ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

FAKULTA STAVEBNÍ, OBOR GEODÉZIE A KARTOGRAFIE

KATEDRA GEOMATIKY

název předmětu

ALGORITMY V DIGITÁLNÍ KARTOGRAFII A GIS

číslo úlohy	název úlohy				
4	Množinové operace s polygony				
školní rok	**studijní skup.**	**číslo zadání**	**zpracoval**	**datum**	**klasifikace**
2019/2020	60		Tomáš Bouček, Jan Šikola	16.1. 2019	

Obsah

1. Zadání

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

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

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: $P_i \cap \bar{P}_j$, resp. $P_j \cap \bar{P}_i$.

Jako vstupní data použijte existující kartografická data (např. konvertované shapefily) 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.

2. Údaje o bonusových úlohách

V rámci bonusových úloh nebyla řešena žádná úloha.

3. Popis a rozbor problému

Cílem úlohy bylo vytvořit aplikaci pomocí Qt Creatoru, která by prováděla množinové operace (sjednocení, průnik, rozdíl) na dvou polygonech (*A*, *B*) zadaných buď ručně „naklikáním“ do prostředí aplikace nebo importovaných ze souboru.

4. Popis algoritmů

Celý algoritmus na provádění množinových operací se dá rozdělit na několik dílčích algoritmů, které ve správném pořadí dají dohromady celek.

a) Výpočet průsečíků A, B + setřídění a update A, B
b) Ohodnocení vrcholů A, B dle pozice vůči B, A
c) Výběr vrcholů dle ohodnocení
d) Vytvoření hran a jejich spojení do oblastí

4.1 Výpočet průsečíků A, B + setřídění

K výpočtu průsečíků byl použit naivní algoritmus. V algoritmu se kontrolují každé dvě hrany (jedna z polygonu A, druhá z polygonu B) na existenci průsečíku. Pokud průsečík existuje, jsou spočteny jeho souřadnice. Dále na základě kritéria (hodnoty α, β) je průsečík vložen do příslušného polygonu na správnou pozici.

Výpočet průsečíku probíhá následovně:

$$\vec{u} = (x_2 - x_1, y_2 - y_1) \qquad \vec{v} = (x_4 - x_3, y_4 - y_3) \qquad \vec{w} = (x_1 - x_3, y_1 - y_3)$$

$$k_1 = v_x \cdot w_y - v_y \cdot w_x \qquad k_2 = u_x \cdot w_y - u_y \cdot w_x \qquad k_3 = v_y \cdot u_x - v_x \cdot u_y$$

$$\alpha = \frac{k_1}{k_3} \qquad \beta = \frac{k_2}{k_3}$$

Dále mohou nastat čtyři možnosti:

- úsečky jsou kolineární → průsečíků je nekonečně mnoho
- úsečky jsou lineární → průsečík neexistuje
- úsečky jsou různoběžné → průsečík existuje
- úsečky jsou mimoběžné → průsečík neexistuje

Pokud $k_1 = 0,\ k_2 = 0\ \ a\ \ k_3 = 0$, tak jsou úsečky kolineární.

Pokud $k_1 = 0\ \ a\ \ k_2 = 0$, tak jsou úsečky rovnoběžné

Pokud $\alpha \in \langle 0, 1\rangle$ a $\beta \in \langle 0, 1\rangle$, tak existuje průsečík a jsou spočteny jeho souřadnice:

$$x_b = x_1 + \alpha \cdot u_x \ \ ,$$
$$y_b = y_1 + \alpha \cdot u_y \ \ .$$

Dále je potřeba vypočtený průsečík zařadit (či nezařadit) do polygonu. O to se stará funkce *processIntersection*, jejímž vstupem je vypočtený průsečík, hodnota α či β, příslušný polygon, do kterého chceme průsečík zařadit, a index. Tato funkce má následující podobu:

processIntersection

1. if (|t| > ε && ||t| - 1| > ε): //Za *t* dosazujeme α či β
2. i ← i + 1 //Inkrementace pozice
3. P ← (b, i) //Přidání průsečíku *b* do polygonu na pozici *i*

