

České vysoké učení technické v Praze
Fakulta stavební



155ADKG Algoritmy v digitální kartografii

Množinové operace s polygony

Bc. Lukáš Kettner Bc. Martin Hulín
17.12.2019

Obsah

1	Zadanie	2
1.1	Bonusové úlohy	3
2	Popis a rozbor problému	4
3	Popis použitých algoritmov	5
3.1	Výpočet priesečníkom, zotriedenie a aktualizácia	5
3.1.1	Implementácia metódy processIntersection	5
3.1.2	Implementácia funkcie computePolygonIntersection	5
3.2	Ohodnotenie vrcholov	6
3.2.1	Implementácia metódy setPosition	6
3.3	Ohodnotenie hran	7
3.4	Vytvorenie hran	7
3.4.1	Implementácia metódy selectEdges	7
3.5	Problematické situácie	7
4	Vstupné dáta	8
5	Výstupné - generovanie množinových operácií	8
6	Ukážka vytvorenej aplikácie	8
6.1	Problematické prípady	11
7	Dokumentácia	13
7.1	Trieda Algorithms	13
7.1.1	Metódy	13
7.2	Trieda Draw	16
7.2.1	Členské premenné	16
7.2.2	Metódy	17
7.3	Trieda Edge	18
7.4	Trieda QPointFB	19
7.5	Trieda Types	19
7.6	Trieda Widget	19
7.6.1	Metódy	19
8	Záver	21

1 Zadaníe

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

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

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

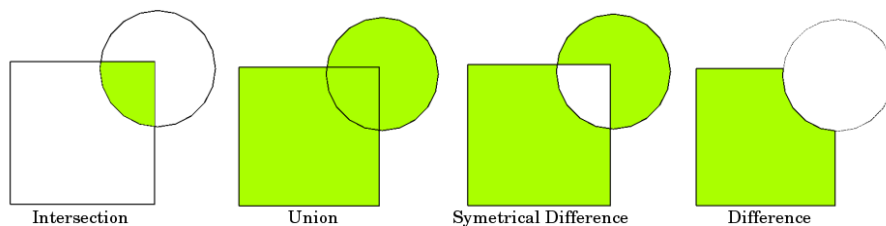
1.1 Bonusové úlohy

V rámci úlohy sú vypracované tieto bonusové úlohy

- Riešenie pre polygóny obsahujúce otvory

2 Popis a rozbor problému

Podstatou úlohy je tvorba aplikácie, v ktorej grafickom rozhraní bude možné prevádzať základné množinové operácie. V rámci úlohy sa zoberáme operáciami prienik, zjednotenie a rozdiel polygónov A a B.



Obr. 1: Typy množinových operácií s polygónmi

3 Popis použitých algoritmov

3.1 Výpočet priesečníkom, zotriedenie a aktualizácia

Využili sme funkciu `get2LinesPosition`. Táto funkcia kontroluje hrany z polygonu A a polygonu B na existenciu priesečníku. V úlohe bol použitý datový typ `QPointF`, ktorý uchováva hodnoty parametrov alfa a beta. Tento typ je odvodený od typu `QPointF`. Pokiaľ priesečník existoval spočítali sme jeho súradnice.

Pri výpočte priesečníku môžu nastať nasledujúce možnosti :

1. úsečky sú kolineárne
2. úsečky sú rovnobežné
3. úsečky sú rôznobežné
4. úsečky sú mimobežné

Tieto hodnoty sa ukladajú do datového typu `map` - kľúč je parameter alfa / beta, hodnota priesečníku. Priesečníky boli ďalej vložené do správneho polygonu na správnu pozíciu pomocou funkcie `processIntersection`.

3.1.1 Implementácia metódy `processIntersection`

1. Nastavenie tolerancie `epsilon`
2. `if((t >= epsilon) && (t <= 1 - epsilon))`
3. `i++ = 1`
4. `polygon.insert(polygon.begin() + i, pi // priradiť priesečník polygonu na pozíciu`

3.1.2 Implementácia funkcie `computePolygonIntersection`

1. Cyklus `for(int i = 0; i < pa.size(); i++)` - prechádzame celý polygon A
2. Vytvorenie `map std::map<double, QPointF> intersections`
3. Cyklus `for(int j = 0; j < pb.size(); j++)` - prechádzame celý polygon B

