České vysoké učení technické v Praze
Fakulta stavební
Katedra geomatiky

Technická zpráva

Algoritmy v digitální kartografii

Úloha č. 1: Geometrické vyhledávání bodu

Bc. Pane Kuzmanov
Bc. František Mužík

Studijní program: Geodézie a kartografie
Specializace: Geomatika

Praha 2021

Úloha č. 1: Geometrické vyhledávání bodu

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

Výstup: P_i, $q \in P_i$.

Nad polygonovou mapou implementujete Winding Number Algorithm pro geometrické vyhledání incidujícího polygonu obsahujícího zadaný bod q.

Nalezený polygon 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
Analýza polohy bodu (uvnitř/vně) metodou Ray Algorithm.	*+5b*
Ošetření singulárního případu u Ray Algorithm: bod leží na hraně polygonu.	*+5b*
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*
Max celkem:	**24b**

Čas zpracování: 1 týden.

1 Údaje o bonusových úlohách

1.1 Analýza polohy bodu (uvnitř/vně) metodou Ray Algorithm (+5b)

Bylo implementováno zjištění polohy bodu s využitím Ray Algorithm. Matematický popis algoritmu včetně vzorců je sepsán v kapitole 3.2. Grafické znázornění funkční implementace algoritmu je na obrázku č. 4.

1.2 Ošetření singulárního případu u Ray Algorithm: bod leží na hraně polygonu (+5b)

Byl ošetřen singulární případ při použití Ray Algorithm, jestliže bod leží na hraně polygonu. Popis řešení problému včetně grafického znázornění, je uveden v kapitole 4.2. Funkčnost ošetření tohoto singulárního případu je zobrazeno v pravé části obr. 4.

1.3 Ošetření singulárního případu u obou algoritmů: bod je totožný s vrcholem jednoho či více polygonů (+2b)

Byla ošetřena situace, ve které bod leží na vrcholu jednoho či více polygonů. Graficky znázorněna je tato situace na obr. 5. Pokud bod leží na vrcholu výsledkem programu je tedy: *Point is on the vertex.*

1.4 Zvýraznění všech polygonů pro oba výše zmíněné singulární případy (+2b)

Zvýraznění polygonů obsahuje možnost, při které se zvýrazní zároveň více polygonů, pokud bod leží na jejich společné hraně či společném vrcholu. Výsledek tohoto určení je zobrazen na obr. 4.

Implementace:

Při určování polohy bodu jedním ze dvou postupů, bylo do *for* cyklu přidáno zapisování pozice bodu vůči polygonu do samostatného vektoru, na základě kterého je následně určeno, zda – li se má daný polygon vybarvit barvou pro odlišení. Závěrem je zadaný bod znovu vykreslen, neboť byl při zvýraznění polygonu překryt.

2 Popis a rozbor problému

Nechť existuje polygon P, který je tvořen jednotlivými vrcholy p_i. V této úloze se konkrétně jedná o načtený polygon (respektive více polygonů) se souřadnicemi z textového souboru ve stanoveném formátu. Následně je nutné určit, zda – li leží uživatelem zadaný bod q uvnitř, vně, na hranici nebo na vrcholu polygonu. Vybraný polygon je graficky odlišen od ostatních. Výpočet polohy bodu vůči polygonu je popsán pomocí níže sepsaného postupu s použitými vzorci.

Možné výsledky:

- Bod q se nachází uvnitř polygonu P.
- Bod q se nachází mimo polygon P.

- Bod q leží na hraně polygonu P.
- Bod q leží vrcholu polygonu P.

3 Popisy algoritmů formálním jazykem

3.1 Winding Number Algorithm

Jestliže pozorovatel stojí na námi určeném bodě q. Při určení polohy bodu vůči polygonu P se následně pozorovatel otáčí na bodě q proti směru hodinových ručiček. Při otočení proti směru hodinových ručiček, se úhlu přiřadí kladné znaménko a naopak, při otočení podél směru hodinových ručiček, je přiřazeno znaménko záporné. Pozorovatel takto zapisuje úhly ω mezi jednotlivými vrcholy polygonu, dokud se nedostane do počátečního bodu. Dále se vypočte suma všech úhlů (tzv. Winding number) s uvedením příslušných znamének a provede se určení polohy:

- Pokud $q \in P$ a pozorovatel chce vidět více $\forall pi \in P$, musí se otočit o úhel 2π.
- Pokud $q \notin P$, je tento úhel menší než 2π.