Celý algoritmus na výpočet průsečíků má následující podobu:

computePolygonIntersection

1. for (i = 0; i < n; i++): //Cyklus pro první polygon
2. M = map<double, QPointFB> //Vytvoření mapy na ukládání průsečíků
3. for (j = 0; j < m; j++): //Cyklus pro druhý polygon
4. if ($b_{ij} \neq 0$): //Existuje průsečík
5. $M[\alpha_i] \leftarrow b_{ij}$ //Přidání průsečíku do mapy
6. *processIntersection(b_{ij}, β, B, j)*
7. if (||M|| > 0) //Alespoň jeden průsečík byl nalezen
8. for ∀ m ∈ M: //Procházení všech průsečíků v mapě
9. b ← m.second //Získání druhé hodnoty z páru
10. *processIntersection(b, α, A, i)*

Datový typ *QPointFB* je odvozenou třídou od třídy *QPointF*, a kromě souřadnic uchovává také hodnoty α a β a pozici bodu vůči polygonu.

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

Ohodnocením se myslí poloha vrcholu v daném polygonu vůči polygonu druhému. Tzn. jestli leží v něm, mimo nebo na hraně. K určení polohy byl použit *Winding Number Algorithm*. Pro polohu bodu byl vytvořen nový datový typ *TPointPolygonPosition* obsahující hodnoty *Inner*, *Outer* a *On*. Algoritmus má následující podobu:

setPositions

1. for (i = 0; i < n; i++): //Projití celého polygonu
2. $m_x = (p_A[i].x() + p_A[i+1].x()) / 2$ //x-ová souřadnice středu hrany
3. $m_y = (p_A[i].y() + p_A[i+1].y()) / 2$ //y-ová souřadnice středu hrany
4. určení pozice pomocí *Winding Number*
5. uložení pozice počátečnímu vrcholu hrany

Tento algoritmus se použije dvakrát, tedy ke zjištění pozic vrcholů polygonu A vůči polygonu B a obráceně.

4.3 Výběr vrcholů dle ohodnocení

Podle požadované množinové operace vybíráme příslušné vrcholy resp. hrany. Kdy a jaké vrcholy vybíráme znázorňuje následující tabulka:

Operace	Polygon A	Polygon B
Průnik	Inner	Inner
Sjednocení	Outer	Outer
Rozdíl A - B	Outer	Inner
Rozdíl B - A	Inner	Outer

4.4 Vytvoření hran

Vrcholy s příslušným ohodnocením jsou dále spojeny do hran a ty jsou ukládány do vektoru hran, který je následně vykreslován. Tvoření hran má následující podobu:

selectEdges

```
1)  for (i = 0; i < n; i++)                                  //Projití celého polygonu
2)      if (pozice vrcholu v polygonu == námi chtěná pozice):  //Výběr pozice
3)          vytvoř hranu e (pol[i], pol[i+1])
4)          přidej hranu do vektoru hran
```

4.5 Řešení singulárních případů

Mezi singulárními případy patří situace, kde při průniku mohou být společné jen hrany anebo společný bod. Pokud žádný z přechozích algoritmů nevrátí žádný výsledek, tak je kontrolováno, jestli polygony nemají společné body. Pokud ano, bod je zvýrazněn. Pokud je bodů víc a jsou sousední, vytvoří se z bodů hrana (případně hrany) a je barevně zvýrazněna.

