ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA STAVEBNÍ, OBOR GEODÉZIE A KARTOGRAFIE
KATEDRA GEOMATIKY

název předmětu

ALGORITMY DIGITÁLNÍ KARTOGRAFIE A GIS

číslo úlohy	název úlohy:			
4	**Množinové operace s polygony**			
školní rok:	semestr:	zpracovali:	datum:	klasifikace:
2019/20	zimní	**David Němec, Jan Šartner**	17. 12. 2019	

1. Zadání

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

2. Doplňující úlohy

Z důvodu nedostatku času nebyly řešeny žádné bonusové úlohy.

3. Problematika

Konečným cílem této úlohy bylo vytvoření aplikace, která je schopná na dvou vstupních polygonech provádět základní množinové operace jako je průnik, rozdíl a sjednocení.

Množinové operace s polygony se používají především v digitální kartografii, je základní součástí většiny softwaru pro práci v GIS. Lze pomocí z nich zjistit, které segmenty a jak velkou částí zasahují do nějakého území, či se nacházejí v určité vzdálenosti od obalové zóny (Bufferu).

3.1 Princip a výpočet

Pracujeme s 2D polygony, na kterých provádíme základní množinové operace. Použitý algoritmus funguje pro jednoduché nekonvexní polygony. Má čtyři možné výsledky: 0D entita, 1D entita, 2D entitace nebo jejich kombinace. Polygony jsou ukládány do kruhových seznamů.

Postup algoritmu:

Výpočet průsečíků A,B a jejich setřídění
Aktualizace A,B
Ohodnocení vrcholů A,B podle pozice vůči B,A
Výběr vrcholů podle ohodnocení
Vytvoření hran
Spojení hran do oblastí

3.1.1 Výpočet průsečíků

Pomocí funkce get2LinesPosition byly hledány průsečíky jednotlivých linií, přičemž platí následující dvě zásady: S□gm□nt □j p□otíná □i nan□jvýš□j□dno□; s□gm□nt □i může□být p□otn□t ví□□s□gm□nty z e'i. Pokud je nalezen průsečík linií, je tento průsečík přidán do datového typu mapa, kam se ukládá také klíč alfa. Složitost algoritmu je O(m,n).

Výpočet:

```
1: Cyklus FOR(počet hran v polygonu n)
2:     Vytvoří se mapa M<double, QpointFB>
3:     Cyklus FOR(počet hran v polygonu m)
4:         IF (existuje průsečík)
5:             Přidání průsečíku do mapy M
6:             Zpracování průsečíků pomocí processIntersection()
7:         IF(byly nalezeny průsečíky ||M||>0)
8:             Procházení všech průsečíků
9:             Zpracování aktuálního průsečíku pomocí processIntersection()
```

3.1.2 Update A, B

Pokud je nalezen nějaký průsečík polygonů, je potřeba aktualizovat seznam bodů obou dvou polygonů, k čemuž slouží funkce processIntersection(). Pokud je nalezena správná pozice, je nový bod vložen do polygonu a příslušné místo.

Výpočet:
1: Cyklus FOR(počet průsečíků)
2: Výpočet koeficientu t s pomocí funkce processIntersection()
3: IF (t se blíží nule)
4: Počáteční vrchol přímky je průsečík. poly[i] is INTERSECT
5: IF (t se blíží jedné)
6: Koncový vrchol přímky je průsečík. poly[i+1] is INTERSECT
7: IF (t se nachází mezi nulou a jednou)
8: Průsečík se vloží na následující pozici po indexu.
9 Insert INTERSECT poly[i+1]

3.1.3 Ohodnocení vrcholů A, B

By vytvořen nový datový typ, který byl použit pro ohodnocení vrcholů podle toho, jestli se bod nachází uvnitř, vně či přímo na polygonu.

Výpočet:
1: Cyklus FOR(počet bodů v polygonu A)
2: Výpočet středu hrany polygonu
3: Výpočet polohy středu hrany vůči polygonu B
4: Nastavení pozice
5: Cyklus FOR(počet bodů v polygonu B)
6: Výpočet středu hrany polygonu
7: Výpočet polohy středu hrany vůči polygonu A
8: Nastavení pozice

3.1.4 Výsledné hrany

Nakonec jsou do vektoru hran vybrány ty hrany, které mají pozici totožnou s požadovanou hodnotou.

3.2 Problematické situace

Při provádění množinových operací může vzniknout celá řada problémů. Jedná se o: Společný vrchol nebo vrcholy polygonů, společná část elementu, společný element, vrchol ležící na hraně polygonu a element ležící v elementu jiného polygonu. Z těchto situací pak mohou vznikat 1D či 0D entity.

Tyto problémy byly v aplikaci ošetřeny a pro jejich testování je k dispozici řada polygonů v podobě textových souborů (viz. kapitola č. 9 - Přílohy).

