České vysoké učení technické v Praze
Fakulta stavební

Algoritmy v digitální kartografii

Množinové operace s polygony

Bc. Petra Pasovská
Bc. David Zahradník

Obsah

1 Zadání

Níže uvedené zadání je kopie ze stránek předmětu.

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**

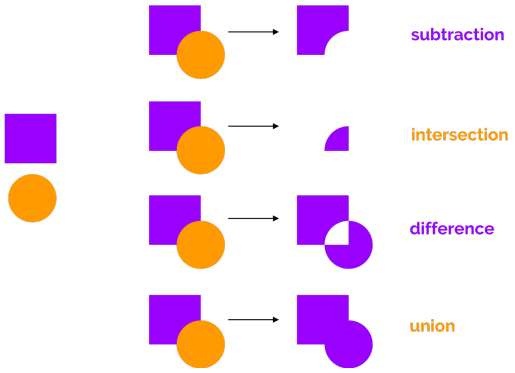
Čas zpracování: 2 týdny

2 Popis a rozbor problému

Hlavním cílem této úlohy je tvorba aplikace, která je schopná na vygenerovaných polygonech provádět základní množinové operace. V rámci této úlohy je tedy možné vypočítat průnik, sjednocení a rozdíl dvou polygonů.

Obecně lze pro určení vztahů použít tzv. Booleovské operátory. Jsou pojmenovány po Georgovi Booleovi, který je použil ve své knize z roku 1854. Základní Booleovské operátory jsou AND (průnik - logický součin), OR (sjednocení - logický součet) a NOT (negace). V Booleovské logice mohou nabývat datové typy bool dvou hodnot, a to 1 (TRUE) či 0 (FALSE).

Množinové operace mají v kartografii velké využití, zejména v prostředí GIS softwarů. Ve většině těchto programů jsou již naimplementovány základní funkce včetně jejich rozšíření. Mezi jedny z nejčastějších rozšířených funkcí patří funkce buffer, která vytváří obalovou zónu kolem vybraných dat. Z praktického hlediska je poté buffer využíván například pro hledání objektů v určité vzdálenosti apod.



Obrázek 1: Přehled základních množinových operací [zdroj: 1]

3 Popis použitých algoritmů

V této úloze byl vytvořen nový datový typ - QPointFB. Pro tvorbu této aplikace bylo použito několik dílčích algoritmů. Obecně lze postup rozdělit na tyto fáze:

1. Výpočet průsečíků A, B + setřídění
2. Update A, B
3. Ohodnocení vrcholů A, B, dle pozice vůči B, A
4. Výběr vrcholů dle ohodnocení
5. Vytvoření fragmentů
6. Vytvoření oblastí z fragmentů

3.1 Výpočet průsečíků A, B + setřídění

V rámci výpočtu průsečíků se prochází jednotlivé body polygonů a porovnává se jejich vzájemný vztah. K určení vztahu byla použita již existující funkce s názvem Get2LinesPosition. Pokud byly linie vyhodnoceny tak, že se protínají, byl průsečík uložen do nově vytvořené proměnné o datovém typu mapa. V rámci proměnné mapa se ukládá výsledek a tzv. klíč, který k dané hodnotě odkazuje. Klíč je označen jako α. Časová náročnost tohoto výpočtu je rovna O(m,n).

3.1.1 Implementace metody

1. Pro všechna i: $for(i = 0; i < n; i++)$
2. Vytvoření mapy: $M = map < double, QPointFB >$
3. Pro všechna j: $for(j = 0; j < m; j++)$

   Pokud existuje průsečík: $if(b_{ij} = (p_i, p_{(i+1)\%n}) \cap (q_j, q_{(j+1)\%m}) \neq \emptyset)$

   Přidání do mapy: $M[\alpha_i] \leftarrow b_{ij}$

   Zpracování prvního průsečíku: $ProcessIntersection(b_{ij}, \beta, B, j)$
4. Při nalezení průsečíků: $if(\|M\| > 0)$

   Procházení všech průsečíků: $for \forall m \in M$

   Zpracování aktuálního průsečíku: $ProcessIntersection(b, \alpha, A, i)$

3.2 Update A, B

S nalezením každého průsečíku je nutné updatovat seznam bodů obou polygonů. K tomu slouží funkce ProcessIntersection. Při nalezení správné pozice je nový bod vložen za pomoci defaultní funkce insert, do níž vstupuje nejprve pozice umístění a následně hodnota, která se vkládá.

