České vysoké učení technické v Praze

Fakulta stavební



Algoritmy v digitální kartografii

Úloha č. 4: Množinové operace s polygony

Skupina:

Sabina Kličková

Martin Vajner

Zimní semestr 2021/2022

Obsah

[1. Zadání 3](#_Toc92705384)

[2. Bonusové úlohy 3](#_Toc92705385)

[3. Popis a rozbor problémů 3](#_Toc92705386)

[4. Popisy algoritmů 4](#_Toc92705387)

[6. Vstupní data, formát vstupních dat, popis. 5](#_Toc92705388)

[7. Výstupní data, formát výstupních dat, popis 6](#_Toc92705389)

[Výstupem úlohy je aplikace, ve které lze vyhodnocovat vzájemnou polohu dvou polygonů spolu s výsledky množinových operací. **Chyba! Záložka není definována.**](#_Toc92705390)

[8. Dokumentaci: popis tříd, datových položek a jednotlivých metod 8](#_Toc92705391)

[10. Citovaná literatura 10](#_Toc92705392)

[11. Seznam obrázků 10](#_Toc92705393)

# Zadání

Vstup: množina n polygonů P={P1, ..., Pn}.

Výstup: množina m polygonů P 0 = {P 0 1 , ..., P0 m}.

S využitím algoritmu pro množinové operace s polygony implementujte pro libovolné dva polygony Pi , Pj ∈ P následující operace:

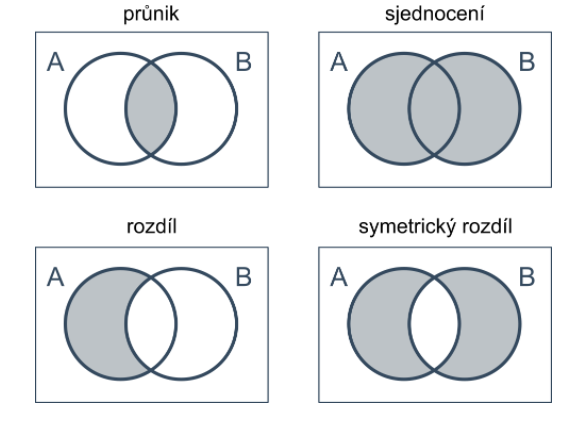
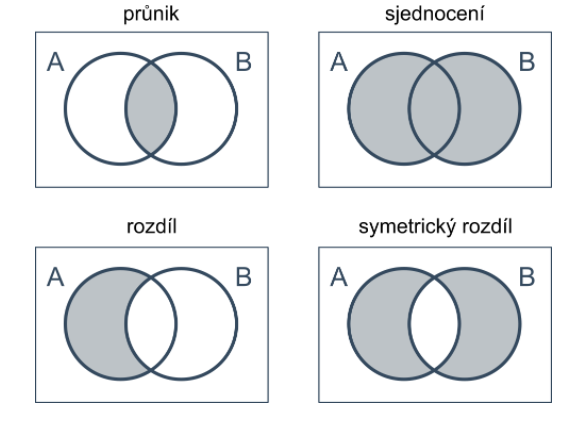
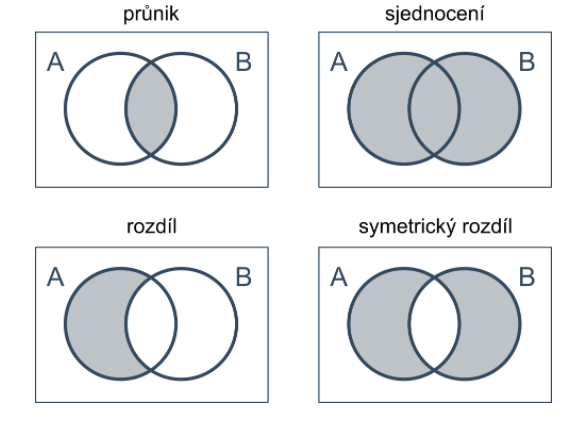
* Průnik polygonů Pi ∩ Pj ,
* Sjednocení polygonů Pi ∪ Pj ,
* Rozdíl polygonů: Pi ∩ Pj , resp. Pj ∩ Pi .

