<table>
<tr><td colspan="5">ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA STAVEBNÍ, OBOR GEODÉZIE A KARTOGRAFIE
KATEDRA GEOMATIKY</td></tr>
<tr><td colspan="5">název předmětu:
ALGORITMY DIGITÁLNÍ KARTOGRAFIE A GIS</td></tr>
<tr><td>číslo úlohy:
1.</td><td colspan="4">název úlohy:
Geometrické vyhledávání bodu</td></tr>
<tr><td>školní rok:
2020/21</td><td>semestr:
3.</td><td>zpracovali:
Eva Frommeltová, Lucie Hnilicová</td><td>datum:
23.10. 2020</td><td>klasifikace:</td></tr>
</table>

Obsah

1 Zadání

Vstup: Souvislá polygonová mapa n polygonů $\{P_1, \ldots, P_n\}$, analyzovaný bod q.
Výstup: P_i, $q \subset P_i$.
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.

2 Doplňující úlohy

- Ošetření singulárního případu u Winding Number Algorithm: bod leží na hraně polygonu
    - řešeno
- Ošetření singulárního případu u obou algoritmů: bod je totožný s vrcholem jednoho či více polygonů.
    - řešeno
- Zvýraznění všech polygonů pro oba výše uvedené singulární případy
    - řešeno
- Algoritmus pro automatické generování nekonvexních polygonů

3 Popis a rozbor problémů

Cílem úlohy je načíst polygonovou mapu na kreslící plátno aplikace. Pomocí kliknutí do kreslícího plátna je vyobrazen bod *q*. Pro tento bod je pomocí algoritmů *Winding Number* a *Ray Crossing* určeno, zda se nachází v nekonvexním polygonu/polygonech nebo mimo něj. Pokud je bod v polygonu, tento polygon je zvýrazněn. Může nastat i několik dalších případů:

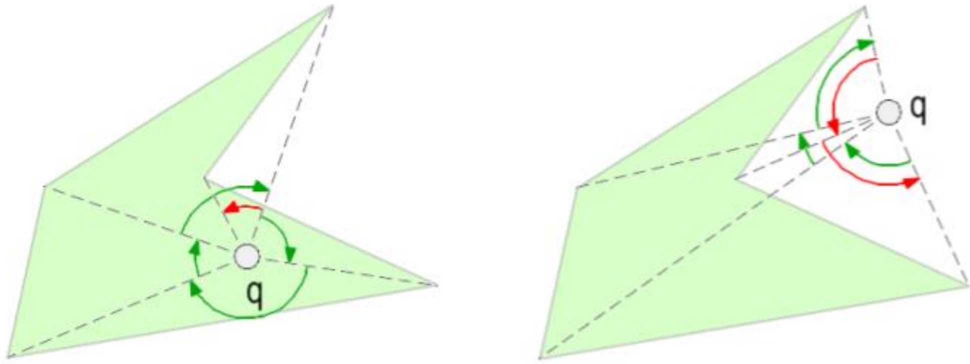
- bod *q* leží na hraně polygonu,
- bod *q* je totožný s vrcholem jednoho či více polygonů.

4 Popis algoritmů

Algoritmy popsané v této kapitole jsou využívány pro řešení nekonvexních mnohoúhelníků.

4.1 Winding Number Algorithm

Algoritmus *Winding Number* je také znám pod názvem *Metoda ovíjení*. Při této metodě je pozorovatel na bodě *q*. Pokud je pozorovatel uvnitř polygonu *P* je možné vidět všechny vrcholy polygonu – součet všech rotací je tedy roven úhlu *2π* – je tedy potřeba, aby se pozorovatel otočil o celý kruh. Pokud je pozorovatel vně polygonu *P* je také možné vidět všechny vrcholy polygonu, ale součet všech rotací je menší než *2π* – pozorovatel se neotočí o celý kruh.