Zadáno: bod q, polygon P tvořený vrcholy p_i
Určováno: úhly mezi vrcholy $\omega(p_i, q, p_{i+1})$
Implementace algoritmu:
Ze souřadnic bodů jsou vypočteny vektory $\vec{u}_i = (q, p_i)$ a $\vec{v}_i = (q, p_{i+1})$.
Dále probíhá výpočet jednotlivých úhlů: $\cos\omega = \dfrac{\vec{u}_i \cdot \vec{u}_i}{|\vec{u}_i| \cdot |\vec{v}_i|}$.

1. Inicializace $\Omega = 0$, tolerance ϵ. Natavení tolerance vychází z nutnosti porovnávání reálných, nikoliv celých, čísel.
2. Opakování pro $\forall$ trojici (p_i, q, p_{i+1}).
3. Určení polohy q vzhledem k hranici polygonu $p = (p_i, p_{i+1})$. Tedy jestli bod leží vlevo, vpravo nebo na úsečce (hranici polygonu).
4. Určení úhlu $\omega_i = \angle p_i, q, p_{i+1}$.
5. Zjištění do jaké poloroviny bod patří. Pokud $q \in \overline{\Omega_l}$, pak $\Omega = \Omega + \omega_i$. Bod bude v ležet v levé polorovině.
6. Jinak $\Omega = \Omega - \omega_i$. Pak bod bude ležet v pravé polorovině.
7. Závěrem je proveden test na odchylku od 2π. Pokud $||\Omega| - 2\pi| < \epsilon$, pak se bod q nachází uvnitř polygonu P.
8. Jinak bod q leží mimo polygon P.

3.2 Ray Algorithm

Bodem q je vedena polopřímka r (tedy paprsek, tzv. ray) ve směru osy X. Jednotlivé průsečíky polopřímky r s polygonem P jsou sčítány jen v kladném nebo jen v záporném směru osy X. Jestliže je výsledný počet průsečíků lichý, pak bod q leží uvnitř polygonu P. Jestliže je výsledný počet průsečíků sudý, pak bod q leží vně polygonu P.

Z několika důvodů je vhodné použít upravenou variantu algoritmu s redukcí ke q. Mezi tyto důvody patří snadnější detekce hran protínajících $r(q)$, jednodušší výpočet průsečíku M, vyšší numerická stabilita a nezávislost na orientaci P.

Zadáno: bod q, polygon P tvořený vrcholy p_i
Určováno: počet průsečíků s polygonem P
Implementace algoritmu:

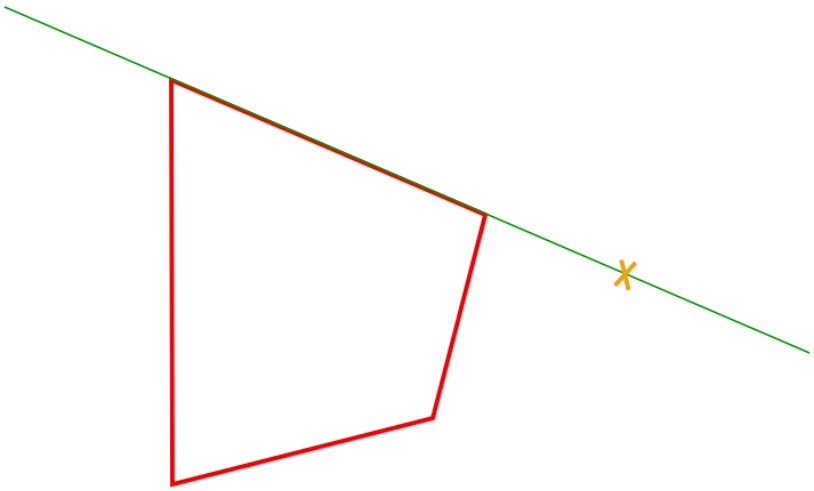
1. Inicializace $k = 0$. Počet průsečíků.
2. Opakování pro $\forall$ body $p_i \in P$:
3. $x_i^{'} = x_i - x_q$. Redukce x – ové souřadnice.
4. $y_i^{'} = y_i - y_q$. Redukce y – ové souřadnice.
5. Pokud $(y_i^{'} > 0)\&\&(y_{i-1}^{'} <= 0) \| (y_{i-1}^{'} > 0)\&\&(y_i^{'} <= 0)$.
6. $$x_m^{'} = \frac{x_i^{'} y_{i-1}^{'} - x_{i-1}^{'} y_i^{'}}{y_i^{'} - y_{i-1}^{'}}$$
7. Pokud $(x_m^{'} > 0)$, pak $k = k + 1$.
8. Pokud $(k\%2) \neq 0$, pak $q \in P$.
9. Jinak $q \notin P$.