3.3 Ohodnocení vrcholů A, B, dle pozice vůči B, A

Pro možnost ohodnocení vrcholů vůči jednotlivým polygonům byl vytvořen nový datový typ. Tento datový typ nabývá hodnot podle toho, zda se bod nachází uvnitř polygonu, na hranici či mimo polygon.

3.4 Vytvoření fragmentů

Vrcholy se stejným ohodnocením byly přidány do polylinie, respektive do fragmentu. Body jsou uloženy včetně pozice, na níž se nachází. Každý fragment začíná průsečíkem a končí prvním bodem s jiným ohodnocením. Pro diferenci má fragment opačné pořadí vrcholů - po směru hodinových ručiček.

3.4.1 Implementace metody

Do této metody vstupuje polygon P o n velikosti, ohodnocením vrcholů g, změnou orientace s a seznamem fragmentů F.

1. Dokud P[i] není průsečík s orientací g: $g(P[i]) \neq g \vee P[i] \neq inters)$
   $i \leftarrow i + 1$
2. Žádný bod s touto orientací neexistuje: $if(i \equiv n)return$
3. Uložení startovního indexu prvního průsečíku: $i_s \leftarrow i$
   Vytvoření prázdného fragmentu: $f = \emptyset$
   Při nalezení fragmentu:
   Swapování prvků je-li potřeba: $f.reverse()$
   Přidání fragmentu do mapy s klíčem počátečního bodu: $F[f[0]] \rightarrow f$
   Přejdi k dalšímu bodu přes index: $i \leftarrow (i + 1)\%m$
4. Opakování dokud se nevrátí zpět k počátečnímu průsečíku: $while(i \neq i_s)$

Následně byla vytvořena ještě jedna funkce pro tvorbu fragmentů, do níž vstupuje index startovního bodu, polygon P, orientace g, index vrcholů i a již vytvořený fragment f.

1. Bod není průsečíkem s orientací g: $if g(P[i]) \neq g \vee P[i] \neq inters \rightarrow returnFALSE$
2. Nekonečný cyklus: $for(;;)$
   Přidání bodu do fragmentu: $f \leftarrow P[i]$
   Následující bod ze seznamu: $i \leftarrow (i + 1)\%n$
   Při navrácení ke startovnímu bodu: $if(i \equiv i_s) \rightarrow returnFALSE$
   Při nalezení prvního bodu s rozdílnou orientací: $if(g(P[i]) \neq g)$
   Přidání bodu do seznamu a úspěšné ukončení: $f \leftarrow P[i] \rightarrow returnTRUE$

3.5 Vytvoření oblastí z fragmentů

Následně je nutné projít všechny fragmenty a sestavit z nich oblasti. Vstupem do funkce jsou vzniklé fragmenty F a výstupem seznam polygonů C.

3.5.1 Implementace metody

1. Pro všechna f: $for \forall f \in F$
2. Vytvoření prázdného polygonu: $P \leftarrow \emptyset$
3. Nalezení startovního bodu fragmentu: $s \leftarrow f.first$
4. Při nezpracování fragmentu: $if(!f.second.first)$
   Přidání polygonu do seznamu: $C \leftarrow P$

Z oblastí jsou následně vytvářeny polygony funkcí createPolygonFromFragments. n značí následující bod (next), s startovní bod (start).

1. Inicializace následujícího bodu: $QPointFBn \leftarrow s$
2. Nekonečný cyklus k procházení všech fragmentů: $for(;;)$
3. Nalezení navazujícího fragmentu: $f \leftarrow F.find(n)$
4. Při neexistenci fragmentu s takovýmto počátečním bodem: $if(f \equiv F.end) \rightarrow returnFALSE$
5. Fragment označen za zpracovaný: $f.second.first \leftarrow TRUE$
6. Následující bod: $n \leftarrow f.second.second.back()$
7. Přidání bez počátečního bodu: $P \leftarrow f.second.second - \{f.second.second[0]\}$
8. Při vrácení se na začátek: $if(n \equiv s) \rightarrow returnTRUE$

3.6 Výsledný algoritmus

Po vytvoření zmíněných dílčích algoritmů jsou funkce postupně volány.

