České vysoké učení technické v Praze
Fakulta stavební
Katedra geomatiky

**Technická zpráva**

# Algoritmy v digitální kartografii

## Úloha č. 3: Digitální model terénu

**Bc. Pane Kuzmanov**
**Bc. František Mužík**

Studijní program: Geodézie a kartografie
Specializace: Geomatika

Praha 2021

## Úloha č. 3: Digitální model terénu

*Vstup: množina* $P = \{p_1, ..., p_n\}$, $p_i = \{x_i, y_i, z_i\}$.

*Výstup: polyedrický DMT nad množinou* $P$ *představovaný vrstevnicemi doplněný vizualizací sklonu trojúhelníků a jejich expozicí.*

Metodou inkrementální konstrukce vytvořte nad množinou $P$ vstupních bodů 2D Delaunay triangulaci. Jako vstupní data použijte existující geodetická data (alespoň 300 bodů) popř. navrhněte algoritmus pro generování syntetických vstupních dat představujících významné terénní tvary (kupa, údolí, spočinek, hřbet, ...).

Vstupní množiny bodů včetně níže uvedených výstupů vhodně vizualizujte. Grafické rozhraní realizujte s využitím frameworku QT. Dynamické datové struktury implementujte s využitím STL.

Nad takto vzniklou triangulací vygenerujte polyedrický digitální model terénu. Dále proveďte tyto analýzy:

- S využitím lineární interpolace vygenerujte vrstevnice se *zadaným krokem* a v *zadaném intervalu*, proveďte jejich vizualizaci s rozlišením zvýrazněných vrstevnic.
- Analyzujte sklon digitálního modelu terénu, jednotlivé trojúhelníky vizualizujte v závislosti na jejich sklonu.
- Analyzujte expozici digitálního modelu terénu, jednotlivé trojúhelníky vizualizujte v závislosti na jejich expozici ke světové straně.

Zhodnoťte výsledný digitální model terénu z kartografického hlediska, zamyslete se nad slabinami algoritmu založeného na 2D Delaunay triangulaci. Ve kterých situacích (různé terénní tvary) nebude dávat vhodné výsledky? Tyto situace graficky znázorněte.

Zhodnocení činnosti algoritmu včetně ukázek proveďte alespoň na **3 strany** formátu A4.

**Hodnocení:**

| **Krok** | **Hodnocení** |
|---|---|
| Delaunay triangulace, polyedrický model terénu. | 10b |
| Konstrukce vrstevnic, analýza sklonu a expozice. | 10b |
| *Triangulace nekonvexní oblasti zadané polygonem.* | *+5b* |
| *Výběr barevných stupnic při vizualizaci sklonu a expozice.* | *+3b* |
| *Automatický popis vrstevnic.* | *+3b* |
| *Automatický popis vrstevnic respektující kartografické zásady (orientace, vhodné rozložení).* | *+10b* |
| *Algoritmus pro automatické generování terénních tvarů (kupa, údolí, spočinek, hřbet, ...).* | *+10b* |
| *3D vizualizace terénu s využitím promítání.* | *+10b* |
| *Barevná hypsometrie.* | *+5b* |
| **Max celkem:** | **65b** |

Čas zpracování: 4 týdny

# 1 Údaje o bonusových úlohách

## 1.1 Triangulace nekonvexní oblasti zadané polygonem (+5b)

## 1.2 Výběr barevných stupnic při vizualizaci sklonu a expozice (+3b)

Byla implementována možnost výběru dvou barevných palet pro každou z vizualizací. Pro sklon terénu je možné zvolit jednu z variant "Grayscale" – odstíny šedé nebo "Yellow to Red" – žluto – červená paleta. Ve výběru pro expozici lze najít dvě možnosti: KM – zvolená námi (obr. 3) nebo barevnou paletu, kterou používá ArcGIS (obr. 4). Detailnější popis je v kapitolách 3.2 a 3.3.

## 1.3 Automatický popis vrstevnic (+3b)

Byl implementován automatický popis hlavních vrstevnic. Hlavní vrstevnice je každá pátá vrstevnice, která je zároveň výrazněji vykreslená. Více informací je v kapitole 3.4.1.