Obr. 1 – Znázornění otáčení pozorovatele na bodě q uvnitř a vně polygonu P

U tohoto algoritmu je potřeba vypočítat počet otočení Winding Number Ω. Počet otočení je roven sumě všech rotací ω_i měřených v CCW (protisměru hodinových ručiček), které musí průvodič mezi *q* a p_i opsat nad všemi body p_i v polygonu *P.*

$$\Omega = \sum_{i=1}^{n} \frac{\omega_i}{2\pi}.$$

Pokud je úhel *ω* načítán po směru hodinových ručiček je ω_i > 0 a má kladné znaménko. Pokud je úhel *ω* načítán proti směru hodinových ručiček je ω_i <0 a má záporné znaménko. Pak tedy hodnota Ω určuje polohu bodu *q* vůči polygonu *P*.

$$\Omega(q,P) = \begin{cases} 1, & q \in P, \\ 0, & q \notin P. \end{cases}$$

4.1.1 Výpočet úhlu ω

Pro výpočet úhlu *ω* je potřeba vypočítat vektory $\vec{u}\ a\ \vec{v}$ mezi bodem *q* a jednotlivými vrcholy P_i*:*

$$\vec{u} = |P_1 q|,$$
$$\vec{v} = |q P_2|.$$

Z těchto vektorů je možné určit úhel *ω:*

$$\omega = \left|\mathrm{acos}\left(\frac{u*v}{|u|\cdot|v|}\right)\right|.$$

V dalším kroku je potřeba určit pozici vzhledem k hraně polygonu – test vzájemné polohy:

$$\vec{u} = (x_{P2} - x_{P1}, y_{P2} - y_{P1}),$$

$$\vec{v} = (x_q - x_{P1}, y_q - y_{P1}),$$

$$t = \begin{vmatrix} u_x & u_y \\ v_x & v_y \end{vmatrix} = \begin{cases} > 0, & q \in \sigma_L \\ < 0, & q \in \sigma_P \\ = 0, & q\ leží\ na\ hraně. \end{cases}$$

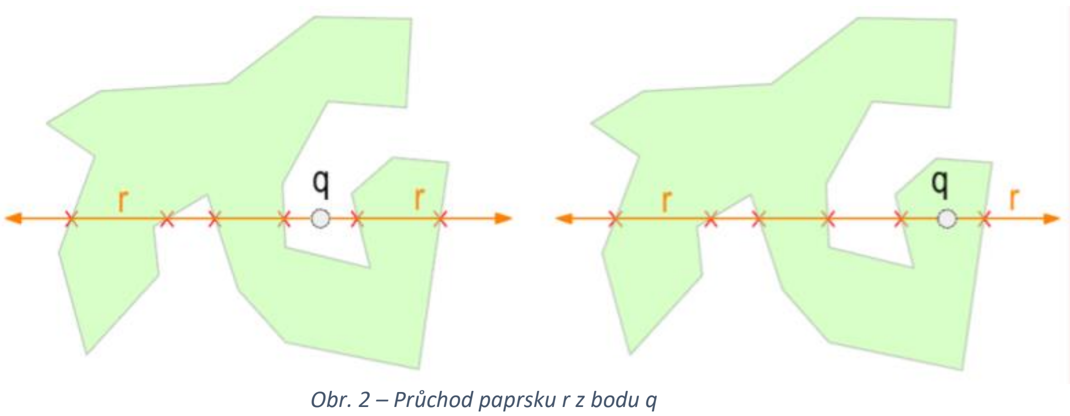
4.1.2 Slovní popis algoritmu WN

1: Inicializuj Ω = 0, tolerance ε
2: Opakuj pro všechny body v trojici p_i, q, p_{i+1}:
3: Urči polohu q vzhledem k $p = (p_i, p_{i+1})$.
4: Urči úhel ω mezi body q, počátečním bodem p_i a koncovým bodem p_{i+1}
5: Pokud se bod q nalézá v levé polorovině – $\Omega = \Omega + \omega_i$
6: Jinak, když se bod q nalézá v pravé polorovině – $\Omega = \Omega - \omega_i$
7: Pokud $(|\Omega \pm 2\pi| < \varepsilon)$ – bod q leží uvnitř polygonu P
8: Jinak bod q leží vně polygonu P

Tento algoritmus však neřeší problémy singularity.

4.2 Ray Crossing Algorithm

Algoritmus *Ray Crossing* je také znám pod názvem *Paprskový algoritmus.* Bodem q je vedena polopřímka r (paprsek) nezávisle na směru. Polopřímka r je často volena $y = y_q$. Je počítán počet průsečíků s polygonem P. Když se bod q nachází v polygonu P je počet průsečíků s r roven lichému číslu. Pokud leží bod q vně polygonu P je počet průsečíků s r roven sudému číslu.