3.2.1 Řešení problematických situací – hrany (1D entity)

Aplikace byla odladěna, aby nedávala špatné výsledky i v případě výsledků v podobě 1D entit. Těmi se rozumí, pokud by polygony měly společnou hranu, její část nebo více společných hran či jejich částí.

Řešením tohoto problému bylo zavedení hran s označením pozice *On* (tj. když hrana nebo její část leží na hranici druhého polygonu). V takovém případě vrací funkce *get2LinesPosition* pozici s názvem *Identical*. Pro tuto pozici byla upravena funkce *computePolygonIntersection* tak, aby pro body identických přímek A_1, A_2 (v polygonu A) a B_1, B_2 (v polygonu B) byly vypočteny koeficienty *alpha* a *beta*. Na základě koeficientů se zjistí, které body leží mezi dvojicí bodů opačného polygonu a zařadí se do polygonů jako nové průsečíky. Pomocí funkcí *setPositionaAB* a *booleanOperations* se takovýmto hranám přidělí označení pozice *On* a tyto hrany jsou vybrány v případě průniku a diferencí (nikoliv sjednocení). Vzniklé polygony jsou pak topologicky korektní a v případě průniku je označena pouze společná hrana.

Toto řešení postihuje případy společné linie, společné části linií, více společných linií.

Výpočet:
1: Když koeficienty dvou hran $k_1 = k_2 = k_3 = 0$
2: Označení *Identical*
3: Výpočet koeficientů *alpha*, *beta*
4: Zařazení průsečíků na kolineárních liniích do polygonů
5: Označení hran dle jejich polohy vůči druhému polygonu
6: Když hrana označena jako *On*
7: Výběr pouze pro operace *Intersect*, *Difference*

3.2.2 Řešení problematických situací – body (0D entity)

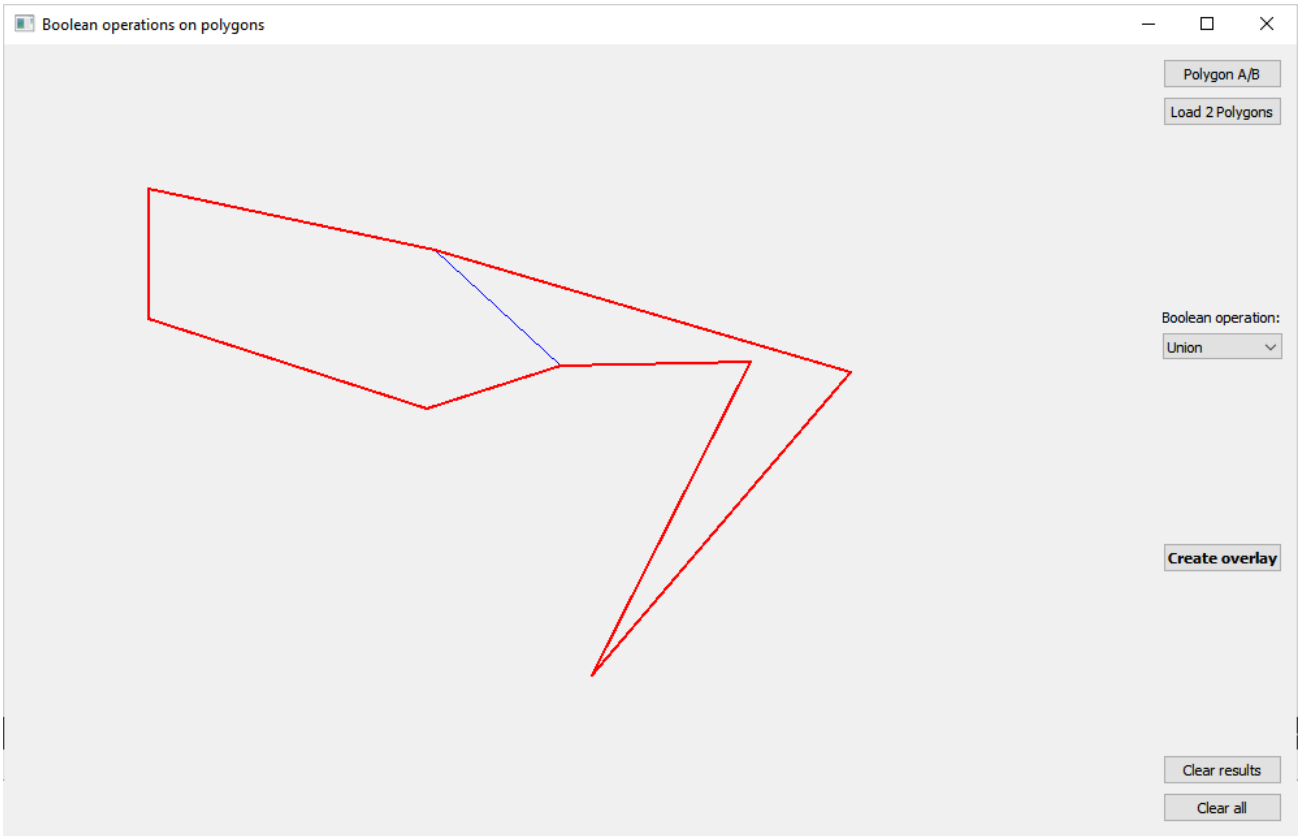
V aplikaci byly provedeny úpravy i pro některé případy, kdy by jako výsledek měly být samostatné body apod. Zde vyvstává problém, např. při rozdílu polygonů, kdy by společná hrana neměla patřit mezi hrany výsledné, ale je třeba výsledný polygon nějak ohraničit. Podobné problémy platí i pro výsledky skládající se ze samostatných bodů. Z hlediska matematiky, geometrie prostorových vztahů a kartografie není jasné, jak by mělo řešení přesně vypadat, a proto se tato úprava vztahuje pouze na společné body polygonu, které nejsou součástí hran rovněž zahrnutých v řešení.

