PŘÍRODOVĚDECKÁ
FAKULTA
Univerzita Karlova

Algoritmy v počítačové kartografii

Úloha č. 4
Množinové operace s polygony

Čelonk Marek, Sýkora Matúš

1. Zadání

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

Výstup: množina m polygonů $P = \{P_1',\ldots,P_n'\}$

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 $P_i \cap P_j$

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 framework QT.

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

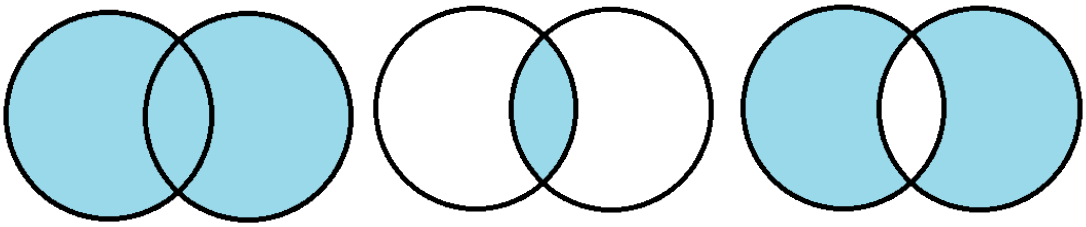
Pro výše uvedené účely je nutné mít řádně odladěny algoritmy z úlohy 1. Postup ošetření těchto případů diskutujte v technické zprávě, zamyslete se nad dalšími singularitami, které mohou nastat.

2. Popis a rozbor problému + vzorce

V prostředí GIS jsou množinové operace základními metodami pro analýzu polygonů, či jejich editaci. Jedná se například o polygony reprezentující plošné jevy ve fyzické geografii, jako je například krajinný pokryv (vodní plochy, lesy, pole) nebo jevy v sociální geografii, jako je například administrativní členění (obce, kraje, státy).

Základní operace (viz obr. č. 1), které lze provádět při práci s polygony jsou

- Sjednocení (Union)
- Průnik (Intersection)
- Rozdíl (Difference)



obr. č. 1: Základní množinové operace (zleva sjednocení, průnik, rozdíl)

Tyto operace lze zapsat pomocí booleovské algebry. V případě sjednocení se jedná o zápis $C = A \cup B$ a výsledkem je spojení obou polygonů. Zápis funkce průnik je $C = A \cap B$ a výsledkem je část polygonu totožná pro oba vstupní polygony. Poslední funkcí je rozdíl, který lze zapsat jako $C = A \Delta B$. Výsledek je opak funkce průniku, jedná se tedy o polygon, či polygony, které nejsou totožné pro oba vstupní polygony.

3. Popisy algoritmů formálním jazykem

Vytvorený algoritmus je zostavený pre nekonvexné polygóny ktoré aplikácia vygeneruje. Ako vstupné dáta sa používajú dva polygóny (A,B). Zložitosť algoritmu je O(m*n)

Medzi hlavné body algoritmu patria

- Výpočet priesečníkov vstupných polygónov
- Aktualizácia polygónov
- Ohodnotenie vrcholov množín A,B podľa pozície B voči A
- Výber vrcholov podľa ohodnotenia a zostavenie fragmentov

Výpočet priesečníkov vstupných polygónov

Najprv algoritmus vypočíta priesečníky vstupných polygónov A a B. Využili sme naivný algoritmus, ktorý testuje kombinácie všetkých strán $e_i \cap e'_j$, kde $e_i = (p_i, p_{i+1})$ a $e_j = (q_j, q_{j+1})$. Predstavuje najzložitejšiu časť celého výpočtu

Výpočet priesečníku hrán

$$b_{ij} = p_i + \alpha\mathbf{u} = p_j + \beta\mathbf{v}$$

Parametre α a β definujú priesečník v smere vektorov **u, v.**

Výpočet smerových vektorov

$$\mathbf{u} = (x_{i+1} - x_i,\ y_{i+1} - y_i),$$
$$\mathbf{v} = (x_{j+1} - x_j,\ y_{j+1} - y_j),$$
$$\mathbf{w} = (x_i - x_j,\ y_i - y_j).,$$

Zo smerových vektorov vypočítame koeficienty k_1, k_2 a k_3.