4 Problematické situace a jejich rozbor

4.1 Winding Number Algorithm

Tento algoritmus problematicky řeší případ, když je bod q shodný s vrcholem polygonu p_i. Tato singularita je ošetřena přímým porovnáním souřadnic určovaného bodu s body tvořícími danou hranu. Pozice bodu je vypočtena metodou getPoinLinePosition.

Dále bylo nutné ošetřit situaci, ve které je bod q na hraně polygonu. V tomto případě je potřeba zjistit, zda – li bod q leží na přímce, která prochází hranou a zároveň se nenachází až za koncem hrany. Názorněji je problém popsán na obr. 1.



Obrázek 1: Problematická situace při určování polohy bodu na hraně.

```
int Algorithms::getPointLinePosition(QPoint &a,QPoint &p1,QPoint &p2)
{
...
double m = vx / ux;

//Point is on the vertex
if (((a.x()==p1.x()) && (a.y()==p1.y())) || ((a.x()==p2.x()) && (a.y()==p2.y())))
    return -2;

//Point in the left halfplane
else if (t > eps)
    return 1;

//Point in the right halfplane
else if (t < -eps)
    return 0;

//Point on the line and making sure that the point a is between p1 and p2
else if (t >= -eps and t <= eps and m >= 0 and m <= 1)
    return -1;
}
```

4.2 Ray Algorithm

Může nastat problém se singularitami, tudíž je vhodné použít upravenou variantu algoritmu, která je redukovaná k bodu q (popis výše v kapitole 3.2). Je provedena redukce do lokálního souřadnicového systému s počátkem právě v bodě q. Osa x' lokálního souřadnicového systému je zvolena rovnoběžně s osou x. Osa y' je kolmá na osu x'.

Řešení případu, kdy určovaný bod q je totožný s bodem tvořícím hranu polygonu, je stejné jako v předchozím případě (kapitola 4.1).

Jestliže bod q leží na hraně polygonu, je tento případ ošetřen následovně:

```
int Algorithms::getPositionRayCrossing(QPoint &q, std::vector<QPoint> &pol)
{
...
    for (int i = 1; i < n+1; i++)
    {
        ...
        double xi = pol[i].x() - q.x();
        double yi = pol[i].y() - q.y();

        double xr = pol[i-1].x() - q.x();
        double yr = pol[i-1].y() - q.y();

        double d = sqrt((xr-xi)*(xr-xi) + (yr-yi)*(yr-yi));
        double dq = sqrt((xr*xr + yr*yr)) + sqrt((xi*xi + yi*yi));

        //Point is on the border
        if(fabs(d-dq) < eps)
            return -1;

        //Find point position
        if (((yr > 0) && (yi <= 0)) || ((yi > 0) && (yr <= 0)))
        {
            double xm = (xi * yr - xr * yi) / (yi - yr);

            //Point is in the right halfplane
            if (xm > eps)
                k += 1;
        }
    }
    return k% 2;
}
```

5 Vstupní data

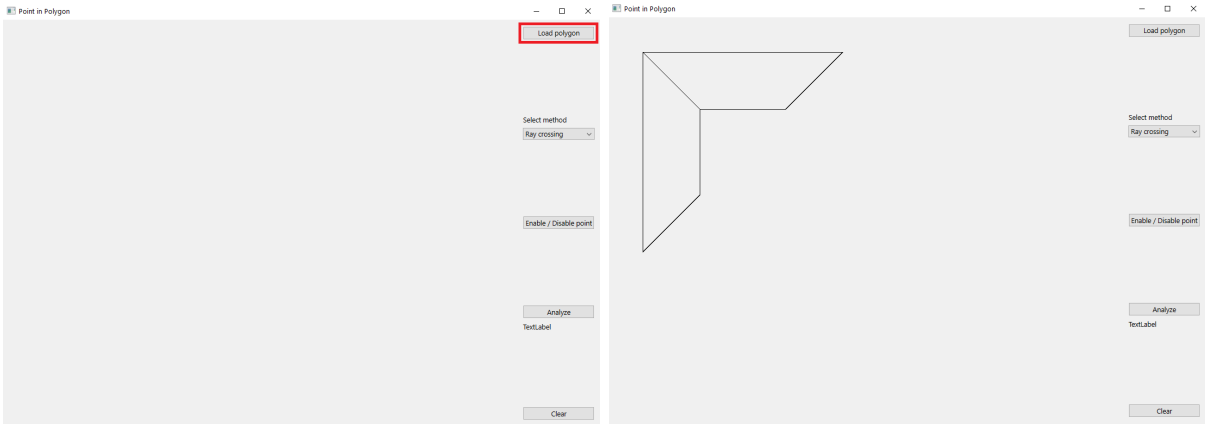
Vstupními daty jsou souřadnice jednotlivých polygonů, které se načítají do aplikace. Jedná se o textový soubor se třemi sloupci (ID bodu, X, Y) a tolika řádky, kolik je celkových bodů pro polygony. Souřadnice bodů jsou v lokálním souřadnicovém systému přímo pro aplikaci.