## 1.4 Automatický popis vrstevnic respektující kartografické zásady (+10b)

## 1.5 Algoritmus pro automatické generováních terénních tvarů (+10b)

## 1.6 3D vizualizace terénu s využitím promítání (+10b)

## 1.7 Barevná hypsometrie (+5b)

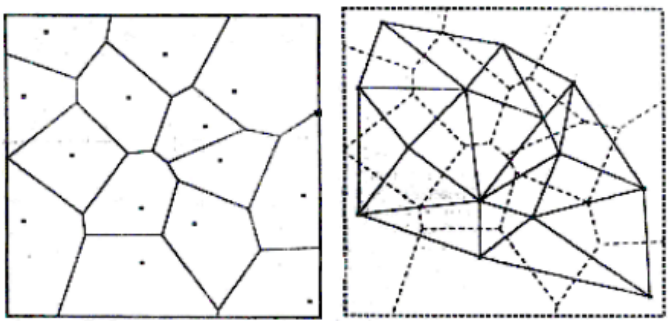
# 2 Popis a rozbor problému

Je zadaná množina bodů s prostorovými souřadnicemi X,Y,Z. Nad těmito body je vytvořen polyedrický digitální model terénu s využitím Delaunayho triangulace. Součástí řešení je taktéž tvorba vrstevnic, včetně jejich popisu. Dále je možné zobrazit sklon terénu (tedy sklon jednotlivých trojúhelníků v triangulaci) a expozici ke světovým stranám (rozděleno na 8 stran).

# 3 Popisy algoritmů formálním jazykem

## 3.1 Delaunay triangulace

Jedná se o nejčastěji využívanou metodu triangulace. Základní funkčnost metody spočívá v rozdělení množiny bodů tzv. Thiessenovými polygony, jejichž vlastností je ohraničení každého z bodů vstupní množiny tak, aby z každého místa v ohraničeném polygonu byl nejbližším bodem právě ten ohraničený (obr. 2). Hrany polygonů tvoří osu spojnice dvou bodů.



Obrázek 1: Ukázka Thiessenových polygonů (vlevo) a výsledné triangulace (vpravo) [4].

Jestliže opsaná kružnice nad spojenou hranou dvou sousedních bodů neprotne jiný bod nebo jestliže se jiný bod nenachází uvnitř kružnice. Průběžným výpočtem je v algoritmu vytvářen seznam hran (AEL – Active Edge List), pro které nebyl nalezen třetí bod pro triangulaci. Při nalezení nové hrany je potřeba zjistit, zda – se již nenachází v AEL a tedy není případně duplicitní. Duplicitní hrany jsou odstraněny. Ve chvíli, kdy je AEL prázdný, je triangulace hotova.

Nutné podmínky pro vytvoření triangulace:

- Žádný bod se nenachází uvnitř kružnice opsané libovolnému trojúhelníku triangulace.
- Libovolné čtyři body by neměly ležet na kružnici – pak je triangulace nejednoznačná.
- Maximalizace nejmenšího úhlu v trojúhelníku.
- Splnění lokálních i globálních kritérií (vyvarování se tupých a velmi malých úhlů v trojúhelníku, minimální suma délek stran...)

**Implementace algoritmu:**

1. Nalezení náhodného a nejbližšího bodu: $p_1 = rand(P), \|p_1 - p_2\| = min.$
2. Vytvoř hranu $e = (p_1 p_2)$
3. $\underline{p} = arg\ min_{\forall p_i \in \sigma_L(e)} r'(k_i), k_i = (a, b, p_i), e = (a, b)$
4. Pokud neexistuje $\underline{p}$, prohoď orientaci $e \leftarrow (p_2 p_1)$. Jdi na 3)
5. Výpočet zbývajících hran trojúhelníka: $e_2 = (p_2, \underline{p}), e_3 = (\underline{p}, p_1)$
6. Přidání 3 hran do AEL: $AEL \leftarrow e, AEL \leftarrow e_2, AEL \leftarrow e_3$
7. Přidání 3 hran do DT: $DT \leftarrow e, DT \leftarrow e_2, DT \leftarrow e_3$
8. Dokud AEL není prázdný:
9. Vybrání první hrany z AEL: $AEL \rightarrow e, e = (p_1 p_2)$
10. Prohození její orientace: $e = (p_2 p_1)$
11. $\underline{p} = arg\ min_{\forall p_i \in \sigma_L(e)} r'(k_i), k_i = (a, b, p_i), e = (a, b)$
12. Pokud existuje $\underline{p}$:

13. Zbývající hrany trojúhelníka: $e_2 = (p_2, p), e_3 = (\underline{p}, p_1)$

14. Přidání hrany do DT, ale nikoliv do AEL: $DT \leftarrow e$

15. Přidání hrany do DT i do AEL: $add(e_2, AEL, DT), add(e_3, AEL, DT)$

## 3.2 Výpočet sklonu

Výpočet sklonu trojúhelníku je učen jako úhel $\varphi$ mezi svislicí a normálou trojúhelníka $n$. Výpočet probíhá vektorově nad každým z trojúhelníků DMT.

1. Určení vektorů $\overrightarrow{u} = (u_x, u_y, u_z) = (x_1 - x_2, y_1 - y_2, z_1 - z_2)$, $\overrightarrow{v} = (v_x, v_y, v_z) = (x_3 - x_2, y_3 - y_2, z_3 - z_2)$

2. Výpočet normálového vektoru trojúhelníka: $\overrightarrow{n} = (n_y, n_y, n_z) = (u_y \cdot v_z - v_y \cdot u_z, -u_x \cdot v_z + v_x \cdot u_z, u_x \cdot v_y - v_x \cdot u_y)$

3. Sklon $\varphi = \dfrac{n_z}{|n|}$

### 3.2.1 Výběr barevné palety

Trojúhelníky jsou vizualizovány na základě hodnoty $\varphi$. Vizualizace je možná s užitím dvou barevných palet. První možností jsou odstíny šedé

Obrázek 2: Barevné palety pro vizualizaci sklonu terénu. Vlevo odstíny šedé a vpravo žluto – červená stupnice.

## 3.3 Výpočet expozice

Výpočet expozice probíhá obdobným způsobem jako výpočet sklonu. V podstatě jde o úhel mezi x – ovou a y – ovou složkou normály trojúhelníka. Expozice určuje náklon trojúhelníka k jedné ze světových stran. V aplikaci je uvažováno s 8 možnostmi (jih, jihozápad, západ, severozápad, sever, severovýchod, východ a jihovýchod).

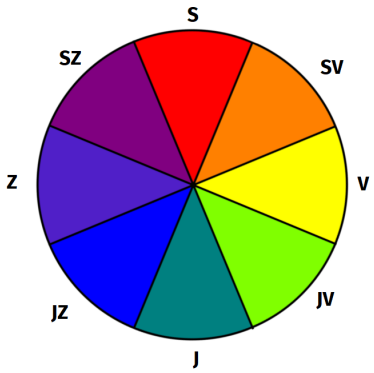
1. Určení vektorů $\overrightarrow{u} = (u_x, u_y, u_z) = (x_1 - x_2, y_1 - y_2, z_1 - z_2)$, $\overrightarrow{v} = (v_x, v_y, v_z) = (x_3 - x_2, y_3 - y_2, z_3 - z_2)$

2. Výpočet normálového vektoru trojúhelníka: $\overrightarrow{n} = (n_y, n_y, n_z) = (u_y \cdot v_z - v_y \cdot u_z, -u_x \cdot v_z + v_x \cdot u_z, u_x \cdot v_y - v_x \cdot u_y)$