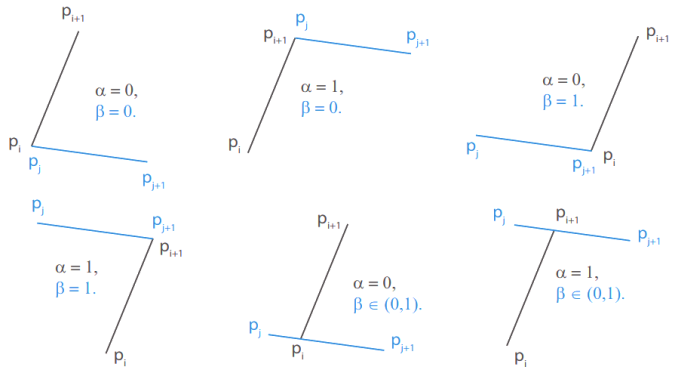
$$k_1 = \mathbf{v_x} \times \mathbf{w_y} - \mathbf{v_y} \times \mathbf{w_x},$$
$$k_2 = \mathbf{u_x} \times \mathbf{w_y} - \mathbf{u_y} \times \mathbf{w_x},$$
$$k_3 = \mathbf{v_y} \times \mathbf{u_x} - \mathbf{v_x} \times \mathbf{u_y}.$$

z vypočítaných koeficientov vypočítame parametre α a β:

$$\alpha = (k_1 / k_3),$$

$$\beta = (k_2 / k_3).$$

koeficienty α a β nadobúdajú hodnoty od (0,1), Na nasledujúcom obrázku sú znázornené rôzne situácie,



obr. č. 2: Hodnoty parametrov α a β zdroj: Bayer (2018)

Aktualizácia polygónov

Následne je nutné vypočítane priesečníky pridať do oboch polygónov, preto je testovaná každá hrana či obsahuje daný priesečník. V miestach priesečníku je hrana rozdelená, pre viacnásobne rozdelenie hrany je potrebné poznať hodnoty parametrov α a β pomocou ktorých sa následne zoradia priesečníky na hrane za sebou

Zápis aktualizácie polygónov

```
Edge e, List<IntersectionPoint> points
List<IntersectionPoint> myInts;

for (IntersectionPoint pt : points) {
        if (pt.e1 == e || pt.e2 == e) {
                myInts.add(pt);
        }
}
Collections.sort(myInts, (IntersectionPoint t, IntersectionPoint t1) -> {
        double c1, c2;
        if (t.e1 == e) {
                c1 = t.alpha;
        } else {
                c1 = t.beta;
        }
        if (t1.e1 == e) {
                c2 = t1.alpha;
        } else {
                c2 = t1.beta;
```

```
            }
            if (c1 < c2) {
                        return -1;
            } else if (c1 > c2) {
                        return 1;
            }
            return 0;
})
List<Edge> divided;
if (myInts.size == 0) {
            divided.add(e);
            return divided;
}
divided.add(new Edge(e.start, myInts.get(0).point));
for (int i = 0; i < myInts.size() - 1; i++) {
            divided.add(new Edge(myInts.get(i).point, myInts.get(i + 1).point));
}
divided.add(new Edge(myInts.get(myInts.size() - 1).point, e.end));
return divided;
```

Ohodnotenie vrcholov množín A,B podľa pozície B voči A

Ohodnotenie ľubovoľnej hrany $e_i = (p_i, p_{i+1})$k polygónu B je určené polohou ľubovoľného vnútorného bodu, napr. stredu hrany:

$p_{i\text{-}s} = 0.5\ (p_i + p_{i+1})$,
$q_{i\text{-}s} = 0.5\ (q_j + q_{j+1})$.

každému stredu hrany ohodnotenie vzhľadom k polygónu B a následne opačne voči polygónu A. Následne možno hranám pridať informáciu o jej polohe:

Ohodnotenie $p_{i\text{-}s}$ voči polygonu B:
$p_{i\text{-}s} \in B$ … uvnitř (inside),
$p_{i\text{-}s} \in \partial B$ … na hranici (boundary),
$p_{i\text{-}s} \notin B$ … vně (outside).