4. $if(get2LinesPosition(...) == INTERSECTED)$ podmienka ak existuje priesečník
5. Získaj hodnoty alpha, beta, ulož priesečník do mapy na základe alpha $intersections[alpha] = p_i$
6. $processIntersection(pi, beta, pb, j)$
7. Ak bol nájdený aspoň jeden priesečník
8. prejdí mapu $for(std :: pair < double, QPointFB > item : intersections)$
9. získaj druhú hodnotu z páru $QPointFBpi = item.second$
10. $processIntersection(pi, alfa, pa, i)$

3.2 Ohodnotenie vrcholov

Tento algoritmus uplatňuje ako ohodnocovacie pravidlo polohu vrcholu v polygóne voči druhému vrcholu. Rozsah hodnotenia v závislosti na polohe môže byť Inner, Outer, On. Tieto hodnoty boli uložené do nového datového typu TPointPolygonPosition. K určeniu polohy sme využili Winding Number algoritmus.

3.2.1 Implementácia metódy setPosition

1. Cyklus $for(int i = 0; i < n; i++)$ - prechádzame celý polygón A
2. Výpočet stredového bodu hrany
3. $double mx = (pa[i].x() + pa[(i + 1) \% n].x()) / 2;$
4. $double my = (pa[i].y() + pa[(i + 1) \% n].y()) / 2;$
5. Uloženie bodu $QPointFBm(mx, my);$
6. Určenie polohy metódou Winding Number
 $TPointPolygonPosition position = positionPointPolygonWinding(m, pb);$
7. Uloženie pozície počiatočného vrcholu hrany

3.3 Ohodnotenie hran

Výber hran pre množinové operácie znázorňuje nasledujúca tabuľka.

Operace	A	B
$C = A \cap B$	Inner	Inner
$C = A \cup B$	Outer	Outer
$C = A \cap \bar{B}$	Outer	Inner
$C = B \cap \bar{A}$	Inner	Outer
$C = A \triangle B$	Inner + Outer	Inner + Outer

Pro operaci sjednocení(UNION) jsou vybrány vnější(OUTER) hrany polygonu A a polygonu B. Při výběru operace průnik(INTERSECT) jsou vybrány vnitřní(INNER) hrany polygonu A a polygonu B. U rozdílu AB(DIFFERENCE AB) jsou vybrány vnější hrany polygonu A a vnitřní hrany polygonu B plus kolinéární segmenty polygonu A – atribut hrany ON. U opačného případu - rozdílu BA(DIFFERENCE BA) jsou vybrány vnější hrany polygonu B a vnitřní hrany polygonu A plus kolinéární segmenty polygonu B.

3.4 Vytvorenie hran

Vrcholy, ktorým náleží príslušne ohodnotenie sme následne spojili do hrán a uložili do vektoru, ktorý je vykreslovaný.

3.4.1 Implementácia metódy selectEdges

1. Cyklus *for(int i = 0; i < n; i++)* prechádzame celý polygon
2. Nájdenie vhodnej hrany
3. *Edgee(pol[i], pol[(i + 1)%pol.size()]);* vytvorenie hrany
4. *edges.push_back(e);* pridanie hrany do vektoru hran

3.5 Problematické situace

Průsečík dvou polygonů nemusí být vždy jednoznačný. Pokud je středový bod přímo na hraně polygonu nelze jednoznačně říct jestli je vnitř nebo vně. Stejně tak pro hrany. Výsledek operací průnik a sjednocení má méně

než tři segmenty a řešením je tedy prázdná množina. V Případě společného bodu je délka "hrany" nulová.

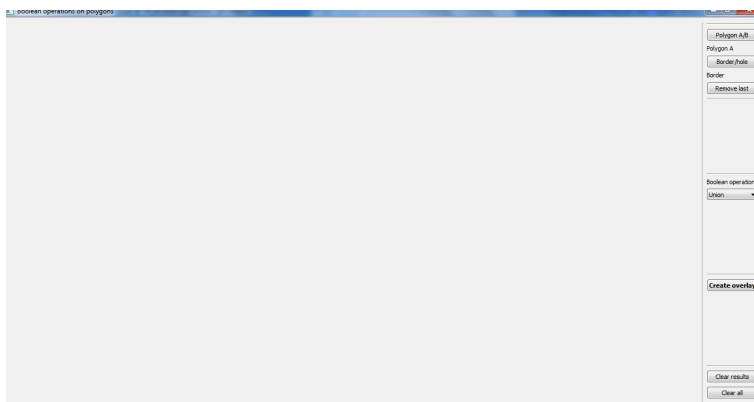
4 Vstupné dáta

Vstupnými dátami sú dva polygóny, naklikané ručne v grafickom rozhraní aplikácie. Pomocou tlačítka PolygonA/B je možné prepínať medzi kresbou jednotlivých polygónov.