Myšlenka úpravy je taková, že (po zpracování všech průsečíků) výše zmíněným bodům po analýze pozice vůči druhému polygonu bude přiřazena pozice *On*, ale jeho sousední hrany budou označeny jako *Outer*. Pro jejich identifikaci byla vytvořena funkce *solveODProblems*, která je vloží do samostatného vektorů bodů. Nakonec jsou vybrané body vykresleny v závislosti na zvolené množinové operaci.

Úprava postihuje případy, kdy společný bod polygonů leží kdekoliv na hraně, i když se jedná o více společných bodů.

Výpočet:
1: Zpracování všech průsečíků polygonů
2: Inicializace vektoru bodů I
3: Když pozice bodu je *On* a zároveň pozice sousedních hran je *Outer*
4: Když bod není v I
5: Přidání bodu do I
6: Když operace = *Intersect*
7: Vykresli body jako součást řešení
8:Jinak když operace = *DifferenceAB*
9: Vykresli body jako součást polygonu B
10:Jinak když operace = *DifferenceBA*
11: Vykresli body jako součást polygonu A

3.2.2 Ukázky řešení problematických situací



4. Vstupní data

Do aplikace je možné nahrát textový soubor, který obsahuje souřadnice dvou polygonů. Tyto polygony je možné nahrát a provádět na nich všechny základní množinové operace, které byly naprogramovány a jsou popsány výše. Informace jsou zapsány ve tvaru:

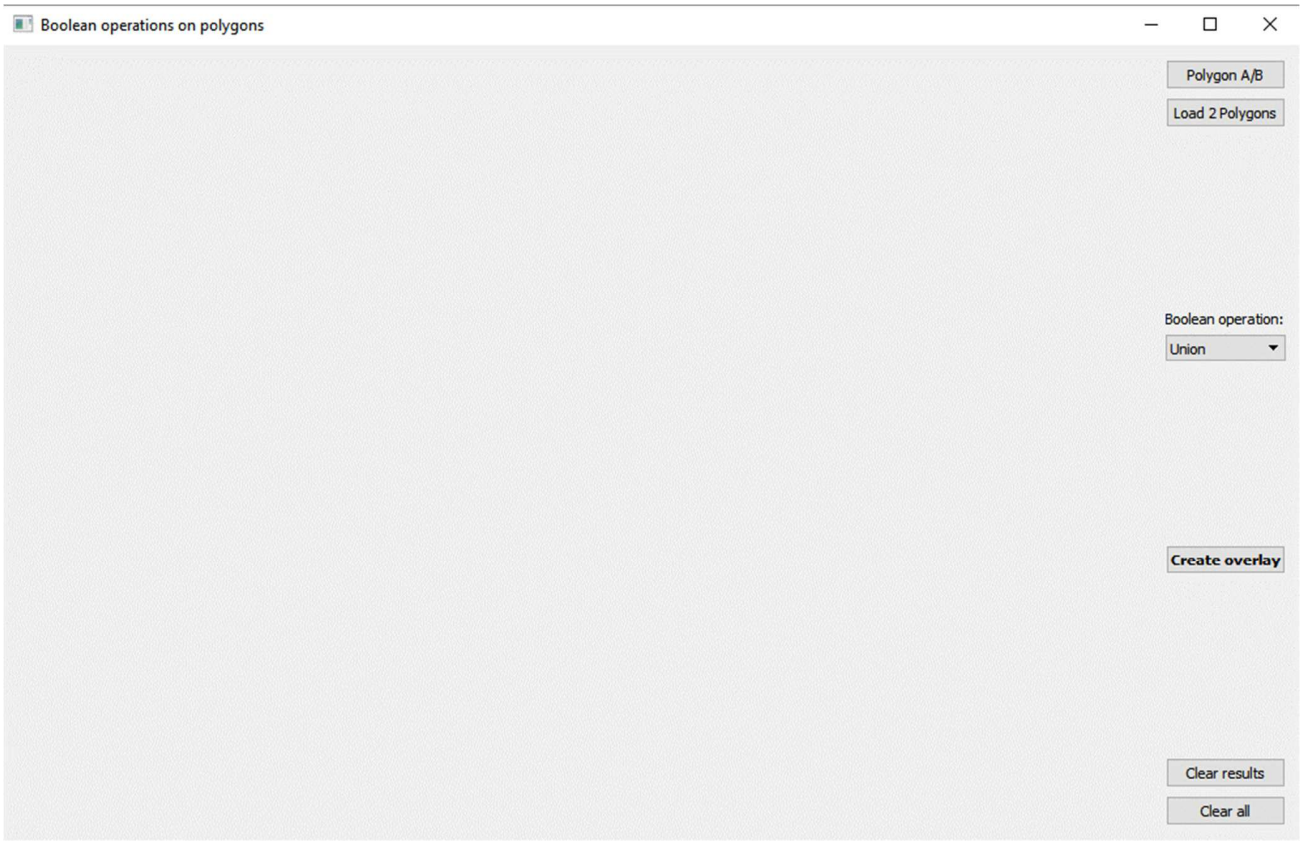
P1_x1 P1_y1
P1_x2 P1_y2
P1_xn P1_yn
xx
P2_x1 P2_y1
P2_x2 P2_y2
P2_xn P2_yn