Ohodnotenie $q_{j\text{-}s}$ voči polygonu A:
$q_{j\text{-}s} \in A$ … uvnitř (inside),
$q_{j\text{-}s} \in \partial A$ … na hranici (boundary),
$q_{j\text{-}s} \notin A$ … vně (outside).

Zápis ohodnotenia vrcholov

```
Edge e;
Point2D middle;
middle = new Point2D(
            (e.start.getX() + e.end.getX()) / 2,
            (e.start.getY() + e.end.getY()) / 2);
e.side = pointPolygonWinding(middle, poly);

function PositionEnum pointPolygonWinding (Point2D pt, Polygon poly) {
            double sumAngle = 0;
            for (Edge e : poly.edges) {
                        ux = e.start.getX - pt.getX;
                        uy = e.start.getY - pt.getY;
                        vx = e.end.getX - pt.getX;
                        vy = e.end.getY - pt.getY;
                        if (orientation of (pt, e.start, e.end) == OrientationEnum.CCW) {
                                    sumAngle = sumAngle - angle(ux, uy, vx, vy);
                        } else {
                                    sumAngle = sumAngle + angle(ux, uy, vx, vy);
                        }
```

```
        }
        if (abs(sumAngle) >= (2 * PI))
                return PositionEnum.INSIDE;
        return PositionEnum.OUTSIDE;
}
```

Výber vrcholov podľa ohodnotenia a zostavenie fragmentov

Na zostavenie jednotlivých fragmentov využijeme ohodnotenie hrán z predchádzajúceho kroku, ktoré tvoria konečnú množinu hrán výsledného polygónu podľa vybraného typu množinovej operácie. Všetky fragmenty začínajú a končia priesečníkom.

Ohodnotenie hrán polygónov pre jednotlivé operácie sú zobrazené v tabuľke č.1

Množinová operácia	A	B
C = A ∩ B	Inner	Inner
C = A ∪ B	Outer	Outer
C = A ∩	Outer	Inner
C = B ∩ A	Inner	Outer
C = A Δ B	Inner + Outer	Inner + Outer

tab. č. 1: ohodnotené hrany

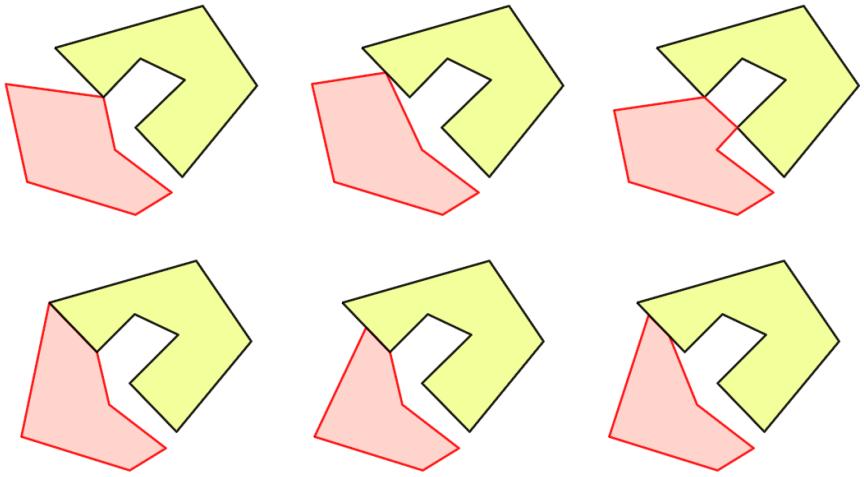
Zápis výberu vrcholov podľa ohodnotenia a zostavenie fragmentov

```
List<Edge> edges
List<Polygon> out = new LinkedList;
Polygon poly = new Polygon;

if (edges.isEmpty)
        return out;

Edge cure = edges.get(0);
edges.remove(0);
poly.edges.add(cure);

while (!edges.isEmpty) {
        Edge next = null;
        for (Edge e : edges) {
                if (cure.end.equals(e.start)) {
                        next = e;
                        break;
                }
                else if (cure.end.equals(e.end)) {
                        e.swap; // swap start and end point
                        next = e;
                        break;
                }
        }

        if (next.end.equals(poly.edges.get(0).start)) {
                poly.edges.add(next);
                edges.remove(next);

                if (orientation of (poly) == OrientationEnum.CCW) {
                        poly.side = PositionEnum.OUTSIDE;
                } else {
                        poly.side = PositionEnum.INSIDE;
                }

                out.add(poly);
                poly = new Polygon;

                if (!edges.isEmpty) {
                        cure = edges.get(0);
                        edges.remove(cure);
                        poly.edges.add(cure);
```

```
                    }
                } else {
                    edges.remove(next);
                    poly.edges.add(next);
                    cure = next;
                }
            }
            return out;

            function polyUnion(Polygon polyA, Polygon polyB) {
                List<Edge> edges = new LinkedList;

                edges.addAll(filterEdges(polyA, PositionEnum.OUTSIDE));
                edges.addAll(filterEdges(polyB, PositionEnum.OUTSIDE));

                return buildRings(edges);
            }
```

