České vysoké učení technické v Praze
Fakulta stavební

155ADKG: Množinové operace s polygony

Michael Kala
Anna Zemánková

1 Zadání

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 Údaje o bonusových úlohách

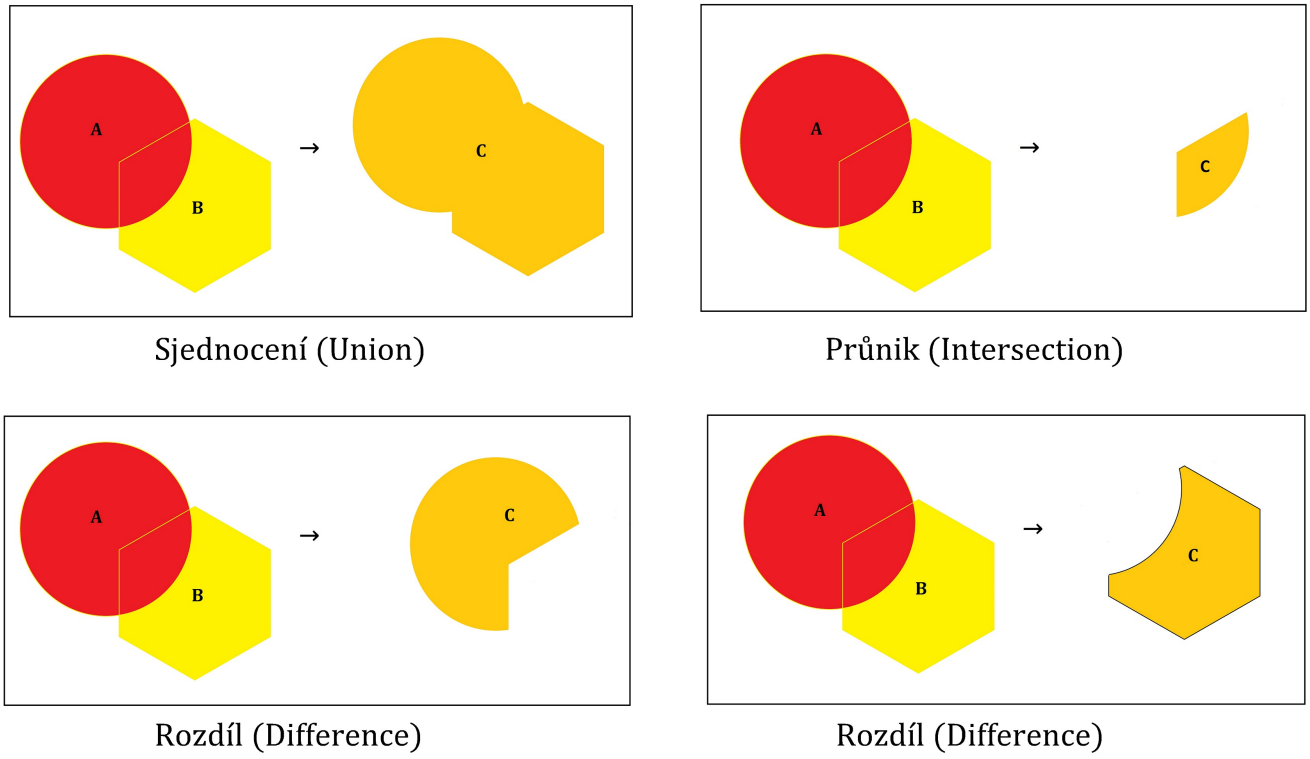
V rámci této úlohy nebyly zpracovány žádné bonusy.

3 Popis a rozbor problému

Mějme dva polygony A $\{p_i\}$ a B $\{p_j\}$.

Množinové operace:

- Sjednocení (Union): $C = A \cup B$
- Průnik (Intersection): $C = A \cap B$
- Rozdíl (Difference): $C = A \cap \bar{B}$ nebo $C = B \cap \bar{A}$



Obrázek 1: Příklady množinových operací

4 Popis algoritmů

Nejprve bylo třeba určit průsečíky množin A a B a setřídit je. Následně byly ohodnoceny vrcholy množiny A resp. B podle pozice vůči B resp. A. Poté byly dle ohodnocení vybrány vrcholy a z nich vytvořeny fragmenty. Nakonec byl(y) z fragmentů vytvořen(y) výsledný/é polygon(y).

