1]

## 4 Vstup dat do aplikace

### ***Vstup dat lze provádět:***

Grafickým vstupem, kdy jsou snímány souřadnice bodů kurzorem myši v grafickém rozhraní aplikace.

Kombinovaným vstupem, kde jsou souřadnice polygonů načítány z textového souboru v předepsaném formátu a souřadnice analyzovaného bodu jsou snímány kurzorem myši v grafickém rozhraní aplikace.

Třetím způsobem vstupu dat do aplikace, konkrétně polygonu, je automatická generace. Tento způsob vstupu je obsahem bonusových úloh. Tlačítkem *Generate Random Polygon* lze vygenerovat v grafickém okně polygon o náhodném počtu vrcholů od 4 do 20.

## 4.1 Grafický vstup

Grafický vstup vrcholů polygonu je aktivován tlačítkem Polygon v sekci Draw Action. Pro ukončení snímání aktuálního polygonu je třeba aktivovat tlačítko None. Poté lze opět po aktivaci tlačítka Polygon snímat a vykreslovat nový polygon nebo aktivací tlačítka Analyze point snímat souřadnice analyzovaného bodu.

## 4.2 Kombinovaný vstup

Pro načtení polygonů z textového souboru je určena sekce Import s tlačítkem Import polygons. Po aktivaci tlačítka lze v dialogovém okně vybrat příslušný soubor a nahrát polygony hromadně do aplikace.

## 4.3 Formát souboru pro import dat

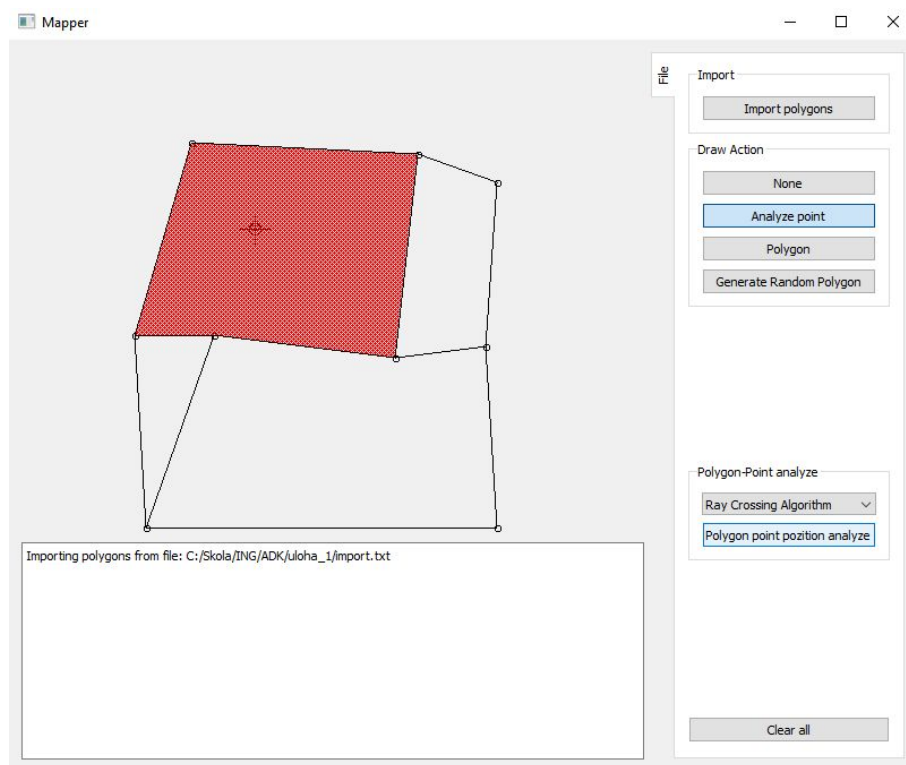
Každý polygon začíná číslem vrcholu od čísla 1 po odsazení tabulátorem následuje souřadnice x, po dalším odsazením souřadnice y. Jednotlivé vrcholy jsou odřádkovány. Nový polygon začíná opět číslem vrcholu 1.

```
//Č.V.    X      Y
  1      100    500    // 1. vrchol prvního polygonu
  2      120    550    // 2. vrchol prvního polygonu
  .
  .
  .
  N      XXX     YYY    // N. Vrchol prvního polygonu
  1      120    550    // 1. vrchol druhého polygonu
```

