České vysoké učení technické v Praze
Fakulta stavební
Katedra geomatiky

Technická zpráva

Algoritmy v digitální kartografii

Úloha č. 4: Množinové operace s polygony

Bc. Pane Kuzmanov
Bc. František Mužík

Studijní program: Geodézie a kartografie
Specializace: Geomatika

Praha 2021

Úloha č. 4: Množinové operace s polygony

Vstup: množina n polygonů P=$\{P_1, ..., P_n\}$.

Výstup: množina m polygonů $P' = \{P'_1, ..., P'_m\}$.

S využitím algoritmu pro množinové operace s polygony implementujte pro libovolné dva polygony $P_i, P_j \in P$ následující operace:

- Průnik polygonů $P_i \cap P_j$,
- Sjednocení polygonů $P_i \cup P_j$,
- Rozdíl polygonů: $P_i \cap \overline{P}_j$, resp. $P_j \cap \overline{P}_i$.

Jako vstupní data použijte existující kartografická data (např. konvertované shape fily) či syntetická data, která budou načítána z textového souboru ve Vámi zvoleném formátu.

Grafické rozhraní realizujte s využitím frameworku QT.

Při zpracování se snažte postihnout nejčastější singulární případy: společný vrchol, společná část segmentu, společný celý segment či více společných segmentů. Ošetřete situace, kdy výsledkem není 2D entita, ale 0D či 1D entita.

Pro výše uvedené účely je nutné mít řádně odladěny algoritmy z úlohy 1. Postup ošetření těchto případů diskutujte v technické zprávě, zamyslete se nad dalšími singularitami, které mohou nastat.

Hodnocení:

Krok	**Hodnocení**
Množinové operace: průnik, sjednocení, rozdíl	20b
Konstrukce offsetu (bufferu)	*+10b*
Výpočet průsečíků segmentů algoritmem Bentley & Ottman	*+8b*
Řešení pro polygony obsahující holes (otvory)	*+6b*
Max celkem:	**44b**

Čas zpracování: 2 týdny

1 Údaje o bonusových úlohách

1.1 Konstrukce offsetu (bufferu) (+10b)

1.2 Výpočet průsečíků segmentů algoritmem Bentley & Ottoman (+8b)

Tato bonusová úloha nebyla řešena.

1.3 Řešení pro polygony obsahující holes (otvory) (+6b)

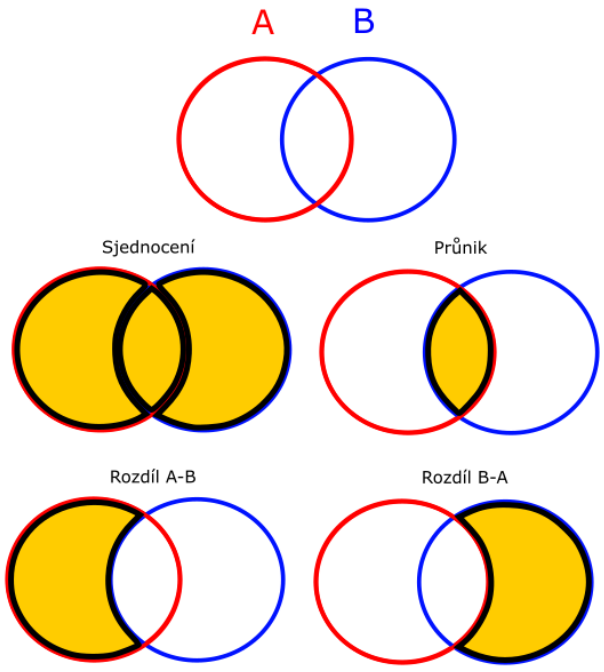
Tato bonusová úloha nebyla řešena.

2 Popis a rozbor problému

Nechť existují dva polygony A, B tvořené body P_1 až P_n. Pro tyto polygony je potřeba zjistit jejich vzájemnou polohu užitím následujících množinových operací: průnik $A \cap B$, sjednocení $A \cup B$ a rozdíl$(A - B, B - A)$ (viz. obr. 1). Uživateli je zobrazen výsledek zvolené množinové operace s využitím grafického zvýraznění částí polygonů.

Popis jednotlivých množinových operací včetně jejich implementace je podrobně rozepsán v kapitole 3.

Při řešení problému je potřeba postihnout některé singulární případy (viz. kapitola 4).



Obrázek 1: Množinové operace.

Popis množinových operací:

- **Sjednocení:**Výsledkem je taková množina, která obsahuje každý prvek, který se nachází alespoň jedné ze sjednocovaných množin, a žádné další prvky.