4.1 Výpočet průsečíků

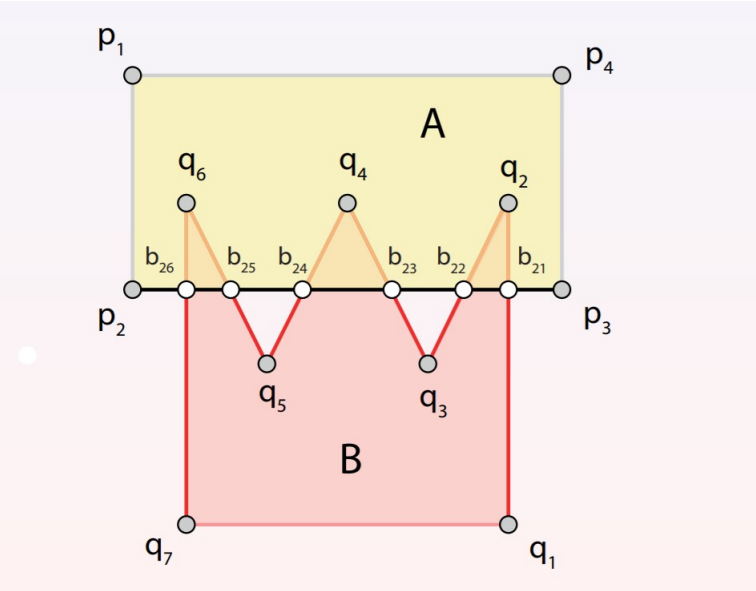
V rámci tohoto algoritmu jsou vypočteny průsečíky polygonů a ukládány do datovéto typu map - kde klíč je α/β a hodnota průsečík. Po každém nalezení průsečíku je aktualizován seznam souřadnic bodů polygonu a to právě za využití parametru α/β tak, aby byly body ve správném pořadí. Nakonec jsou vrcholy množiny A resp. B ohodnoceny podle pozice vůči B resp. A za využití algoritmu *getPositionWinding* z první úlohy - vždy je vypočten střed hrany a určena jeho poloha vůči druhému polygonu (viz setPositions).

4.1.1 Implementace metody

1. $for(i = 0; i < n, i++)$
2. $M = map(double, QPointFB)$ // Vytvoření mapy
3. $for(j = 0; j < m, j++)$
4. if $b_{ij} = (p_i, p_{(i+)1\%n} \cap q_j, q_{(j+)1\%m}) \neq 0$ // Existuje průsečík
5. $M[\alpha_i] \longleftarrow b_{ij}$ //Přidej do M
6. ProcessIntersection (b_{ij}, β, B, j) //Zpracuj první průsečík pro e_j
7. if $(||M|| > 0)$ // Nějaké průsečíky jsme nalezli
8. $for \forall m \in M :$ // Procházej všechny průsečíky v M
9. $b \longleftarrow m.second$ // Získej 2. hodnotu z páru
10. ProcessIntersection (b, α, A, i) //Zpracuj první průsečík pro e_i

Process Intersection

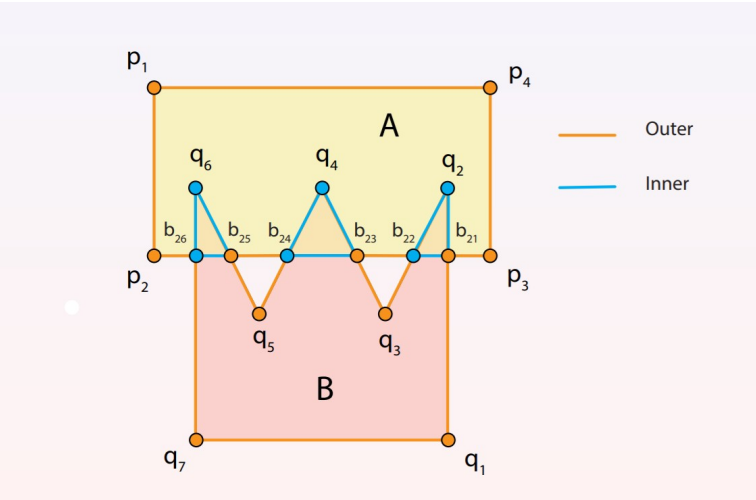
1. $if(|t| < \epsilon) :$
2. $P[i] \longleftarrow inters$ //Startovní bod průsečíkem
3. $if(|t - 1| < \epsilon) :$
4. $P[(i + 1)\%m] \longleftarrow inters$ // Koncový bod průsečíkem
5. $else :$
6. ProcessIntersection $i \longleftarrow i + 1$ //Inkrementuj pozici
7. if $P \longleftarrow (b, i)$ // Přidej průsečík na pozici i+1



Obrázek 2: Průsečíky množin A a B [1]

setPositions

1. for $(i = 0; i < n, i++)$:
2. $M = \frac{p_i(x,y)+p_{i+1}(x,y)}{2}$ // Výpočet středu hrany
3. pozice = $GetPositionWinding(\bar{p}, B)$ //Určení polohy M vůči množině B
4. $p_i[pozice]$ = pozice



Obrázek 3: Ohodnocení hran [1]

4.2 Fragmenty

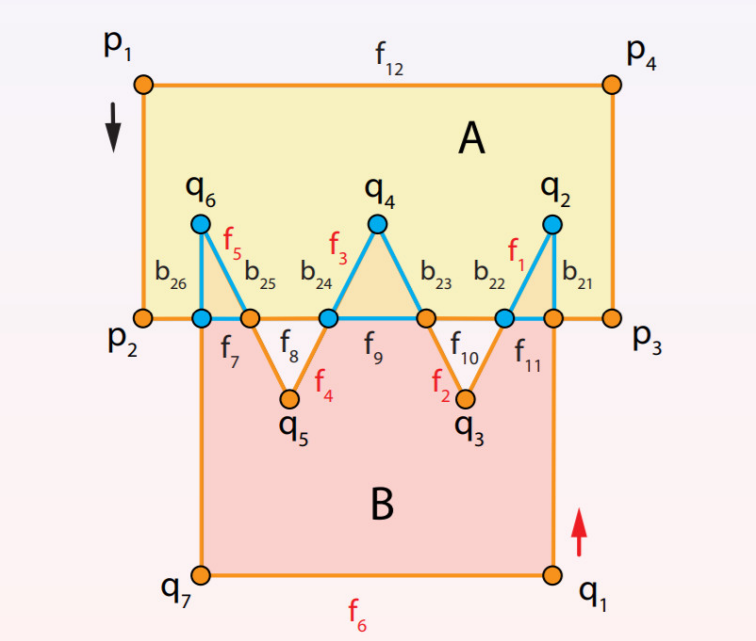
Nejprve jsou podle ohodnocení hran vytvořeny fragmenty = seznamy po sobě jdoucích bodů se stejným ohodnocením, počáteční bod fragmentu je vždy průsečík a koncový bod je první vrchol s jiným ohodnocením (další průsečík). Následně jsou fragmenty spojeny do oblastí.