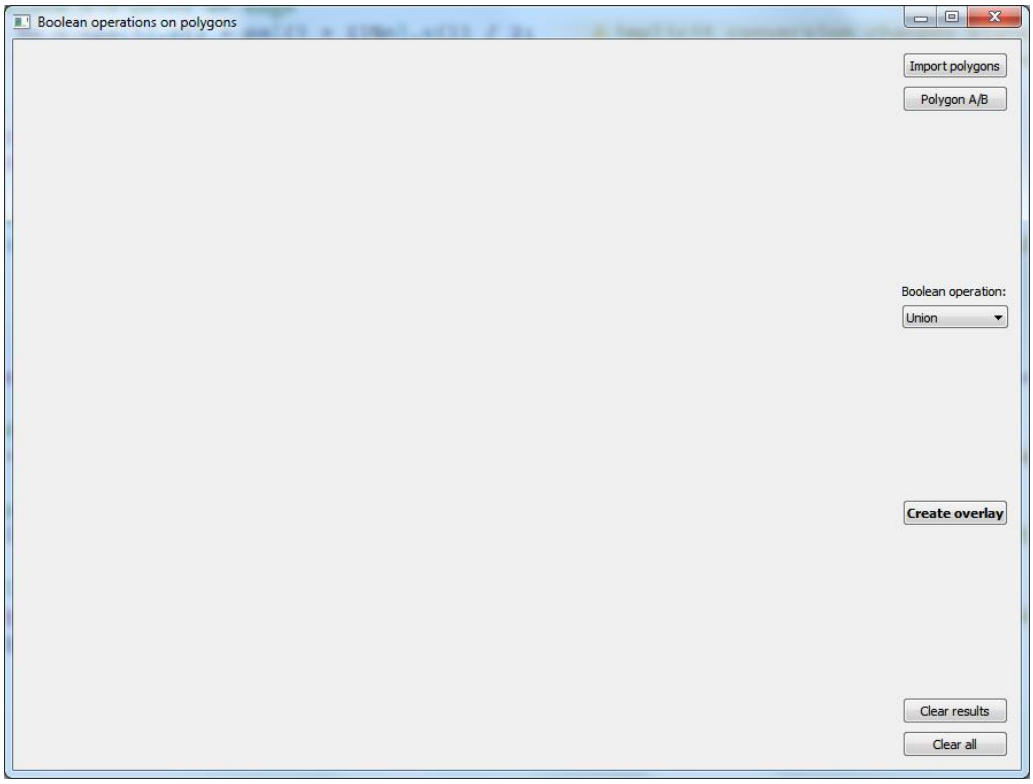
5. Vstupní data

Vstupními daty jsou dva polygony, buď zadané ručně anebo naimportovány ze souboru. Importovaný soubor je ve formě textového souboru. Soubor obsahuje *n* + *m* řádků, kde *n* je počet vrcholů prvního polygonu a *m* je počet vrcholů druhého polygonu. Každý řádek obsahuje tři hodnoty: *id*, *X*, *Y*. Hodnota *id* má dvě možnosti. Pro první polygon má hodnotu 1 a pro druhý polygon hodnotu 2.