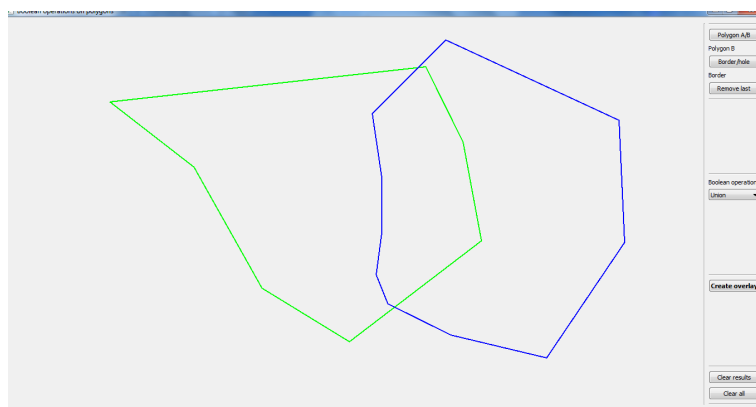
5 Výstupné - generovanie množinových operácií

Výstupnými dátami je graficky reprezentované zobrazenie množinových operácií.

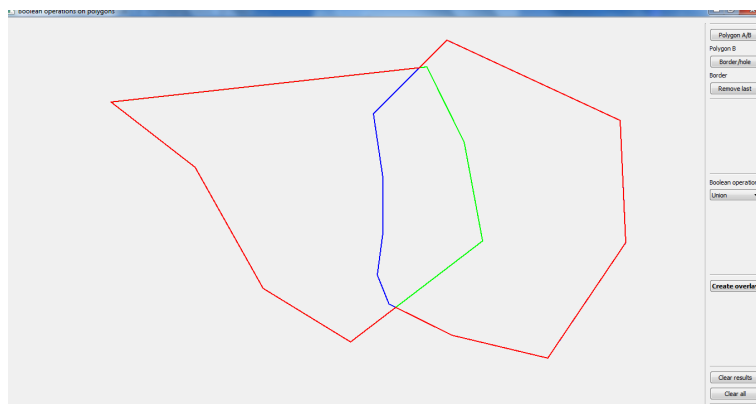
6 Ukážka vytvorenej aplikácie



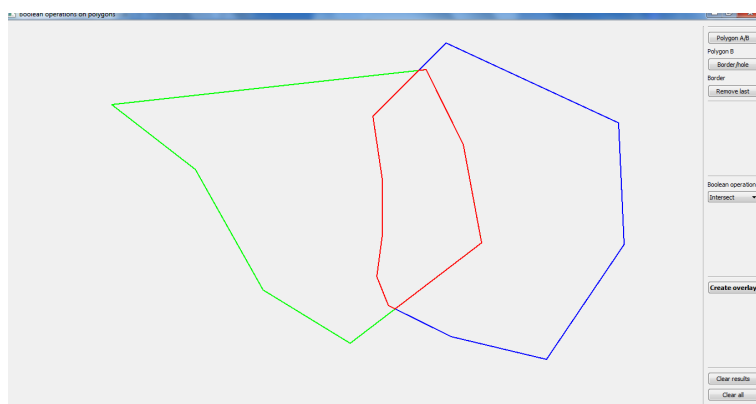
Obr. 2: Ukážka grafického rozhrania aplikácie



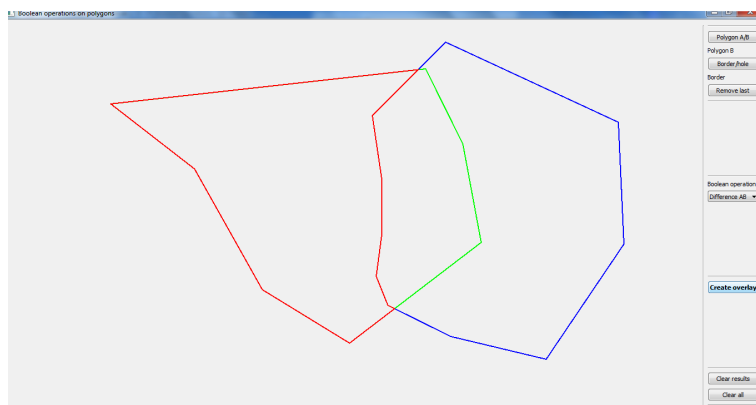
Obr. 3: Ukážka grafického rozhrania aplikácie - 2 polygóny