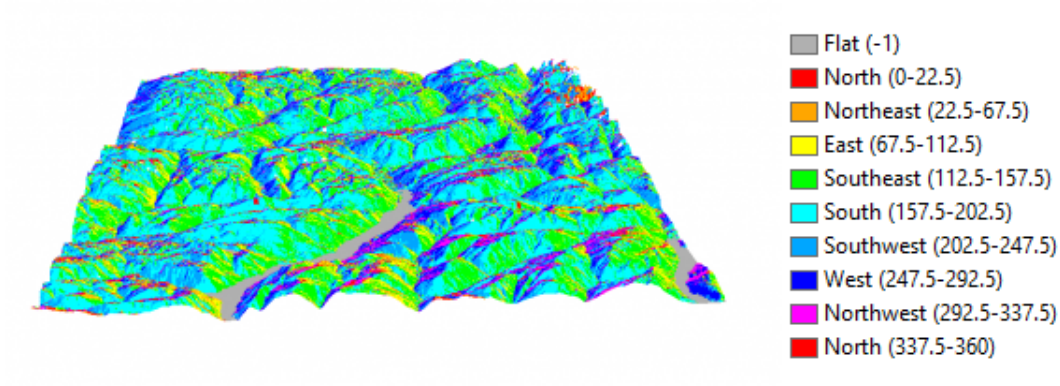
3. Určení expozice $exp = arccos \dfrac{n_z}{|n|}$

### 3.3.1 Výběr barevné palety

Pro expozici jsou na výběr dvě možnosti barevných palet. Námi zvolená KM paleta a standardní paleta, kterou využívá např. ArcGIS.




Obrázek 3: KM barevná paleta pro vizualizaci expozice terénu.




Obrázek 4: Vizualizace expozice terénu dle ArcGIS [1].

## 3.4 Výpočet vrstevnic

Výpočet vrstevnic probíhá na základě vstupní Delaunayho triangulace, minimální a maximální výšky a rozestupu mezi vrstevnicemi. Jedná se o určení vrstevnic lineární interpolací, předpoklad je tedy takový, že spád terénu je mezi podrobnými body konstantní. Jedná se o nejjednodušší a rozšířený způsob tvorby vrstevnic.

Nejprve je nutné zjistit, zda rovina $\rho$ protíná stranu $(p_i, p_{i+1})$ trojúhelníka:

1. Jestliže $(z - z_i)(z - z_{i+1}) < 0$, pak trojúhelník protíná rovinu $\rho$ v obecných bodech.
2. Jestliže $(z - z_i)(z - z_{i+1}) > 0$, pak trojúhelník neprotíná rovinu $\rho$.
3. Jestliže $(z - z_i)(z - z_{i+1}) = 0$, pak jedna ze stran trojúhelníka leží v rovině $\rho$.

Výpočet souřadnic průsečíku A,B vrstevnice o nadmořské výšce $z$ s trojúhelníkem tvořeným body 1,2,3:

$$x_A = \frac{x_3 - x_1}{z_3 - z_1}(z - z_1) + x_1$$
$$y_A = \frac{y_3 - y_1}{z_3 - z_1}(z - z_1) + y_1$$
$$x_B = \frac{x_2 - x_1}{z_2 - z_1}(z - z_1) + x_1$$
$$y_B = \frac{y_2 - y_1}{z_2 - z_1}(z - z_1) + y_1$$

### 3.4.1 Popis vrstevnice

Automatický popis vrstevnic byl implementován pro každou hlavní vrstevnici (tedy každou pátou vrstevnici). Výškový údaj je vypsán nad vrstevnicí a umístěn v polovině triangulačního trojúhelníka.

Ve třídě Draw je implementace následovná:

```
//Draw main contours label
 for (Edge mcl : main_contours_label)
 {
    //Start and end point of the contour edge
    QPoint3D sl_point = mcl.getStart();
    QPoint3D el_point = mcl.getEnd();

    //Get label coordintes
    QPoint3D label_point( (sl_point.x()+el_point.x())/2,
                          (sl_point.y()+el_point.y())/2);

    //Draw Z coordinate
    QString z = QString::number(sl_point.getZ());
    qp.drawText(label_point, z);
 }
```

# 4 Problematické situace a jejich rozbor

## 4.1 Automatické nastavení přechodné barevné stupnice mezi žlutou a červenou

Pro zobrazení sklonu terénu byl zvolen přechod žluté do červené jako optimální varianta, nicméně vyladění správně plynule vykreslovaných barev bylo poměrně náročné k představení. Naštěstí existují různé vizualizační online aplikace, které umí velice názorně pomoci – například [2].