6 Výstupní data

Za výstup je považován výpis polohy bodu vůči polygonu v aplikaci a grafické znázornění bodu s polygony v okně aplikace.

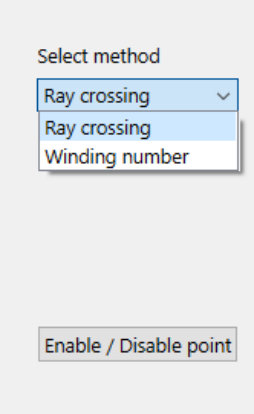
7 Snímky obrazovky vytvořené aplikace a její popis

Po zapnutí aplikace se zobrazí prázdné okno pouze s ikonami a výběrem algoritmu na pravé straně (obr. 2). Stiskem tlačítka *Load polygon* se otevře možnost vybrat textový soubor ve stanoveném formátu se souřadnicemi polygonů z disku. Tímto je import polygonů hotov.



Obrázek 2: Aplikace po zapnutí (vlevo) a po načtení polygonů (vpravo).

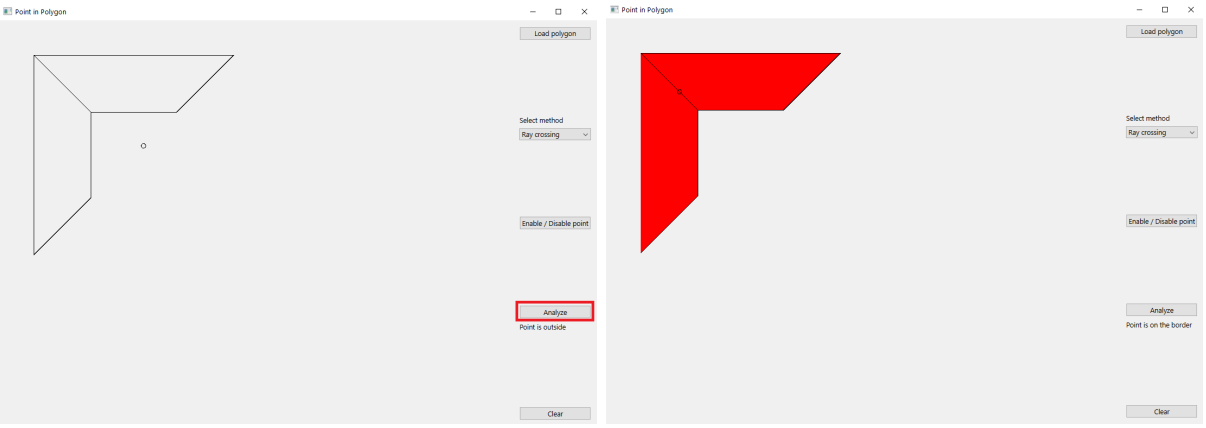
Následně je potřeba stisknout tlačítko *Enable/Disable point*, které umožňuje vkládání bodu do okna aplikace. Nyní je potřeba zvolit algoritmus pro určení polohy bodu přes otevření rozevírací nabídky pod názvem *Select method* (obr. 3). Defaultním nastavením je Ray crossing algoritmus.



Obrázek 3: Výběr algoritmu.

Pomocí tlačítka *Analyze* je provedeno vyhodnocení výpočtu. Vpravo dole je vypsán výsledek. Možnosti jsou následující:

- *Point is inside* $\longrightarrow$ Bod q se nachází uvnitř polygonu P.
- *Point is outside* $\longrightarrow$ Bod q se nachází mimo polygon P.
- *Point is on the border* $\longrightarrow$ Bod q leží na hraně polygonu P.
- *Point is on the vertex* $\longrightarrow$ Bod q leží na vrcholu polygonu P.



Obrázek 4: Analýza polohy bodu vůči polygonu.