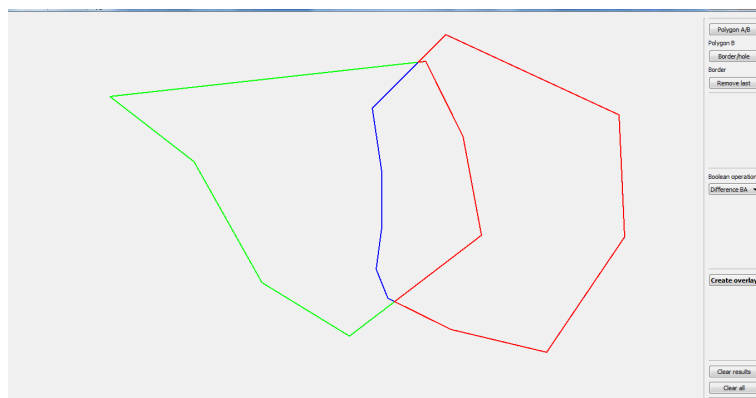
Obr. 4: Ukážka grafického rozhrania aplikácie - Union



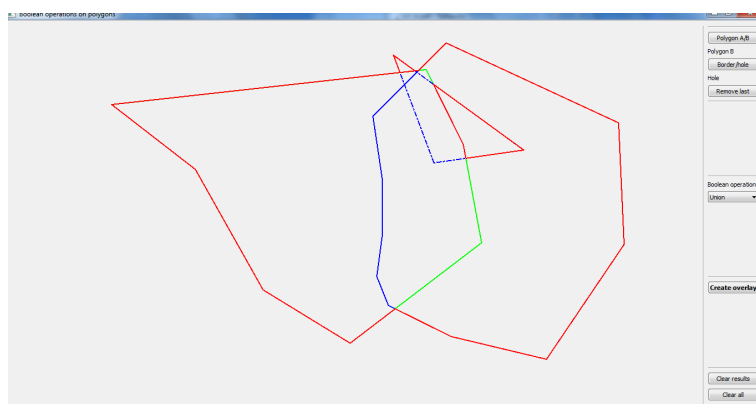
Obr. 5: Ukážka grafického rozhrania aplikácie - Intersect



Obr. 6: Ukážka grafického rozhrania aplikácie - Difference AB



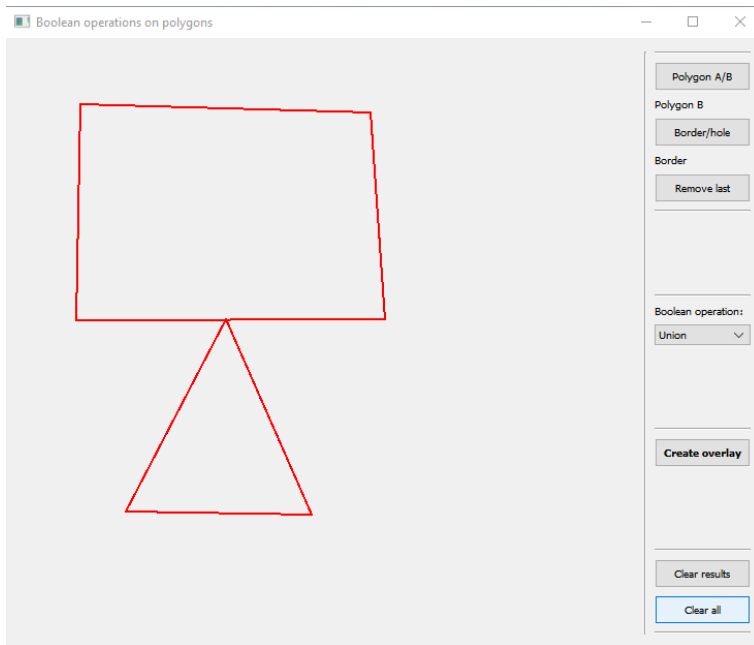
Obr. 7: Ukážka grafického rozhrania aplikácie - Difference BA