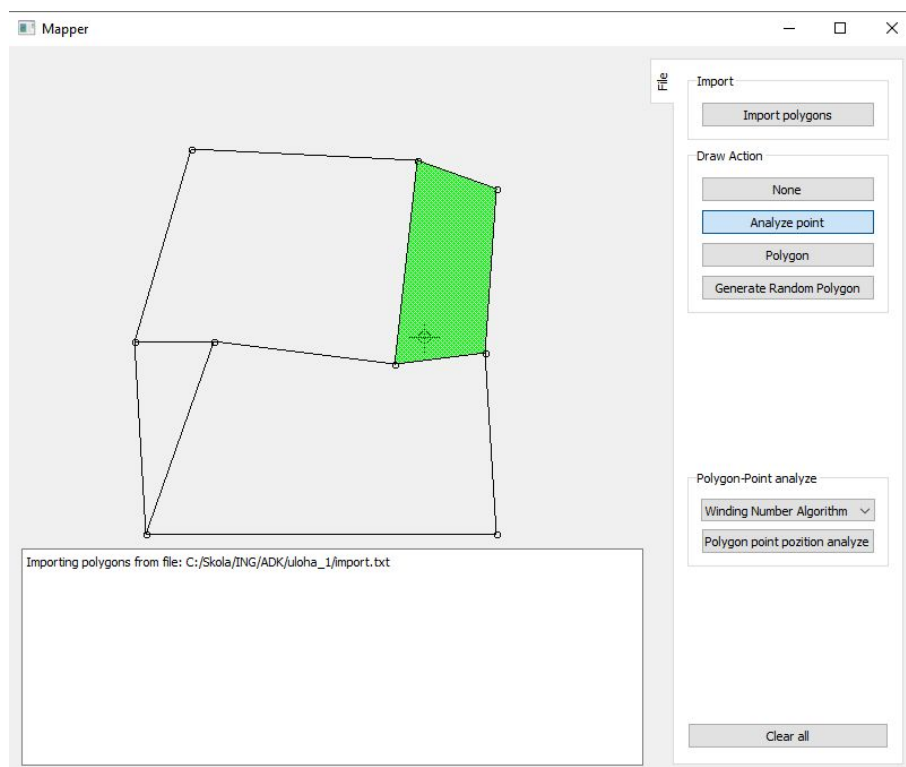
Obrázek 4: Příklad formátu vstupních souřadnic

## 5 Výstup aplikace

Výstupní data jsou prezentována grafickou cestou. Po zadání souřadnic polygonů a analyzovaného bodu jsou vstupní data vykreslena v grafickém okně. Po aktivaci tlačítka Polygon point position analyze je graficky zvýrazněn červenou barvou polygon, ve kterém se bod nachází. Nachází-li se bod na hraně či vrcholu polygonu, je polygon zvýrazněn zelenou barvou.