1. Nastavení správné orientace obou polygonů.
2. Výpočet průsečíků A, B: $ComputeIntersections(A, B)$
3. Určení polohy vrcholů vůči oblastem: $setPositions(A, B)$
4. Tvorba mapy fragmentů: $mapF$
5. Určení pozice: $pos = (oper \equiv Intersection \vee oper \equiv DifBA?Inner : Outer)$
6. Swapnutí: $swap = (oper \equiv DifAB) : TRUE : FALSE$
7. Tvorba fragmentů: $createFragments(A, pos, swap, F)$
8. Propojení fragmentů: $mergeFragments(A, B, C)$

4 Vstupní data

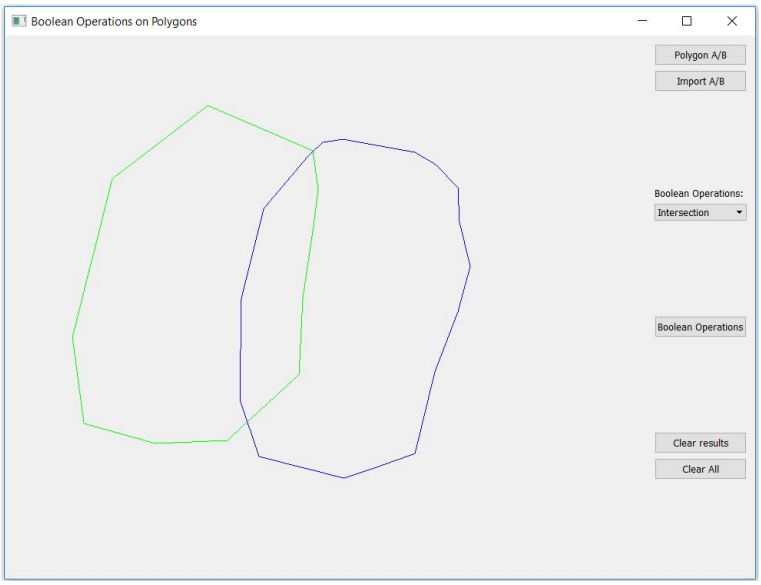
Vstupní data musí být seřazená, tedy každý polygon v souboru musí začínat bodem jedna a končit n-tým bodem. Ve vstupních datech je polygon A dán číslem 1 a polygon B číslem jiným než 1. Jednotlivé body se sekvenčně ukládají do proměnné QPointFB a následně polygonu A/B.

Struktura vstupních dat: [číslo polygonu, souřadnice X, souřadnice Y]

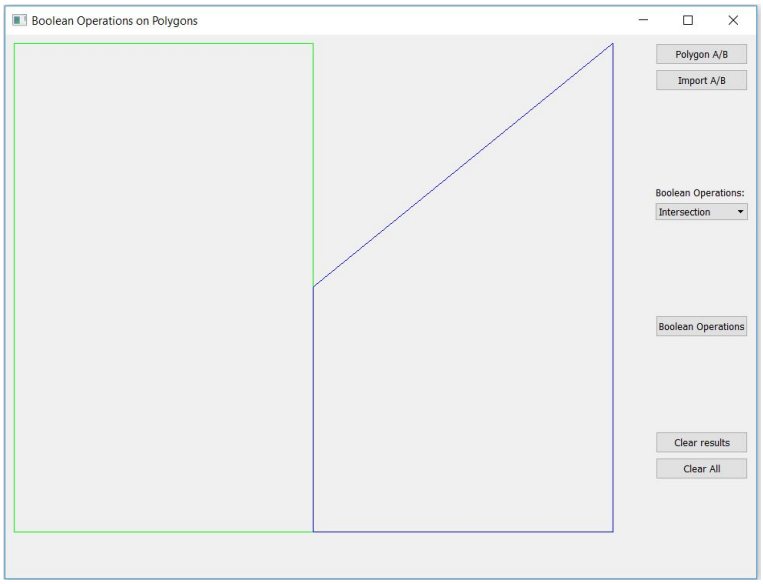
5 Výstupní data

Výstupem aplikace je grafické znázornění jednotlivých booleovských metod na načtených polygonech, případně ručně vložených polygonech.

