

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

FAKULTA STAVEBNÍ

PROGRAM GEODÉZIE A KARTOGRAFIE

OBOR GEOMATIKA



MNOŽINOVÉ OPERACE S POLYGONY

AUTOŘI: Bc. LINDA KLADIVOVÁ, Bc. JANA ŠPEREROVÁ

PŘEDMĚT: ALGORITMY DIGITÁLNÍ KARTOGRAFIE A GIS

Úloha č. 4: Množinové operace s polygony

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

Obsah

1	Popis a rozbor problému	5
1.1	Údaje o bonusových úlohách	7
2	Popis použitých algoritmů	8
2.1	Výpočet průsečíků obou polygonů A, B	9
2.1.1	Slovní zápis algoritmu	9
2.2	Vsunutí průsečíků do polygonů A, B	9
2.2.1	Slovní zápis algoritmu	9
2.3	Ohodnocení vrcholů polygonů A resp. B dle pozice vůči B resp. A	10
2.3.1	Slovní zápis algoritmu	10
2.4	Výběr hran podle pozice	10
2.4.1	Slovní zápis algoritmu	10
2.5	Sestavení hran pro zadanou operaci	11
2.5.1	Problematické situace a jejich rozbor	11
2.6	Winding Number Algorithm	12
2.6.1	Slovní zápis algoritmu	12
2.6.2	Výsledný algoritmus	13
3	Vstupní data	14
4	Výstupní data	14
5	Ukázka aplikace	15
6	Technická dokumentace	26
6.1	Třídy	26
6.1.1	Hlavičkový soubor Types	26
6.1.2	QPointFB	27
6.1.3	Algorithms	28
6.1.4	Draw	31
6.1.5	Widget	32
6.1.6	Edge	33
7	Závěr	34
8	Náměty na vylepšení	34
8.1	Import polygonů	34
8.2	Označení polygonu	34
	Literatura	35

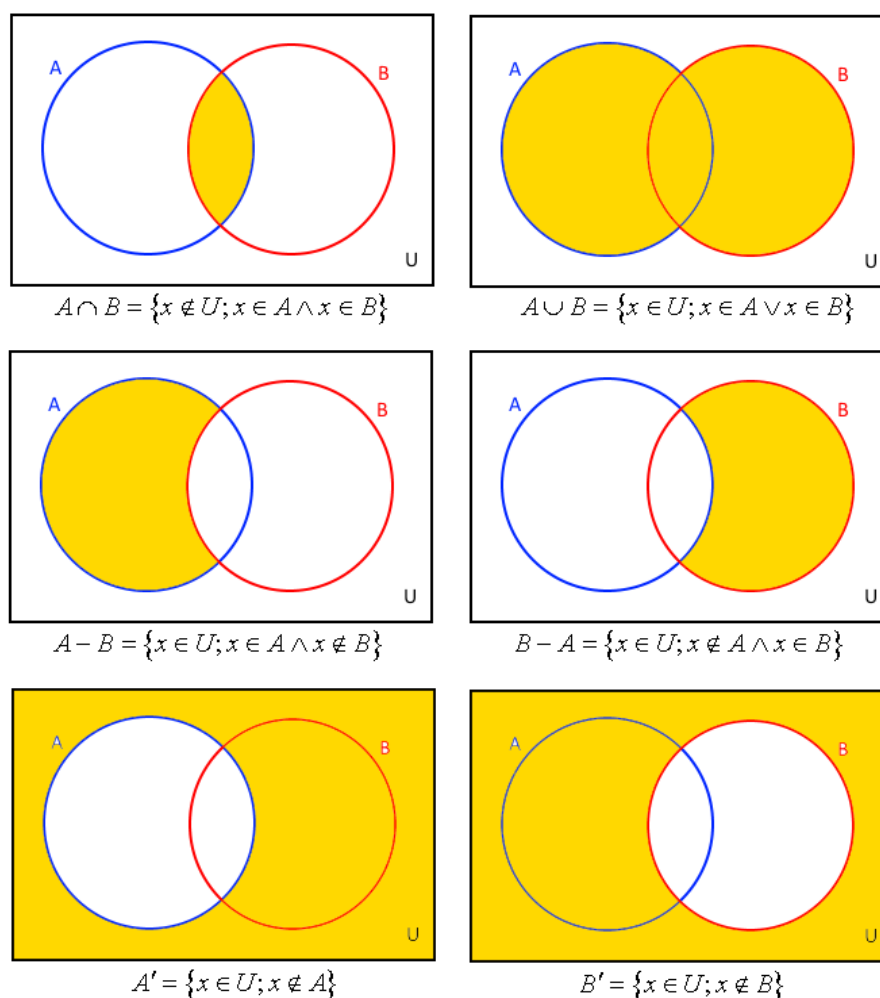
Seznam obrázků

1	Matematické operace - průnik, sjednocení, rozdíl [?]	5
2	Problematické situace u množinových operací	7
3	Ukázka vstupního textového souborů	14
4	Ukázka aplikace po importu polygonů	15
5	Sjednocení dvou polygonů	15
6	Průnik dvou polygonů	16
7	Rozdíl A-B dvou polygonů A, B	16
8	Rozdíl B-A dvou polygonů A, B	16
9	Situace A - společný bod - sjednocení	17
10	Situace A - společný bod - průnik	17
11	Situace A - společný bod - rozdíl A-B	18
12	Situace A - společný bod - rozdíl B-A	18
13	Situace B - více společných bodů - sjednocení	18
14	Situace B - více společných bodů - průnik	19
15	Situace B - více společných bodů - rozdíl A-B	19
16	Situace B - více společných bodů - rozdíl B-A	19
17	Situace C - body ve hraně druhého polygonu - sjednocení	20
18	Situace C - body ve hraně druhého polygonu - průnik	20
19	Situace C - body ve hraně druhého polygonu - rozdíl A-B	21
20	Situace C - body ve hraně druhého polygonu - rozdíl B-A	21
21	Situace D - hrana obou polygonů se shoduje - sjednocení	22
22	Situace D - hrana obou polygonů se shoduje - průnik	22
23	Situace D - hrana obou polygonů se shoduje - rozdíl A-B	23
24	Situace D - hrana obou polygonů se shoduje - rozdíl B-A	23
25	Situace E - společná část hrany - sjednocení	23
26	Situace E - společná část hrany - průnik	24
27	Situace E - společná část více hran - průnik	24
28	Situace E - společná část více hran - sjednocení	24
29	Situace E - společná část hrany - rozdíl A-B	25
30	Situace E - společná část hrany - rozdíl B-A	25
31	Situace F - hrana ležící v hraně druhého polygonu - sjednocení	25
32	Situace F - hrana ležící v hraně druhého polygonu - průnik	26

1 Popis a rozbor problému

Hlavním cílem této úlohy je vytvoření aplikace, která bude umět provádět z vygenerovaných polygonů základní matematické operace – sjednocení, rozdíl a průnik, viz obrázek 1.

Tyto operace mají velké využití a to nejen v matematice. Používáme je například v GIS, v kartografii a ve spoustě dalších odvětví. Vyskytují se všude kolem nás. Pro určení vztahů těchto operací můžeme použít Booleovské operátory, kdy datové typy mají pouze dvě hodnoty, buď TRUE nebo FALSE. Tedy máme průnik (AND), kdy platí všechna kritéria zároveň, máme sjednocení (OR), kdy platí jedno nebo druhé kritérium nebo poté máme negaci (NOT) a tedy, že něco neplatí.



Obrázek 1: Matematické operace - průnik, sjednocení, rozdíl, doplněk [?]

Sjednocení

Sjednocení je množinová operace, jejímž výstupem je oblast vyplněná všemi daty z množiny A a zároveň i všemi daty z množiny B. Čili, všechna data z obou množin spadají do požadovaného výsledku.

Průnik

Průnik je množinová operace, kde výstupem je pouze ta oblast, která obsahuje data, která spadají jak do množiny A, tak zároveň i do množiny B. Jinými slovy, výstupem je pouze ta část, kterou obě množiny mají společnou.

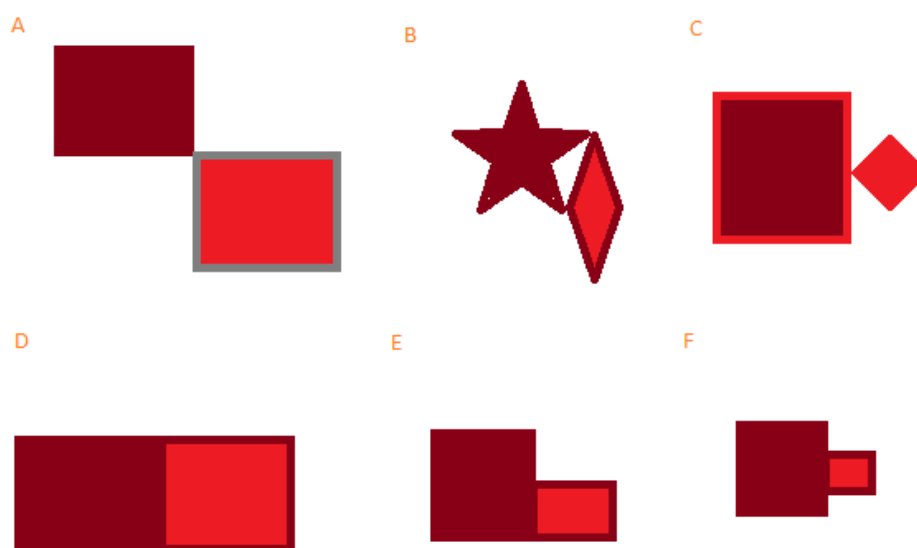
Rozdíl

Rozdíl je množinová operace, která má na výstupu oblast množiny, od které „odečítáme“ druhou množinu, avšak bez oblasti průniku těchto dvou množin. Jinak řečeno, od sjednocení těchto dvou množin odebereme celou jednu množinu.