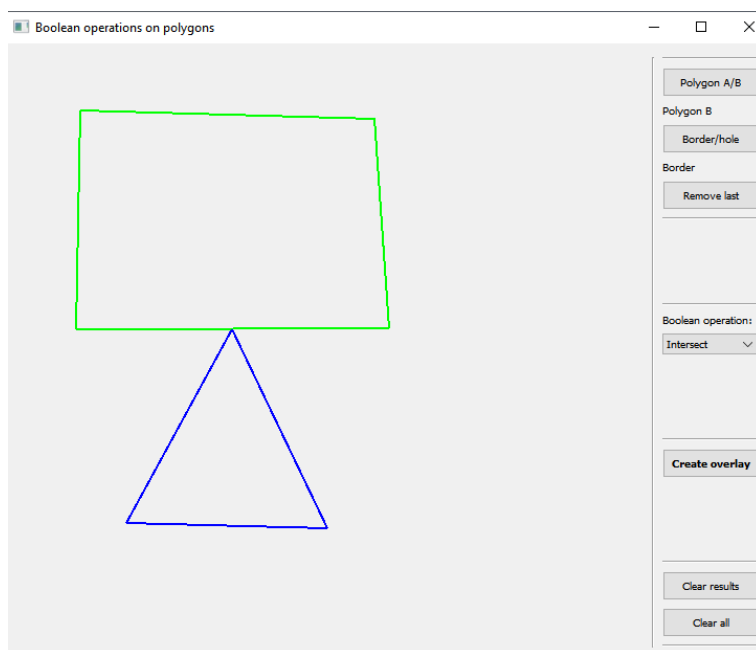
Obr. 8: Ukážka grafického rozhrania aplikácie - Holes pri operácii Union

6.1 Problematické případy

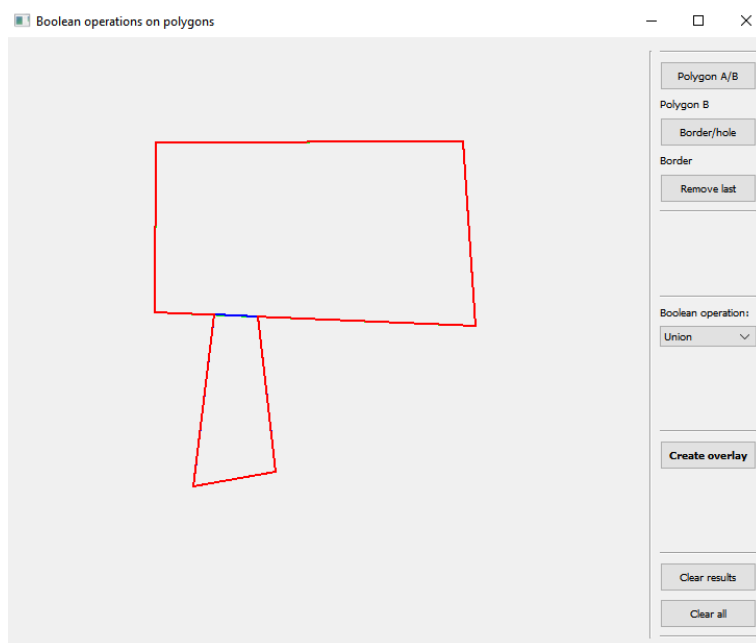
Aplikace nemá implementované importování polygonů, takže je nasimulování problematických situací mírně komplikovanější a neumožňuje jednoduché opakování.