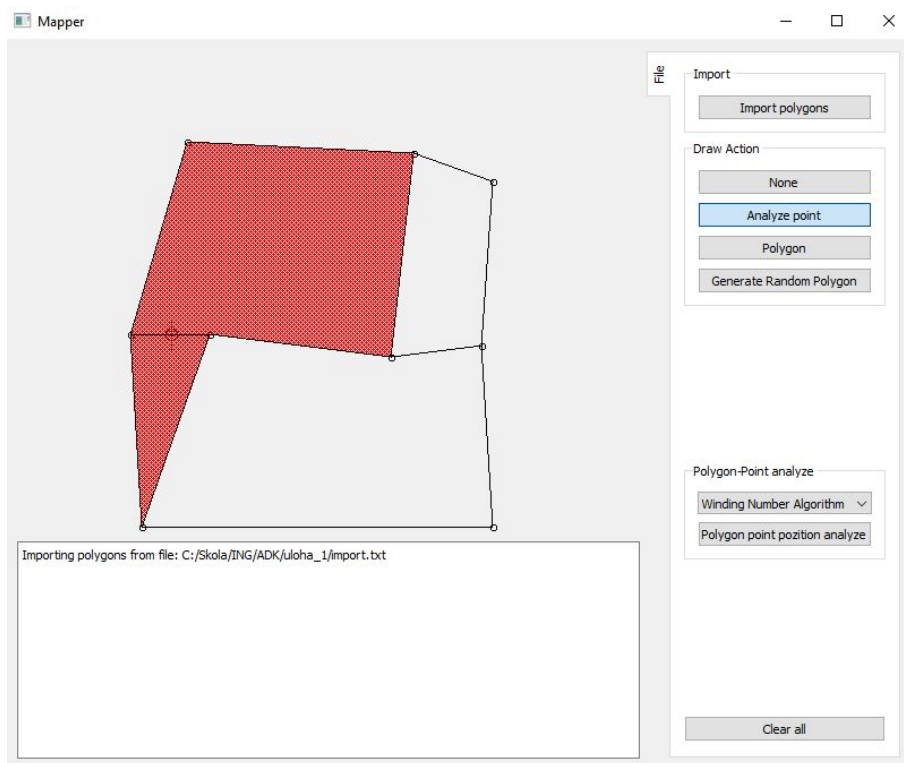


Obrázek 5: Výstup aplikace pro analýzu metodou Ray Crossing

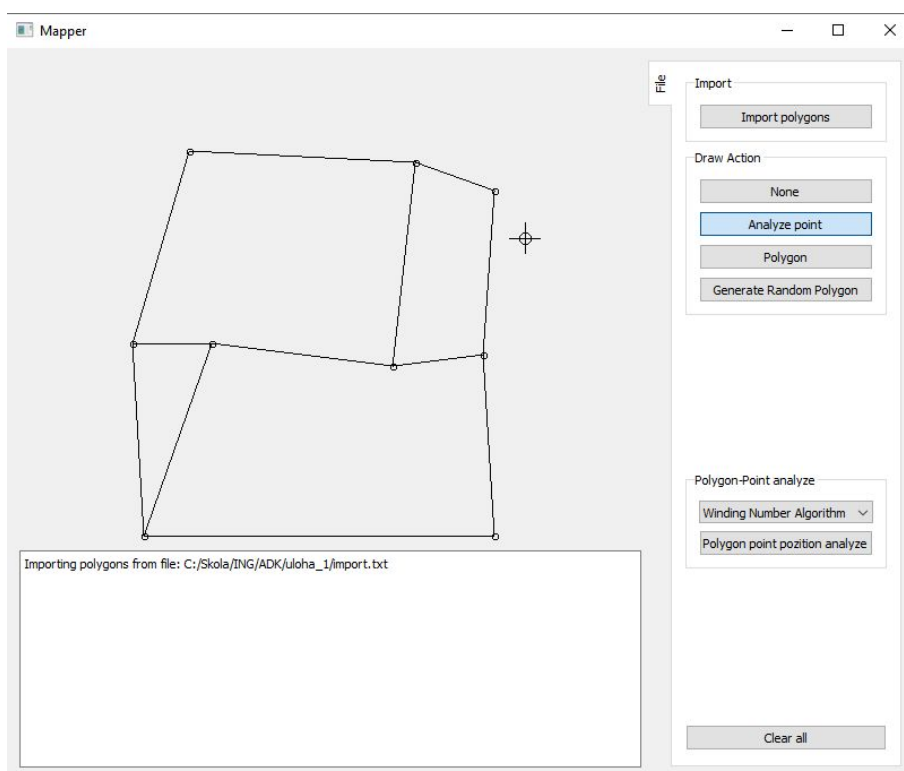


Obrázek 6: Výstup aplikace pro analýzu metodou Winding number





Obrázek 7: Výstup aplikace pro analýzu, kdy se bod nachází na hraně dvou polygonů



Obrázek 8: Výstup aplikace pro analýzu bodu, který nenáleží žádnému polygonu

## 6 Dokumentace

### 6.1 Třídy

#### 6.1.1 Algorithms

Třída obsahující metody potřebné pro výpočet polohy bodu vzhledem k polygonu.