Při bližším zkoumání výše jmenovaných matematických operací, je zřejmé, že existují singulární situace. Výsledkem operací může být linie, polygon, nebo také entita dimenze 0 (bod). Existují následující singulární situace, viz obrázek 2:

1. Jeden společný vrchol [A]
2. Vrchol leží na hraně druhého polygonu [B]
3. Více společných vrcholů [C]
4. Společná hrana [D]
5. Společná část hrany [E]
6. Hrana leží v hraně druhého polygonu [F]

Při variantách A, B (tedy 1 nebo více společných vrcholů) bude výsledkem operace INTERSECT prázdná množina, jelikož typ výsledku našich operací je vektor hran. Nicméně reálně by průnikem měl být jeden nebo více společných bodů. Při operaci UNION dojde ve všech případech ke sjednocení obou polygonů, v tomto případě součtu vínového a růžového polygonu. Pokud by měly polygony společný vrchol/-vrcholy na hraně, tak výsledkem operace UNION bude znovu součet vínového a



Obrázek 2: Problematické situace u množinových operací

růžového polygonu. Průnik bude znovu pouze v bodě, v naší aplikaci není průnik pouze v bodě řešen.

V případě variant C. D a E je výsledkem operace INTERSECTION linie. Do sjednocení tato společná hrana nepatří. Polygon, který by měl vrchol větší než stupeň dva, by byl nekorektní. Výsledek operace DIFFERENCE výše zmíněných situací by měl být jeden ze vstupních polygonů.

1.1 Údaje o bonusových úlohách

V rámci úlohy nebyly řešeny žádné bonusové úlohy.

2 Popis použitých algoritmů

Pro nakreslenou nebo naimportovanou dvojici polygonů chceme určit výsledek množinových operací. Pro tvorbu této aplikace bylo použito několik dílčích algoritmů. Výsledkem tohoto postupu jsou dílčí oblasti, na něž je pak relativně snadné aplikovat množinové operace.

Základem bylo vytvoření nového datového typu `QPointFB`, který si nese informace o parametrech α a β , které popisují, zda se jedná o průsečík, a dále si nese informaci o poloze bodu vůči polygonu.

Aby byla celá práce přehlednější, byla založena třída *types*, do které byly nadefinovány typy, jež jsou pak speciálně použity jako návratové typy různých funkcí. Jedná se o výčetové typy `TPointLinePosition` (`LeftHp = 0`, `RightHp = 1`, `Colinear = 2`), `TPointPolygonPosition` (`Inner`, `Outer`, `On`), `TBooleanOperation` (`Union`, `Intersect`, `DifferenceAB`, `DifferenceBA`) a `T2LinePosition` (`Parallel`, `Identical`, `NonIntersected`, `Intersected`). Tyto typy jsou použity ve funkcích, které byly vytvořeny pro různé fáze algoritmu.

Soupis dílčích fází algoritmu pro určení výsledku různých množinových operací:

1. Výpočet průsečíků obou polygonů A, B
2. Vsunutí průsečíků do polygonů A, B
3. Ohodnocení vrcholů polygonů A resp. B dle pozice vůči B resp. A
4. Výběr hran podle pozice
5. Sestavení hran pro zadanou množinovou operaci

Jednotlivé dílčí algoritmy jsou dále podrobně rozebrány.

2.1 Výpočet průsečíků obou polygonů A, B

Výpočet průsečíků obou polygonů A, B + seřídění V tomto algoritmu se prochází body obou polygonů a pomocí funkce `get2LinesPosition` se hledá zda se vytvořené úsečky z daného a následujícího bodu protínají. K tomuto účelu je na začátku vytvořena proměnná typu Map (slovník). Pokud je nalezen průsečík, je uložen spolu s hodnotu alfa do slovníku. Tento bod je uložen do příslušného polygonu na příslušné místo (dovnitř polygonu) pomocí funkce `ProcessIntersection`. Obdobně jsou zpracovány průsečíky u polygonu A.

2.1.1 Slovní zápis algoritmu

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: $ProcessIntersection(b, \alpha, A, i)$

2.2 Vsunutí průsečíků do polygonů A, B

Do metody `ProcessIntersection` vstupuje bod, který je průsečíkem, koeficient přímky alfa nebo beta označen t, polygon a index pořadí počátečního bodu úsečky, na které leží průsečík.

2.2.1 Slovní zápis algoritmu

1. Pokud se koeficient t blíží nule. $if(abs(t) < eps)$

Vlož průsečík za počáteční bod úsečky

2.3 Ohodnocení vrcholů polygonů A resp. B dle pozice vůči B resp. A

Počítá už s polygony, do kterých byly vloženy průsečíky. Metoda `setPosition` prochází prvně zadaný polygon a počítá středový bod daného bodu a následujícího (střed úsečky). Dále určí pozici druhotně zadaného polygonu a středu úsečky pomocí funkce `positionPointPolygonWinding`. Ke každému procházenému bodu nastaví pozici (Inner, Outer, On).

2.3.1 Slovní zápis algoritmu

1. Pro všechny body polygonu A. *for(allpointsinpolygonA)*

Vypočti střed hrany polygonu. $x/y = (poly[i] + poly[i + 1])/2$

Zjisti pozici středu oproti polygonu B . *positionPointPolygonWinding()*

Nastav pozici počátečnímu bodu hrany . *setPosition = INNER/OUTER/ON*

2. Pro všechny body polygonu B. *for(allpointsinpolygonB)*

Vypočti střed hrany polygonu. $x/y = (poly[i] + poly[i + 1])/2$

Zjisti pozici středu oproti polygonu A . *positionPointPolygonWinding()*

Nastav pozici počátečnímu bodu hrany . *setPosition = INNER/OUTER/ON*

2.4 Výběr hran podle pozice

V této části jsou vybírány hrany, které mají vůči zadanému polygonu určitou pozici. Pokud má bod v polygonu zadanou pozici vůči polygonu je sestavena hrana.

2.4.1 Slovní zápis algoritmu

1. Pro všechny body polygonu. *for(allpointsinpolygonA)*

Pokud se pozice bodu rovná zadané pozici (INNER, OUTER, ON), vytvoř hranu.

2.5 Sestavení hran pro zadanou operaci

Pokud je vybrána operace UNION, jsou vybrány vnější hrany k polygonu A (OUTER) a vnější hrany z polygonu B. Pokud je vybrána operace INTERSECT, jsou vybrány vnitřní hrany k polygonu A (INNER) a vnitřní hrany z polygonu B. Pokud je vybrána operace DIFFERENCEAB, jsou vybrány vnější hrany k polygonu A a vnitřní hrany z polygonu B a kolineární segmenty polygonu A. Pokud je vybrána operace DIFFERENCEBA, jsou vybrány vnější hrany k polygonu B a vnitřní hrany z polygonu A a kolineární segmenty polygonu B.

2.5.1 Problematické situace a jejich rozbor

Pokud je středový bod přímo na hraně polygonu, tedy nelze rozhodnout, zda je bod uvnitř nebo mimo polygon, je pozice bodu ON. To samé platí pro hrany. Pozice hrany, která není uvnitř ani mimo polygon, je rovněž ON. V obou těchto zmíněných případech jsou výsledkem operací průnik a sjednocení méně než tři segmenty. Pokud je výstupem množina definovaná méně než třemi segmenty, výstupem je potom prázdná množina.

U případů D, E, F u operací průnik i sjednocení nebyly tedy zahrnuty kolineární segmenty. Důležité je si uvědomit, že tyto kolineární hrany se nenacházejí uvnitř množiny, ale pouze na její hranici. U operace union byly zahrnuty pouze vnější segmenty, zatímco u operace intersect pouze ty vnitřní.

U průsečíků A, B, C, kde se nachází kolineární segment v podobě jednoho nebo více bodů rovněž nebyly kolineární segmenty uvažovány. U operací rozdíl kolineární segmenty zahrnuty byly, a to u operace A-B kolineární segment k polygonu A, a u operace B-A kolineární segment k polygonu B.

2.6 Winding Number Algorithm

Tento algoritmus byl využit ve funkci `setPosition` při zjišťování pozice středu úsečky polygonu A vůči polygonu B a naopak.

Tento algoritmus je také nazýván Metoda ovíjení. Půjdeme postupně po vrcholech polygonu P a budeme sčítat nebo odečítat úhly mezi daným bodem q a vrcholy P (po směru hodinových ručiček přičítáme, proti směru odečítáme). Pokud je výsledný úhel roven 2π , byl dokončen celý kruh, lze tedy prohlásit, že bod q náleží polygonu P. Pokud je výsledný úhel roven 0, bod q polygonu P nenáleží. Princip Winding algoritmu je zřejmý z obrázku. Při této metodě byla uvažována malá tolerance, jelikož je pravděpodobné, že výsledný úhel není roven přesně 2π nebo 0.