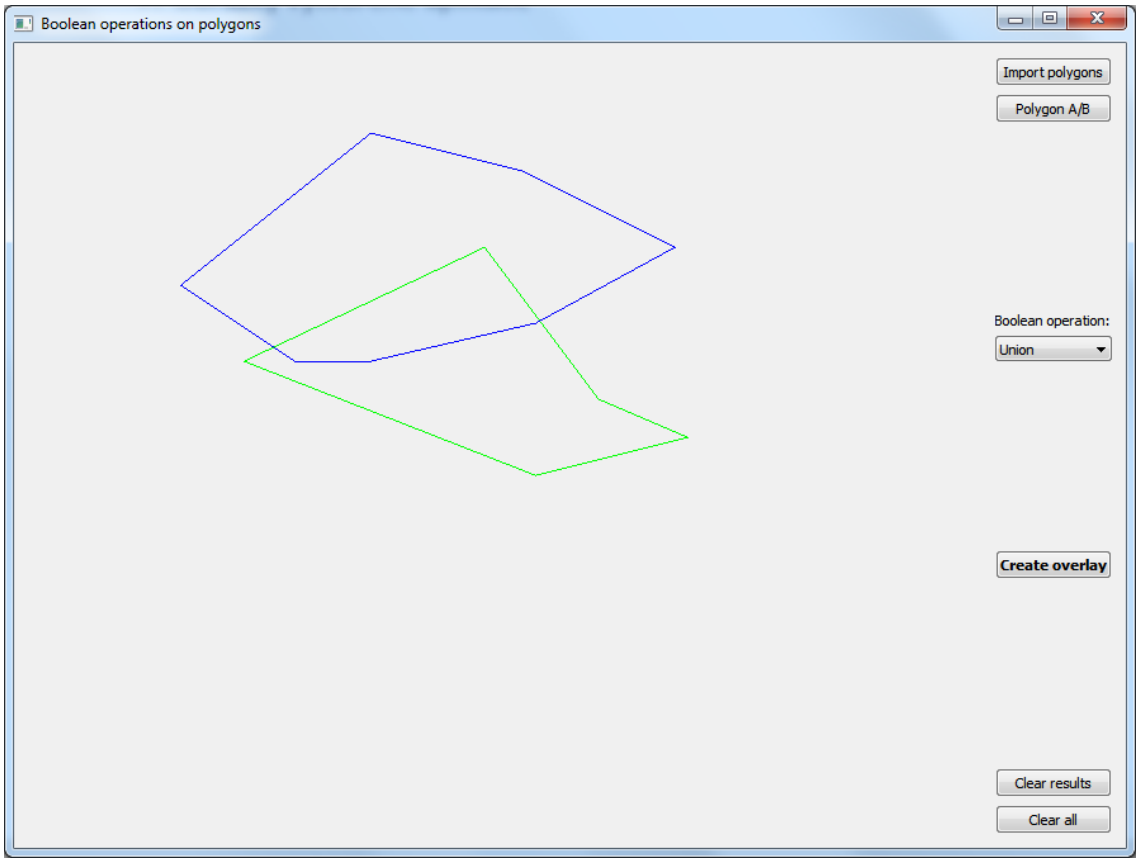
6. Výstupní data

Za výstupní data lze považovat grafické znázornění výsledku dané množinové operace.