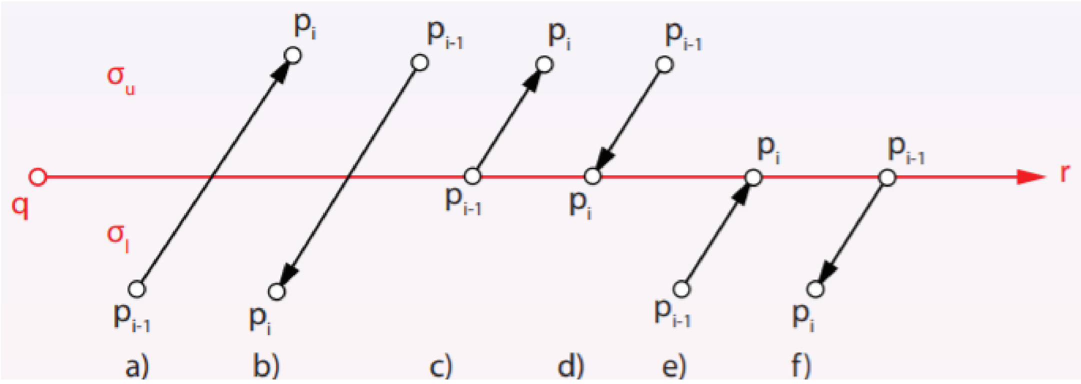


Obr. 2 – Průchod paprsku r z bodu q

Pro přehlednost lze určit, zda zbytek po celočíselném dělení počtu průsečíků *k* je roven:

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

Paprsek je volen buď vodorovně nebo svisle z bodu *q*. Tím je polygon rozdělen a je testována jen jedna polovina nebo segmenty které jsou kolineární – těchto případů je třeba se zbavit. Podle souřadnice *y* je určeno, kde leží bod p_i vůči paprsku *r*. Když je větší, než y_q je na *r* a naopak.



Obr. 3 – Zobrazení případů polohy bodu p_i a p_{i-1}

Průsečík je započítáván pokud:

Hrana v obou polorovinách nebo jen v horní:

i) je bod p_i nad paprskem a p_{i-1} pod nebo na paprsku → a), c)

ii) je bod p_{i-1} nad paprskem a p_i pod nebo na paprsku → b), d)

$$(y_i > y_q) \wedge (y_{i-1} \leq y_q) \vee (y_{i-1} > y_q) \wedge (y_i \leq y_q).$$

Hrana v obou polorovinách nebo jen v dolní

iii) je p_i pod paprskem a p_{i-1} je nad nebo na paprsku → a), e)

iv) je bod p_{i-1} pod paprskem a p_i nad nebo na paprsku → b), f)

$$(y_i < y_q) \wedge (y_{i-1} \geq y_q) \vee (y_{i-1} < y_q) \wedge (y_i \geq y_q).$$

4.2.2 Slovní popis algoritmu WN

Pro redukci souřadnic polygonu *P* k bodu *q* jsou použity tyto vzorce:

$$x'_p = x_p - x_q,$$

$$y'_p = y_p - y_q.$$

Dále je vypočten průsečík s osou x‘ = 0:

$$x'_m = \frac{(x'_i y'_{i-1} - x'_{i-1} y'_i)}{(y'_i - y'_{i-1})}.$$

Poté je proveden test, zda existuje průsečík:

$$(y'_i > 0) \wedge (y'_{i-1} \leq 0) \vee (y_{i-1} > 0) \wedge (y_i \leq 0).$$

Průsečík se nachází v kladném směru osy x‘ pokud je $x'_m > 0$.

$$k(r,P) = \begin{cases} k(r,P) + 1, & x'_m > 0, \\ k(r,P), & x'_m \leq 0. \end{cases}$$

4.2.2 Slovní popis algoritmu WN

1: Inicializuj k = 0,
2: Opakuj pro všechny body p_i polygonu *P*
3: Redukuj souřadnice $x'_i = x_i - x_q, y'_i = y_i - y_q$
4: Když $(y'_i > 0) \wedge (y'_{i-1} \leq 0) \vee (y_{i-1} > 0) \wedge (y_i \leq 0)$ – je segment vhodný
5: $x'_m = \frac{(x'_i y'_{i-1} - x'_{i-1} y'_i)}{(y'_i - y'_{i-1})}$ – vhodný průsečík
6: Když ($x'_m > 0$) pak k = k+1
7: Když (k%2) $\neq 0$ pak je bod *q* v polygonu *P*
8: Jinak bod *q* leží vně polygony *P*

5 Problematické situace a jejich rozbor

Za problematické situace jsou považovány případy, kdy bod *q* leží na hraně polygonu *P* nebo je bod *q* totožný s vrcholem jednoho či více polygonů. Pro tyto situace je potřeba ošetřit singulární případy.