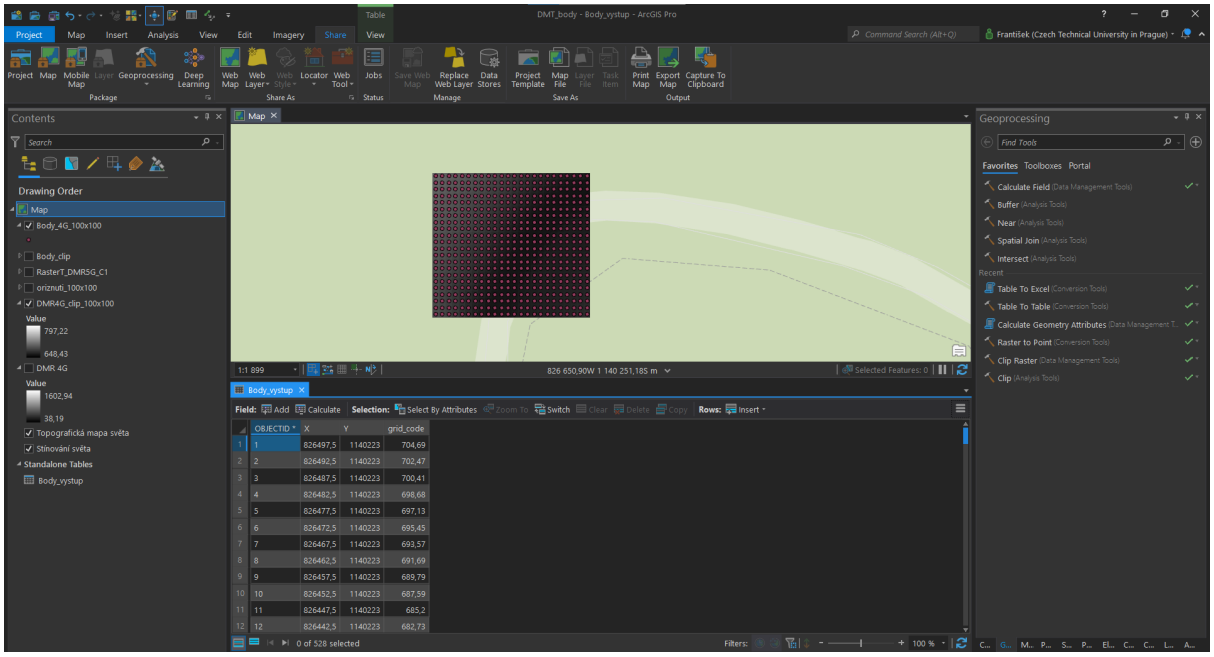
```
void Draw::paintEvent(QPaintEvent *event)
{
...
        if (yelred == TRUE)
        {
        //Convert to color
        int col = 255 - k * slope;
```

```
        int colg = k * slope;
        QColor color(255, colg, col);

        //Set pen and brush
        qp.setBrush(color);
        qp.setPen(color);
        }
...
}
```

# 5 Vstupní data

Vstupem je textový soubor s prostorovými souřadnicemi X,Y,Z geodetických bodů. Tyto body jsou považovány za vstupní množinu P. Body byly získány exportem z Digitálního modelu reliéfu 4. generace [3] v prosředí ArcGIS Pro [1] (viz. obr. 5).



Obrázek 5: Export bodů v ArcGIS Pro z DMR4G na vybraném území.

Formát vstupních dat je následující (ID, X, Y, Z):

```
1 846308.0 1129497.0 1141.2
2 846288.0 1129497.0 1131.8
3 846268.0 1129497.0 1123.2
...
```

# 6 Výstupní data

Za výstup je považována grafický výstup vytvořené aplikace. Grafickým výstupem je polyedrický DMT nad množinou P představovaný vrstevnicemi doplněný vizualizací sklonu trojúhelníků, jejich expozicí a vrstevnicemi s popisem.