- **Průnik:** Výsledkem je taková množina, která obsahuje pouze ty prvky, které se nalézají v obou množinách a obsahuje všechny takové prvky.
- **Rozdíl:** Výsledkem je taková množina, která obsahuje každý prvek, který se nachází v první z množin, ale nenachází se ve druhé z nich, a žádné další prvky.

3 Popisy algoritmů formálním jazykem

3.1 Určení polohy bodu pomocí 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 Výpočet průsečíků polygonů A,B

Následující algoritmus slouží k výpočtu průsečíků dvou hran polygonů A,B. Tyto průsečíky jsou ukládány do mapy s klíčem α, β, které určují polohu průsečíku dvou hran a hodnotou danou průsečíkem. Po nalezení každého dalšího průsečíku je mapa aktualizována a body v ní seřazeny dle hodnot α, β. Závěrem je s užitím algoritmu Winding Number (viz. kapitola 3.1) určena poloha vrcholů jednoho polygonu vůči druhému (a naopak), z čehož poté vychází výsledky jednotlivých množinových operací, jež jsou sepsány v tabulce 1.
Implementace algoritmu:

1. $for(i = 0; i < n; i++)$
2. Vytvoření mapy: $M = map(double, QPointFBO)$
3. $for(j = 0; j < m; j++)$
4. Pokud existuje průsečík: $if b_{ij} = (p_i, p_{(i+1)\%m}) \cap (q_j, q_{(j+1)\%m}) \neq 0$
5. Přidání do mapy M: $M[\alpha_i] \leftarrow b_{ij}$
6. Zpracování prvního průsečíku pro e_j': $processIntersection(b_{ij}, \beta, B, j)$
7. Pokud jsou nalezeny nějaké průsečíky: $if(\|M\| > 0)$
8. Procházení všech průsečíků v mapě: $for(\forall m \in M)$:
9. Získání 2. hodnoty páru: $b \leftarrow m.second$
10. Zpracování aktuálního průsečíku pro e_i: $processIntersection(b, \alpha, A, i)$

processIntersection:

1. Jestliže ϵ je minimální hodnota: $if(|t| > \epsilon \;\&\&\; ||t| - 1| > \epsilon)$:
2. Inkrementace pozice: $i \leftarrow i + 1$
3. Přidání průsečíku na pozici i+1: $P \leftarrow (b, i)$

3.3 Ohodnocení hran

Algoritmus setEdgePositions rozděluje pozice středů hran vůči polygonu. Pro pozice bodů je využit algoritmus Winding Number (kapitola 3.1). Výsledkem je buď vnitřní (inner) nebo vnější (outer) pozice.
Implementace algoritmu:

1. $for(i = 0; i < n; i++)$, kde n je počet vrcholů polygonu A
2. Určení středu hrany $M(x_m, y_m)$:
3. $x_m = \frac{x_i + x_{i+1}}{2}$
4. $y_m = \frac{y_i + y_{i+1}}{2}$
5. Určení pozice vůči polygonu B: $pos = getPositionWinding(M, B)$
6. Aktualizováni pozice: $A(i) \leftarrow pos$

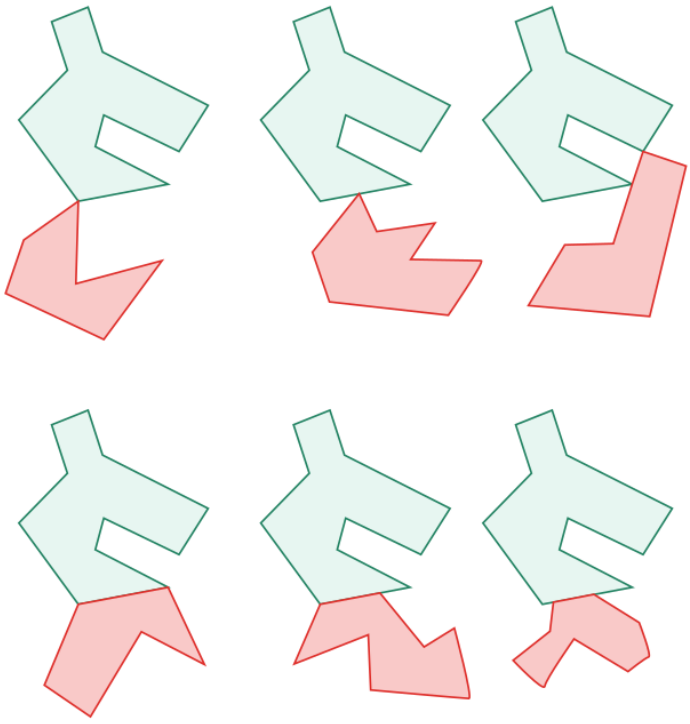
3.4 Množinové operace dle polohy bodů

Tabulka 1: Množinové operace

Operace	**Polygon A**	**Polygon B**
Sjednocení	Vnější	Vnější
Průnik	Vnitřní	Vnitřní
Rozdíl A-B	Vnější	Vnitřní
Rozdíl B-A	Vnitřní	Vnější