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.

|  |  |
| --- | --- |
| Množinové operace, průnik, sjednocení, rozdíl | +20b |

# Bonusové úlohy

V této úloze nebyly zpracovány žádné bonusové úlohy:



1. Popis a rozbor problémů

Existují-li dva polygony Pi,Pj, je třeba zjistit jejich vzájemnou polohu. K tomu se použijí následující množinové operace:

* Průnik Pi ∩ Pj ,
* Sjednocení Pi ∪ Pj ,
* Rozdíl p: Pi ∩ Pj , resp. Pj ∩ Pi .

Popis jednotlivých operací:

* **Průnik:** Výsledkem operace je množina, která obsahuje pouze prvky, které se nachází v obou množinách zároveň.
* **Sjednocení:** Výsledkem operace je množina, která obsahuje všechny prvky, nacházející se alespoň v jedné z množin, se kterými pracujeme.

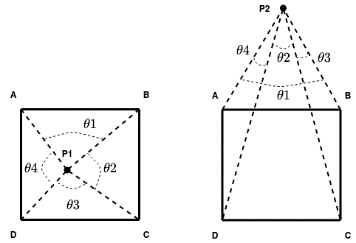
Možinové operace

* **Rozdíl:** Výsledkem operace je množina, která obsahuje prvky nacházející se v jedné množině, ale nenacházející se v te druhé.

1. Popisy algoritmů
2. **Winding Number**

Algoritmus Winding Number je definován jako počet rotací křivky, či v našem případě polygonu, v jednom směru okolo daného bodu q. Výpočet je založen na zjištění hodnot úhlů ω mezi daným bodem q a body polygonu a na poloze bodu q vůči segmentu polygonu. Pro všechny segmenty je třeba zjistit, jestli se bod nachází nalevo, či napravo. Pokud jsou nalevo, tak se úhly přičítají, pokud napravo tak se odečítají. Toto platí pro směr ccv (counter clockwise). Pro opačný směr se znaménka prohodí. Pokud se suma všech úhlů nerovná 0, leží bod uvnitř a pokud je suma rovna 0, leží bod mimo polygon. (1)

Princip algoritmu:

1. Inicializuj Ω=0, tolerance ε.

1

Princip Winding Number algoritmu 1

Princip Winding Number algoritmu

1. Opakuj pro ϒ trojici (pi, qi, pi+1)

Urči polohu q vzhledem k p = (pi, pi+1)

Urči úhel ωi = ∠pi, qi, pi+1

If , pak Ω = Ω + ωi //Bod v levé polorovině

Else Ω = Ω - ωi //Bod v pravé polorovině

1. If ||ω|-2π| < ε, pak
2. **Výpočet prúsečíku polygonů A,B**

Algoritmus slouží k výpočtu polohy průsečiku hran polygonů. Ty jsou ukládány společně s hodnotami α,β, které určují polohu a hodnotu danou průsečákem. S každým dalším nalezeým bodem je seznam aktualizován a řazen podle hodnot α,β. Poledním krokem je implementace algoritmu Winding number, který určuje vzájemnou polohu vrcholů polygonů.

Princip algoritmu:

1. for(i=0; i<n; i++)
   1. Vytvoření mapy: M=map(double,QPointFBO)
   2. for(j=0; j<m; j++)
      1. Pokud existuje průsečík:
      2. Přidání do mapy M:
      3. Zpracování prvního průsečíku pro :processIntersection(
2. Pokud jsou nalezeny nějaké průsečíky: if()
   1. Procházení všech průsečíků v mapě: for():
      1. Získání 2. hodnoty páru: b<-m.druhá
      2. Zpracování aktuálního průsečíku pro : $processIntersection()

**processIntersection:**

1. Jestliže je minimální hodnota: if()
   1. Inkrementace pozice: i<-i+1
   2. Přidání průsečíku na pozici i+1: P<-(b,i)
2. Ohodnocení hran

Pro ohodnocení hran byl vytvořen algoritmus setEdgePositions. Rozdeluje pozice středů hran ku polygonu na vnitřní či vnější.

1. for(i=0; i<n; i++) n..počet vrcholů prvního polygonu
2. Určení středu hrany :
   1. Určení pozice: pos=GetPositionWinding(M,B)
   2. Úprava pozice A(i)<- pos
3. Problematické situace

Problematickou situací je stav, kdy dva vstupní polygony mají právě jeden/jednu, čí více společných vrcholů nebo hran.

Ukázka problematické situace

Další problematickou situací mohou být nekonvexní polygony s „otvorem“. U těch může docházek k chybnému vyhodnocení množinových operací. Výsledek napravo je ten správný. Naše aplikace není schopna tento problém ovšem řešit, jelikož tato bonusová úloha nebyla řešena.