5.1 Bod ležící na hraně polygonu – Winding Number Algorithm

Pro ošetření tohoto případu je využita funkce *getPointLinePosition*. Díky této funkci je možno určit orientaci bodu *q* vůči hraně polygonu. Jako návratové hodnoty jsou použity -1, 0, 1. V případě, že je vrácena hodnota -1, je bod *q* na hraně polygonu.
Dalším krokem pro ověření je porovnat souřadnice *x* koncových bodů linie a souřadnici *x* bodu *q*.

5.2 Bod je totožný s vrcholem jednoho či více polygonů

Pro ošetření tohoto případu je využito porovnávání souřadnic x_q a y_q se souřadnicemi vrcholů polygonu. Když jsou souřadnice bodu *q* totožné s vrcholem, který je společný pro jeden nebo více polygonů, jsou tyto polygony zvýrazněny.

5.3 Zvýraznění všech polygonů pro oba výše uvedené případy

Při umístění bodu *q* do polygonu, na jeho hranu nebo na vrchol polygonů jsou tyto případy zvýrazněny šrafováním.

6 Vstupní data, formát vstupních dat, popis

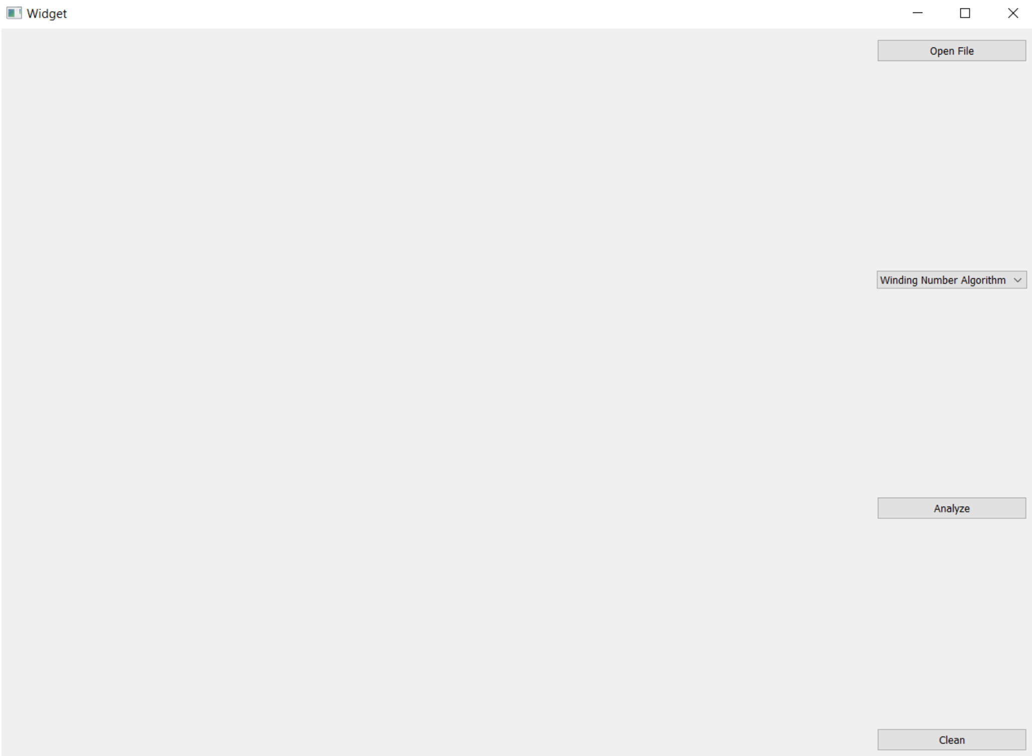
Jako vstup je použit textový soubor *polygons.txt.* Textový soubor je strukturován tak, že v prvním sloupci je číslo vrcholu, v druhém je souřadnice *x* a ve třetím je souřadnice *y*. Soubor obsahuje *i* řádků. Každý dílčí polygon je číslován od 1 a může obsahovat *n* vrcholů.