Je také možné vykreslit dva vlastní polygony, které se vytvářejí v grafickém vstupu pomocí kliknutí myši a snímání její polohy.

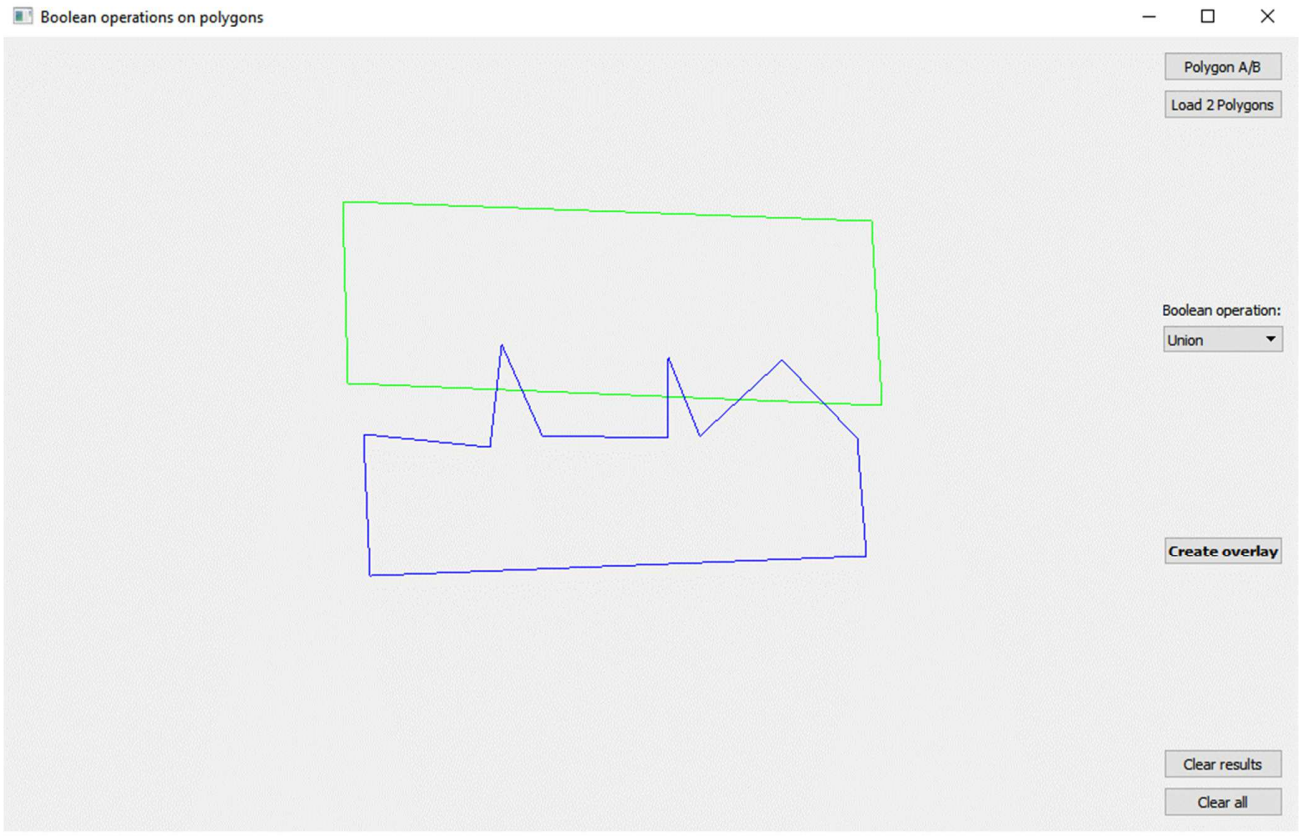
5. Výstupní data

Výstupem aplikace je grafické zobrazení provedených základních množinových operací nad dvěma vstupními polygony. Jedná se o rozdíl, spojení a průnik.