4. Problematické situace a jejich rozbor

Mezi problematické situace různé singularity, které mohou při práci s polygony vzniknout. Jedná se například o polygony, které mají společný jeden či více vrcholů. Problematická situace nastává, i když se polygony dotýkají jednou, či více stran nebo pouze částí strany.



obr. č. 3: Problematické situace mezi polygony, zdroj: Bayer (2018)

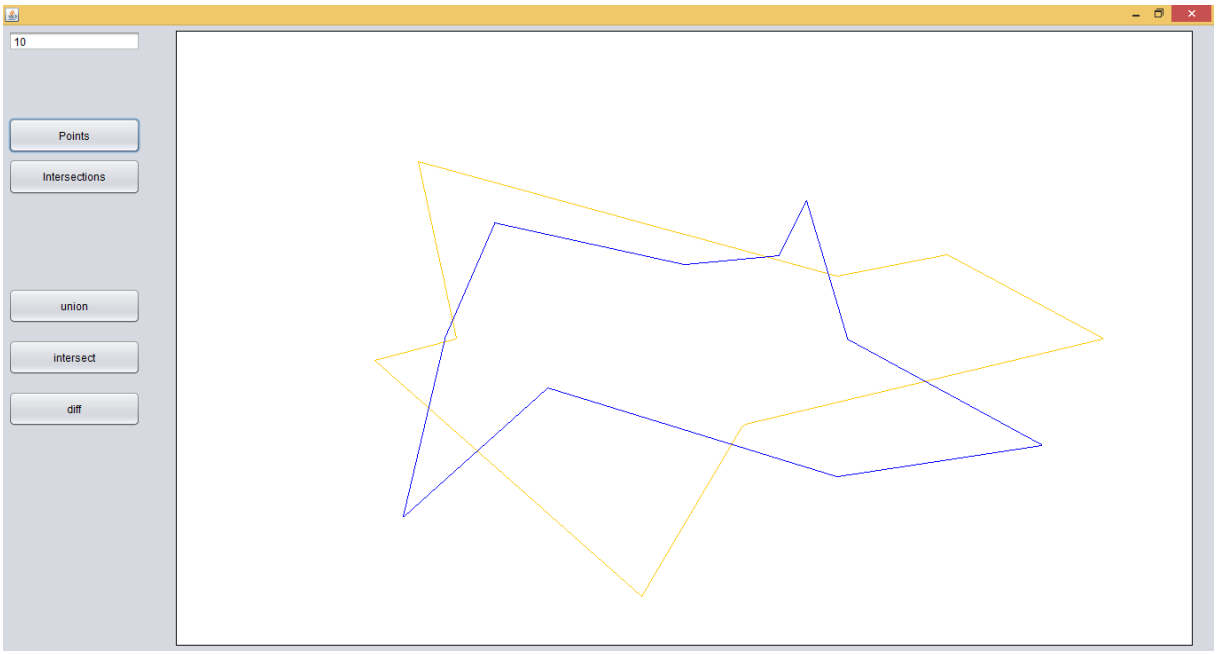
5. Vstupní data, formát vstupních dat, popis

Prvotními vstupními daty je množina náhodných bodů se souřadnicemi x a y. Je využita třída *Point2D*. Tyto body jsou propojeny liniemi ze třídy *Edge* a vzniknou dva polygony třídy *Polygon*. Výsledným vstup jsou tedy dva různé polygony vytvořené z náhodných bodů.