4.2.1 Vytvoření fragmentů - implementace metody

1. $i \longleftarrow 0$
2. $while(g(P[i]) \neq g \vee P[i] \neq inters)$ //Dokud P[i] není průsečík s orientací g
3. $\quad i \longleftarrow i + 1$
4. $if(i \equiv n)$ return; //Žádný bod s touto orientací neexistuje
5. $i_s \longleftarrow i$ //Zapamatuj startovní index prvního průsečíku
6. do
7. $\quad f = \emptyset$ //Prázdný fragment
8. $\quad if(createFragmentFromVertices(i_s, P, g, if))$ //Nalezen fragment
9. $\quad\quad if(s)f.reverse()$ //Swapuj prvky fragmentu, je-li třeba
10. $\quad\quad F[f[0]] \longrightarrow f$ //Přidej fragment do mapy, klíč poč. bod
11. $\quad i \longleftarrow (i + 1)\%m$
12. $while(i \neq i_s)$ //Dokud nedojdeme k počátečnímu průsečíku

createFragmentFromVertices

1. $if(g(P[i]) \neq g \vee P[i] \neq inters)$ //Bod není průsečíkem s orientací g
2. $\quad return\ false$
3. for (;;)
4. $\quad f \longleftarrow P[i]$ // Přidej bod do fragmentu
5. $\quad i \longleftarrow (i + 1)\%n$
6. $\quad if(i \equiv i_s)$ //Obešli jsme celý polygon
7. $\quad\quad return\ false$
8. $\quad if(g(P[i]) \neq g)$ // První bod s rozdílou orientací
9. $\quad\quad f \longleftarrow P[i]$ //Přidej ho do seznamu
10. $\quad\quad return\ true$



Obrázek 4: Fragmenty [1]

4.2.2 Sestavení oblastí z fragmentů - implementace metody

1. $for \forall f \in F$
2. $P \longleftarrow \emptyset$ //Vytvoř prázdný polygon
3. $s \longleftarrow f.first$ //Najdi startovní bod fragmentu
4. $if(!f.second.first)$ //Pokud fragment již nebyl zpracován
5. $if(createPolygonFromFragments(s, F, P))$
6. $C \longleftarrow P$ //Přidej polygon do seznamu

createPolygonFromFragments

1. $QPointn \longleftarrow s$ //Inicializuj následující bod
2. for (;;) //Projdi všechny fragmenty tvořící polygon
3. $f \longleftarrow F.find(n)$ //Najdi navazující fragment
4. $if(f \equiv F.end)$ // Fragment s takovým poč. bodem neexistuje
5. $return\ false$
6. $f.second.first \longleftarrow true$ //Fragment označen za zpracovaný
7. $n \longleftarrow f.second.second.back()$ // Najdi následující bod
8. $P \longleftarrow f.second.second - \{f.second.second[0]\}()$ // Přidej bez poč. bodu
9. $if(n \equiv s)$ // Obešli jsme polygon, jsme na startu
10. $return\ true$

4.3 Množinové operace

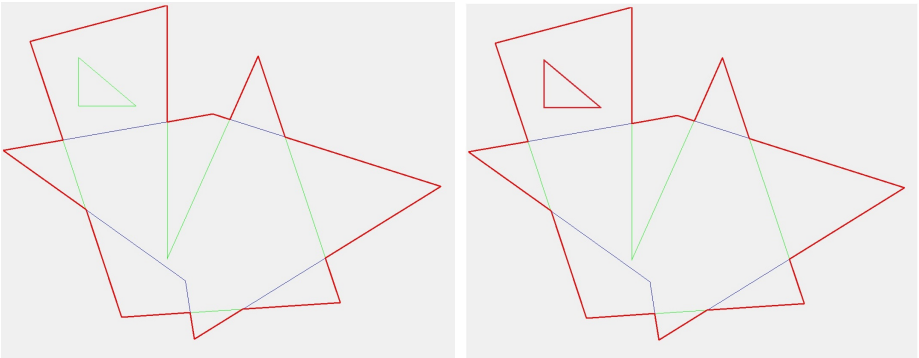
Výsledný algoritmus spojuje výše popsané kroky a