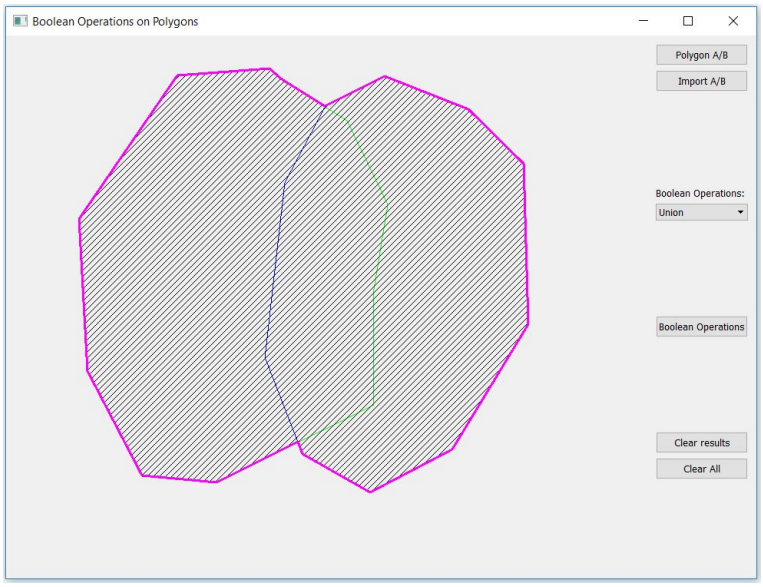
6 Aplikace



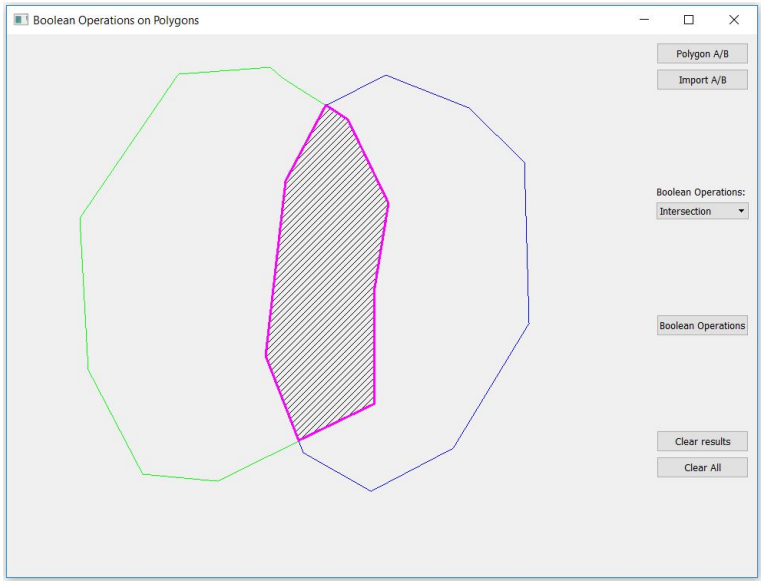
Obrázek 2: Vzhled aplikace a ruční vložení obou polygonů