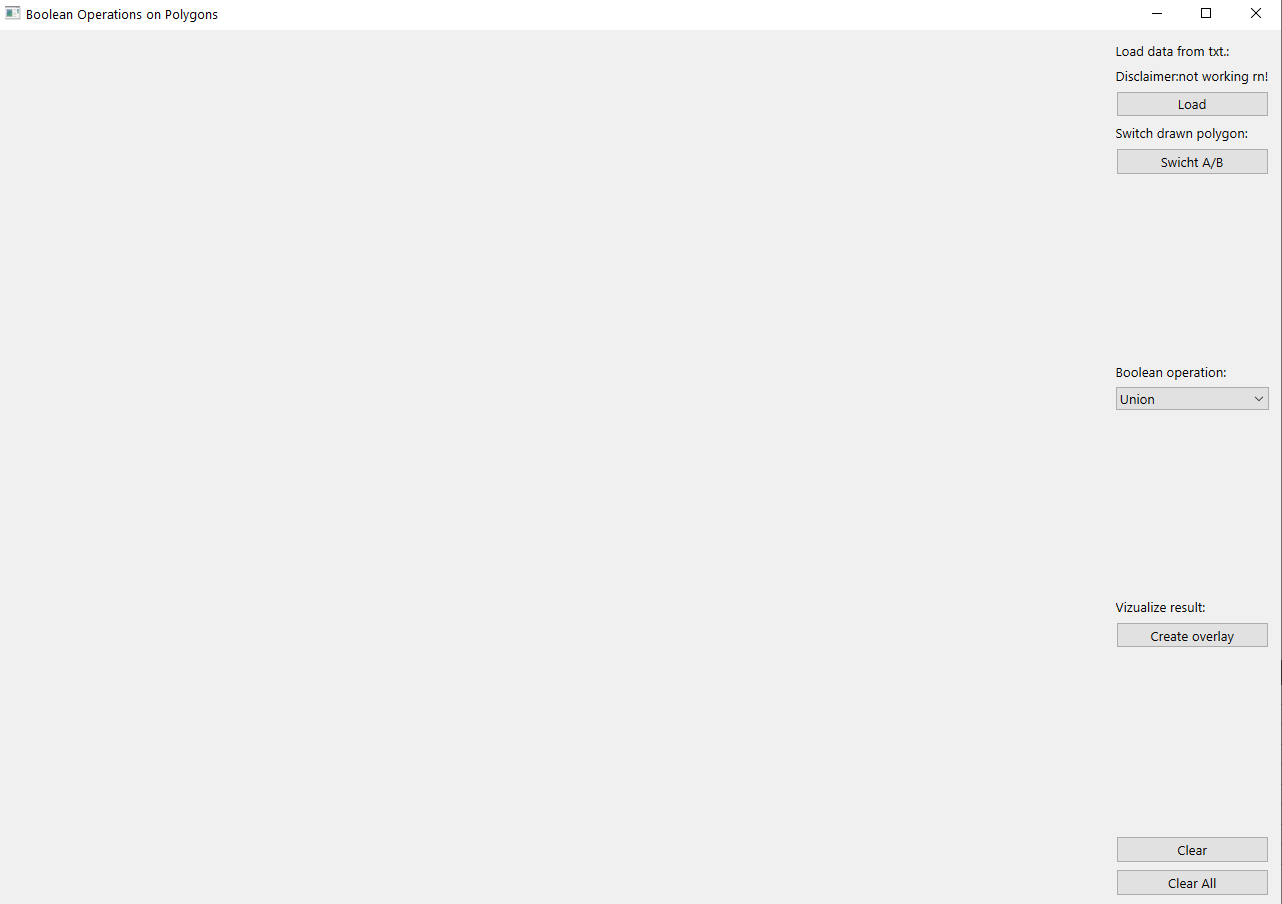
Ukázka problematické situace

1. Vstupní data, formát vstupních dat, popis.

Vstupní data jsou datového typu bod. V současné době lze zadávat data pouze ručním kreslením.