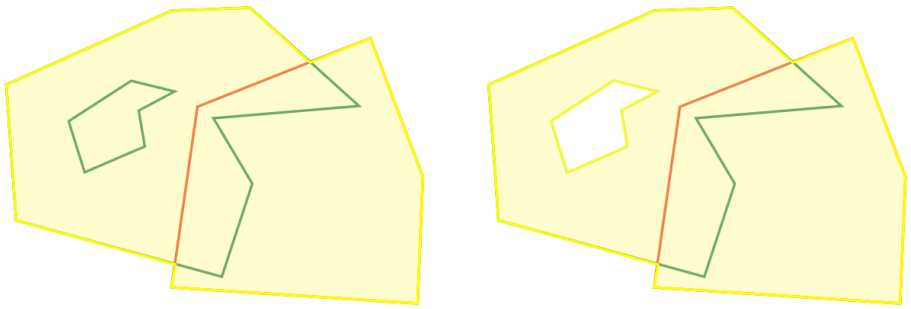
4 Problematické situace a jejich rozbor

Problematické situace nastávají, jestliže polygony mají společný jeden či více společných vrcholů, úseků hrany a nebo celé společné hrany. Ukázky je možné prohlédnout níže na obr. 2.



Obrázek 2: Problematické situace.

Další problém nastává u polygonů s otvory. Na obr. 3 je v levé části zřetelné chybné určení sjednocení zeleného a červeného polygonu, jestliže některý polygon (zde zelený) obsahuje otvor. Vpravo je správné určení. Aplikace bohužel neumí tento problém správně vyřešit, tato bonusová úloha nebyla řešena.



Obrázek 3: Problematické situace – polygony s otvory.

5 Vstupní data

Vstupními daty je textový soubor obsahující lomové body polygonů. Jedná se o generalizovaná data (s tolerancí 2 km) z datasetu ArcČR [2]. Konkrétně jde o Ústecký, Liberecký kraj a území CHKO České středohoří. Výběr a export dat proběhl v softwaru ArcGIS Pro [1]. Každý bod je zadaný id polygonu (Ústecký, Liberecký kraj či České středohoří) a souřadnicemi X,Y v modifikovaném systému JTSK:

```
1 736266 965664
1 729273 967523
1 727720 975632
...
2 736358 972787
2 739509 978352
2 732697 995088
...
3 705750 965801
3 702423 960319
3 690445 962329
...
```

6 Výstupní data

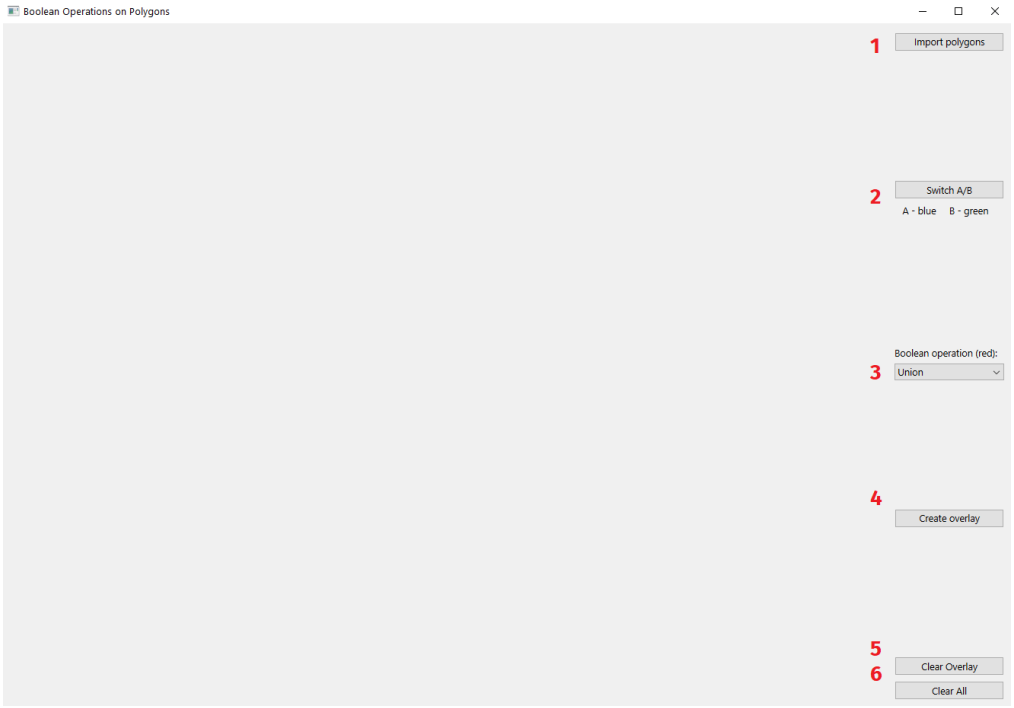
Za výstupní data je považováno grafické rozhraní vytvořené aplikace. S jejím využitím lze posuzovat vzájemné polohy dvou polygonů a výsledky množinových operací nad nimi.