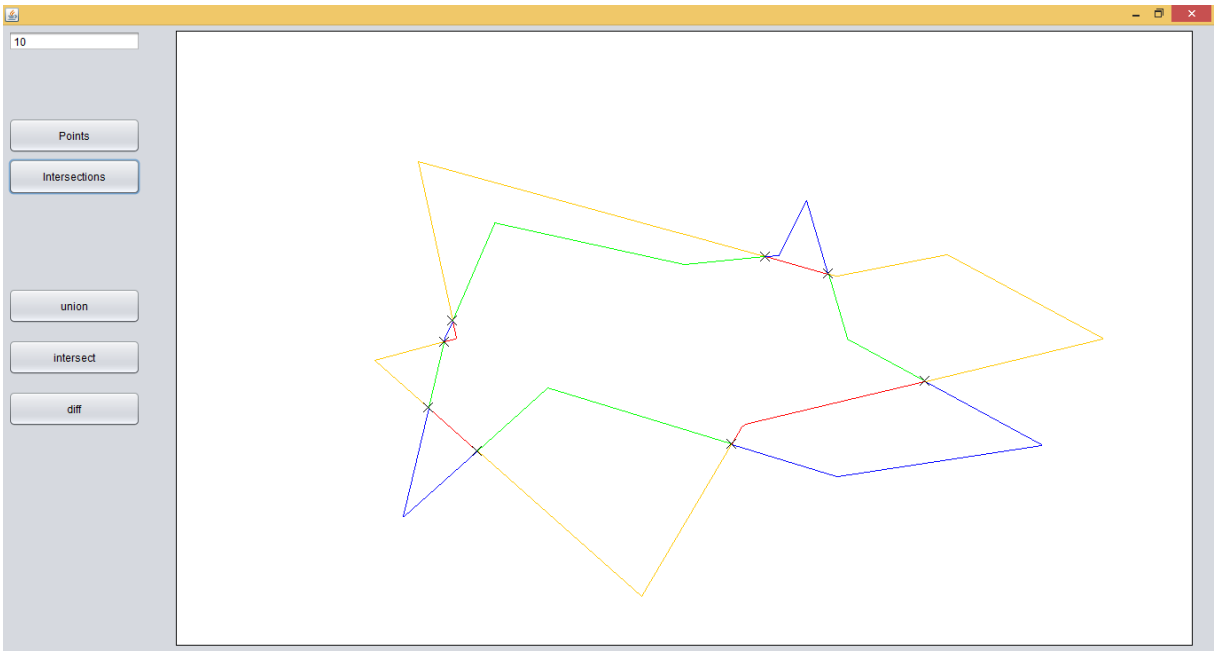
6. Výstupní data, formát výstupních dat, popis

Hlavním výstupem je vizuální zobrazení výsledků vzniklých z použitých funkcí. Toho je docíleno pomocí vykreslení vyplněných polygonů. Výsledné polygony jsou třídy *Polygon*. Pro výsledek funkcí *Union* a *Intersect* jsou použity polygony s tyrkysovou výplní. V případě výsledku funkce *Difference* je pro každý polygon použita odlišná barva (tyrkysová a žlutá)