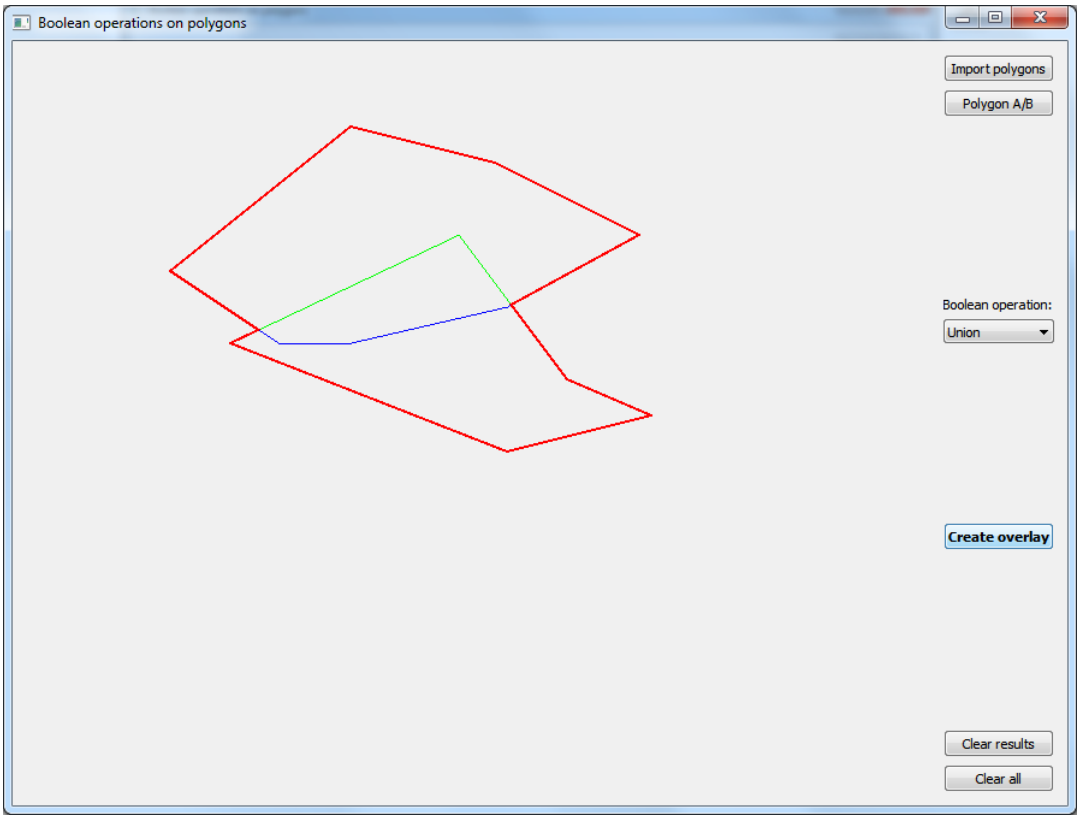
7. Obrázky vytvořené aplikace



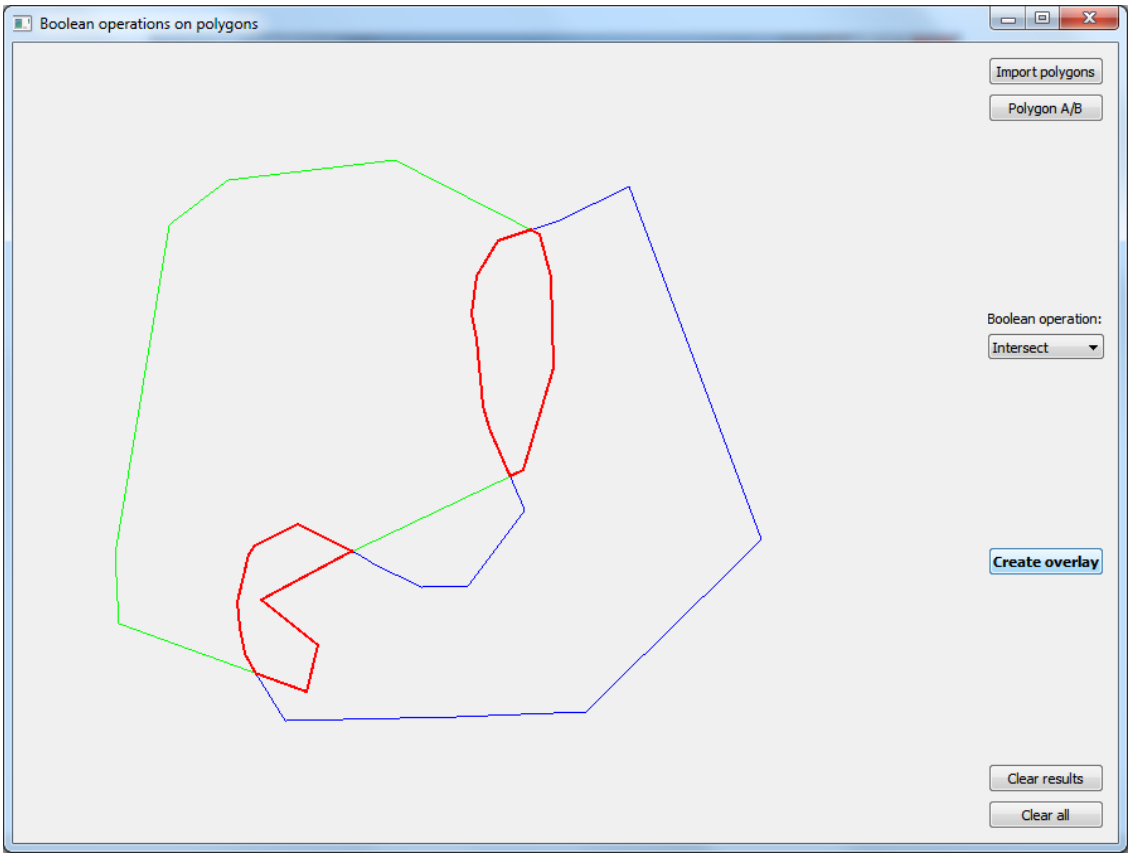
Obrázek 1: Aplikace po spuštění



Obrázek 2: Načtení polygonů



Obrázek 3: Sjednocení polygonů



Obrázek 4: Průnik

Obrázek 5: Rozdíl

8. Dokumentace

```
class Algorithms

public:

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

- funkce na určení polohy bodu vůči linii

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

- funkce na výpočet úhlu mezi dvěma vektory

```
    static TPointPolygonPosition positionPointPolygonWinding(QPointFB &q,
std::vector<QPointFB> &pol);
```

- funkce na určení polohy bodu vůči polygonu pomocí Winding Number Algorithm

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

- funkce na určení polohy dvou přímek vůči sobě

```
    static std::vector<Edge> booleanOperations(std::vector<QPointFB>
&polygonA,std::vector<QPointFB> &polygonB, TBooleanOperation operation);
```

- funkce na provedení množinových operací

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

- funkce na zařazení vypočteného průsečíku na správnou pozici v příslušného polygonu

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

- funkce na výpočet průsečíků dvou polygonů

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

- pomocná funkce pro booleanOperations

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

- funkce na určení pozice hrany

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

- funkce na vybrání příslušných hran

```
class Draw
{
private:
    std::vector<QPointFB> a, b;
```

- deklarace proměnných pro polygony A a B

```
    std::vector<Edge> res;
```

- deklarace proměnné pro výsledek množinové operace

```
    bool ab;
```

- deklarace proměnné pro volbu polygonu A/B

```
public:
    void changePolygon(){ab = !ab;}
```

- funkce na změnu polygonu při vykreslování