7 Výstupní data, formát výstupních dat, popis

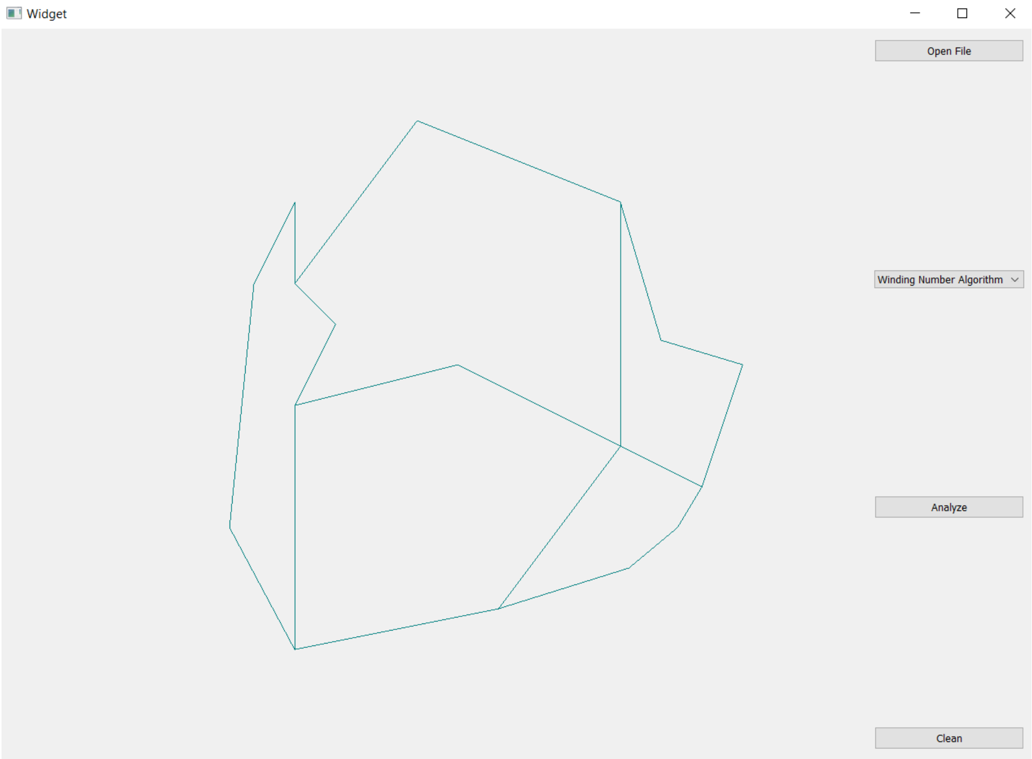
Výstupní data nemají žádný formát. Za výstup je považováno vykreslení polygonů a bodu na kreslící plátno aplikace. Pomocí tlačítka „Analyze“ je zvýrazněn polygon, ve kterém se nachází bod q.

8 Vzhled vytvořené aplikace

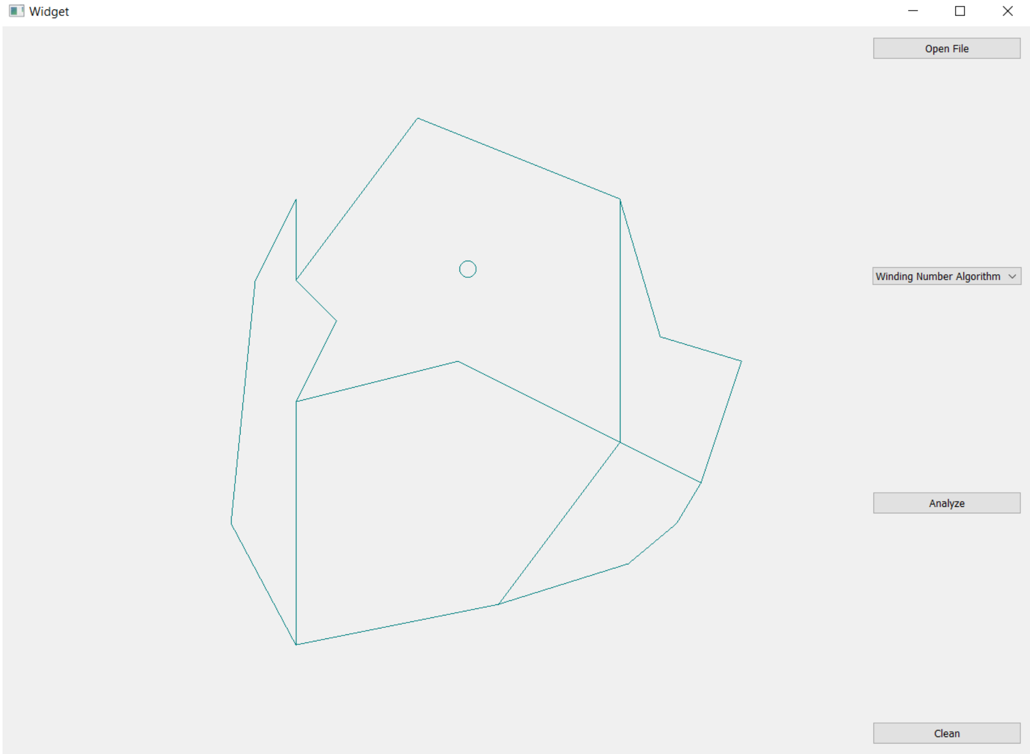
V této kapitole jsou vloženy ukázky vzhledu aplikace s různými funkcionalitami.