4.3.1 Implementace metody

1. if $(o(A) != CCW)$
2. $\quad A.switchOr()$ // Změň orientaci na CCW
3. if $(o(B) != CCW)$
4. $\quad B.switchOr()$ // Změň orientaci na CCW
5. $ComputeIntersections(A, B)$ // urči průsečíky A,B
6. $SetPositions(A, B)$ //Urči polohu vrcholů vůči oblastem
7. $map < QPointFB, pair < bool, vector < QPointFB >>> F$
8. $pos1 = (oper \equiv intersection \vee oper \equiv DifAB?Inner : Outer)$
9. $pos2 = (oper \equiv intersection \vee oper \equiv DifAB?Inner : Outer)$
10. $swap1 = (oper \equiv DifAB? : true : false)$
11. $swap2 = (oper \equiv DifAB? : true : false)$
12. $CreateFragments(A, pos1, swap1, F)$
13. $CreateFragments(B, pos2, swap2, F)$
14. $MergeFragments(A, B, C)$

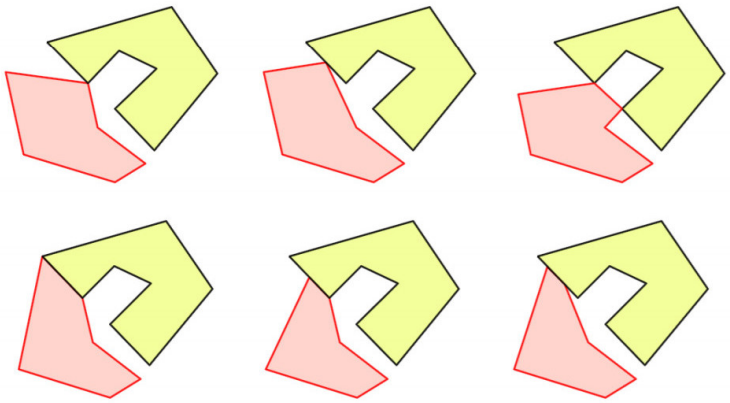
5 Problematické situace

Problematické jsou polygony obsahující otvory - bylo by třeba speciální ošetření pro tyto případy. Jinak by mohlo dojít k tomu, že například při opraci Union bude do výsledku zahrnuta celá oblast včetně otvorů jakoby tam nebyly (obrázek vlevo). Správně řešení je zobrazeno na obrázku vpravo.



Obrázek 5: Problematické situace - polygon s otvoren [1]

Dále jsou problematické situace, kdy množiny mají společný vrchol (nebo více vrcholů), společnou hranu, její část nebo segment hran.



Obrázek 6: Problematické situace [1]

6 Vstupní data

Dva polygony mohou být "nakresleny"klikáním do kanvasu pokud je zakšrnutá možnost *Draw polygons*, pomocí tlačítka *Polygon A/B* lze přepínat mezi polygony.

Dále mohou být polygony nahrány pomocí tlačítka *Load polygon*, v tom případě jsou data nahrána ze vstupního souboru *.txt, kde jsou souřadnice lomových bodů polygonů v tomto pořadí:

1. počet bodů v prvním polygonu
2. souřadnice X a Y (s desetinnou tečkou, pokud se nejedná o celé číslo) bodů prvního polygonu
3. počet bodů v druhém polygonu
4. souřadnice X a Y bodů druhého polygonu atd.

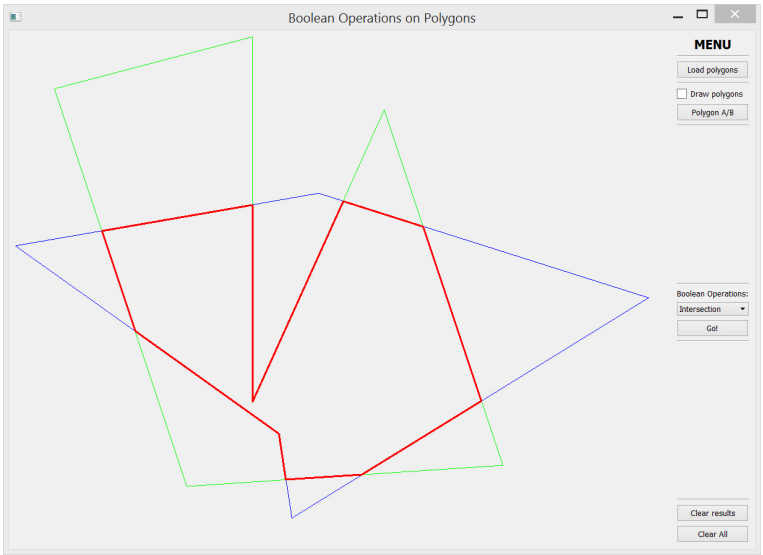
Př.: Vstupní soubor se 2 polygony, kde první polygon obsahuje 6 bodů a druhý polygon 5 bodů bude vypadat následovně:

6 5 10 20 5 20 40 30 12 39 46 15 48
5 2 25 25 20 50 30 23 51 22 43

Vše může být zapsáno do jednoho řádku, odděleno mezerami nebo jako v příkladu výše může každý útvar začínat na novém řádku.