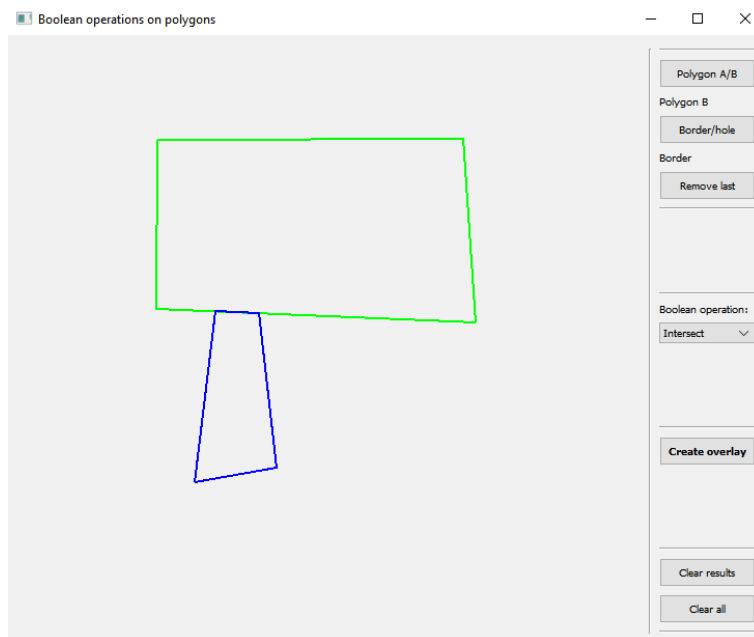
Obr. 9: Problematické případy - společný body - sjednocení



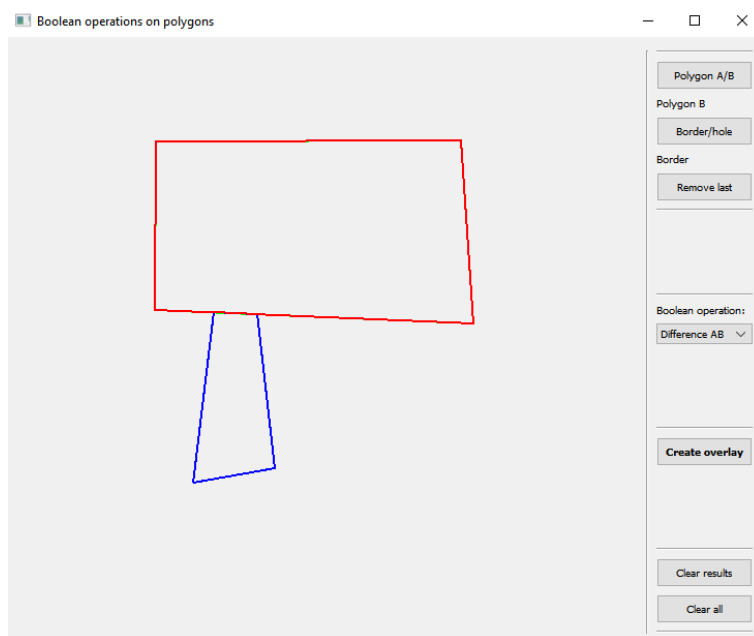
Obr. 10: Problematické případy - společný body - průnik



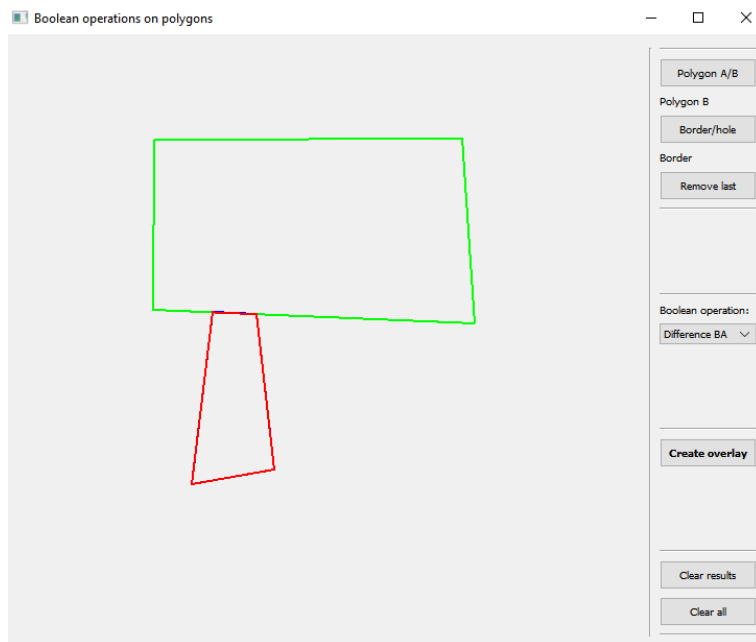
Obr. 11: Problematické případy - společná hrana - sjednocení



Obr. 12: Problematické případy - společná hrana - průnik



Obr. 13: Problematické případy - společná hrana - rozdíl A - B



Obr. 14: Problematické prípady - spoločná hrana - rozdiel B - A

7 Dokumentácia

7.1 Trieda Algorithms

Triedu Algorithms sme použili pre deklarovanie funkcií pre výpočtové algoritmy tvorby množinových operácií s polygónmi.