1. Výstupní data, formát výstupních dat, popis

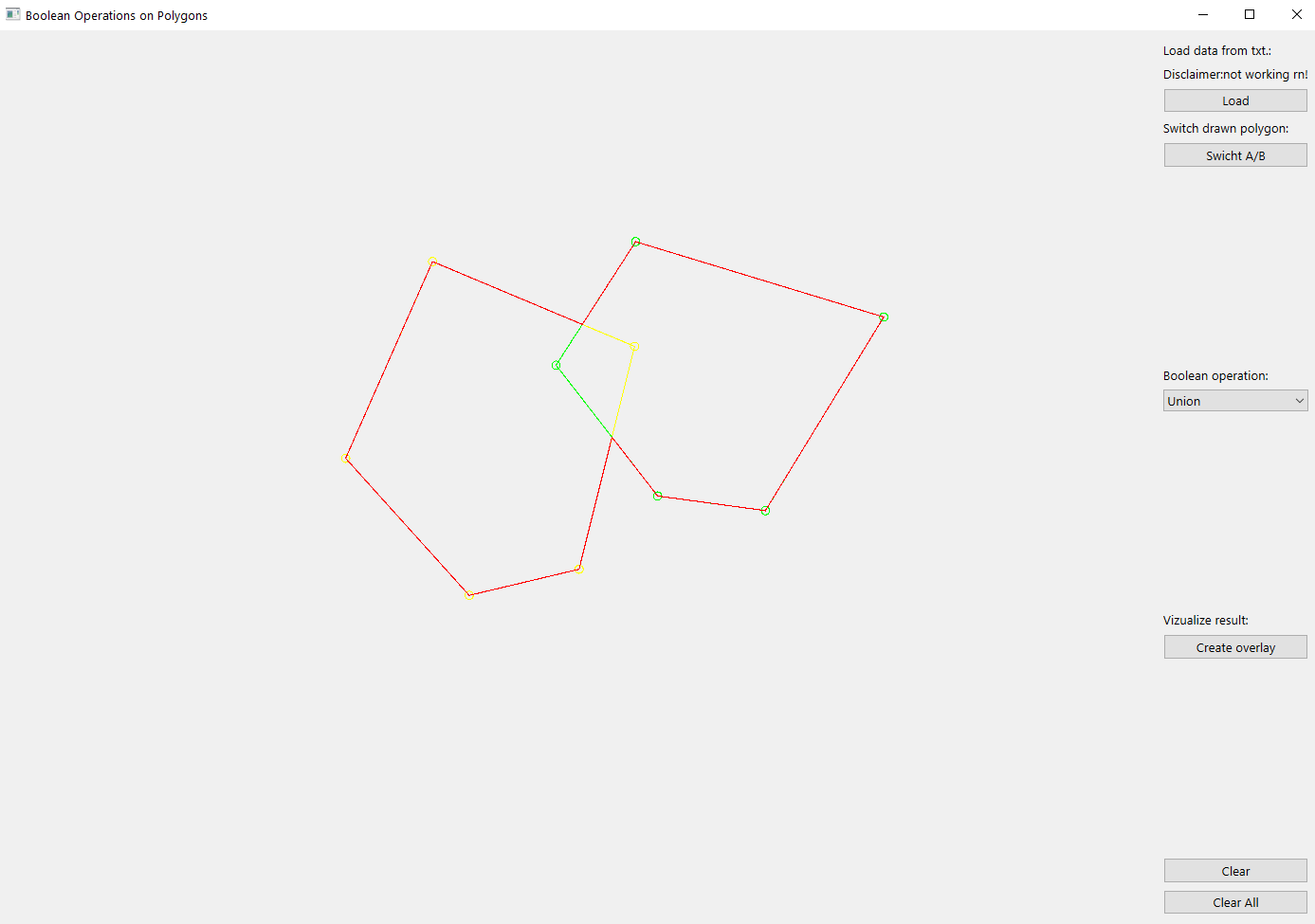
Výstupem je aplikace, která dokáže nad nakreslenými daty spočítat množinové operace a ty následně vizuálně zobrazit. Obsahuje tlačítko Switch pro změnu kreslení polygonu A/B, tlačítko pro výběr množinové operace, tlačítko pro vizuální zobrazení výsledku a dvě tlačítka na smazání obsahu okna a smazání všech hodnot proměnných.