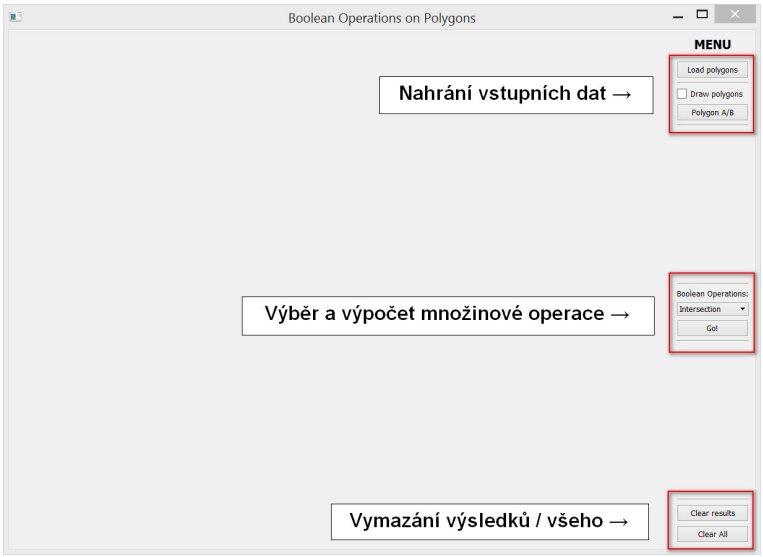
7 Výstupní data

Výsledek aplikace je zobrazen graficky na kanvasu (červenou barvou).