Při této metodě je zapotřebí počítat Winding Number Ω . Pro toto číslo platí, že je rovno sumě všech rotací ω proti směru hodinových ručiček, které průvodič opíše nad všemi body:

$$\Omega = \frac{1}{2\pi} \sum_{i=1}^n \omega_i$$

Platí, že pokud je úhel $\angle p_i, q, p_{i+1}$ orientován ve směru hodinových ručiček, pak $\omega_i > 0$. Naopak pokud je úhel $\angle p_i, q, p_{i+1}$ orientován proti směru hodinových ručiček, pak $\omega_i < 0$. V závislosti na výsledné hodnotě Ω lze rozhodnout o poloze bodu q vůči polygonu P:

- pokud je $\Omega = 1$, platí $q \in P$,
- pokud je $\Omega = 0$, pak platí $q \notin P$.

2.6.1 Slovní zápis algoritmu

1. Nastavení výchozího úhlu $\omega = 0$, volba tolerance $\epsilon = 1e - 6$.
2. Určení úhlu: $\omega_i = \angle p_i, q, p_{i+1}$.
3. Určení orientace o_i bodu q ke straně p_i, p_{i+1} .
4. Volba podmínky - pokud bod vlevo: $\omega = \omega + \omega_i$, v opačném případě: $\omega = \omega - \omega_i$.
5. Volba podmínky - pokud rozdíl: $||\omega| - 2\pi| < \epsilon$, pak platí: $q \in P$, v opačném případě: $q \notin P$.

2.6.2 Výsledný algoritmus

Po vytvoření zmíněných dílčích algoritmů jsou funkce postupně volány ve funkci `booleanOperations`:

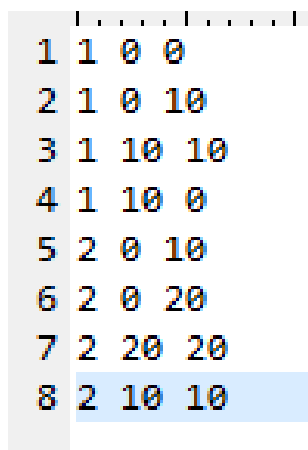
1. Výpočet průsečíků A, B : *ComputeIntersections*(A, B)
2. Určení polohy vrcholů vůči oblastem: *setPosition*(A, B)
3. Určení polohy vrcholů vůči oblastem: *setPosition*(B, A)
4. Výběr hran podle zadané operace *selectEdges*(*polygon*, *position*, *result*)

3 Vstupní data

Vstupní data jsou dva polygony, jež porovnáváme. První z nich musí být označen číslem 1 a druhý libovolným číslem různým od 1. Dále pak body polygonu musí být seřazeny od bodu č. 1 až po bod č. n, jelikož tyto body se sekvenčně ukládají do naší vytvořené proměnné `QPointF` a následně do příslušného polygonu. Vstupní data mají následující strukturu: [číslo polygonu, X-souřadnice, Y-souřadnice]

4 Výstupní data

Výstupem aplikace je grafické znázornění jednotlivých množinových operací nad importovanými nebo nad ručně vloženými polygony.

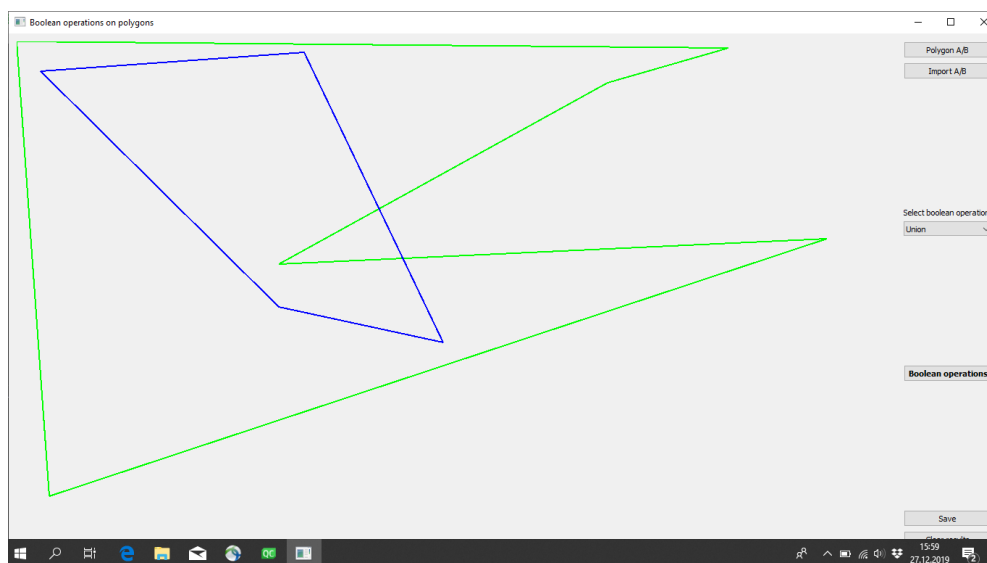


```
1 1 0 0
2 1 0 10
3 1 10 10
4 1 10 0
5 2 0 10
6 2 0 20
7 2 20 20
8 2 10 10
```

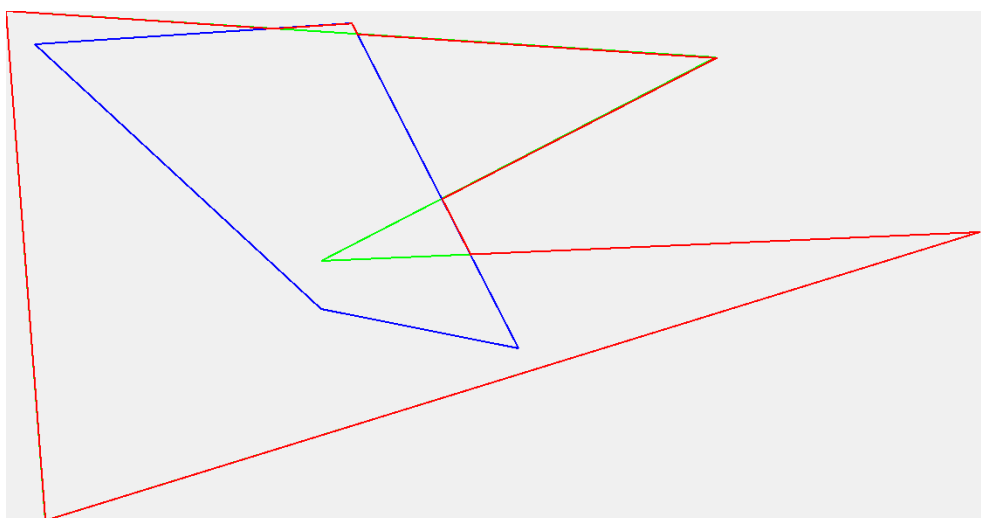
Obrázek 3: Ukázka vstupního textového souboru

5 Ukázka aplikace

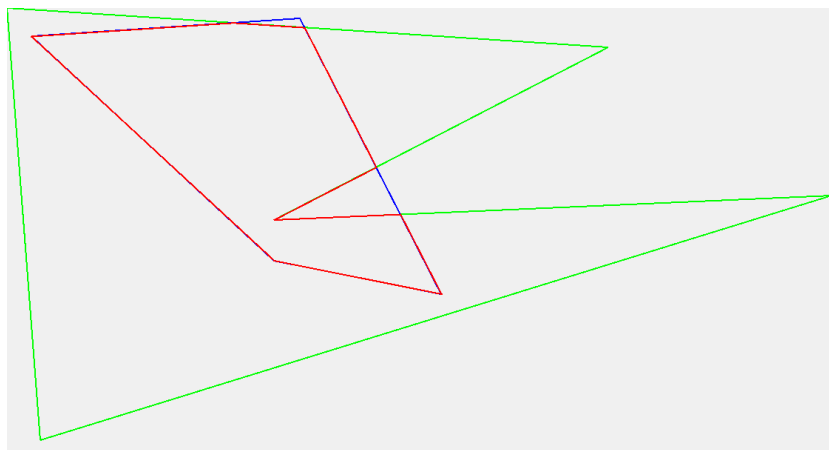
Do této kapitoly je zahrnuto několik ukázek vytvořené aplikace. Nejprve jsou ukázány případy na datech, které nevykazují žádnou singularitu. Polygon A je vyznačen zelenou barvou, polygon B modrou barvou.