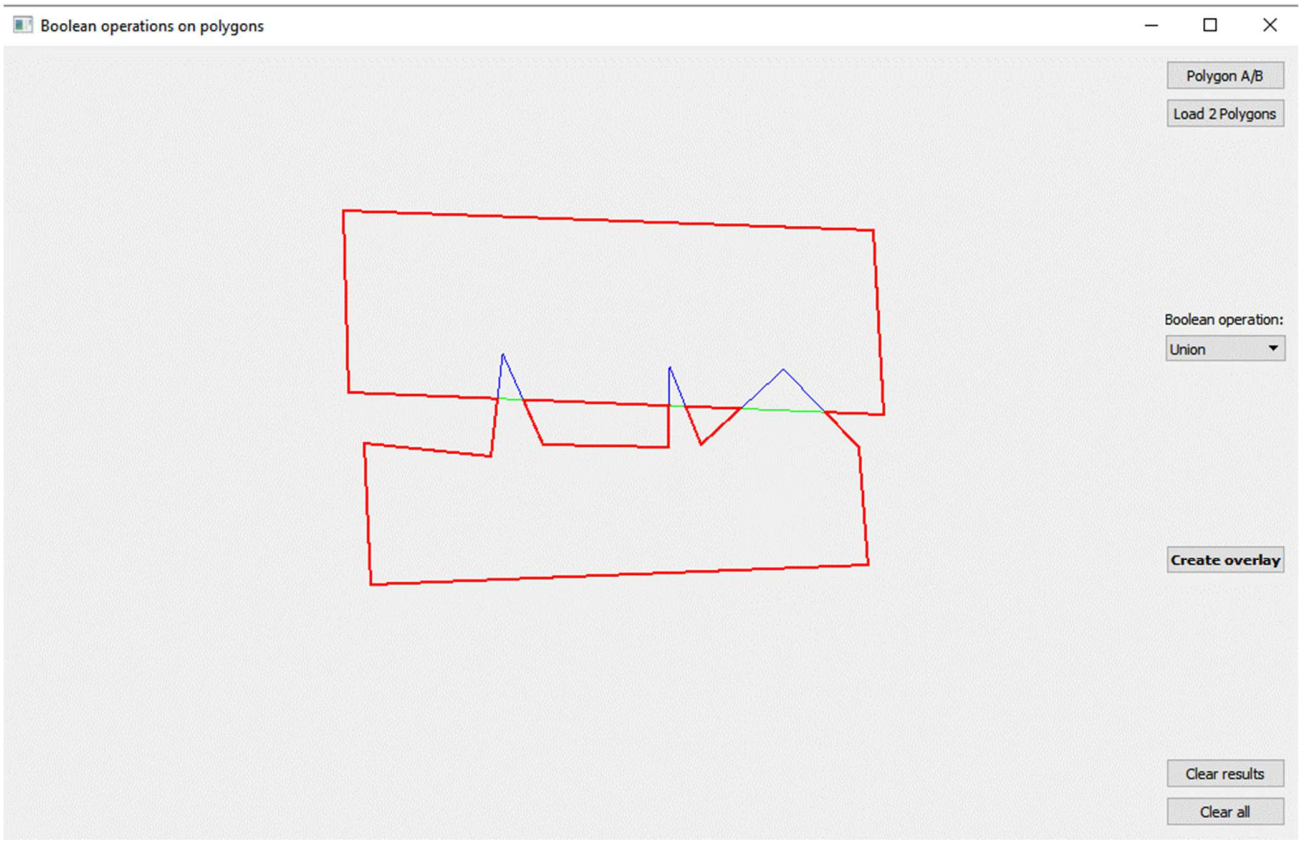
6. Vzhl□d aplikac□



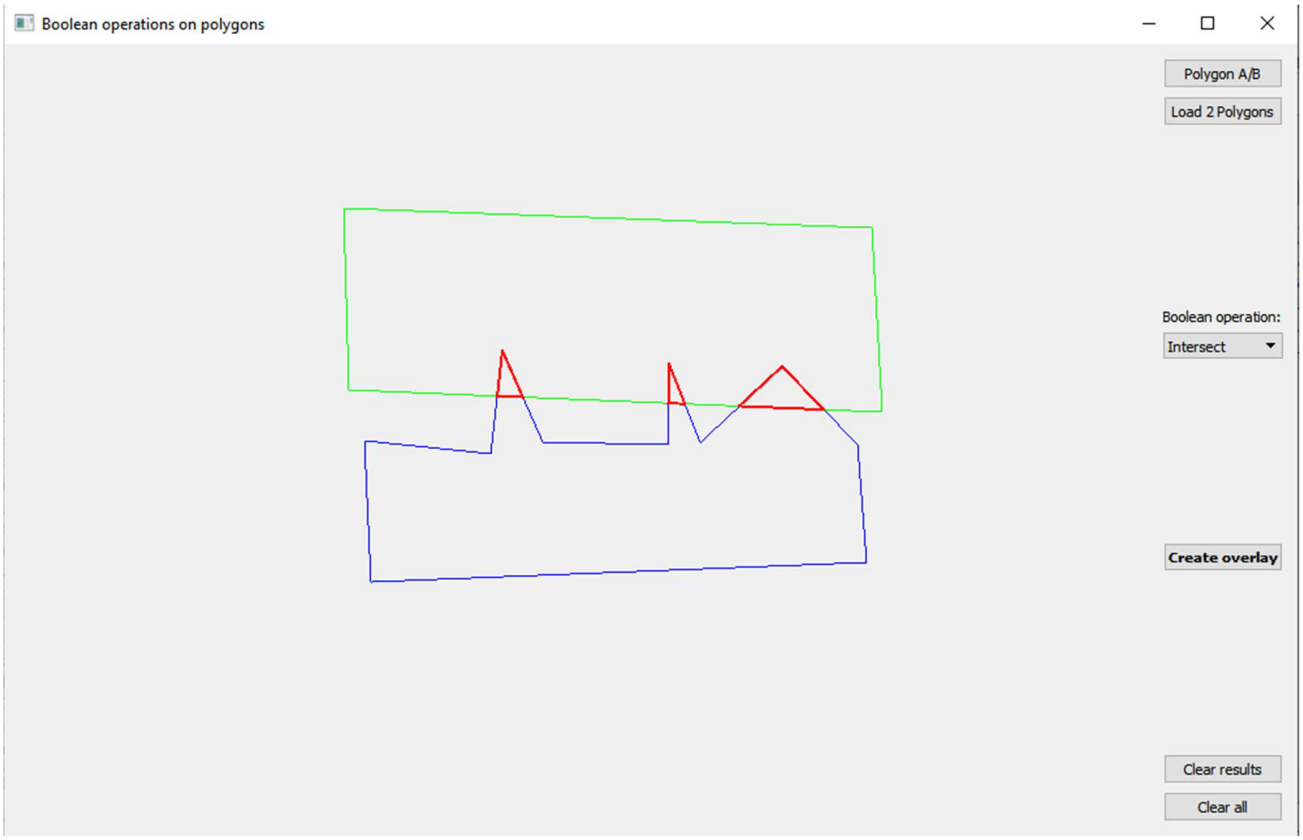
Ob□ázek 1: Základní g□afi□ké □ozh□aní



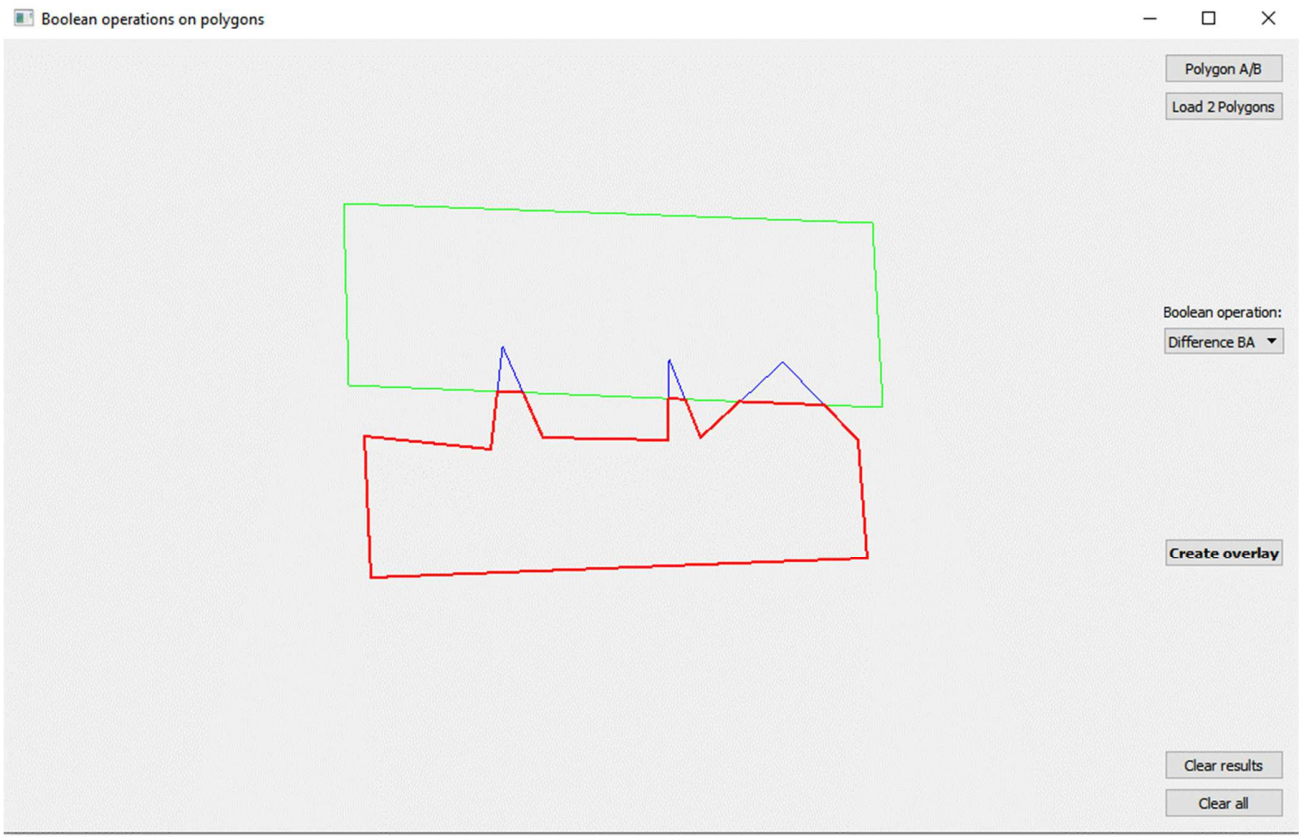
Obrázek 2: Vložené polygony



Obrázek 3: sjednocení polygonů



Obrázek 4: průnik polygonů



Obrázek 5: rozdíl polygonů A-B

Obrázek 6: Rozdíl polygonů A-B

7. Dokumentace

Třídy:

Algorithms:

Třída obsahuje 11 funkcí pro výpočet a analýzu polygonů

static TPointLinePosition getPointLinePosition(QPointFB &q, QPointFB &p1, QPointFB &p2);

Funkce zjišťuje, zda bod q leží vlevo či vpravo od vybrané úsečky spojující body p1 a p2. Také určuje, zda neleží přímo na úsečce. Funkce vrací hodnoty *TpointLinePosition(LeftHp když je vlevo, RightHp když je v pravo, Colinear když je na přímce).*

static double getAngle2Vectors(QPointFB &p1, QPointFB &p2, QPointFB &p3, QPointFB &p4);

Funkce vrací velikost úhlu mezi dvěma vektory, které jsou určeny body p1,p2 a p3,p4. Úhel je vracen v radiánech.

static TPointPolygonPosition positionPoiintPolygonWinding(QPointFB &q, std::ve□to□<QPointFB> &pol);