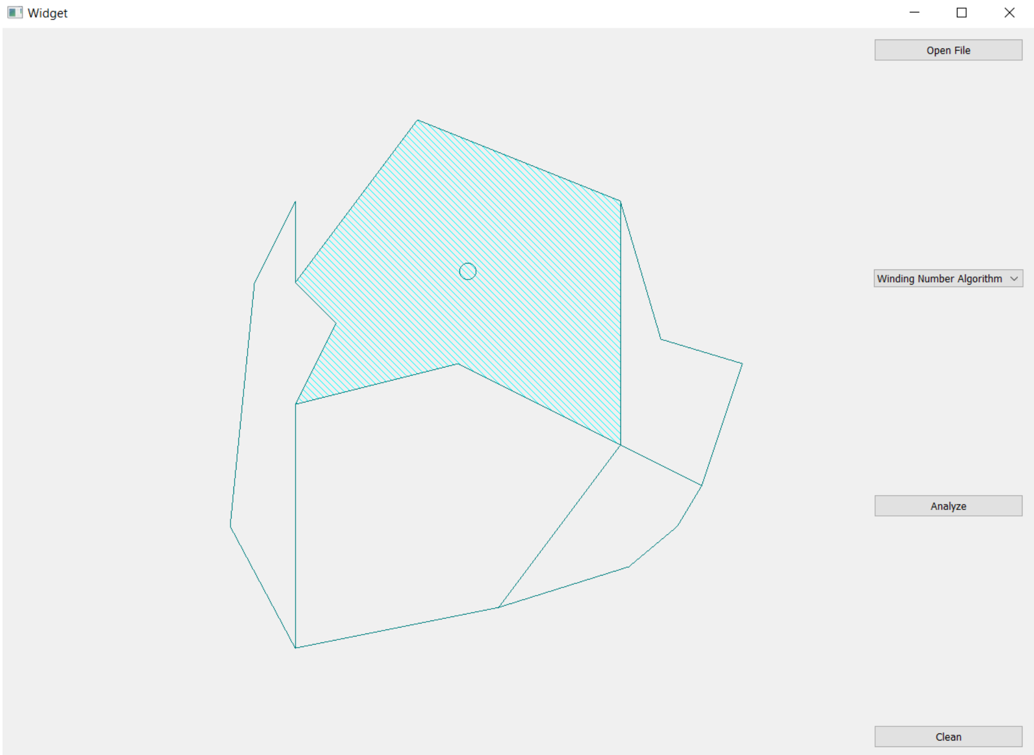
Obr. 4 - Základní vzhled aplikace



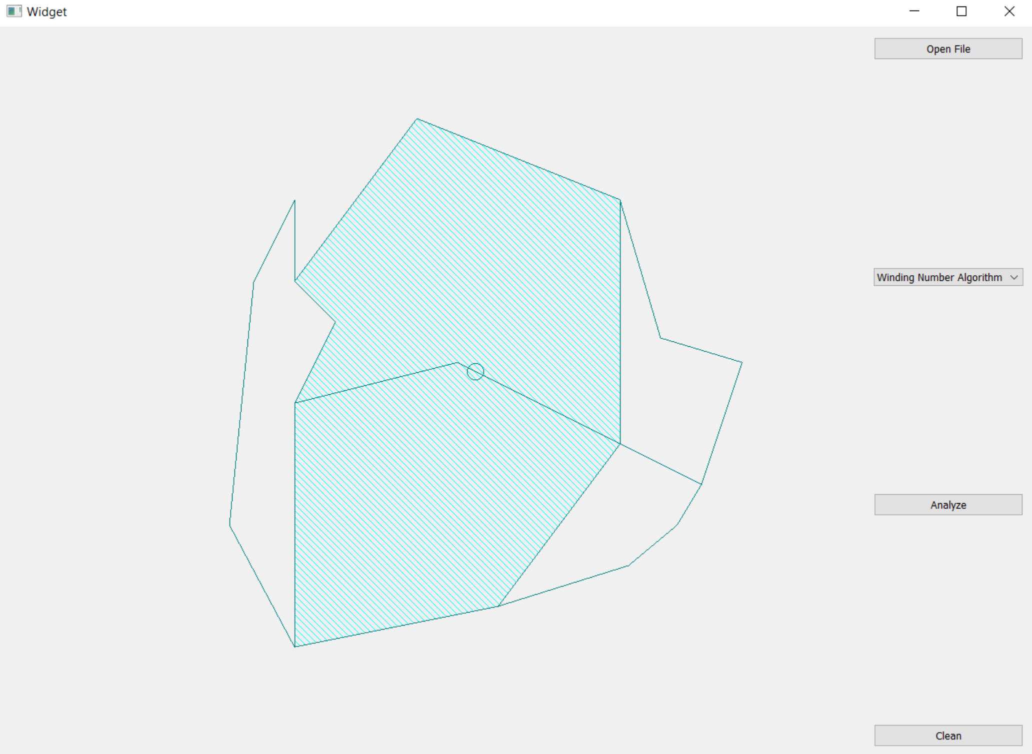
Obr. 5 - Vykreslení polygonu po otevření souboru



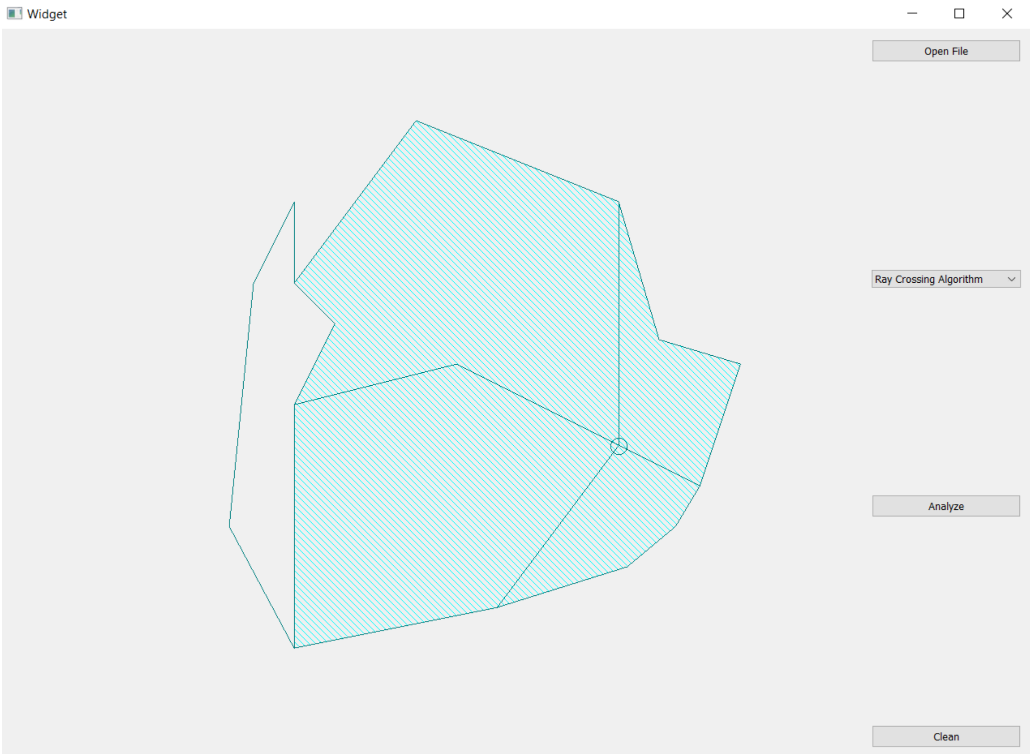
Obr. 6 - Vykreslení bodu do polygonu po kliknutí myší



Obr. 7 - Zvýraznění polygonu při spuštění analýzy – kliknutí na tlačítko „Analyze"



Obr. 8 – Ošetření singulárního případu – bod na hraně Winding Number Aalgorythm



Obr. 9 – Ošetření singulárního případu – bod totožný s vrcholem polygonů

9 Dokumentace

V této kapitole jsou pospány jednotlivé třídy, které zajišťují chod aplikace.