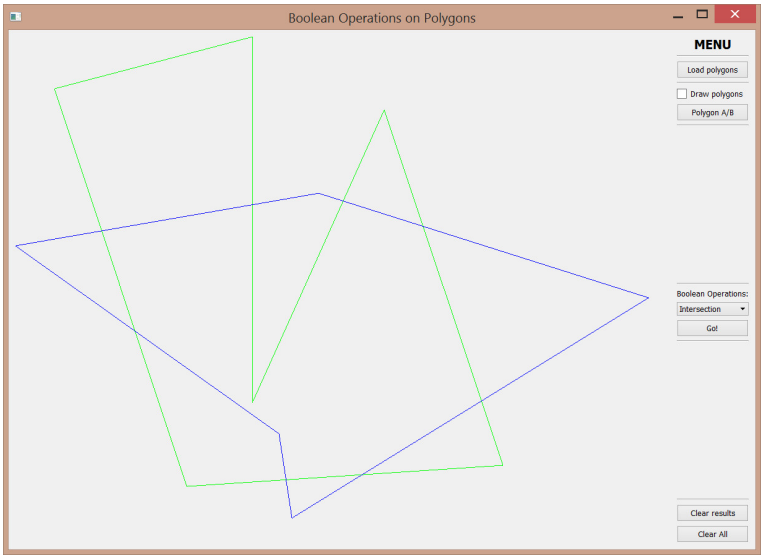


Obrázek 7: Intersection

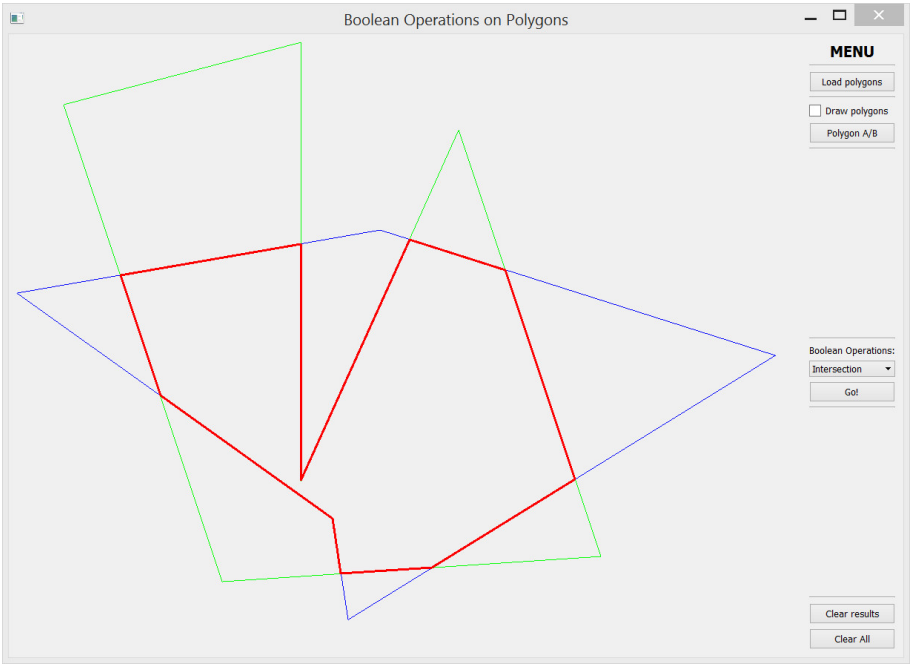
8 Ukázka vytvořené aplikace



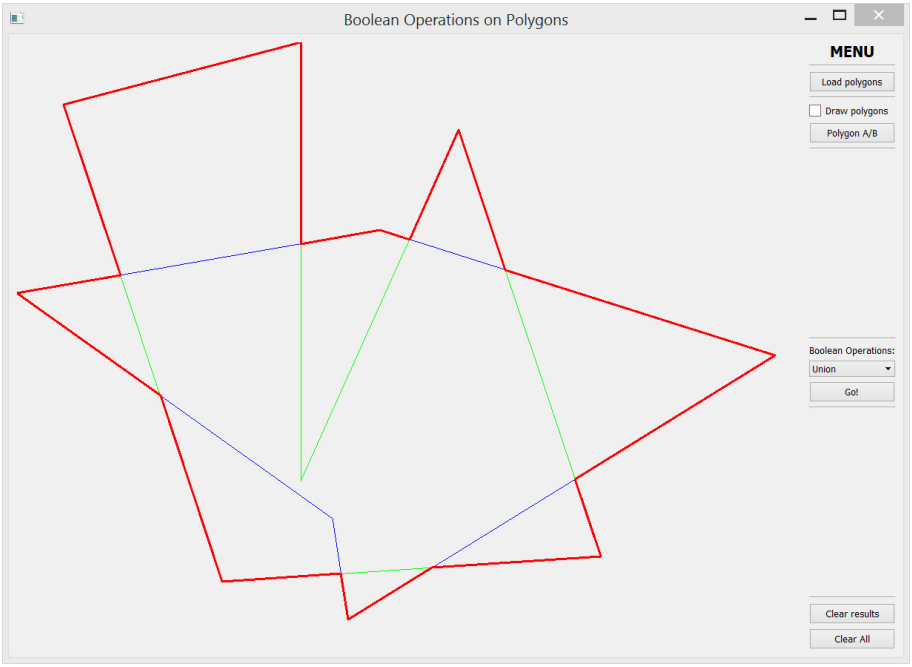
Obrázek 8: Aplikace



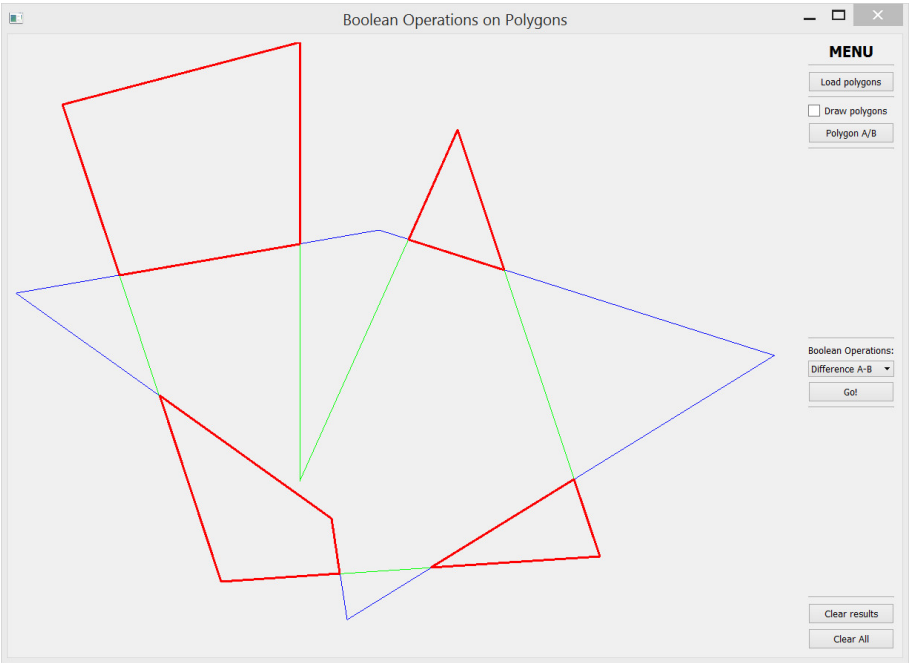
Obrázek 9: Nahrání vstupních dat



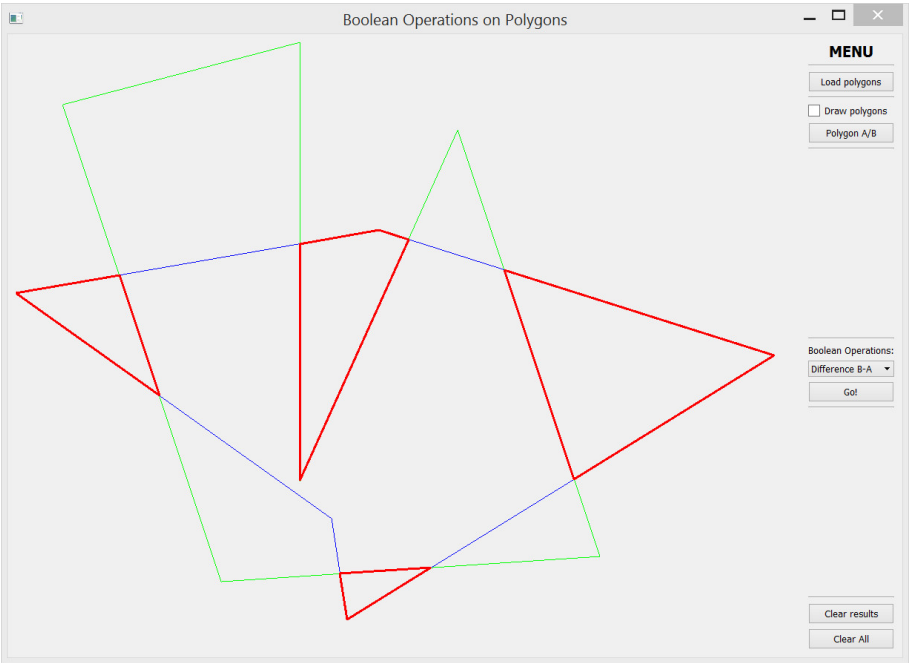
Obrázek 10: Intersection



Obrázek 11: Union



Obrázek 12: Difference A-B



Obrázek 13: Difference B-A

9 Dokumentace

9.1 Types

Tato třída je využita k definici výčtových typů použitých jako návratové hodnoty v algoritmech použitých v aplikaci.