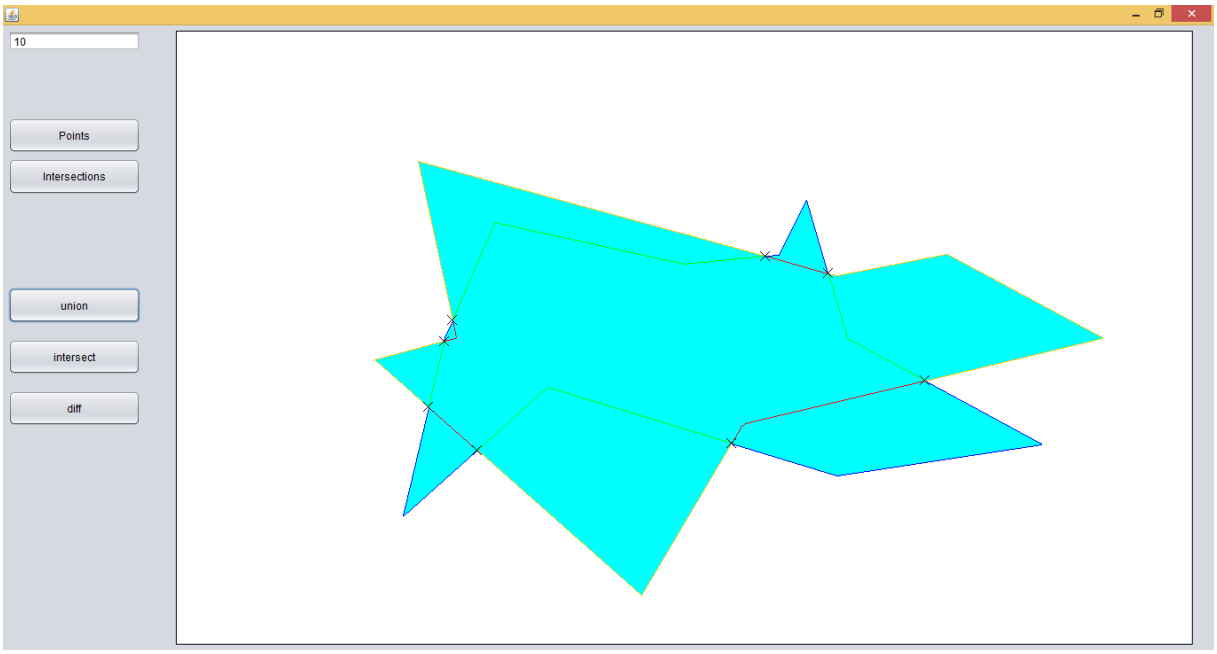
7. Printscreen vytvořené aplikace



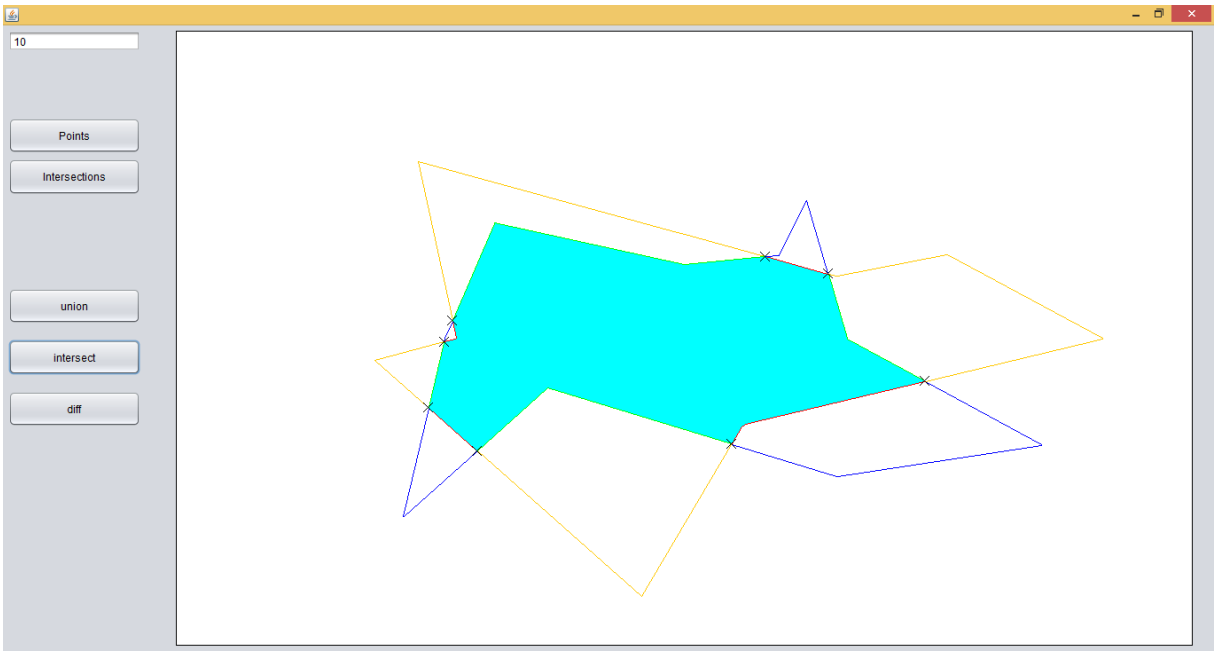
obr. č. 4: Náhodně vygenerované dva polygony



