

# Algoritmy v digitální kartografii

Digitální model terénu a jeho analýzy

Zimní semestr 2018/2019

Tereza Kulovaná  
Markéta Pecenová

# Obsah

<b>1</b>	<b>Zadání</b>	<b>2</b>
<b>2</b>	<b>Popis a rozbor problému</b>	<b>3</b>
<b>3</b>	<b>Algoritmy</b>	<b>3</b>
3.1	Delanuayova triangulace . . . . .	3
3.2	Vrstevnice . . . . .	4
3.3	Sklon . . . . .	5
3.4	Orientace . . . . .	5
<b>4</b>	<b>Vstupní data</b>	<b>5</b>
<b>5</b>	<b>Výstupní data</b>	<b>6</b>
<b>6</b>	<b>Aplikace</b>	<b>7</b>
<b>7</b>	<b>Zhodnocení činnosti algoritmu</b>	<b>10</b>
7.1	Celý model . . . . .	10
7.2	Plošina . . . . .	11
7.3	Hrana . . . . .	12
7.4	Údolnice . . . . .	13
<b>8</b>	<b>Dokumentace</b>	<b>14</b>
8.1	Algorithms . . . . .	14
8.2	Draw . . . . .	17
8.3	Edge . . . . .	18
8.4	QPoint3D . . . . .	18
8.5	SortByXAsc . . . . .	19
8.6	Triangle . . . . .	19
8.7	Widget . . . . .	19
<b>9</b>	<b>Závěr</b>	<b>21</b>
<b>10</b>	<b>Zdroje</b>	<b>22</b>

# 1 Zadání

Zadání úlohy bylo staženo ze stránek předmětu 155ADKG.

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

*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. navrhnete 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.

Zhodnot'te 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 tři 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
<b>Max celkem:</b>	<b>65b</b>

Čas zpracování: 3 týdny

V rámci této úlohy nebyly implementovány žádné bonusové úlohy.

## 2 Popis a rozbor problému

Úloha **Digitální model terénu a jeho analýzy** se zabývá vytvořením aplikace, která Delaunayovou triangulací nad vstupní množinou bodů  $P$  vytvoří trojúhelníkovou síť, pro kterou se lineární interpolací vypočítají vrstevnice. Aplikace dále počítá a vhodně vizualizuje sklon a orientaci trojúhelníků ke světovým stranám.

Způsobů, jak geometricky zkonstruovat trojúhelníkovou síť, je více. Pro účely této úlohy byla vybrána Delaunayova triangulace, protože poskytuje optimální trojúhelníky z hlediska tvaru, což je zejména v kartografii velmi důležité. Delaunayova triangulace má čtyři základní vlastnosti:

1. Uvnitř kružnice opsané trojúhelníku  $t_i \in DT$  neleží žádný jiný bod množiny  $P$ .
2.  $DT$  maximalizuje minimální úhel v  $\forall t_i$ , avšak  $DT$  neminimalizuje maximální úhel v  $t_i$ .
3.  $DT$  je lokálně optimální i globálně optimální vůči kritériu minimálního úhlu.
4.  $DT$  je jednoznačná, pokud žádné čtyři body neleží na kružnici.<sup>1</sup>

## 3 Algoritmy

Tato kapitola se zabývá popisem algoritmů, které byly v aplikaci implementovány.

### 3.1 Delaunayova triangulace

Delaunayova triangulace byla realizována metodou inkrementální konstrukce, která je založena na postupném přidávání bodů do již vytvořené triangulace. Během výpočtu je používána struktura *AEL* (Active Edge List), která obsahuje všechny hrany, proto které ještě nebyl nalezen třetí bod trojúhelníku. Hrana, pro kterou byl bod nalezen, je vzápětí ze seznamu odstraněna. Před přidáním hran do seznamu je kontrolováno, zda se v něm již nenachází hrana s opačnou orientací. V takovém případě není hrana do seznamu přidána.