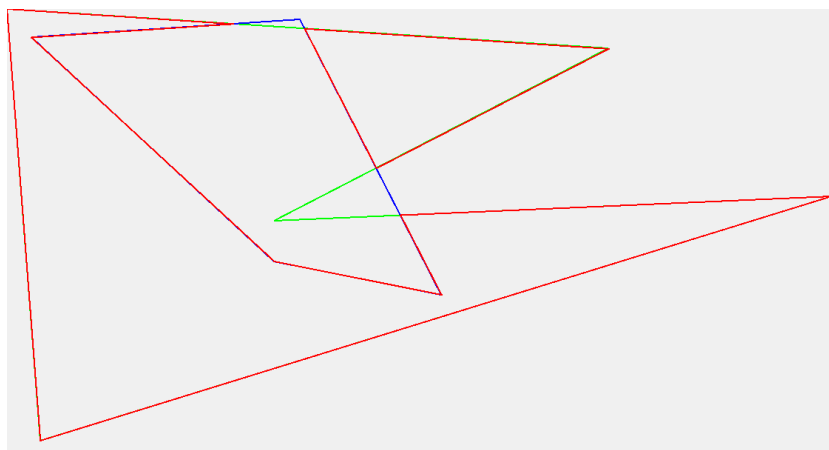
Obrázek 4: Ukázka aplikace po importu polygonů



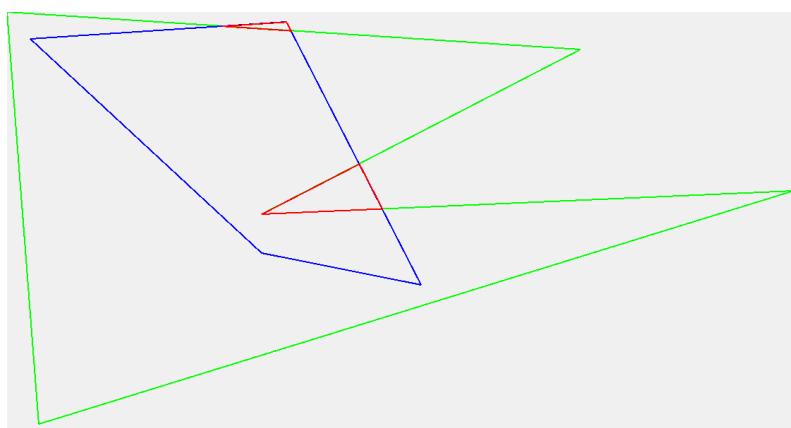
Obrázek 5: Sjednocení dvou polygonů



Obrázek 6: Průnik dvou polygonů

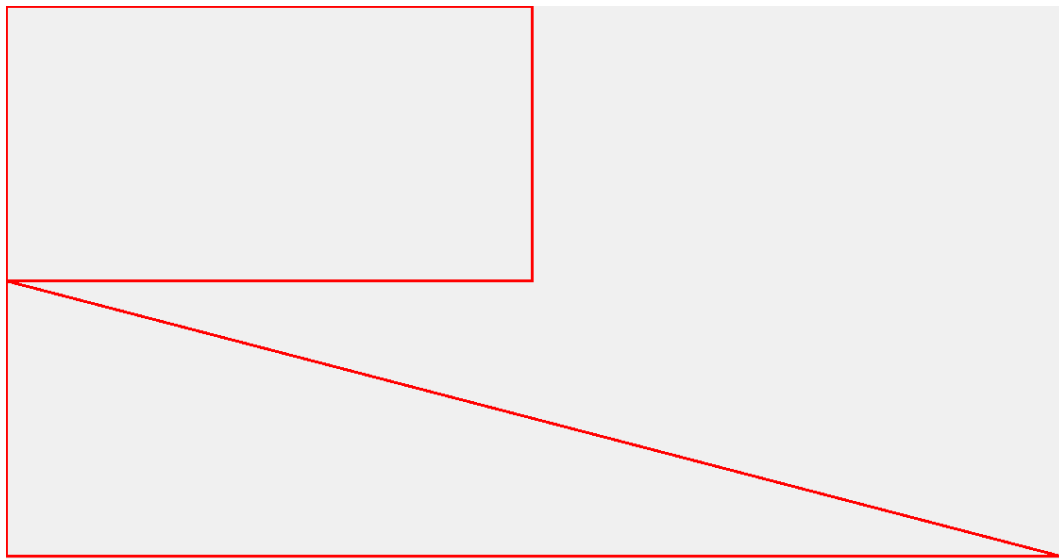


Obrázek 7: Rozdíl A-B dvou polygonů A, B

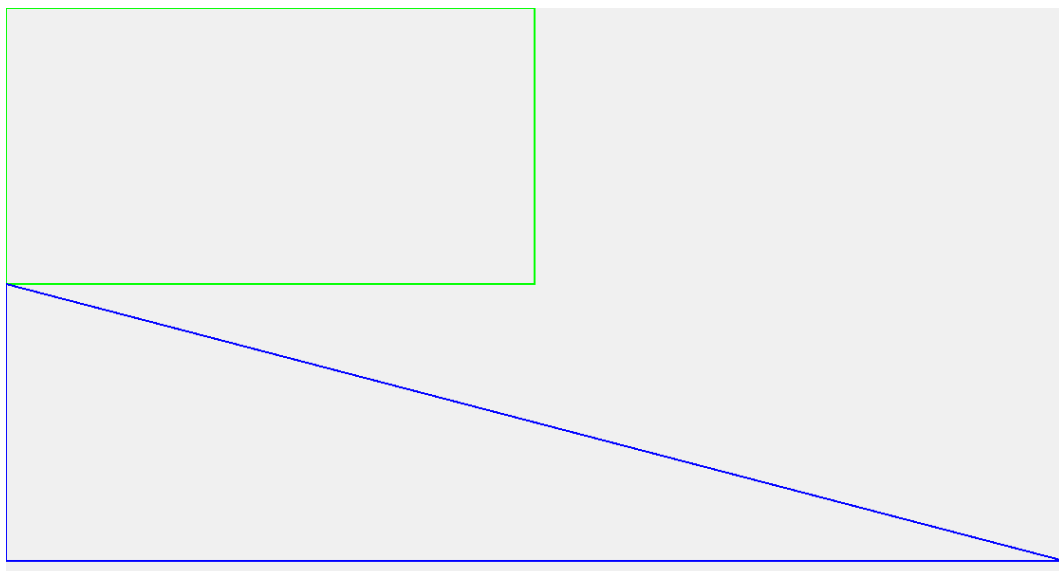


Obrázek 8: Rozdíl B-A dvou polygonů A, B

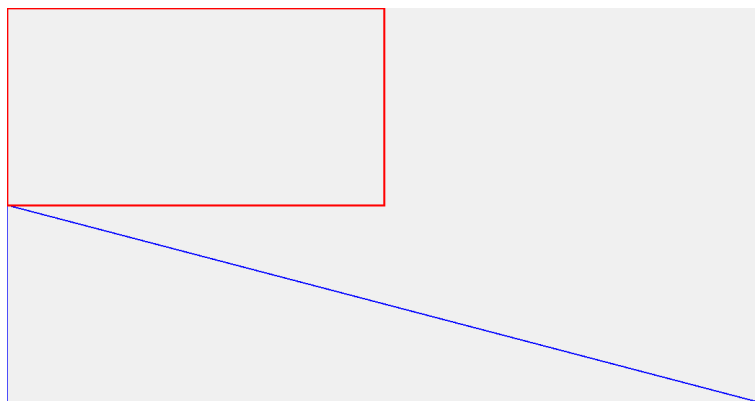
V následujících ukázkách jsou ukázány příklady singulárních situací, tak jak byly ukázány na obrázku 2. První obrázek A ukazuje situaci, kdy oba polygony mají jeden společný bod. Pokud je průsečíkem pouze 0D entita, není v aplikaci zobrazeno nic. Aplikace se soustředí pouze na společné úsečky nebo polygony. Při průniku tedy není žádná část obou polygonů zobrazena červeně, viz obrázek 10.



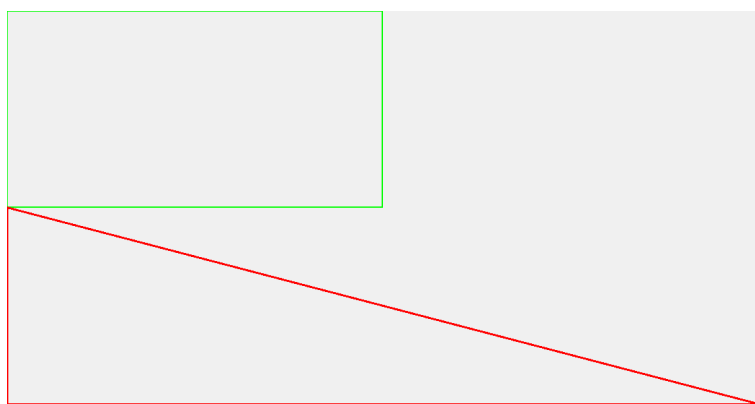
Obrázek 9: Situace A - společný bod - sjednocení