Funkce zjišťuje zda bod leží na hraně, uvnitř či vně polygonu. Používá přitom metodu Winding. Vrací tři možné hodnoty: *TpointPolygonPosition(On, když je bod na hraně polygonu, Inner, když je bod uvnitř polygonu, Outer, když je bod vně polygonu).* Vstupní hodnoty jsou: určený bod q a vektor bodů, které prozkoumávaný polygon obsahuje: pol.

static T2LinesPosition get2LinesPosition(QPointFB &p1, QPointFB &p2, QPointFB &p3, QPointFB &p4, QPointFB &pi);

Funkce analyzuje vzájemnou polohu dvou linií. Vrací 4 možné výsledky v typu *T2LinesPosition (Identi□al, když jso□linie totožné; Pa□alel, když jso□linie □ovnoběžné; Inte□se□ted, když se linie p□otínají; NonInte□se□ted, když se linie nep□otínají).*

static std::vector<Edge> booleanOperations(std::vector<QPointFB> &polygonA,std::ve□to□<QPointFB> &polygonB, TBooleanOpe□ation ope□ation);

Funkce dle vstupních parametrů spouští další analyzující funkce a vrací vektor hran, které tvoří výsledek analytických operací.

static void processIntersection(QPointFB &pi, double &t, std::vector<QPointFB> &polygon, int &i);

Funkce vloží do vektoru *„polygon“* bod pi na pozici i+1, pokud se nachází na hraně polygonu a není jeho vrcholem.

static void computePolygonIntersection(std::vector<QPointFB> &pa, std::vector<QPointFB> &pb);

Funkce spočítá průsečíky dvou vstupních polygonů.

stati□void setPositionsAB(std::ve□to□<QPointFB> &pa, std::ve□to□<QPointFB> &pb);

Funkce nastaví bodům polygonu hodnotu *postion*.

static void setPositions(std::ve□to□<QPointFB> &pa, std::ve□to□<QPointFB> &pb);

Funkce spočítá bodům polygonů jejich *position*.

static void selectEdges(std::vector<QPointFB> &pol, TPointPolygonPosition position, std::vector<Edge> &edges);

Funkce vybírá linie v závislosti na požadované pozici.

static std::vector<QPointF> solve0DProblems(std::vector<QPointFB> &points, std::vector<QPointFB> &pol, std::vector<QPointF> &intersections_alone);

Funkce vybere body polygonu, které odpovídají pozici *On* a zároveň jejich sousední hrany mají pozici *Outer*. Tyto body pak vrátí jako vektor.