Obrázek 5: Ukázka funkčnosti algoritmu pro výběr bodu na vrcholu dvou polygonů.

Při stisknutí tlačítka *Clear* se provede vymazání všech načtených polygonů.

8 Dokumentace

8.1 Třída algorithms

Tato třída obsahuje výpočetní vzorce pro použité algoritmy.
Třída algorithms obsahuje následující veřejné metody:

```
int getPointLinePosition(QPoint &a,QPoint &p1,QPoint &p2)
```

Metoda určuje, zda – li bod leží v levé či v pravé polorovině od přímky (hrany polygonu). Vstupními parametry jsou určovaný bod a a body p_1, p_2, které tvoří přímku.

```
double get2LinesAngle(QPoint &p1, QPoint &p2, QPoint &p3, QPoint &p4)
```

Metoda vypočte velikost úhlu, který svírají dvě přímky. První přímku tvoří body p_1, p_2 a druhou přímku body p_3, p_4.

```
int getPositionWinding(QPoint &q, std::vector<QPoint> &pol)
```

Metoda určuje polohu bodu vůči polygonu za pomocí algoritmu Winding Number. Vstupními parametry jsou souřadnice bodu q a souřadnice polygonů uložené ve vectoru *pol*.

```
int getPositionRayCrossing(QPoint &q, std::vector<QPoint> &pol)
```

Metoda určuje polohu bodu vůči polygonu za pomocí algoritmu Ray Crossing. Vstupními parametry jsou souřadnice bodu q a souřadnice polygonů uložené ve vectoru *pol*.

8.2 Třída draw

Tato třída umožňuje vykreslování bodu a polygonů.

Třída draw obsahuje následující privátní metody a proměnné:

```
QPoint q
```

Proměnná se souřadnicemi bodu q, jehož poloha vůči polygonu je určována.

```
std::vector<QPolygon> polygons
```

Proměnná, do které se ukládají načtené polygony z textového souboru.

```
bool enable_draw
```

Proměnná, která povoluje vykreslování bodu.

Třída draw obsahuje následující veřejné metody a proměnné:

```
explicit Draw(QWidget *parent = nullptr)
```

Prvotní vykreslení bodu q mimo okno aplikace.

```
void loadPolygon(std::string &path)
```

Metoda, která načítá polygony z vybraného textového souboru na disku.

```
void paintEvent(QPaintEvent *event)
```

Metoda, která vykresluje bod či polygony.

`void mousePressEvent(QMouseEvent *event)`

Metoda určující souřadnice určeného bodu.

`void clear()`

Metoda, která vymaže vybrané polygony z okna aplikace.

`void changeStatus(){enable_draw = !enable_draw;}`

Metoda, která mění status kreslení bodu.

`QPoint getPoint(){return q;}`

Vrací souřadnice bodu q.

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

Vrací polygony při analyzování pozice.

8.3 Třída widget

Tato třída propojuje uživatelské rozhraní aplikace s kódem. Je vytvořena v sekci *Design*.

Třída widget obsahuje následující privátní metody a proměnné:

`void on_pushButtonClear_clicked()`

Metoda, která určuje, co následuje po stisknutí tlačítka *Clear*.

`void on_pushButton_clicked()`

Metoda, která určuje, co následuje po stisknutí tlačítka *Enable/Disable point*.

`void on_pushButtonAnalyze_clicked()`

Metoda, která určuje, co následuje po stisknutí tlačítka *Analyze*.

`void on_pushButtonLoad_clicked()`

Metoda, která určuje, co následuje po stisknutí tlačítka *Load*.

9 Závěr

Cílem zadané úlohy bylo vytvoření aplikace, která uživateli umožňuje načíst souřadnice bodů z textového souboru ve stanoveném formátu a následně určit, zda – li leží uživatelem zadaný bod uvnitř, vně, na hranici nebo na vrcholu polygonu. K výpočtu polohy bodu vůči polygonu byly použity dva algoritmy: Winding Numbers a Ray Crossing. Výsledná aplikace umožňuje načíst vlastní polygony a určit všechny čtyři výše zmíněné polohy bodu.

9.1 Možné či neřešené problémy

Viz níže v kapitole 9.2.

9.2 Náměty na vylepšení

Použitý soubor s polygony slouží pouze pro ukázku, jistě by bylo možné jej dále rozvinout.

Dále by bylo možné implementovat například nastavení přesnosti při určování nebo zápis souřadnic čísly namísto klikání myší.

V Praze 12.11.2021

Bc. Pane Kuzmanov
Bc. František Mužík