Obrázek 10: Situace A - společný bod - průnik

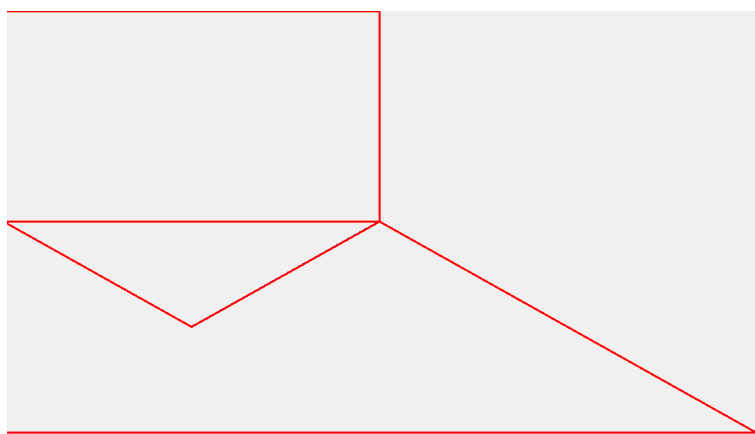


Obrázek 11: Situace A - společný bod - rozdíl A-B

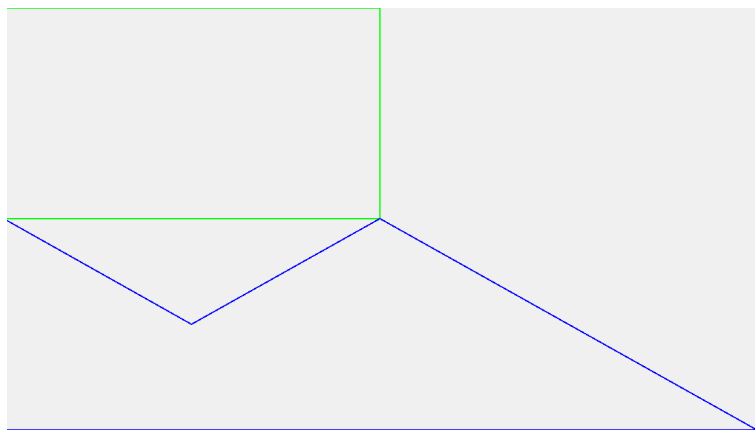


Obrázek 12: Situace A - společný bod - rozdíl B-A

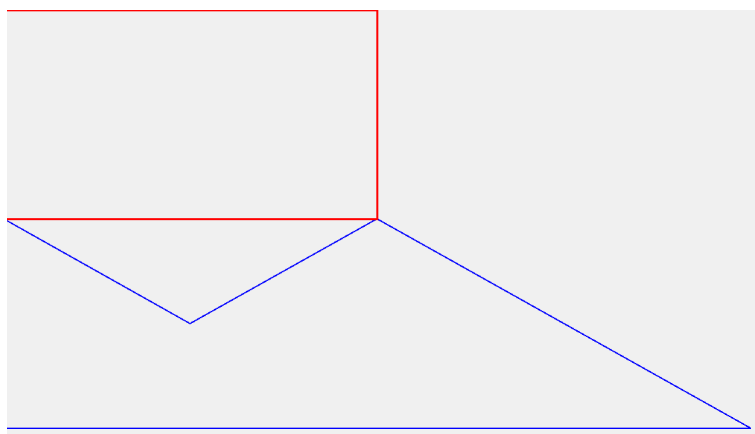
Velice podobná je situace při více společných bodech.



Obrázek 13: Situace B - více společných bodů - sjednocení



Obrázek 14: Situace B - více společných bodů - průnik

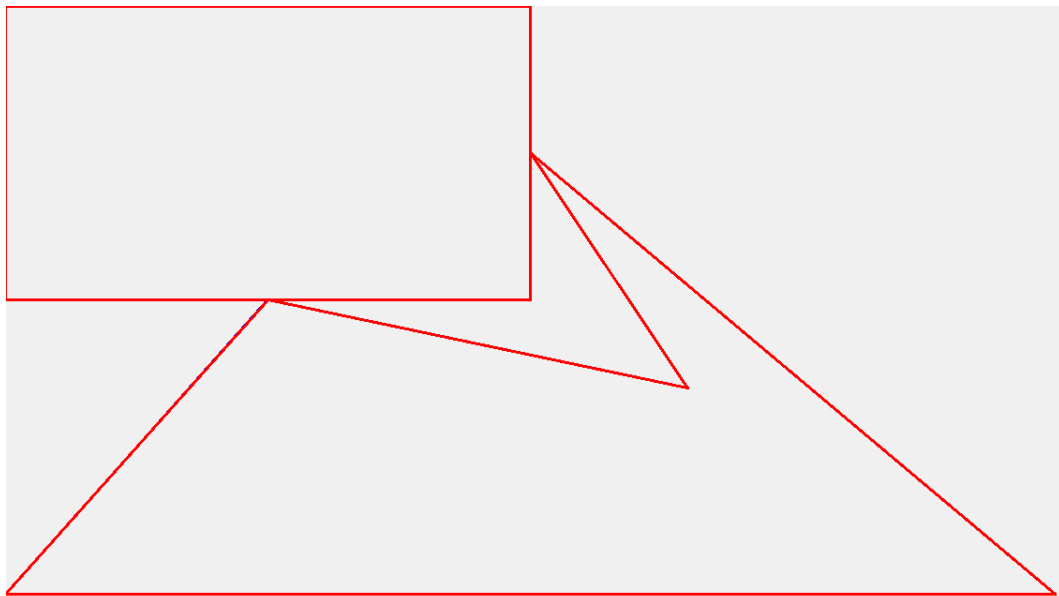


Obrázek 15: Situace B - více společných bodů - rozdíl A-B

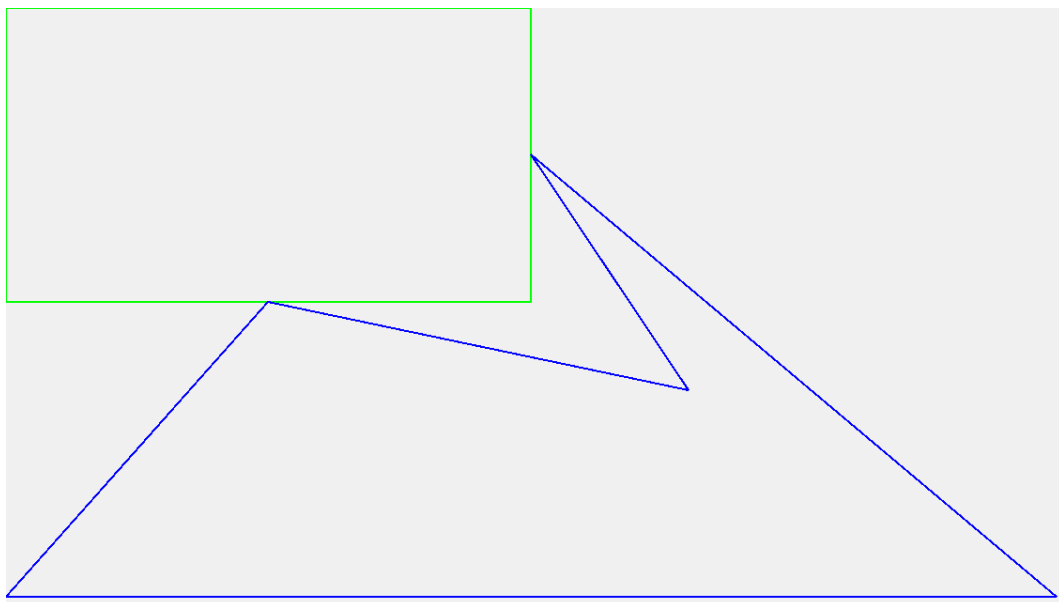


Obrázek 16: Situace B - více společných bodů - rozdíl B-A

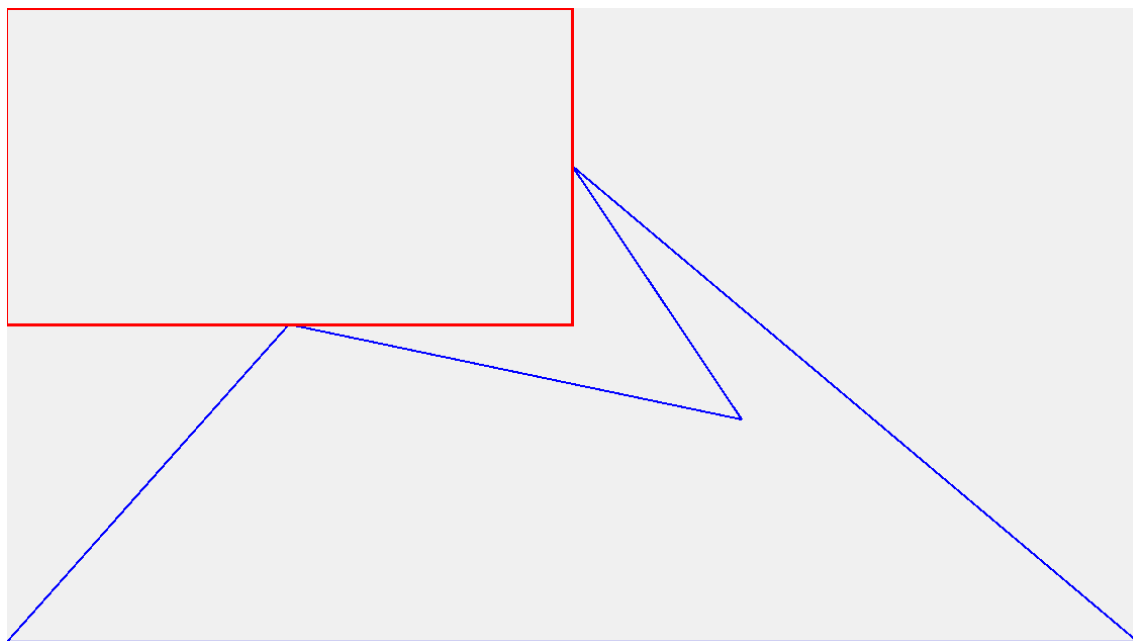
Polygony nemusí mít společný bod ve vrcholu, ale i v hraně. To je v tom případě, pokud se vrchol jednoho polygonu dotýká hrany druhého polygonu. V tomto případě je opět průnikem bod, který není v aplikaci vyznačen.