7 Snímky obrazovky vytvořené aplikace a její popis

Na obrázku 4 je znázorněn popis uživatelského rozhraní aplikace v momentu po jejím spuštění. Tlačítka mají následující využití:

1. Import polygon umožňuje načtení polygonů z textového souboru
2. Pomocí Switch A/B lze změnit kreslení polygonů A nebo B
3. Combo box umožňující výběr dané množinové operace

4. Create overlay vykreslí hranice polygonu zvolené množinové operace
5. Clear overlay vymaže předchozí vykreslení
6. Clear all vymaže celé okno



Obrázek 4: Popis uživatelského rozhraní aplikace.

Obrázek 5: Aplikace po vykreslení tří polygonů.

Obrázek 6: Vyhodnocení sjednocení polygonů.

Obrázek 7: Vyhodnocení průniku polygonů.

Obrázek 8: Vyhodnocení rozdílu polygonů A-B.

Obrázek 9: Vyhodnocení rozdílu polygonů B-A.

8 Dokumentace

8.1 Třída Algorithms

Tato třída obsahuje následující veřejné metody:

```
TPointLinePosition getPointLinePosition
     (QPointFBO &a,QPointFBO &p1,QPointFBO &p2)
```

Určení pozice bodu a přímky.

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

Určení úhlu mezi dvěma přímkami.

```
TPointPolygonPosition getPositionWinding(QPointFBO &q, TPolygon &pol)
```

Určení pozice bodu a polygonu.

```
std::tuple<QPointFBO,T2LinesPosition> get2LinesIntersection
     (QPointFBO &p1, QPointFBO &p2, QPointFBO&p3, QPointFBO &p4)
```

Nalezení průsečíku dvou přímek.

```
void updatePolygons(TPolygon &A, TPolygon &B)
```

Aktualizace polygonů po nalezení průsečíků.

```
void processIntersection(QPointFBO &b, double t, int &index, TPolygon &P)
```

Vkládání společných bodů do polygonu.

```
void setEdgePositions(TPolygon &A, TPolygon &B)
```

Určení pozice hrany vůči polygonu.

```
void selectEdges(TPolygon &P, TPointPolygonPosition pos, TEdges &edges)
```

Výběr hran na základě pozice.

```
TEdges createOverlay(TPolygon &A, TPolygon &B, TBooleanOperation &op)
```

Vytvoření překryvné vrstvy polygonů na základě zvolené množinové operace.

8.2 Třída Draw

Tato třída obsahuje následující privátní proměnné:

```
TPolygon A, B
```

Oba zadané či nakreslené polygony.

```
TEdges res
```

Uchovávání vybraných hran.

```
bool addA
```

Změna pro kreslení polygonu A nebo B.

Tato třída obsahuje následující veřejné metody:

`void paintEvent(QPaintEvent *event)`

Vykreslování na Canvas.

`void mousePressEvent(QMouseEvent *event)`

Metoda pro sledování kliknutí myši.

`void switchSource(){addA = !addA;}`

Změna při kreslení polygonů A nebo B.

`void drawPolygon(TPolygon &pol, QPainter &qp)`

Vykreslování polygonů.

`TPolygon getA(){return A;}`

Předání polygonu A.

`TPolygon getB(){return B;}`

Předání polygonu B.

`void setEdges(TEdges &edg){res = edg;}`

Nastavení hran.

`void clear(){res.clear();}`

Vymazání obsahu mapového okna.

`void clearAll(){A.clear(); B.clear(); res.clear();}`

Vymazání celého obsahu mapového okna.