Draw:

Třída obsahuje 14 funkcí pro vykreslování proměnných.

void changePolygon();

Funkce přepíná kreslení polygonu A na polygon B a naopak.

void setA (std::vector<QPointFB> &a_);

Funkce zajišťuje převod polygonu A do Canvasu.

void setB (std::vector<QPointFB> &b_);

Funkce zajišťuje převod polygonu B do Canvasu.

void setRes (std::vector<Edge> res_);

Funkce zajišťuje převod výsledných hran do Canvasu.

void setPath(std::string &path_)

Funkce zajišťuje převod cesty k souboru s daty do Canvasu.

std::vector<QPointFB> getA();

Funkce zajišťuje převod polygonu A z Canvasu.

std::vector<QPointFB> getB();

Funkce zajišťuje převod polygonu B z Canvasu.

std::vector<Edge> getRes();

Funkce zajišťuje převod výsledných hran z Canvasu.

void clearResults();

Funkce smaže výsledné hrany.

void clearAll();

Funkce smaže vše.

*void mousePressEvent(QMouseEvent *event);*

Funkce ukládá souřadnice po kliknutí myši.

*void paintEvent(QPaintEvent *event);*

Funkce vykresluje zadané a vypočtené proměnné.

void drawPolygon(QPainter &painter, std::vector<QPointFB> &polygon)
Funkce vykresluje polygon.

void loadTxtFile(std::vector<QPointFB> &a_, std::vector<QPointFB> &b_, std::string &path);
Funkce načítá souřadnice dvou polygonů z textového souboru.

Edge:
Třída obsahuje 4 funkce. Byla vytvořena ze dvou FB bodů a vytváří tak hranu.

QPointFB & getStart();
Funkce vrací bod počátku hrany.

void setStart(QPointFB &s);
Funkce nastavuje bod počátku hrany.

QPointFB & getEnd();
Funkce nastavuje bod konce hrany.

void setEnd(QPointFB &e);
Funkce vrací bod konce hrany.

QPointFB:
Třída obsahuje 6 funkcí. Třída byla odvozena z třídy QPointF. Navíc obsahuje koeficienty alfa, beta a pozici bodu.

double getAlpha ();
Funkce vrací koeficient alfa.

double getBeta ();
Funkce vrací koeficient beta.

TPointPolygonPosition getPosition ();
Funkce vrací pozici bodu.

void setAlpha (double alpha_);
Funkce nastavuje koeficient alfa.

void setBeta (double beta_);
Funkce nastavuje koeficient beta,

void setPosition (TPointPolygonPosition position_);
Funkce nastavuje pozici bodu.

Widget:

void on_pushButton_clicked();
Po stisknutí tlačítka Polygon A/B se změní kresba z polygonu A na B a obráceně.

void on_pushButton_2_clicked();
Po stisknutí tlačítka Create overlay se provede analýza dvou polygonů dle předem nastavených parametrů.

void on_pushButton_3_clicked();
Po stisknutí tlačítka Clear results se smaže výsledný polygon.

void on_pushButton_4_clicked();
Po stisknutí tlačítka Clear all se smaže vše.

void on_pushButton_5_clicked();
Po stisknutí tlačítka Load 2 Polygons se objeví okno pro procházení v adresářích a načte vybraný .txt soubor se dvěma polygony.

8. Závěr

Dle zadání úlohy byla vytvořena aplikace, která nad zadanými polygony provádí základní množinové operace. Polygony je možné načíst z externího textového souboru nebo je přímo vložit pomocí grafického rozhraní v aplikaci. Je možné provádět následující operace: sjednocení, rozdíl a průnik. Výsledky operací se zobrazí v grafickém okně. Z důvodu nedostatku času nebyly vyhotoveny žádné bonusové úlohy.

9. Přílohy

1) Zdrojový kód aplikace (algorithms.h / .cpp; BooleanOperations.pro drawing.h / .cpp; mainwindow.h / .cpp; main.cpp; qpointfb.h/ .cpp; types.h; widget.h / .cpp / .ui)
2) polygon_klasicky.txt (ukázka klasických polygonů)
3) polygon_uvnitr.txt (polygon uvnitř druhého polygonu)
4) spolecna_hrana.txt (polygon pro testování společné hrany)
5) spolecna_hrana_cast.txt (polygon pro testování části společné hrany)
6) spolecny_bod.txt (polygon pro testování společného bodu)

17.12.2019 v Praze
Bc. David Němec, Bc. Jan Šartner

Oprava: 22. 1. 2020