7.1.1 Metódy

getAngle2Vectors

- Slúži k určeniu uhlu medzi dvoma priamkami. Jej návratovou hodnotou je double
- na vstupe má : súradnice bodov p_1, p_2, p_3, p_4 určujúcich prvú a druhú priamku
- výstupom je hodnota uhlu medzi priamkami

getPointLinePosition

- Slúži na určenie polohy bodu voči priamke. Jej návratovou hodnotou je integer.
- na vstupe má : súradnice určovaného bodu q , súradnice bodov priamky $p_1 p_2$
- na výstupe hodnoty :
 - LeftHp
 - RightHp
 - Colinear

positionPointPolygonWinding

- slúži k určeniu polohy bodu prostredníctvom Winding Number algoritmu. Jej návratový typ je integer.
- na vstupe má : QPointFB q – bod ktorého polohu určujeme, std::vector<QPointFB> pol – polygon, voči ktorému určujeme polohu bodu q
- výstupom je hodnota :
 - Outer
 - Inner

get2LinesPosition

- funkcia slúžia k výpočtu polohy dvoch priamok voči sebe.
- na vstupe je sú body QPointFB p_1, p_2, p_3, p_4, p_i
- na výstupe hodnoty :
 - Identical
 - Paralel
 - Intersected
 - NonIntersected

booleanOperations

- funkcia slúžia k prevedeniu množinových operácií..
- na vstupe je sú polygóny bodov QPointFB $polygonA, polygonB$ a typ operácie

- na výstupe je vektor hrán odpovedajúci zvolenej operácii pre polygóny

processIntersection

- funkcia slúžia k prevedeniu zaradenia vypočítaného priesečníku na správnu pozíciu v príslušnom polygóne. Jej návratovým typom je void.

computePolygonIntersection

- funkcia slúžia k výpočtu priesečníku. Jej návratovým typom je void.

setPositionsAB

- funkcia je pomocnou funkciou pre booleanOperations. Jej návratovým typom je void.

setPositions

- funkcia slúži k určeniu pozície hrany. Jej návratovým typom je void.

selectEdges

- funkcia slúži k vybraniu príslušných hrán. Jej návratovým typom je void.

booleanOperationsHoles

- funkcia slúžia k prevedeniu množinových operácií pri zadaní Holes.
- na vstupe je sú polygóny bodov QPointFB polygonA, polygonB a typ operácie
- na výstupe je vektor hrán odpovedajúci zvolenej operácii pre polygóny

mergeVectors

- funkcia slúži k zlúčeniu vektorov hrán. Jej návratovým typom je void.

7.2 Trieda Draw

Trieda Draw slúži ku grafickému vykresleniu množinových operácií.

7.2.1 Členské premenné

std::vector<QPointFB>a, b

- vektory bodov, ktoré tvoria jednotlivé polygóny A a B. Polygón A sa vykresluje plnou čiarou zelenej farby, polygon B plnou čiarou modrej farby.

std::vector<QPointFB>inA, inB

- vektory bodov, ktoré tvoria diery v jednotlivých polygónoch. U každého polygónu je možná len jedna diera. Diery sa vykresľujú rovnakou farbou ako polygóny, no čiarkovanou čiarou.

std::vector<Edge>res

- je to vektor, ktorý obsahuje hrany tvoriace výsledok množinovej operácie. Tá je vykreslená červenou farbou.

std::vector<Edge>removeEdges

- je to vektor, ktorý obsahuje hrany, ktoré majú byť vo výsledku zakryté / zafarbené. Ich vykreslenie je prevedené bilou farou, čo je mierne vidieť ale v rámci praktického použitia to bolo najjednoduchšie možné riešenie.

bool ab

- táto premenná indikuje v akom bode je kreslenie, či užívateľ kreslí polygón A alebo B. V default nastavení sa kreslí polygon A.

bool inout

- premenná indikuje kreslenie hranice / diery. Defaultne nastavenie je kresba hranice.