Obrázek 3: Ukázka naimportovaných polygonů



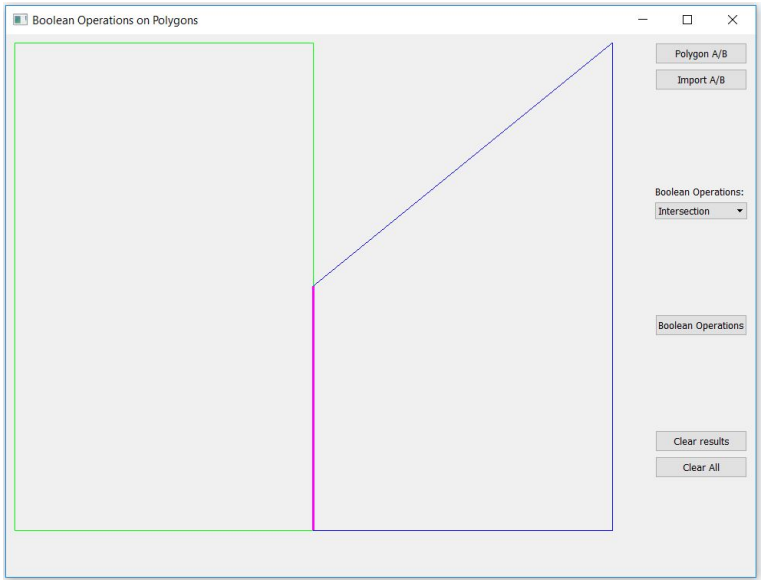
Obrázek 4: Výsledek operace sjednocení (UNION)



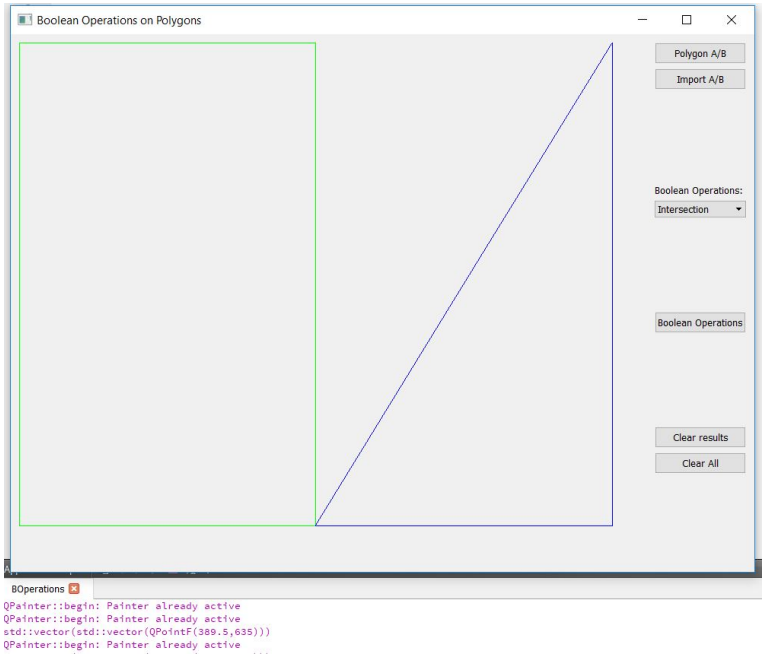
Obrázek 5: Výsledek operace průnik (INTERSECTION

Obrázek 6: Výsledek operace rozdílu A a B (DIFFERENCE A - B)

Obrázek 7: Výsledek operace rozdílu B a A (DIFFERENCE B - A)



Obrázek 8: Ukázka aplikace v případě, že výsledek je linie (1D entita)



Obrázek 9: Ukázka aplikace v případě, že výsledek je bod (0D entita)

7 Dokumentace

7.1 Třídy

7.1.1 Algorithms

Třída Algorithms obsahuje několik metod. Metody jsou určeny pro výpočty použitých algoritmů.

TPointPolygon getPositionWinding(QPointFB q, std::vector<QPointFB>pol)
Metoda, která vrátí vztah polohy bodu q a polygonu pol. Návratové hodnoty jsou INSIDE, OUTSIDE, ON .

TPointLinePosition getPointLinePosition(QPointFB &q, QPointFB &a, QPointFB &b)
Metoda, která vrátí vztah polohy bodu q a přímky tvořenou body a a b. Návratové hodnoty jsou LEFT, RIGHT, COL (na hraně).

double get2LinesAngle(QPointFB &p1,QPointFB &p2,QPointFB &p3, QPointFB &p4)
Tato metoda slouží k vypočetní hodnoty úhlu mezi dvěma přímkami <p1,p2>a <p3,p4>.