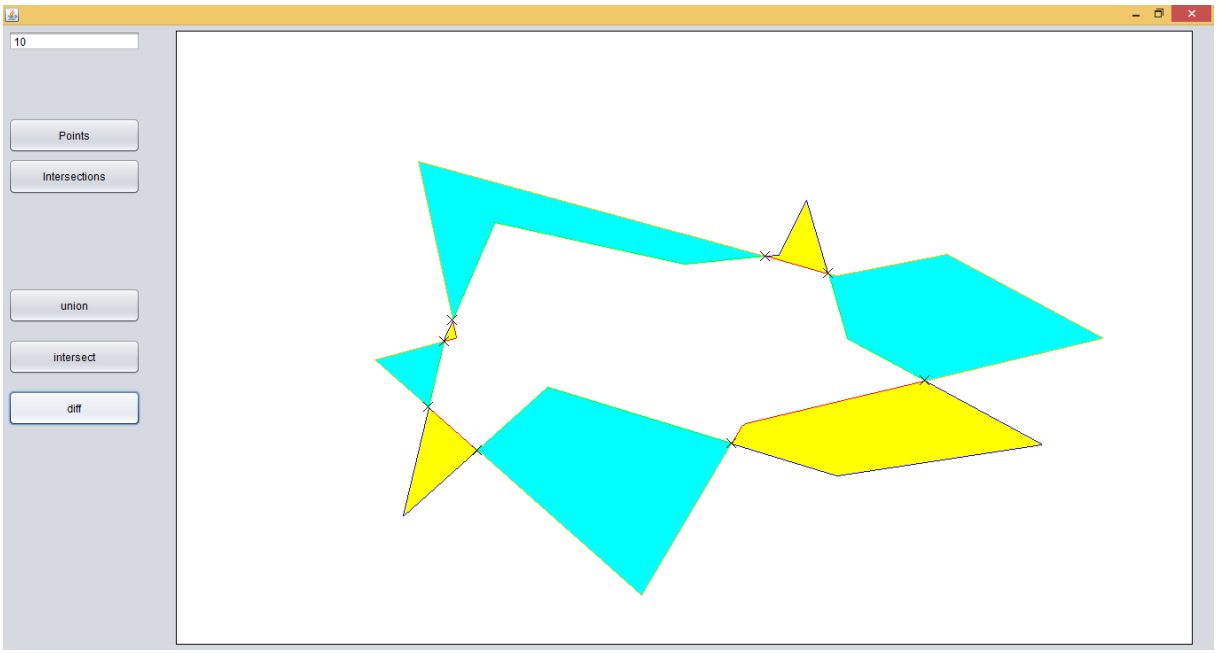
obr. č. 5: Nalezení protínajících se hran obou polygonů



obr. č. 6: Výsledek funkce Union



obr. č. 7: Výsledek funkce Intersect



obr. č. 8: Výsledek funkce Difference

8. Dokumentaci: popis tříd, datových položek a jednotlivých metod

Vytvořená aplikace obsahuje celkem šest tříd. Všechny použité algoritmy a výpočty jsou obsaženy v třídě *Algorithms*. Třída *GUI* představuje uživatelské rozhraní a slouží k volání jednotlivých funkcí. Třída *drawPanel* umožňuje vykreslení vygenerovaných polygonů a jednotlivých výsledků. Další zbývající třídy (*Edge, IntersectionPoint, Polygon)* patří vytvořeným objektům hrany, průsečíku a polygonu.

Třída *Algorithms* se skládá celkem ze sedmnácti metod a třech proměnných. V případě proměnných se jedná o *Epsilon* datového typu *double*, která reprezentuje hraniční přesnost při výpočtu orientace jednoho bodu vůči dalším dvěma tvořícím přímku. *OrientationEnum* datového typu *enum* definuje možné hodnoty orientace jednoho bodu vůči dalším dvěma. *PositionEnum* datového typu *enum* definuje možné hodnoty polohy hrany vůči druhému polygonu.