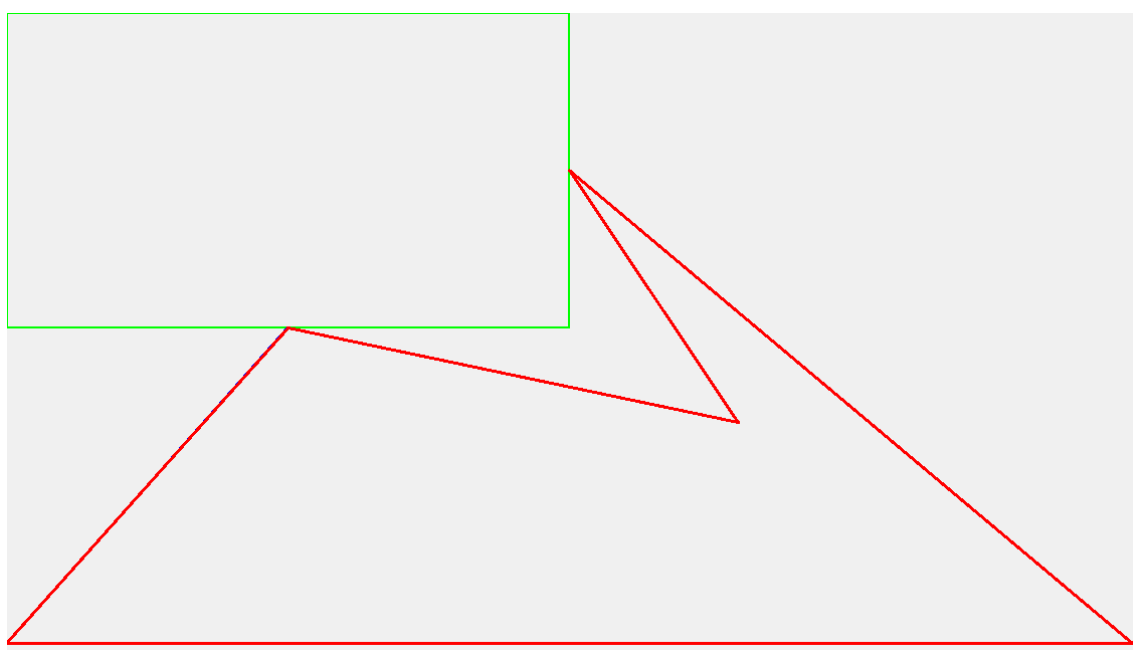
Obrázek 17: Situace C - body ve hraně druhého polygonu - sjednocení



Obrázek 18: Situace C - body ve hraně druhého polygonu - průnik

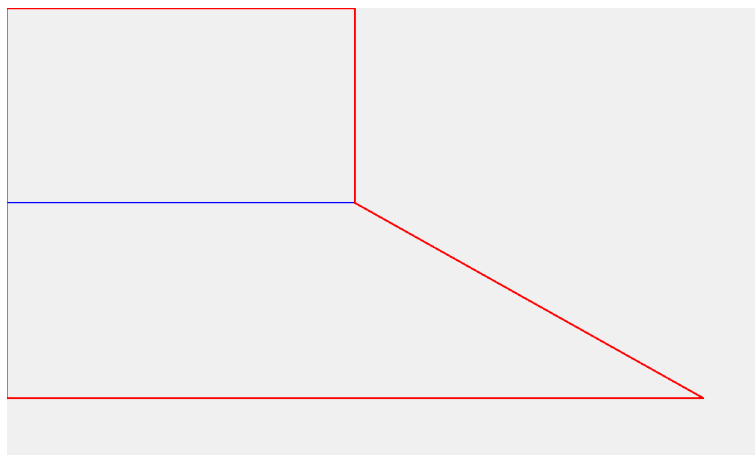


Obrázek 19: Situace C - body ve hraně druhého polygonu - rozdíl A-B

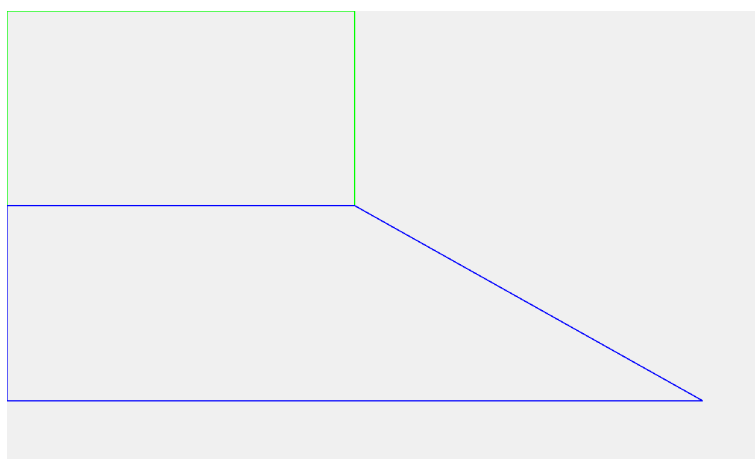


Obrázek 20: Situace C - body ve hraně druhého polygonu - rozdíl B-A

Dále může nastat případ, kdy se hrana jednoho polygonu přesně shoduje s hranou druhého polygonu. Průnikem takové situace je potom ona hrana. Do sjednocení obou polygonů tato hrana patří taktéž.

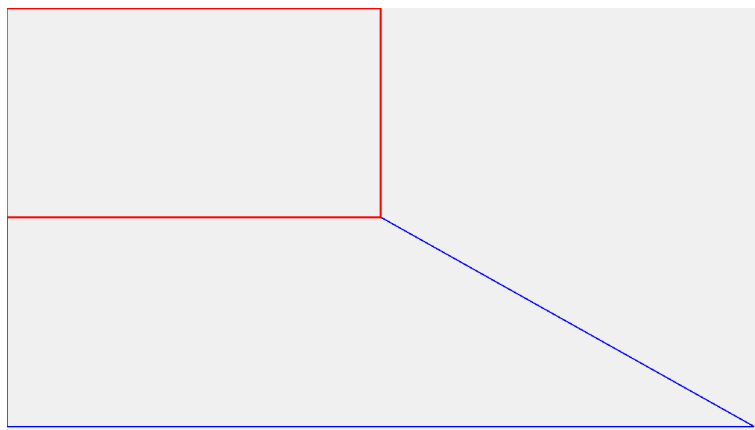


Obrázek 21: Situace D - hrana obou polygonů se shoduje - sjednocení



Obrázek 22: Situace D - hrana obou polygonů se shoduje - průnik

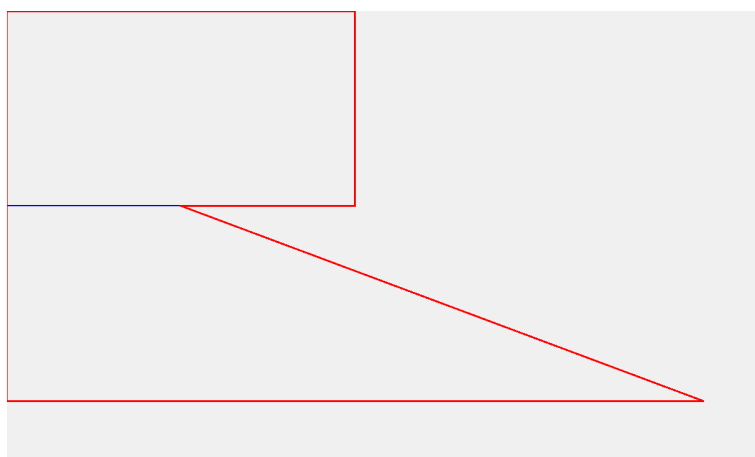
Dalšími singulárními případy je společná část hrany [E] nebo hrana ležící v hraně druhého polygonu [F]. Oba tyto případy jsou velmi podobné případu D.



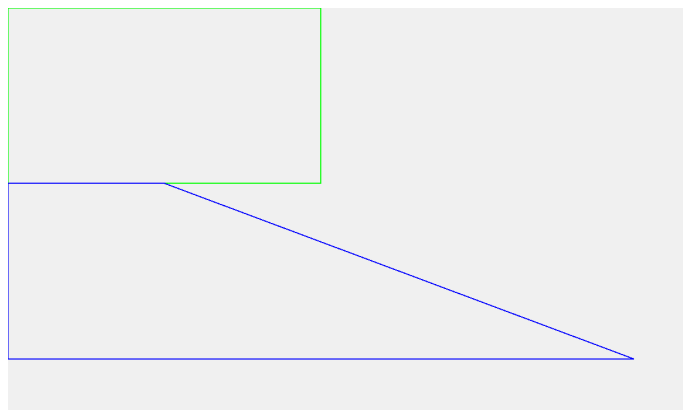
Obrázek 23: Situace D - hrana obou polygonů se shoduje - rozdíl A-B



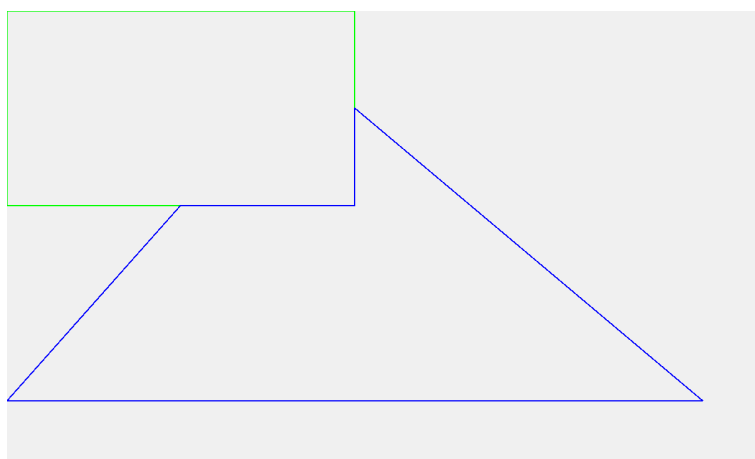
Obrázek 24: Situace D - hrana obou polygonů se shoduje - rozdíl B-A