7.2.2 Metódy

paintEvent

- slúži k vykresleniu naklikaných hran v tvare polygónov. Návrátovým typom je void.

void mousePressEvent

- slúži k vykresleniu bodu stlačením tlačidla myši, v okamihu stlačenia tlačidla na myši sa uložia súradnice bodu. Návrátovým typom je void.

void drawPolygon

- slúži k vykresleniu polygónu. Návrátovým typom je void.

void clearResult

- slúži k vymazaniu výsledku vykonanej množinovej operácie. Návrátovým typom je void.

void removeLast

- slúži k vymazaniu poslednej pridanej hrany polygónu. Návrátovým typom je void.

void clearAll

- slúži k vymazaniu obsahu celého grafického okna aplikácie. Návrátovým typom je void.

getA

- slúži k vráteniu členskej premennej a

getB

- slúži k vráteniu členskej premennej b

getAHole

- slúži k vráteniu členskej premennej inA

getBHole

- slúži k vráteniu členskej premennej inB

getRes

- slúži kvráteniu členskej premennej res

getRemoveEdges

- slúži k vráteniu členskej premennej removeEdges

bool getPolygonStatus

- slúži k vráteniu členskej premennej ab

bool getDrawStatus

- slúži k vráteniu členskej premennej inout

setA

- slúži k nastaveniu členskej premennej a

setB

- slúži k nastaveniu členskej premennej b

setRes

- slúži k nastaveniu členskej premennej res

setRemoveEdges

- slúži k nastaveniu členskej premennej removeEdges

void changePolygon

- slúži k zmene polygónu pri vykreslovaní

7.3 Trieda Edge

Je to definície datového typu Edge. Reprezentuje usporiadanú dvojicu vrcholov, ktoré tvoria hranu.

7.4 Trieda QPointFB

Je to odvodená trieda od triedy QPointF. Pridali sme k nej hodnotu parametru alpha/beta.

7.5 Trieda Types

Trieda Types slúži pre definovanie nových datových typov, ktoré nám umožnili ľahšiu a prehľadnejšiu implementáciu algoritmov.

7.6 Trieda Widget

Trieda Widget obashuje metódy ktoré sú odkazom na sloty umožňujúce vykonávať príkazy z grafického rozhrania aplikácie. Nemajú žiadne vstupné hodnoty, návratovým typom je void.

7.6.1 Metódy

on_pushButton_switch_clicked

- tlačidlo **Polygon A/B** po kliknutí naň sa zmení vykresľovanie polygonu A na B a opačne

on_pushButton_createOverlay_clicked

- tlačidlo **Create overlay** po kliknutí naň sa vygeneruje zvolená množinová operácia

on_pushButton_clearResult_clicked

- tlačidlo **Clear result** po kliknutí naň sa vymaže výsledok množinovej operácie

on_pushButton_clearAll_clicked

- tlačidlo **Clear all** po kliknutí naň sa vymaže celé grafické okno Canvasu

on_pushButton_changeInOut_clicked

- tlačidlo **Border / Hole** kliknutím naň je možné meniť kresbu polygónu / hole

on_pushButton_removeLast_clicked

- tlačidlo **Remove last** kliknutím naň sa odstráni posledná vytvorená hrana

8 Záver

Výsledkom úlohy je funkčná aplikácia a grafická prezentácia množinových operácií. V aplikácii je zakomponovaná tvorba dier (holes). Hrany, ktoré majú byť vo výsledku skyté sú vykresľované bielou farbou, čo si pozorný užívateľ aplikácie určite všimne. Bohužiaľ vzhľadom na časové možnosti autorov bola táto možnosť najvhodnejším možným riešením. Tento bod je určite námetom na vylepšenie aplikácie, kedy by hrany, ktoré nemajú byť vykresľované boli vyhládané a odstránené.

Zoznam obrázkov

1	Typy množinových operácií s polygónmi	4
2	Ukážka grafického rozhrania aplikácie	8
3	Ukážka grafického rozhrania aplikácie - 2 polygóny	9
4	Ukážka grafického rozhrania aplikácie - Union	9
5	Ukážka grafického rozhrania aplikácie - Intersect	9
6	Ukážka grafického rozhrania aplikácie - Difference AB	10
7	Ukážka grafického rozhrania aplikácie - Difference BA	10
8	Ukážka grafického rozhrania aplikácie - Holes pri operácii Union	10
9	Problematické prípady - spoločný body - sjednocení	11
10	Problematické případy - společný body - průnik	12
11	Problematické případy - společná hrana - sjednocení	12
12	Problematické případy - společná hrana - průnik	13