T2LinesPosition get2LinesPosition(QPointFB &p1,QPointFB &p2,QPointFB &p3, QPointFB &p4, QPointFB &intersection)
Metoda, která vrátí vztah dvou přímek a do proměné intersection, pokud existuje, spočte průsečík a přiřadí jemu hodnoty do typu QPointFB. Návratové hodnoty jsou PARALLEL, COLINEAR, INTERSECTING, NONINTERSECTING .

void computePolygonIntersections(std::vector<QPointFB>&p1, std::vector<QPointFB>&p2)
Metoda spočte průsečíky polygonů p1 a p2 a vloží je do oněh polygonů (spustí se metoda processIntersection).

void processIntersection(QPointFB &b, double t, std::vector<QPointFB>&poly, int &i)
Metoda vloží bod b do polygonu poly na pozici i+1 , pokud je na hraně daného polygonu a není vrcholem.

void setPositions (std::vector<QPointFB>&pol1,std::vector<QPointFB>&pol2)
Metoda nastaví bodu polygonu pol1 třídy QPointFB hodnotu pos, podle vztahu s polygonem pol2 a opačně.

void createFragments(std::vector<QPointFB>&pol, TPointPolygon posit, bool rev, std::map<QPointFB,std::vector<QPointFB>>&F)
Metoda vytvoří fragmenty se stejnou hodnotou posit a uloží je do hasovací tabulky F.

void mergeFragments(std::map<QPointFB, std::vector<QPointFB>>&Fa, std::map<QPointFB, std::vector<QPointFB>>&Fb, std::vector<std::vector

<QPointFB>>&C)
Metoda sjednotí fragmenty z polygonu A a polygonu C a uloží je do vektoru polygonů C.

double getPolygonOrientation(std::vector<QPointFB>&pol)
Metoda vrátí plochu polygonu pol. Pokud je výměra záporná polygon na CCV orientaci.

std::vector<std::vector<QPointFB>>BooleanOper(std::vector<QPointFB>&A, std::vector<QPointFB>&B, TBooleanOperation oper)
Metoda nad polygony A a B provede metodu oper: INTERSECTION, UNION, DIFFAB, DIFFBA.

7.1.2 Draw

Třída Draw obsahuje několik metod. Metody jsou určeny pro generování a vykreslování proměných.

void paintEvent(QPaintEvent *e)
Metoda pro kreslení do vykreslovacího okna.

void drawPol(std::vector<QPointFB¿ &pol, QPainter &painter)
Metoda pro vykreslení polygonu.

void mousePressEvent(QMouseEvent *e)
Metoda po kliknutí do zobrazovacího okna uoží bod do polygonu podle setAB.

void setAB()
Metoda nastaví, kam se budou ukládat vložené body.

void clearAll();
Metoda smaže vše ze zobrazovacího okna.

void clearResults();
Metoda smaže výsledky Boolovských operací ze zobrazovacího okna.

void setRes(std::vector<std::vector<QPointFB¿ ¿ result)
Metoda nastaví vektor polygonů s výsledky Boolovských operací.

void setA(std::vector<QPointFB>polA_)
Metoda nastaví polygonu A.

void setB(std::vector<QPointFB>polB_)
Metoda nastaví polygonu B.

std::vector<QPointFB>getA()
Metoda vrátí polygonu A.

std::vector<QPointFB>getB()
Metoda vrátí polygonu B.

7.1.3 QPointFB

Nová třída odvozená od třídy QPointF.(double alfa, double beta, bool inters, TPointPolygon pos). Alfa je koeficient alfa, pokud bod leží na přímce A. Beta je koeficient beta, pokud bod leží na přímce B. Inters je true, pokud bod je průsečíkem dvou přímek. Pos nabývá hodnot INSIDE, OUTSIDE, ON vůči danému polygonu.

double getAlfa()
Metoda vrátí hodnotu alfa bodu třídy QPointFB.

void setAlfa(double alfa_)
Metoda nastaví hodnotu alfa bodu třídy QPointFB.

double getBeta()
Metoda vrátí hodnotu beta bodu třídy QPointFB.

void setBeta(double beta_)
Metoda nastaví hodnotu beta bodu třídy QPointFB.

bool getInters()
Metoda vrátí hodnotu inters bodu třídy QPointFB.

void setInters(bool inters_)
Metoda nastaví hodnotu inters bodu třídy QPointFB.