```
    void setA (std::vector<QPointFB> &a_){a = a_;}
```

- funkce na nastavení polygonu A

```
    void setB (std::vector<QPointFB> &b_){b = b_;}
```

- funkce na nastavení polygonu B

```
    void setRes (std::vector<Edge> res_){res = res_;}
```

- funkce na nastavení výsledku

```
std::vector<QPointFB> getA(){return a;}
```

- funkce na získání polygonu A

```
std::vector<QPointFB> getB(){return b;}
```

- funkce na získání polygonu B

```
std::vector<Edge> getRes(){return res;}
```

- funkce na získání výsledku

```
void clearResults() {res.clear();}
```

- funkce na smazání výsledku

```
void clearAll() {res.clear(); a.clear(); b.clear();}
```

- funkce na smazání výsledku a obou polygonů

```
void mousePressEvent(QMouseEvent *event);
```

- funkce na získání souřadnic po kliknutí do aplikace

```
void paintEvent(QPaintEvent *event);
```

- funkce na vykreslení polygonů a hran

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

- pomocná funkce na vykreslování polygonů

```
void loadFile(std::string &path);
```

- funkce na načtení polygonů

Kód dále pracuje s vytvořenou třídou *Edge*, která slouží k práci s hranami. Umožňuje získat/nastavit počáteční a koncový bod hrany. Další vytvořenou třídou je *QPointFB*, která je odvozená od třídy *QPointF*. Z ní dědí souřadnice *X* a *Y*, byly přidány hodnoty α, β a pozice. S těmito parametry je možno pracovat pomocí „*get*“ a „*set*“ funkcí. Hlavičkový soubor *Types* slouží k vytvoření nových datových typů pro snadnější práci v kódu.

9. Závěr

Byla vytvořena aplikace na množinové operace s polygony. Polygony je možné kreslit do „Canvasu“ anebo načtení polygonů z textového souboru v příslušném formátu.

10. Náměty na vylepšení

- Bylo by vhodné množinové operace plošně vybarvit kvůli přehlednosti.
- Načtené polygony by se mohli přizpůsobovat zobrazovacímu oknu.