Dále je sepsán jednoduchý popis použitých metod:

- *allIntersection* – výpočet všech průsečíků
- *angle* – výpočet úhlu dvou vektorů
- *buildRings* – seřazení hran do správného pořadí
- *calcIntersection* – výpočet průsečíků pouze dvou hran
- *divideAll* – rozdělení všech hran na části podle průsečíků
- *divideEdge* – rozdělení hrany podle průsečíků
- *dotProd* – výpočet skalárního součinu
- *filterEdges* – vložení hran do seznamu podle polohového atributu
- *getOrientation* – zjištění orientace bodu vůči dvěma předchozím bodům
- *getPolygonOrientation* – zjištění orientace bodů polygonu
- *len* – výpočet délky vektrou
- *pointPolygonWinding* – zjištění zda bod leží na hraně
- *polyDiff* – výpočet rozdílu
- *polyIntersect* – výpočet průniku
- *polyUnion* – výpočet sjednocení
- *setInside* – přiřazení objektu hrany, zda se nachází v polygonu

- *setInside* – přiřazení všem objektům hran, zda se nachází uvnitř polygonu

Jak bylo zmíněno výše, třída GUI představuje uživatelské rozhraní a nachází se v ní metody pro práci s komponenty uživatelského rozhraní jako je textové okno, tlačítko nebo vykreslovací panel. Jednotlivá tlačítka volají metody zapsané ve třídě *Algorithms.* Samostatnou metodou je *generateStar*, která vytváří polygony ve vykreslovacím okně.

Ve třídě *drawPanel* se nachází celkem pět metod a pět proměnných. Mezi metody patří načtení komponentů *initComponenst*, vykreslení požadované grafiky *drawComponents*. Metoda *affineTransform* provede afiní trasformaci za účelem správného vykreslení. Další funkce *drawPoly* převede informace z datového typu *Polygon* do *Path2D*, poslední metoda *drawSinglePoly* vytvoří polygon datového typu *Path2D*. Mezi proměnné patří *polyA* a *polyB*, jedná se o dva vstupní polygony datového typu *Polygon*. Proměnná *intersec* je list objektů datového typu *IntersectionPoint* a obsahuje průsečíky hran. Poslední dvě proměnné jsou *polyOut* a *PolyOut2*, které představují list obsahující objekty datového typu *Polygon* a jedná se o výsledné polygony.

Třída *Edge* obsahuje jednu metodu a tři proměnné. Použitá metoda je *swap*, která změní orientaci hrany. Proměnné *start* a *end* jsou datového typu *Point2D* a obsahují počáteční a koncový bod, který tvoří hranu. Proměnná *side* je datového typu *enum* podle *Algorithms.PositionEnum* a nese informaci o poloze hrany vůči druhému polygonu.

Třída *IntersectionPoint* obsahuje pouze pět proměnných. První je *point* datového typu *Point2D* a obsahuje informaci o poloze průsečíku. Další jsou *e1* a *e2* datového typu *Edge*, které nesou informaci o hranách, které tento průsečík rozdělením hrany vytvoří. Proměnné *alpha* a *beta* datového typu *double* obsahují hodnoty stejnojmenných parametrů.

Poslední třídou je třída *Polygon* obsahující proměnnou *edges*, která plní list datového typu *Edge*. Jedná se o všechny hrany tvořící polygon. Druhou proměnnou je *side* datového typu *enum* podle *Algorithms.PositionEnum*, která nese informaci o orientaci.

9. Závěr, možné či neřešené problémy, náměty na vylepšení

Vytvořená aplikace splňuje základní nároky a umožňuje použití množinových operací jako je sjednocení, průnik nebo rozdíl. Aplikace by v budoucnu mohla být vylepšena o řešení singulárních situací nebo při řešení polygonů obsahující holes (otvory). Dalším možným vylepšením by bylo rozšíření uživatelského rozhraní o možnost nahrání vlastích polygonů.

10. Seznam literatury

BAYER, Tomáš. Operace s uzavřenými oblastmi v GIS: Booleovské operace s uzavřenými oblastmi. Minkowského suma. Offset polygonu. [online]. [citováno 2019-1-14]. Praha, 2018. Dostupné z: https://web.natur.cuni.cz/~bayertom/images/courses/Adk/adk9.pdf