# 7 Snímky obrazovky vytvořené aplikace a její popis

# 8 Dokumentace

## 8.1 Třída Algorithms

Tato třída obsahuje výpočetní vzorce pro použité algoritmy.
**Třída algorithms obsahuje následující veřejné metody:**

```
double get2LinesAngle(QPoint &p1, QPoint &p2, QPoint &p3, QPoint &p4)
```

Metoda vypočte velikost úhlu, který svírají dvě přímky. První přímku tvoří body $p_1, p_2$ a druhou přímku body $p_3, p_4$.

```
int getPointLinePosition(QPoint &a,QPoint &p1,QPoint &p2)
```

Metoda určuje, zda – li bod leží v levé či v pravé polorovině od přímky (hrany polygonu). Vstupními parametry jsou určovaný bod $a$ a body $p_1, p_2$, které tvoří přímku.

```
double getPointLineDistance(QPoint &a, QPoint &p1, QPoint &p2)
```

Metoda určuje vzdálenost bodu $a$ od přímky tvořenou body $p_1, p_2$.

```
QPolygon cHullJarvisScan(std::vector <QPoint> &points)
```

Výpočet konvexní obálky načtených bodů na základě algoritmu Jarvis Scan (kapitola **??**).

```
std::vector<QPoint> rotate(std::vector<QPoint> &points, double sigma)
```

Otočení načtených bodů o zadaný úhel *sigma*.

```
std::tuple<std::vector<QPoint>, double> minMaxBox(std::vector<QPoint> &points)
```

Proměnná vracející min – max box, tedy vrcholy obdélníka a jeho výměru. Vstupem jsou načtené body.

```
QPolygon minAreaEnclosingRectangle(std::vector<QPoint> &points)
```

Určení hlavního směru polygonu s využitím algoritmu Minimum Area Eclosing Rectangle (kapitola **??**). Na vstupu jsou body polygonu. Výstupem je generalizovaný a pootočený polygon.

```
QPolygon wallAverage(std::vector<QPoint> &points)
```

Určení hlavního směru polygonu s využitím algoritmu Wall Average (kapitola **??**). Na vstupu jsou body polygonu. Výstupem je generalizovaný a pootočený polygon.

```
double LH(std::vector<QPoint> &points)
```

Výpočet plochy obecného mnohoúhelníků pomocí LH vzorce.

```
std::vector<QPoint> resizeRectangle(std::vector<QPoint> &points, std::vector<QPoint
```

Přepočet velikosti ohraničujícího obdélníku na základě výměry původního polygonu.

```
QPolygon longestEdge(std::vector<QPoint> &points)
```

Určení hlavního směru polygonu s využitím algoritmu Longest Edge (kapitola **??**). Na vstupu jsou body polygonu. Výstupem je generalizovaný a pootočený polygon.

```
QPolygon weightedBisector(std::vector<QPoint> &points)
```

Určení hlavního směru polygonu s využitím algoritmu Weighted Bisector (kapitola **??**). Na vstupu jsou body polygonu. Výstupem je generalizovaný a pootočený polygon.

```
QPolygon cHullQuickHull(std::vector <QPoint> &points)
```

Výpočet konvexní obálky načtených bodů na základě algoritmu Quick Hull (kapitola **??**).

```
void quickHullLocal(int ps, int pe, std::vector<QPoint> &points, QPolygon &ch)
```

Metoda pro pomocný výpočet rekurentního určení konvexní obálky bodů algoritmem Quick Hull, lokální procedura (kapitola **??**). Vstupními parametry jsou indexy počátečního a koncového bodu *ps* a *pe* vstupní hrany, načtené body a konvexní obálka.

## 8.2 Třída Draw

Tato třída umožňuje vykreslování bodu a polygonů.
**Třída draw obsahuje následující privátní metody a proměnné:**

```
std::vector<QPoint> points
```

Proměnná se souřadnicemi načtených bodů.

```
QPolygon ch, er;
```

Polygony pro ukládání konvexní obálky a ohraničujícího obdélníku.