TPointPolygon getPosition()
Metoda vrátí hodnotu pos bodu třídy QPointFB.

void setPosition(TPointPolygon pos_)
Metoda nastaví hodnotu pos bodu třídy QPointFB.

bool operator <(const QPointFB &p)
Přetížený operátor pro porovnání bodů třída QPointFB podle x.

7.1.4 Widget

void on_pushButton_clicked()
Po stisknutí tlačítka Polygon A/B, se nastaví kreslení polygonu A nebo B.

void on_pushButton_2_clicked()
Po stisknutí tlačítka Boolean Operations, se vypočte a zobrazí výsledek Boolovské operace, dle výběru v combo boxu.

void on_pushButton_3_clicked()
Po stisknutí tlačítka Clear results, se vymaže výsledek Boolovské operace.

void on_pushButton_4_clicked()
Po stisknutí tlačítka Clear All, se vymaže vše co je v obrazovém okně.

void on_pushButton_5_clicked()
Po stisknutí tlačítka Buffer, se vypočte a zobrazí buffer nad objekty v obrazovém okně.

void on_pushButton_6_clicked()
Po stisknutí tlačítka Import A/B, lze naimportovat dva polygony pro testování aplikace.

8 Závěr

V rámci úlohy byla vytvořena aplikace pro výpočet Boolovských operací nad dvěma polygony. Polygony lze ručně naklikat v obrazovém okně, nebo naimportovat z textového souboru. Následně je možné vypočítat průnik, sjednocení, rozdíl A-B a rozdíl B-A polygonů. Výsledek se vykreslí v zobrazovací okně a je vyšrafován a ohraničen silnější linií, aby byl na první pohled jasně zřejmý.

V této aplikaci bylo zapotřebí ošetřit i případy, kdy je výsledkem linie či bod. Pro testování těchto případů byly vytvořeny testovací textové soubory, na kterých si může uživatel sám vyzkoušet funkčnost. Přesto pro případ, kdy je výsledkem 0D entita - bod, tak není bod zřetelně vyznačen. Po spouštění aplikace a volbě jednotlivých operací pro tento případ je bod správně vyhodnocen, což lze vidět na ukázce aplikace, kde je ve frameworku Qt v konzolovém řádku po příkazu qDebug vypsán.

9 Náměty na vylepšení

9.1 Import polygonů

Načítání polygonů ze souboru by mohlo být oddělené. Uživatel by načetl polygony ze dvou různých textových souborů a přitom by si zvolil, kam chce načtený polygon uložit A/B.

9.2 Buffer

Bohužel operace buffer nebyla z důvodu nedostatku času zprovozněna, proto byla z aplikace vyřazena, aby v ní nezůstávaly nefunkční části. Přesto by bylo určitě vhodné buffer dokončit.

9.3 Vykreslení 0D výsledku

Bylo by vhodné ošetřit, že v případě, že výsledkem je 0D entita (bod), tak bude graficky lépe zvýrazněn, aby byl uživateli na první pohled výsledek zřejmý.

9.4 Přehlednost

Pro přehlednost by bylo určitě vhodné, aby byly jednotlivé polygony popsány, zda se jedná o polygon A nebo B. Případně aby bylo i v aplikaci vidět, zda uživatel vykresluje polygon A nebo B. V případě, že je poté zvolena operace rozdílu, tak nemusí být uživateli na první pohled jasné, od kterého polygonu se který odečítá.

10 Reference

1. TSAGARIS, Antonis. Vector ilustration basics for Android developers [online][cit. 21. 12.2018].
   Dostupné z: https://hackernoon.com/vector-illustration-basics-for-android-developers-part-3-boolean-operations-8a0ced922030

2. BAYER, Tomáš. Množinové operace s polygony [online][cit. 3. 1. 2019].
   Dostupné z: https://web.natur.cuni.cz/ bayertom/images/courses/Adk/adk9.pdf