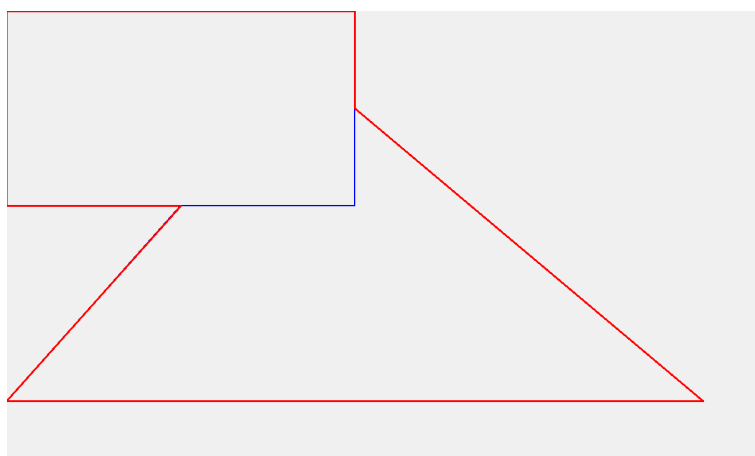
Obrázek 25: Situace E - společná část hrany - sjednocení



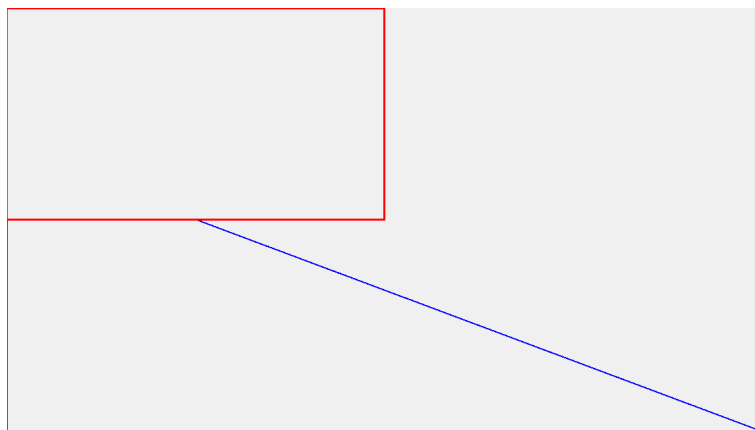
Obrázek 26: Situace E - společná část hrany - průnik



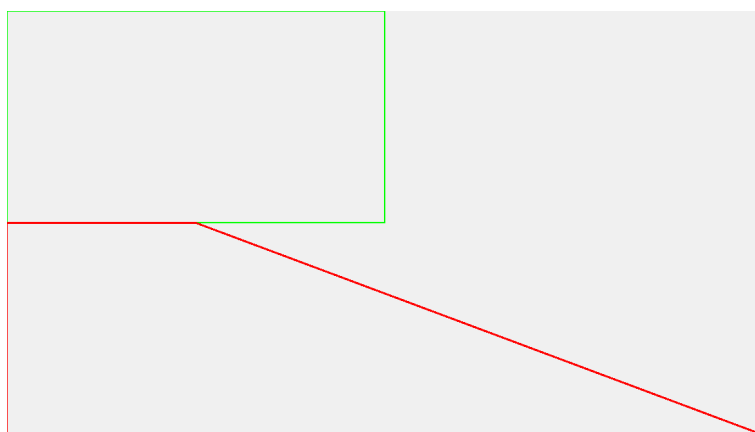
Obrázek 27: Situace E - společná část více hran - průnik



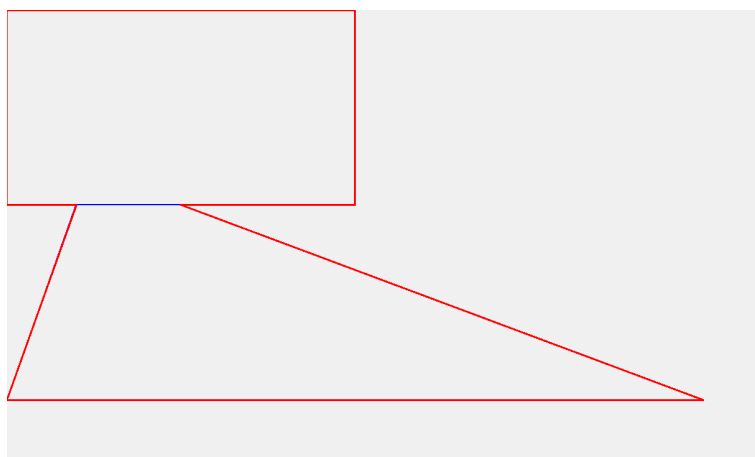
Obrázek 28: Situace E - společná část více hran - sjednocení



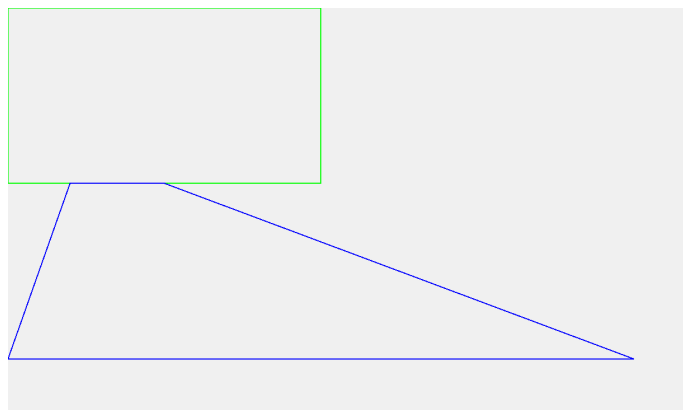
Obrázek 29: Situace E - společná část hrany - rozdíl A-B



Obrázek 30: Situace E - společná část hrany - rozdíl B-A



Obrázek 31: Situace F - hrana ležící v hraně druhého polygonu - sjednocení



Obrázek 32: Situace F - hrana ležící v hraně druhého polygonu - průnik

6 Technická dokumentace

6.1 Třídy

V aplikaci se nachází celkem pět tříd `QPointFB`, `Types`, `Algorithms`, `Draw` a `Widget`. Dále se zde nachází hlavičkový soubor `Types` s definovanými výčetovými typy.

6.1.1 Hlavičkový soubor `Types`

V tomto hlavičkovém souboru je definováno několik výčetových typů.

Prvním je typ definující polohu bodu vůči přímce. Je používán ve funkci *getPointLinePosition*, která je používána ve funkci *positionPointPolygonWinding*.

```
typedef enum {LeftHp = 0, RightHp = 1, Colinear = 2} TPointLinePosition
```

Druhým je typ definující polohu bodu vůči polygonu. Tento typ je návratovým typem funkce *positionPointPolygonWinding*. Je rozhodujícím pro výběr hran.

```
typedef enum {Inner, Outer, On} TPointPolygonPosition
```

Definuje typ množinové operace. Podle toho, která je zvolena, dojde ve funkci *booleanOperation* k výběru příslušných hran, které budou poté červeně zobrazeny

jakožto výsledek množinové operace.

```
typedef enum {Union, Intersect, DifferenceAB, DifferenceBA} TBoleeanOperation
```

Čtvrtým je typ definující polohu bodu vůči přímce. Je použit ve funkci *get2LinesPosition*, která je použita ve funkci *computePolygonIntersection*. Tento typ je tedy důležitý při výpočtu průsečíků obou polygonů.

```
typedef enum {Parallel, Identical, NonIntersected, Intersected} T2LinePosition
```

6.1.2 QPointFB

Tato třída dědí od třídy *PointF*. Konstruktor se skládá ze souřadnic x, y , které jsou odvozeny od rodičovské třídy *PointF*. Navíc jsou v konstruktoru inicializovány hodnoty α a β a pozice bodu vůči polygonu. Konstruktor má tvar:

```
QPointFB(double x, double y):PointF(x,y), alpha(0), beta(0), position(On){}
```

Ve třídě je dále definováno několik setterů a getterů:

```
void setAlpha (double alpha_)
```

```
void setBeta (double beta_)
```

```
void setPosition (TPointPolygonPosition position_)
```

```
double getAlpha()
```

```
double getBeta()
```

```
TPointPolygonPosition getPosition ()
```

6.1.3 Algorithms

Třída Algorithms obsahuje konstruktor a metody, které jsou určeny pro výpočet algoritmů používaných v digitálním GIS. Datové typy u bodu a polygonu byly zvoleny s plovoucí desetinnou čárkou (floating point).