Mějme množinu bodů  $P$  a orientovanou hranu  $e_i$ . Hledáme takový bod  $p_i \in P$ , který se nachází v levé polorovině vymezené hranou  $e_i$ , pro který dále platí, že poloměr kružnice jemu a hraně opsané je minimální. Během výpočtu jsou upřednostňovány body, jejichž středy opsaných kružnic se nachází v pravé polorovině. Je-li bod splňující výše uvedená kritéria nalezen, vytvoří se dvě nové orientované hrany  $e_{i+1}$  a  $e_{i+2}$ , které se přidají do triangulace a do *AEL*. Původní hrana  $e_i$  je z *AEL* odstraněna. Není-li žádný vhodný bod nalezen, dochází k prohození orientace hrany  $e_i$  a postup je opakován. Celý proces je ukončen ve chvíli, kde se v *AEL* nenachází již žádná hrana.

Zjednodušený zápis algoritmu lze zapsat způsobem uvedeným níže:

---

<sup>1</sup>Zdroj: <https://web.natur.cuni.cz>, slide 22

1. Nalezení pivota  $a$ :  $a = \min(x)$  a jemu nejbližší bod  $b$
2. Vytvoření  $e_1 = (a, b)$
3. Nalezení Delaunayova bodu:  $r(k_i) = \min, k_i = (e_1, p_i)$
4. Podmínka:  $p_i$  nenalezen  $\rightarrow e_1 = (b, a)$ , opakuj krok 3
5. Vytvoř zbylé hrany trojúhelníku:  $e_2 = (b, p_i), e_3 = (p_i, a)$
6. Přidej hrany do  $AEL$ :  $AEL \leftarrow e_1, AEL \leftarrow e_2, AEL \leftarrow e_3$
7. Přidej hrany do triangulace  $DT$ :  $DT \leftarrow e_1, DT \leftarrow e_2, DT \leftarrow e_3$
8. Dokud  $AEL \neq \emptyset$ :
  - Vezmi první hranu z  $AEL \rightarrow e_1$
  - Prohod' orientaci:  $e_1 = (b, a)$
  - Nalezení Delaunayova bodu:  $r(k_i) = \min, k_i = (e_1, p_i)$
  - Podmínka:  $p_i$  nalezen
  - Vytvoř zbylé hrany trojúhelníku:  $e_2 = (b, p_i), e_3 = (p_i, a)$
  - Přidej hranu do  $DT$ :  $DT \leftarrow e_1$
  - $add(e_2, AEL, DT), add(e_3, AEL, DT)$

Lokální algoritmus  $add$ :

1. Prohod' orientaci:  $e' = (b, a)$
2. Podmínka:  $e' \in AEL \rightarrow$  odstraň  $e'$  z  $AEL$
3. Jinak:  $AEL \leftarrow e$
4.  $DT \leftarrow e$

### 3.2 Vrstevnice

Druhý algoritmus použitý v aplikaci slouží k výpočtu vrstevnic. Vrstevnice byly konstruovány metodou lineární interpolace, která je založena na předpokladu, že spád terénu mezi dvěma body  $p_i$  se mění stejně, tedy konstantně. Výpočet byl proveden postupně pro všechny trojúhelníky a vrstevnice byly ukládány jako seznam hran.

Mějme trojúhelník  $t_i$  tvořený třemi hranami  $e_1(p_1, p_2)$ ,  $e_2(p_2, p_3)$  a  $e_3(p_3, p_1)$  a rovinu  $\rho$  o výšce  $Z$ . Hledáme průsečnici roviny trojúhelníku  $t_i$  s rovinou  $\rho$ . Pro kritérium  $t = (z - z_i)(z - z_{i+1})$  mohou nastat tři základní situace:

1.  $t > 0 \rightarrow e_i \notin \rho$
2.  $t = 0 \rightarrow e_i \in \rho$

3.  $t < 0 \rightarrow e_i \cap \rho$

Pro případy 1 a 2 nebyly vrstevnice řešeny. Nastane-li případ 3 ( $e_i \cap \rho$ ), je pro hranu  $e_i$  a rovinu  $\rho$  níže uvedenými vzorci vypočten průsečík  $a$  o výšce  $z_a$ : (pro přehlednost uvedeno pro hranu  $e_1$ )

$$x_a = \frac{(x_2 - x_1)}{(z_2 - z_1)}(z_a - z_1) + x_1$$
$$y_a = \frac{(y_2 - y_1)}{(z_2 - z_1)}(z_a - z_1) + y_1$$

### 3.3 Sklon

Algoritmus pro výpočet sklonu počítá sklon jednotlivých trojúhelníků  $t_i$ . Sklon je úhel  $\varphi$  mezi svislicí  $n$  a normálou trojúhelníku  $n_t$ . Rovina trojúhelníku  $t_i$  je určena vektory  $u, v$ . Sklon nabývá hodnot  $\langle 0^\circ; 90^\circ \rangle$  a v aplikaci je zobrazen v odstínech šedi.

$$n = (0, 0, 1)$$
$$n_t = \vec{u} \times \vec{v}$$
$$\varphi = \arccos \left( \frac{n_t \cdot n}{|n_t||n|} \right)$$

### 3.4 Orientace

Orientace terénu  $A$  je definována jako azimut průmětu gradientu normálového vektoru roviny trojúhelníku do roviny  $x, y$ . Nabývá hodnot  $\langle 0^\circ; 360^\circ \rangle$  a v aplikaci je zobrazen barevnou škálou.

$$n_t = \vec{u} \times \vec{v}$$
$$A = \arctan 2 \left( \frac{n_x}{n_y} \right)$$

## 4 Vstupní data

Pro účely této úlohy byla použita data, která byla naměřena v rámci geodetické výuky v terénu v Mariánské u Jáchymova. Souřadnice X a Y byly pro tuto úlohu zredukovány na rozumnou velikost, souřadnice Z byla zachována. Body byly zaměřeny metodou GNSS a totální stanicí a znázorňují tamní louku a část silnice. Seznam vstupních bodů je uložen v textovém souboru `testing_data.txt`. Soubor je nutné do aplikace nahrát pomocí tlačítka *Load points*. Struktura textového souboru je následující:

*Sloupec 1*: souřadnice X [m]

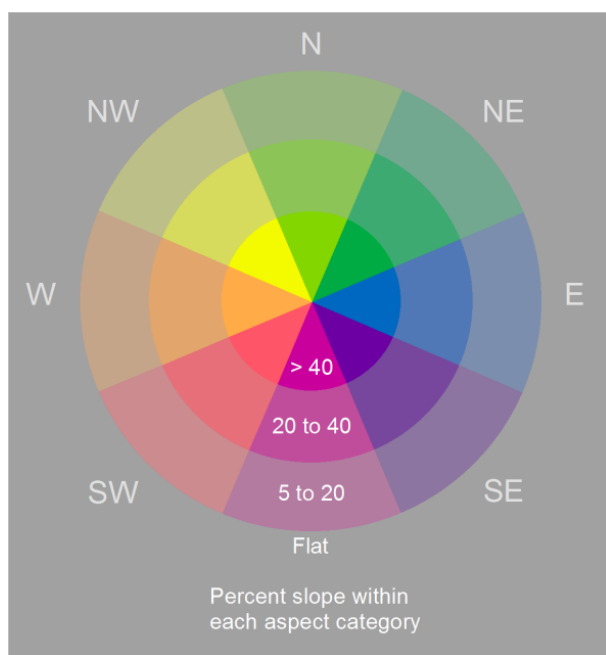
*Sloupec 2*: souřadnice Y [m]

*Sloupec 3*: souřadnice Z [m]

Po úspěšném/neúspěšném nahrání souboru je uživatel upozorněn hláškou. Uživatel nemůže kliknout na žádné jiné tlačítko pro výpočty, nejsou-li nahrána data (tlačítka jsou zašedivělá). Aplikace dále nedovoluje spustit výpočty, jejichž fungování je závislé na vygenerované trojúhelníkové síti, nebyla-li předtím vytvořena. Uživatel má dále možnost zvolit krok, v jakém se budou vykreslovat vrstevnice. Hodnoty lze měnit šipkami nahoru/dolů po 5 m nebo ručně vepsat hodnotu celého čísla v rozmezí 1 m až 100 m. Delaunayova triangulace, vrstevnice, sklon a orientace se generují stisknutím příslušných tlačítek.