9.1 Třída Algorithms

Třída *Algorithms* obsahuje metody, které počítají polohu bodu q vůči polygonům. Metody *getPositionWinding* a *getPositionRay* vrací hodnotu 1 pokud je bod uvnitř polygonu a hodnotu 0 pokud je bod vně polygonu.

```
static int getPointLinePosition(QPointF &q,QPointF &p1,QPointF &p2);
```

- Vstupními parametry této funkce jsou bod *q* a souřadnice koncových bodů hran polygonů (p_1, p_2)
- Funkce je vytvořena pro výpočet pozice bodu *q* vůči hraně polygonu. Určuje, zda se bod nachází v pravé, v levé polorovině nebo na hraně

```
static double getAngle(QPointF &p1,QPointF &p2,QPointF &p3,QPointF &p4);
```

- Vstupními parametry této funkce jsou počáteční a koncové souřadnice vektorů $\vec{u}$ a $\vec{v}$
- Funkce je vytvořena pro výpočet úhlu mezi dvěma vektory

```
static int getPositionWinding(QPointF &q, std::vector<QPointF> &pol);
```

- Vstupními parametry této funkce jsou souřadnice bodu *q* a souřadnice polygonů
- Funkce je vytvořena pro výpočet metody *Winding Number*, která určuje polohu bodu vůči polygonu metodou navíjení

```
static int getPositionRay(QPointF &q, std::vector<QPointF> &pol);
```

- Vstupními parametry této funkce jsou souřadnice bodu *q* a souřadnice polygonů
- Funkce je vytvořena pro výpočet metody *Ray Crossing*, která určuje polohu bodu vůči polygonu paprskovou metodou

9.2 Třída Draw

Třída *Draw* obsahuje funkce a datové položky umožňující vykreslení výsledků. V privátní části jsou vytvořeny datové položky pro bod *q* a vektor souřadnic polygonů. Ve veřejné části jsou vytvořeny funkce pro vykreslování.

```
private:
std::vector<QPolygonF> polygons;
```

- Vytvoření proměnné pro načtení polygonů

```
QPointF q;
```

- Vytvoření proměnné pro bod *q*

```
std::vector<int> result;
```

- Vytvoření proměnné pro zvýraznění polygonů

```
public:
void paintEvent(QPaintEvent *e);
```

- Funkce pro vykreslení a zvýraznění polygonů a vykreslení bodu *q*

```
void mousePressEvent(QMouseEvent *e);
```

- Funkce pro sejmutí bodu z kreslícího plátna po kliknutí myši

```
void loadPolygons(std::string &path);
```

- Funkce pro načtení souboru, ve kterém jsou uloženy souřadnice polygonů (*.txt)

```
void setResult(std::vector<int> res){result = res;}
```

- Funkce pro nastavení hodnot potřebných pro vykreslení zvýraznění polygonů

```
QPointF & getPoint(){return q;}
```

- Funkce pro vrácení hodnoty uložené v proměnné *q*

```
std::vector<QPolygonF> & getPolygons(){return polygons;}
```

- Funkce pro vrácení hodnot uložených v proměnné *polygons*

```
std::vector<int> &getResult(){return result;}
```

- Funkce pro vrácení hodnot uložených v proměnné *result*

9.3 Třída Algorithms

Třída *Widget* obsahuje funkce, které ovládají všechna tlačítka ve výstupní aplikaci.

```
void on_openFile_clicked();
```

- Funkce pro otevření textového souboru s polygony

```
void on_Analyze_clicked();
```

- Funkce, která volá metody pro určení polohy bodů vůči polygonu

```
void on_clear_clicked();
```

- Funkce pro vyčištění kreslícího plátna

10 Závěr

V této úloze bylo cílem vytvořit aplikaci, která určuje vztah bodu vůči polygonu/ům za pomoci dvou algoritmů – *Winding Number Algorithm* a *Ray Crossing Algorithm*. Dále byly řešeny doplňkové úlohy pro ošetření případů, kdy vybraný bod leží na hraně polygonu nebo je totožný s vrcholem polygonu.

Seznam obrázků