- Výčtový typ **TPointPolygon**.
  - *INSIDE, OUTSIDE, ON*
- Výčtový typ **TBooleanOperation**.
  - *INTERSECTION, UNION, DIFFAB, DIFFBA*
- Výčtový typ **T2LinesPosition**.
  - *PARALLEL, COLINEAR, INTERSECTIONG, NONINTERSECTING*
- Výčtový typ **TPointLinePosition**.
  - *LEFT, RIGHT, COL*

9.2 QPointFB

Definice nového datového typu **QPointFB**, který dědí od třídy **QPointF**.

- Členské proměnné **alfa, beta** - parametry popisující polohu bodu vůči dvěma přímkám (implicitně nastavené na 0), jejich typem je **double**
- Členská proměnná **inters** - **bool**ovská proměnná indikující, zda jde o průsečík, implicitně nastavené na *false*
- Členská proměnná **pos** - indikuje polohu bodu vůči polygonu, jejím typem je **TPointPolygon**, impicitně nastavené na *ON*
- Public metody - getery
  - **getAlfa, getBeta, getInters, getPosition** - získání členských proměnných
- Public metody - setery
  - **setAlfa, setBeta, setInters, setPosition** - nastavení hodnot členských proměnných

9.3 Algorithms

V této třídě jsou staticky implementovány metody pro výpočet množinových operací.

- Metoda **getPositionWinding**
  - Tato metoda slouží k určení polohy bodu vůči polygonu pomocí Winding algoritmu. Jejím návratovým typem je **TPointPolygon**. Vstupem je bod **QPointFB q** a polygon **std::vector<QPointFB> pol**, jejichž vzájemná poloha je určována.

- Metoda **getPointLinePosition**
  - Tato metoda slouží k určení polohy bodu vůči linii. Jejím návratovým typem je **TPointLinePosition**. Vstupem jsou body **QPointFB q, a, b**.
- Metoda **get2LinesAngle**
  - Metoda sloužící k výpočtu úhlu mezi 2 liniemi. Jejím návratovým typem je **double**. Vstupem jsou body určující linie: **QPointFB p1, p2, p3, p4**.
- Metoda **get2LinesPosition**
  - Tato metoda slouží k určení vzájemné polohy dvou linií. Jejím návratovým typem je **T2LinesPosition**. Na vstupu jsou body **QPointFB p1, p2, p3, p4** a bod **QPointFB intersection** obsahující případně vypočtený průsečík (předává se referencí).
- Metoda **computePolygonIntersections**
  - Metoda sloužící k výpočtu průsečíků dvou polygonů. Jejím návratovým typem je **void**. Na vstupu jsou 2 polygony **std::vector<QPointFB> p1, p2**.
- Metoda **processIntersection**
  - Metoda pro processing vypočtených průsečíků z metody **computePolygonIntersections**. Slouží pro správné zařazení průsečíků do seznamu. Jejím návratovým typem je **void**. Na vstupu je vypočtený průsečík **QPointFB b**, hodnota jeho parametru **alfa/beta**, oblast zpracování **std::vector<QPointFB> poly** a index počátečního bodu hrany **int i**.
- Metoda **setPositions**
  - Metoda sloužící k určení polohy středních bodů jednotlivých hran vstupních polygonů vzhledem k druhému z polygonů. Návratovým typem je **void**. Na vstupu jsou oba vstupní polygony **std::vector<QPointFB> pol1, pol2**.
- Metoda **createFragments**
  - Metoda sloužící k vytvoření fragmentů ze sousedních bodů o stejném ohodnocení. Fragmenty ukládá do mapy. Jejím návratovým typem je **void**. Na vstupu je polygon bodů **std::vector<QPointFB> pol**, **TPointPolygon pol** - ohodnocení bodů, **bool swap** - orientace fragmentu, **std::map <QPointFB, std::pair<bool, std::vector<QPointFB>> > fragments** - mapa fragmentů.
- Metoda **createFragmentsFromVertices**
  - Pomocná metoda metody **createFragments** pro výpočet fragmentů. Výstupním typem je **bool** detekující, zda-li se fragment vypočetl. Na vstupu je index počátečního bodu fragmentu **int i_start**, body polygonu **std::vector<QPointFB> pol**, ohodnocení bodů **TPointPolygon pos**, index daného bodu **int i** a seznam bodů fragmentu (měněn referencí) **std::vector<QPointFB> fr**.