`void loadData(std::string &path, int height, int width)`

Načtení textového souboru se souřadnicemi polygonů.

8.3 Třída Edge

Tato třída obsahuje následující privátní proměnné:

`QPointFBO start, end`

Startovní a koncový bod hrany.

Tato třída obsahuje následující veřejné metody:

`void setStart(QPointFBO &start_){start=start_;}`

Nastavení počátečního bodu hrany.

`void setEnd(QPointFBO &end_){end=end_;}`

Nastavení koncového bodu hrany.

`QPointFBO getStart(){return start;}`

Vrácení počátečního bodu hrany.

`QPointFBO getEnd(){return end;}`

Vrácení koncového bodu hrany.

8.4 Třída QPointFBO

Tato třída obsahuje následující privátní proměnné:

`double alpha, beta`

Koeficienty α, β, které udávají hodnotu průsečíku hran.

`TPointPolygonPosition pos`

Pozice bodu a polygonu.

Tato třída obsahuje následující veřejné metody:

`double getAlpha(){return alpha;}`

Vrácení α.

`double getBeta(){return beta;}`

Vrácení β.

`TPointPolygonPosition getPosition(){return pos;}`

Předání pozice bodu a polygonu.

`void setAlpha(double alpha_){alpha=alpha_;}`

Nastavení α.

`void setBeta(double beta_){beta=beta_;}`

Nastavení β.

`void setPosition(TPointPolygonPosition pos_){pos=pos_;}`

Nastavení pozice bodu a polygonu.

8.5 Třída Types

Tato třída definuje následující typy proměnných:

```
typedef enum{
    LeftHP,
    RightHP,
    On
} TPointLinePosition
```

Definice možností průsečíku bodu a hrany.

```
typedef enum{
    Inner,
    Outer,
    Boundary
} TPointPolygonPosition
```

Definice možností průsečíku bodu a polygonu.

```
typedef enum {
    Union,
    Intersection,
    DifferenceA_B,
    DifferenceB_A
} TBooleanOperation
```

Definice možností množinových operací.

```
typedef enum{
    Colinear,
    Parallel,
    Intersect,
    NonIntersect
} T2LinesPosition
```

Definice možností průsečíku dvou hran.

```
typedef std::vector<QPointFBO> TPolygon
```

Definice polygonu, který se skládá z bodů QPointFBO.

```
typedef  std::vector<Edge> TEdges
```

Definice více za sebou jdoucích hran.

8.6 Třída Widget

Tato třída obsahuje následující privátní metody:

`void on_pushButton_clicked()`

Určení operace následující po stisku tlačítka Switch A/B.

`void on_pushButton_2_clicked()`

Určení operace následující po stisku tlačítka Create overlay.

`void on_pushButton_3_clicked()`

Určení operace následující po stisku tlačítka Clear.

`void on_pushButton_4_clicked()`

Určení operace následující po stisku tlačítka Clear All.

`void on_pushButton_Import_clicked()`

Určení operace následující po stisku tlačítka Import polygons.

9 Závěr

Byla vytvořena desktopová aplikace, umožňující vyhodnocení základních množinových operací mezi dvěma polygony A,B (sjednocení, průnik, rozdíl A-B, rozdíl B-A). Aplikace umí načítat souřadnice polygonů v souřadnicovém systému JTSK (EPSG:5514) ve formě textového souboru. Uživateli je umožněno polygony také vykreslit ručně. Výsledek množinové operace, zvolené z výběrové nabídky, je červeně zvýrazněn. Vývoj aplikace proběhl v programovacím jazyce C++.

9.1 Možné či neřešené problémy

Nebyly řešeny bonusové úlohy. Problémy mohou nastávat u polygonů s otvory. V těchto případech aplikace nesprávně vyhodnocuje množinové operace.

9.2 Náměty na vylepšení

K vylepšení se nabízí vyřešení problému s polygony, které mají otvory. Stávající verze aplikace neumí správně vyřešit některé takovéto situace. Řešení tohoto problému je součástí bonusové úlohy, která nebyla řešena.

V Praze 6.1.2022

Bc. Pane Kuzmanov
Bc. František Mužík

Použitá literatura

[1] ArcGis Pro. Základ systému ArcGIS. https://www.arcdata.cz/produkty/arcgis/desktopovy-gis/arcgis-pro [cit. 2021-12-03].

[2] ArcČR. Vybraná administrativní a statistická data o České republice. https://www.arcdata.cz/produkty/geograficka-data/arccr-4-0 [cit. 2021-12-26].