**int getPointLinePosition(QPointF q, QPointF p1, QPointF p2)**

Návratová hodnota: *integer*;

Určení polohy bodu vůči přímce. Návratová hodnota 1 pro bod v levé polorovině, 0 pro bod v pravé polorovině, -1 pro bod na přímce.

**double getAngle2Vectors(QPointF p1, QPointF p2, QPointF p3, QPointF p4)**

Návratová hodnota: *double*;

Metoda vrací úhel mezi dvěma vektory.

**int positionPointPolygonWinding(QPointF q, QPolygonF pol)**

Návratová hodnota: *integer*;

Obsahuje Winding number algoritmus pro výpočet polohy bodu vůči polygonu. Návratová hodnota 1 pro body v polygonu, 0 pro bod mimo polygon a -1 pro bod na hraně polygonu.

**int positionPointPolygonRayCrossing(QPointF q, QPolygonF pol)**

Návratová hodnota: *Integer*;

Obsahuje Ray crossing algoritmus pro výpočet polohy bodu vůči polygonu. Návratová hodnota 1 pro body v polygonu, 0 pro bod mimo polygon a -1 pro bod na hraně polygonu.

**QPolygonF createRandomPolygon()**

Návratová hodnota: *QPolygonF*;

Metoda generující náhodný polygon o 4 až 20 vrcholech.

#### 6.1.2 Draw

Třída s metodami zajišťující snímání a ukládání dat grafického vstupu. Metody také zajišťují grafické vykreslení dat. Třída také umožňuje přepínání metod grafického vstupu.

#### 6.1.3 FileReader

Třída obsahující funkce zajišťující import polygonů v podobě textového souboru.

#### 6.1.4 Widget

Třída vytvářející grafické rozhraní aplikace.

## 7 Řešení bonusových úloh

Z bonusových úloh bylo řešeno zadání: *Algoritmus pro automatické generování nekonvexních polygonů* a *Ošetření singulárního případu u Winding Number Algorithm: bod leží na hraně polygonu..*

### 7.0.1 Ošetření singulárního případu u Winding Number Algorithm: bod leží na hraně polygonu

Tento singulární případ byl ošetřen při určování pozice bodu vůči přímce (funkce: *get-PointLinePosition*). Funkce vyhodnotí jako kolineární bod  $q$  takový, který má součet vzdáleností počátečního bodu úsečky do  $q$  ( $p1,q$ ) a  $q$  do koncového bodu úsečky ( $q,p2$ ) krajně blízký k samotné délce úsečky ( $p1,p2$ ).

### 7.0.2 Algoritmus pro automatické generování nekonvexních polygonů

Způsob vstupu polygonů metodou generování náhodných nekonvexních polygonů je naimplementován tak, že generuje náhodný polygon o čtyřech až dvaceti vrcholech. Algoritmus nejprve náhodně vygeneruje počet vrcholů polygonu. V dalším kroku generuje algoritmus vrcholy polygonu. Tvorba vrcholů je naprogramována tak, že se vygeneruje od 3 do 19 vrcholů a jako poslední je použit střed polygonu, pro dosažení nekonvexnosti. Pro vygenerované body polygonu je použit cyklus, který vrcholy seřadí podle velikosti směrníku. Směrník je dán prvním vrcholem, středem a následujícím vrcholem polygonu, nakonec opravený v závislosti na kvadrantu. Takto vygenerovaný polygon se seřazenými vrcholy je doplněn o středový bod a funkcí navrácen.

## 8 Závěr

Pro předpřipravená data aplikace funguje bez problému, problematické situace algoritmu podle zadání byly ošetřeny. Aplikace umožňuje několik metod vstupů dat a pro povinnou část zadání vyhodnocuje analýzu správně.

## 9 Náměty pro vylepšení

Aplikace by byla o mnoho všestrannější, pokud by při importu neupravených polygonů z textového souboru, automaticky redukovala souřadnice vrcholů tak, aby je bylo možné zobrazit v grafickém okně. Tento problém aplikace neřeší a proto je nutné před importem souřadnice upravit jiným způsobem.

## 10 Reference

1. BAYER, Tomáš. Metody konstrukce konvexní obálky [online][cit. 5.11.2019]. Dostupné z: <https://web.natur.cuni.cz/~bayertom/images/courses/Adk/adk4.pdf>