Okno aplikace po spuštění



Okno aplikace po načtení dat



Okno aplikace po vizualizaci množinové operace

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

class **Algorithms**

TPointLinePosition **getPointLinePosition**(QPointFBO &a,QPointFBO &p1,QPointFBO &p2);

* zjištění vzájemné polohy bodu a přímky

double **get2LinesAngle**(QPointFBO &p1, QPointFBO &p2, QPointFBO &p3, QPointFBO &p4);

* zjištění hodnoty úhlu mezi dvěma přímkami

TPointPolygonPosition **getPositionWinding**(QPointFBO &q, TPolygon &pol);

* zjištění polohy bodu ku polygonu

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

* nalezení průsečíku přímek

void **updatePolygons**(TPolygon &A, TPolygon &B);

* aktualizace polygonů novým průsečíkem

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

* vložení společných bodů

void **setEdgePositions**(TPolygon &A, TPolygon &B);

* nastavení pozice hrany ku polygonu

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

* výběr hran

TEdges **createOverlay**(TPolygon &A, TPolygon &B, TBooleanOperation &op);

* vytvoření vrstvy polygonů podle zvolené množinové operace

class **Draw**

private:

TPolygon A, B;

* zhodnocované polygony

TEdges res;

* vybrané hrany

bool addA;

* změna kreslení polygonu A/B

public:

void ***paintEvent***(QPaintEvent \*event);

* vykreslování na plátno

void ***mousePressEvent***(QMouseEvent \*event);

* metoda pro vyhodnocování kliknutí na myši

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

* změna vykresllování polygonů A/B

void **drawPolygon**(TPolygon &pol, QPainter &qp);

* vykreslování polygonů

TPolygon **getA**(){return A;}

* získání polygonu A

TPolygon **getB**(){return B;}

* získání polygonu B

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

-nastavení hran

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

* Smazání vykreslovacího okna

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

* Smazání všech proměnných a okna

void **loadFile**(std::string &path);

* Načtení souřadnicového textového souboru

class Edge

private:

QPointFBO start, end;

* počáteční a koncový bod hrany

public:

void setStart(QPointFBO &start\_){start=start\_;}

* nastavení počátku hrany

void setEnd(QPointFBO &end\_){end=end\_;}

* nastavení konce hrany

QPointFBO getStart(){return start;}

* získání počátku hrany

QPointFBO getEnd(){return end;}

* získání konce hrany

class QPointFBO

private:

double alpha, beta;

* hodnoty průsečíku hran

TPointPolygonPosition pos;

* poloha bodu ku polygonu

public:

double getAlpha(){return alpha;}

* získání hodnoty alfa

double getBeta(){return beta;}

* získání hodnoty beta

TPointPolygonPosition getPosition(){return pos;}

* získání polohy bodu ku polygonu

void setAlpha(double alpha\_){alpha=alpha\_;}

* nastavení hodnoty alfa

void setBeta(double beta\_){beta=beta\_;}

* nastavení hodnoty beta

void setPosition(TPointPolygonPosition pos\_){pos=pos\_;}

* nastavení polohy bodu ku plygonu

class Types

typedef enum{

LeftHP,

RightHP,

On

} TPointLinePosition;

* pozice bodu ku hraně

typedef enum{

Inner,

Outer,

Boundary

} TPointPolygonPosition;

* pozice bodu ku polygonu

typedef enum {

Union,

Intersection,

DifferenceA\_B,

DifferenceB\_A

} TBooleanOperation;

* definice množinové operace

typedef enum{

Colinear,

Parallel,

Intersect,

NonIntersect

} T2LinesPosition;

* pozice dvou linií vůči sobe

typedef std::vector<QPointFBO> TPolygon;

* definice polygonu tvořeného z bodů QPointFBO

typedef std::vector<Edge> TEdges;

* definice hran

1. Závěr

Námi vytvořená aplikace dokáže zhodnotit vzájemnou polohu dvou polygonů Pi, Pj pomocí základních množinových operací. Dokáže pracovat s polygony vytvořenými ručně. Výsledek množinové operace je zvýrazněn červenou barvou.

1. Náměty na vylepšení

V aplikaci se bohužel nepodařilo zprovoznit načítání dat ze souboru. Data se načtou do požadovaných proměnných A,B ale poté se už nevykreslí. Vzhledem k malé dotaci času během zkouškového období nebylo věnováno řešení tohoto problému více času.

1. Citovaná literatura

1. **Tomáš, Bayer.** Perslonal page of Bayer Tomas. *Charles University of Prague.* [Online] [Citace: 05. 12 2021.] https://web.natur.cuni.cz/~bayertom/index.php/teaching/algoritmy-v-digitalni-kartografii.

1. Seznam obrázků

Množinové operace 3

Princip Winding Number algoritmu 4

Ukázka problematické situace 5

Okno aplikace po spuštění 6

Okno aplikace po načtení dat 7

Okno aplikace po vizualizaci množinové operace 7

V Praze dne 10.1.2022