TPointLinePosition **getPointLinePosition(QPointFB &q, QPointFB &p1, QPointFB &p2)**

Tato funkce má za úkol určit polohu bodu q vůči přímce zadané dvěma body $p1$ a $p2$. Nejprve se z vektorů vypočítá determinant. Pokud je determinant větší než tato tolerance, bod se nechází v levé polorovině a funkce vrací výčtový typ *LeftHp*. Pokud je menší, bod se nachází v pravé polorovině a funkce vrací výčtový typ *RightHp*. Pokud nenastane ani jeden z výše uvedených případů, výstupem je *Colinear*.

double **length2Points(QPointFB p, QPointFB q)**

Vrací vzdálenost dvou bodů vypočtenou z Pythagorovy věty.

double **getAngle2Vectors(QPointFB &p1, QPointFB &p2, QPointFB &p3, &QPointFB p4)**

V této funkci je pomocí norem a skalárního součinu počítán úhel mezi dvěma hranami zadanými čtyřmi body typu QPointF. Úhel je vypočten jako arcus cosinus poměru skalárního součinu a součinu obou velikostí. Defaultně se v prostředí počítá v radiánech, což bylo ponecháno.

TPointPolygonPosition **positionPointPolygonWinding(QPointF &q, QPolygonF &pol)**

Tato funkce určí polohu bodu vůči polygonu metodou Winding Number Algorithm. Vstupem je určovaný bod q a polygon P , vůči kterému je poloha určována. Hned na počátku je také zvolena tolerance (minimální hodnota) $eps = 1e-6$. V případě, že výstupem je výčtový typ *Outer*, bod leží mimo polygon, v případě že *Inner*, bod leží uvnitř polygonu. Pokud nenastane žádná z těchto uvedených možností, výstupem je hodnota *On*.

T2LinePosition get2LinesPosition(QPointF &p1, QPointF &p2, QPointF &p3, QPointF &p4, QPointF &pi)

Určí pozici dvou přímek vůči sobě. Pokud najde průsečík, uloží jej do proměnné *pi* typu reference. Nejprve se z vektorů vypočítají dílčí determinanty. Pokud jsou všechny téměř nula, funkce vrátí výčtový typ *Identical*. Pokud je první a druhý dílčí determinant téměř roven nule, funkce vrátí výčtový typ *Parallel*. Pokud nedojde k těmto případům, jsou vypočteny parametry alfa a beta. Pokud je alfa i beta mezi 0 a 1 včetně, je nalezen průsečík a vrácen text *Intersected*. Pokud nenastane žádná z těchto variant, úsečky nemají žádný společný bod a funkce vrátí *NonIntersected*.

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

Tato funkce je volána ve funkci *computePolygonIntersection* a zpracovává nalezený průsečík, který má určité parametry alfa a beta a také svůj index. Podle toho se určí, na jaké místo v polygonu má být vložen.

std::vector< Edge > booleanOperations(std::vector< QPointF > &polygonA, std::vector< QPointF > &polygonB, TBooleanOperation operation)

Vrátí výsledné hrany podle požadované operace. Nejprve vrátí průsečíky obou polygonů a vloží je na své místo do polygonů (funkce *computePolygonIntersection*). Poté vypočte pozici polygonu A vůči polygonu B a naopak. Pokud je vybrána operace UNION, jsou vybrány vnější hrany k polygonu A (OUTER) a vnější hrany z polygonu B pomocí funkce *selectEdges*. Pokud je vybrána operace INTERSECT, jsou vybrány vnitřní hrany k polygonu A (INNER) a vnitřní hrany z polygonu B. Pokud je vybrána operace DIFFERENCEAB, jsou vybrány vnější hrany k polygonu A a vnitřní hrany z polygonu B. Pokud je vybrána operace DIFFERENCEBA, jsou vybrány vnější hrany k polygonu B a vnitřní hrany z polygonu A. Pokud je středový bod přímo na hraně polygonu, nelze rozhodnout, zda je bod uvnitř nebo mimo polygon, pozice bodu ON. Body, které mají tuto pozici jsou vybrány všechny jak k polygonu A, tak k polygonu B.

void computePolygonIntersection(std::vector< *QPointFB* > &polygonA, std::vector< *QPointFB* > &polygonB)

Tato funkce hledá průsečíky pomocí funkce *get2LinesPosition* a pokud najde, uloží průsečík do mapy - klíčem je parametr alfa nebo beta a hodnotou je daný průsečík. Dále nalezený průsečík zpracuje pomocí funkce *processIntersection*.

void setPositions(std::vector< *QPointFB* > &polygonA, std::vector< *QPointFB* > &polygonB)

Po výpočtu průsečíků je třeba nastavit pozici hran vůči polygonům. Pozice se spočte pomocí Winding algoritmu a je ke konkrétnímu bodu polygonu přiřazena pomocí setteru *setPosition*.

void setPositionsAB(std::vector< *QPointFB* > &polygonA, std::vector< *QPointFB* > &polygonB)

Volá funkci *setPositions* nejprve s polygony v pořadí A, B a poté s polygony v pořadí B, A.

void selectEdges(std::vector< *QPointFB* > &pol, TPointPolygonPosition position, std::vector< *Edges* > &edges)

Vybírá hrany, jejichž počáteční bod má zadanou pozici, kterou pro konkrétní množinovou operaci hledáme.

6.1.4 Draw

Třída Draw dědí od třídy QWidget. Je v ní obsaženo několik metod a také konstruktor, který nastavuje počáteční kurzor mimo kreslicí okno.

Ve třídě je dále definováno několik setterů a getterů:

```
void setA (double std::vector< QPointFB > &a_)  
void setB (double std::vector< QPointFB > &b_)  
void setRes ((double std::vector< Edge > result_)  
double std::vector< QPointFB > getA()  
double std::vector< QPointFB > getB()  
double std::vector< Edge > getRes()
```

void changePolygon

Slouží k přepínání polygonů při kreslení kursorem myši.

void mousePressEvent

V této funkci je v závislosti na vybraném polygonu, zadaný bod buď vložen do polygonu A nebo do polygonu B.

void paintEvent

Tato metoda slouží k vykreslení obou polygonů a k vykreslení hran, které byly vybrány množinovou operací. Pro vykreslení polygonů je použita metoda *drawPolygon*.

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

Tato metoda vytváří polygon z definovaných bodů a pak ho vykreslí.

void clearResult

Metoda sloužící k vymazání výsledku množinové operace zobrazené červeně a k překreslení.

void clearAll

Metoda sloužící k vymazání obsahu zobrazovacího okna. Volá se před importem bodů.

```
void importPolygons(std::string &path, std::vector< QPointFB > & A,  
std::vector< QPointFB > &B, QSizeF &canvas_size )
```

Tato funkce slouží k importu textového souboru, ve kterém se nachází body polygonů. Struktura textového souboru je blíže popsána v kapitole Vstupní data.

6.1.5 Widget

void on_pushButton_clicked

Při kreslení bodu kurzorem můžeme změnit, který polygon chceme vykreslovat.

void on_pushButton_2_clicked

Touto funkcí se volá hlavní metoda *booleanOperations*, která se mění podle vybrané množinové operace v comboboxu.

void on_pushButton_3_clicked

Zavolá funkci *ClearResults*.

void on_pushButton_4_clicked

Zavolá funkci *ClearAll*.

void on_pushButton_5_clicked

Zavolá funkci *importPolygons* a naimportované polygony uloží pomocí setterů do privátní proměnné.

void on_Save_clicked

Umožňuje uložit výsledný obrázek množinové operace.

6.1.6 Edge

Tato třída vytváří hranu se startovním a koncovým bodem. Konstruktor má tvar:

```
Edge(QPointF &start, QPointF &end):s(start), e(end){}
```

Ve třídě je dále definováno několik setterů a getterů:

```
void setStart (QPointF &s)
```

```
void setEnd (QPointF &e)
```

```
QPointF & getStart()
```

```
QPointF & getEnd()
```

7 Závěr

Výsledná aplikace má několik funkcionalit. Umí importovat textový soubor se souřadnicemi dvou polygonů, nebo tyto polygony ručně naklikat v obrazovém okně. Následně je možné zobrazit výsledek vybrané množinové operace. V comboboxu si můžeme vybrat z následujících množinových operací: průnik, sjednocení, rozdíl A-B a rozdíl B-A. Výsledné hrany patřící do výsledku jsou zvýrazněny červenou barvou. V aplikaci bylo zapotřebí ošetřit případy, kdy vznikají kolineární segmenty. Tyto případy byly testovány skrze speciální textové soubory, na kterých si může uživatel sám vyzkoušet funkčnost aplikace. Ve výsledku průniku jsou zahrnuty pouze vnitřní hrany, podobně ve výsledku sjednocení jsou zahrnuty pouze vnější hrany. Kolineární segmenty zahrnuty nejsou. To souvisí s tím, že hrana se nenachází uvnitř polygonu, ale pouze na jeho hranici. To znamená, že u operací průnik a sjednocení nejsou společné body ani hrany vyznačeny. U operace rozdílu A-B jsou vyznačeny singulární hrany u polygonu A, u operace rozdílu B-A jsou vyznačeny singulární hrany u polygonu B.

8 Náměty na vylepšení

8.1 Import polygonů

Načítání polygonů ze souboru by mohlo být oddělené. Uživatel by načetl polygony ze dvou různých textových souborů a přitom by si zvolil, do kterého polygonu chce daný polygon uložit (A/B).

8.2 Označení polygonu

Aby byla aplikace přehlednější, bylo by vhodné polygony popsat, aby bylo jasné, zda se jedná o polygon A nebo B. V případě, že je zvolena množinová operace rozdílu, nemusí být na první pohled jasné, od kterého polygonu se který odečítá.