`std::vector<QPolygon> polygons, chs, ers`
Proměnná uchovávající pomocné polygony v průběhu výpočtu a pro vykreslení.

**Třída draw obsahuje následující veřejné metody a proměnné:**
`explicit Draw(QWidget *parent = nullptr)`
Prvotní vykreslení bodu mimo okno aplikace.

`void paintEvent(QPaintEvent *event)`
Metoda, která vykresluje bod či polygony.

`void mousePressEvent(QMouseEvent *event)`
Metoda určující souřadnice určeného bodu.

`void clearAddedData()`
Metoda mazající přidaná data z obrazovky.

`void clearDrawing()`
Metoda mazající nakreslený polygon z obrazovky.

`std::vector<QPoint> getPoints() {return points;}`
Vrací souřadnice lomových bodů polygonu.

`std::vector<QPolygon> getPolygons(){return polygons;}`
Vrací souřadnice polygonů.

`void setCh(QPolygon &ch_) {chs.push_back(ch_);}`
Nastavení polygonu konvexní obálky.

`void setEr(QPolygon &er_) {ers.push_back(er_);}`
Nastavení polygonu ohraničujícího obdélníku.

`void clearChs(){chs.clear();}`
Smazání polygonu konvexní obálky.

`void clearErs(){ers.clear();}`
Smazání polygonu ohraničujícího obdélníku.

`void drawPolygons(std::vector<QPolygon> &data);`
Vykreslení polygonů načtených z textového souboru.

## 8.3 Třída Load

**Třída draw obsahuje následující veřejnou proměnnou:**

`static std::vector<QPolygon> load_file(std::string &filename)`

Umožňuje načítání polygonů z textového souboru.

## 8.4 Třída SortByX

**Třída draw obsahuje následující veřejnou proměnnou:**

`bool operator() (QPoint &p1, QPoint &p2)`

Ražení bodů dle jejich x – ové souřadnice.

## 8.5 Třída SortByY

**Třída draw obsahuje následující veřejnou proměnnou:**

`bool operator() (QPoint &p1, QPoint &p2)`

Ražení bodů dle jejich y – ové souřadnice.

## 8.6 Třída Widget

Tato třída propojuje uživatelské rozhraní aplikace s kódem. Je vytvořena v sekci *Design*.

**Třída widget obsahuje následující privátní metody a proměnné:**

`void on_pushButtonClear_clicked()`

Vymazání kresby. Propojení s tlačítkem *Clear drawing*.

`void on_pushButtonLoad_clicked()`

Načtení lomových bodů polygonů z textového souboru. Propojení s tlačítkem *Load file with polygon*.

`void on_pushButton_clicked()`

Spuštění funkce Building Simplify s vybraným algoritmem. Propojení s tlačítkem *Building Simplify*.

`void on_pushButtonClearData_clicked()`

Vymazání přidaných dat. Propojení s tlačítkem *Clear added data*.

`void processPoints(std::vector<QPoint> &points)`

Volba použitého algoritmu na základě výběru z combo boxů.

# 9 Závěr

## 9.1 Možné či neřešené problémy

## 9.2 Náměty na vylepšení

V Praze 30.11.2021

Bc. Pane Kuzmanov
Bc. František Mužík

# Použitá literatura


[1] ArcGis Pro. Základ systému ArcGIS. https://www.arcdata.cz/produkty/arcgis/desktopovy-gis/arcgis-pro [cit. 2021-12-03].

[2] Computer Science Field Guide. Computer Science Education Research Group at the University of Canterbury, New Zealand. https://www.csfieldguide.org.nz/en/interactives/rgb-mixer/ [cit. 2021-12-04].

[3] Český úřad zeměměřický a katastrální. Základní báze geografických dat České republiky (ZABAGED). https://www.cuzk.cz/ [cit. 2021-12-03].

[4] Západočeská univerzita v Plzni. Katedra geomatiky, Fakulta aplikovaných věd. https://kgm.zcu.cz/ [cit. 2021-12-03].