## 5 Výstupní data

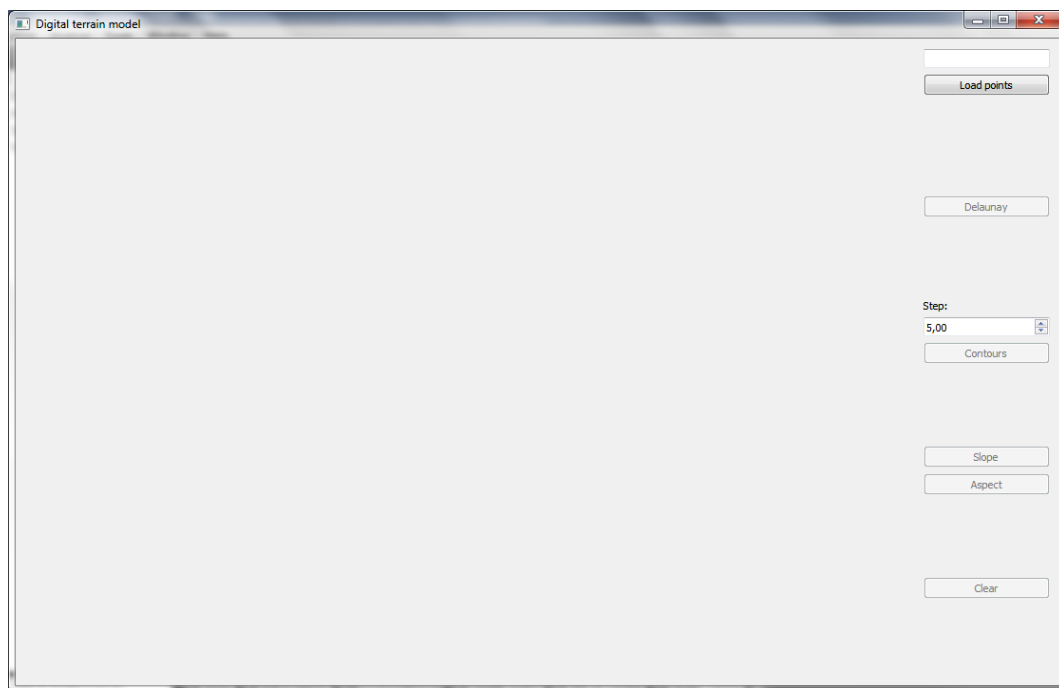
Vstupní množina bodů a nad ní vygenerovaná trojúhelníková síť je zobrazena ve grafickém okně aplikace. Vykreslování vrstevnic, sklonu a orientace je odděleno. U vrstevnic je každá pátá (hlavní) zvýrazněna. Sklon je v odstínech šedi (čím vyšší sklon, tím tmavší barva). Pro zobrazení orientace trojúhelníků ke světovým stranám byla využita prostřední kružnice barvené škály ze stránek společnosti *ESRI*, viz níže. Aplikace je uvedena do výchozího stavu stisknutím tlačítka *Clear*.