- Metoda **mergeFragments**
  - Metoda sloužící ke spojení fragmentů do výsledného polygonu a uložení do seznamu polygonů. Návratovým typem je **void**. Na vstupu je mapa fragmentů **std::map<QPointFB, std::pair<bool, std::vector<QPointFB>>> FR** a vektor vektorů bodů výsledných polygonů **std::vector<std::vector<QPointFB>> res**
- Metoda **createPolygonFromFragments**
  - Metoda sloužící k vytvoření polygonu z fragmentů. Výstupním teypem je **bool** indikující, zda byl polygon vytvořen. Na vstupu je startovní bod fragmentu **QPointFB start**, mapa fragmentů **std::map<QPointFB, std::pair<bool, std::vector<QPointFB>>> FR**, vektor bodů vytvářeného polygonu **std::vector<QPointFB> pol**.
- Metoda **getPolygonOrientation**
  - Tato metoda slouží k výpočtu výměry polygonu pomocí LLH vzorce a tedy k určení orientace bodů polygonu. Typem výstupu je **double** určující výměru a znaménkem orientaci polygonu. Na vstupu je polygon **std::vector<QPointFB> pol**.
- Metoda **BooleanOper**
  - Zastřešující metoda pro výpočet množinových operacích. Na výstupu je vektor zjištěných polygonů typu **std::vector<std::vector<QPointFB>>**. Na vstupu jsou dva polygony, nad nimiž jsou operace počítány: **std::vector<QPointFB> A, B** a zvolený typ operace **TBooleanOperation oper**.
- Metoda **resetIntersections**
  - Metoda, která slouží pro zresetování atributu průsečíku u bodů polygonu (tj. nastavení na *false*). Výstupním typem je **void**. Na vstupu je polygon **std::vector<QPointFB> A**.

9.4 Draw

Třída Draw slouží k vykreslení načtených polygonů a polygonů vypočtených množinovými operacemi. Třída dědí od třídy **QWidget**.

- Členské proměnné **std::vector<QPointFB> polA, polB** - vstupní polygony
- Členská proměnná **std::vector<std::vector<QPointFB>> res** - výsledné polygony.
- Členská proměnná **bool ab** - flag značící, který ze vstupních polygonů je právě "naklikáván".
- Členská proměnná **bool draw_pol** - flag značící, zda-li je zvolena možnost vstupu bodů polygonů pomocí klikání do kanvasu.

- Metoda **paintEvent**
  - Tato metoda slouží k vykreslení načtených (nebo naklikaných) polygonů a výsledných polygonů. Metoda se volá pomocí metody **repaint()**. Návratovým typem je **void**. Na vstupu je **QPaintEvent *e**.
- Metoda **drawPol**
  - Pomocná metoda pro vykreslení polygonu. Výstupním typem je **void**. Na vstupu je vykreslovaný polygon **std::vector<QPointFB> pol** a objekt handlující vykreslování **QPainter painter**.
- Metoda **mousePressEvent**
  - Metoda zajišťující uložení naklikaných bodů do příslušných polygonů. Výstupním typem je **void**, na vstupu je **QMouseEvent *e**.
- Metoda **setAB**
  - Metoda, kterou je zaměňován polygon, do něhož se ukládají naklikané body.
- Metody **clearAll** resp. **clearResults**
  - Vyčištění proměnných obsahujících vstupní a výsledné polygony resp. výsledné polygony. Výstupními typy jsou **void**.
- Členské metody - setery
  - Metody **setRes** resp. **setA** resp. **setB** sloužící pro uložení polygonů do příslušných členských proměnných **res** resp. **polA** resp. **polB**.
- Členské metody - getery
  - Metody **getA** resp. **getB** vracející dané polygony **polA** resp. **polB**.
- Metoda **setDrawPol**
  - Metoda sloužící ke změně flagu **draw_pol**
- Metoda **loadPoints**
  - Metoda sloužící k načení polygonů ze souboru. Výstupním typem je **void**. Na vstupu je **std::string points_path** - cesta k souboru a **QSizeF canvas_size** - velikost kanvasu.

9.5 Widget

Tato třída slouží ke komunikaci s GUI. Třída dědí od třídy QWidget. Všechny její metody slouží jako sloty k signálům z GUI, nemají žádné vstupní hodnoty a jejich návratovým typem je void.

- Metoda **on_pushButton_clicked**

  - Změna polygonu, do něhož se ukládají naklikané body.
- Metoda **on_pushButton_2_clicked**
  - Provedení zvolené množinové operace.
- Metoda **on_pushButton_3_clicked**
  - Vyčištění kanvasu od výsledku množinových operací.
- Metoda **on_pushButton_4_clicked**
  - Vyčištění kanvasu od vstupních a výsledných polygonů.
- Metoda **on_draw_poly_check_clicked**
  - Nastavení, zda-li jde body vkládat klikáním.
- Metoda **on_load_button_clicked**
  - Otevření file dialogu, načtení souboru se vstupními polygony.

10 Přílohy

- Příloha č.1: Vstupní data - *body.txt*

11 Závěr

Tato úloha byla z celého semestru nejobtížnější a (i vzhledem k volnu přes Vánoce) hůře jsme se v kódu orientovali. Také proto jsme nebyli schopní přijít na to, jak vše odladit, i když jsme si nad tím lámali hlavu.

11.1 Návrhy na vylepšení

Jako první se nabízí nedokončený offset, dále pak bonusová úloha pro polygony s otvory a již zvýšené odladění - např. ošetření situací, kdy je výsledkem bod či úsečka.

12 Zdroje

1. BAYER, Tomáš. Konvexní obálky [online][cit. 3.1.2019].
   Dostupné z: https://web.natur.cuni.cz/ bayertom/images/courses/Adk/adk9.pdf

2. BAYER, Tomáš. Konvexní obálky [online][cit. 3.1.2019].
   Dostupné z: https://web.natur.cuni.cz/ bayertom/images/courses/Adk/adkcv4.pdf