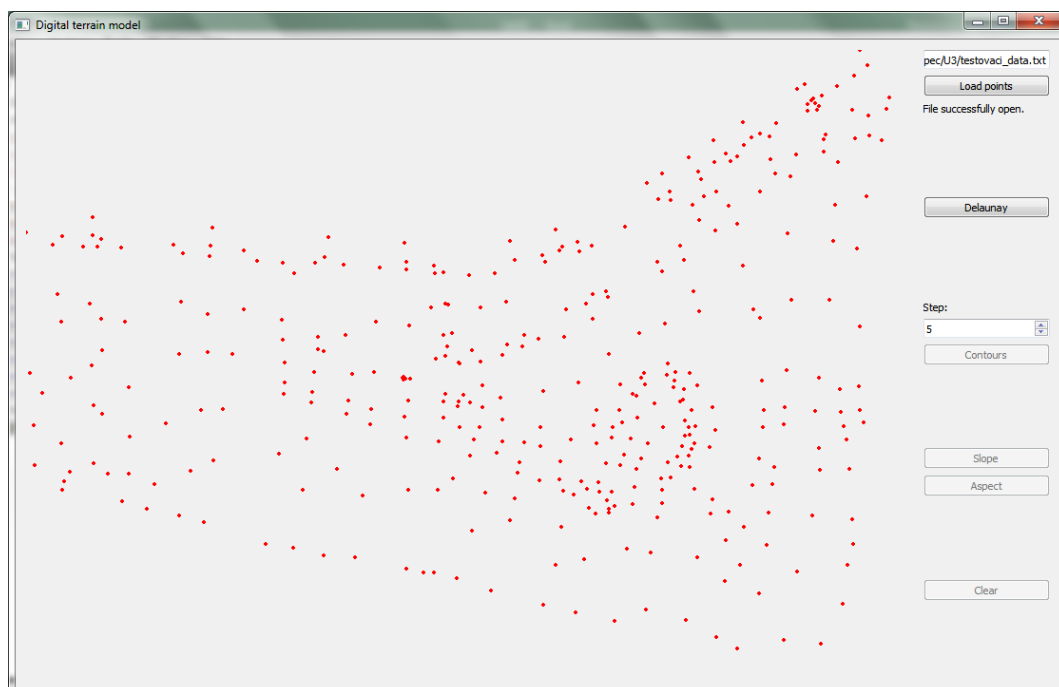
Obrázek 1: Barevná škála orientace trojúhelníků ke světovým stranám [Zdroj]

## 6 Aplikace

V následující kapitole je představen vizuální vzhled vytvořené aplikace tak, jak ji vidí prostý uživatel.

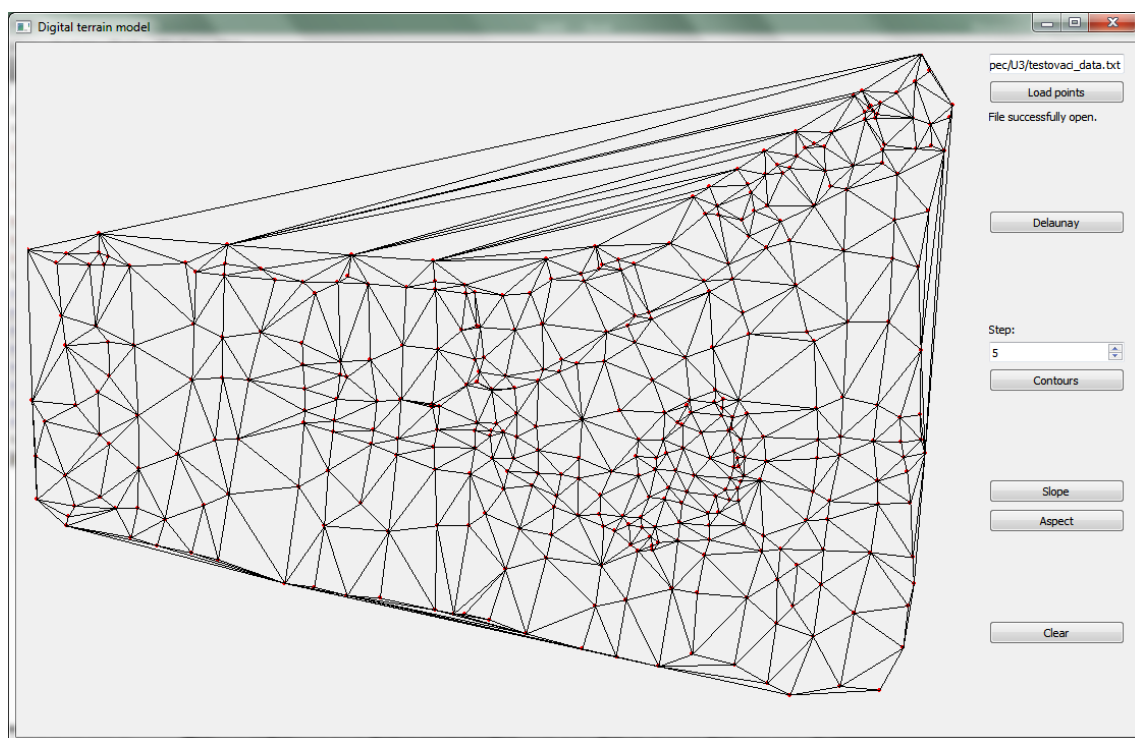


Obrázek 2: Výchozí vzhled aplikace po spuštění

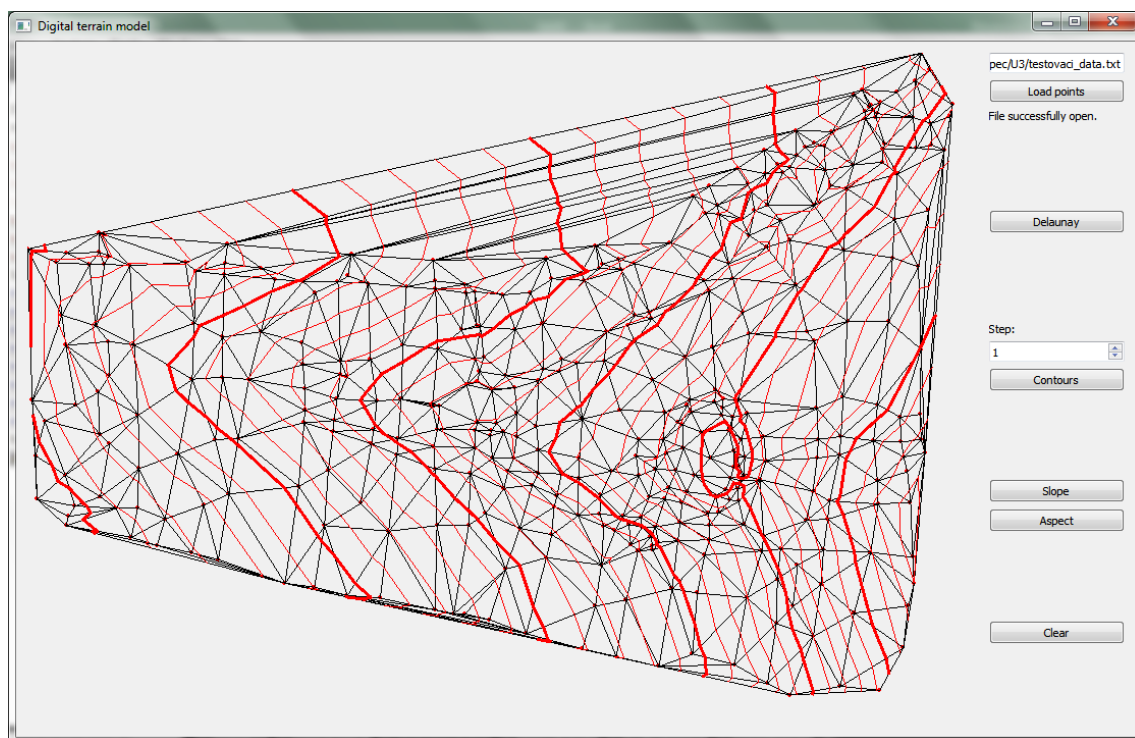


Obrázek 3: Aplikace po nahrání vstupních dat

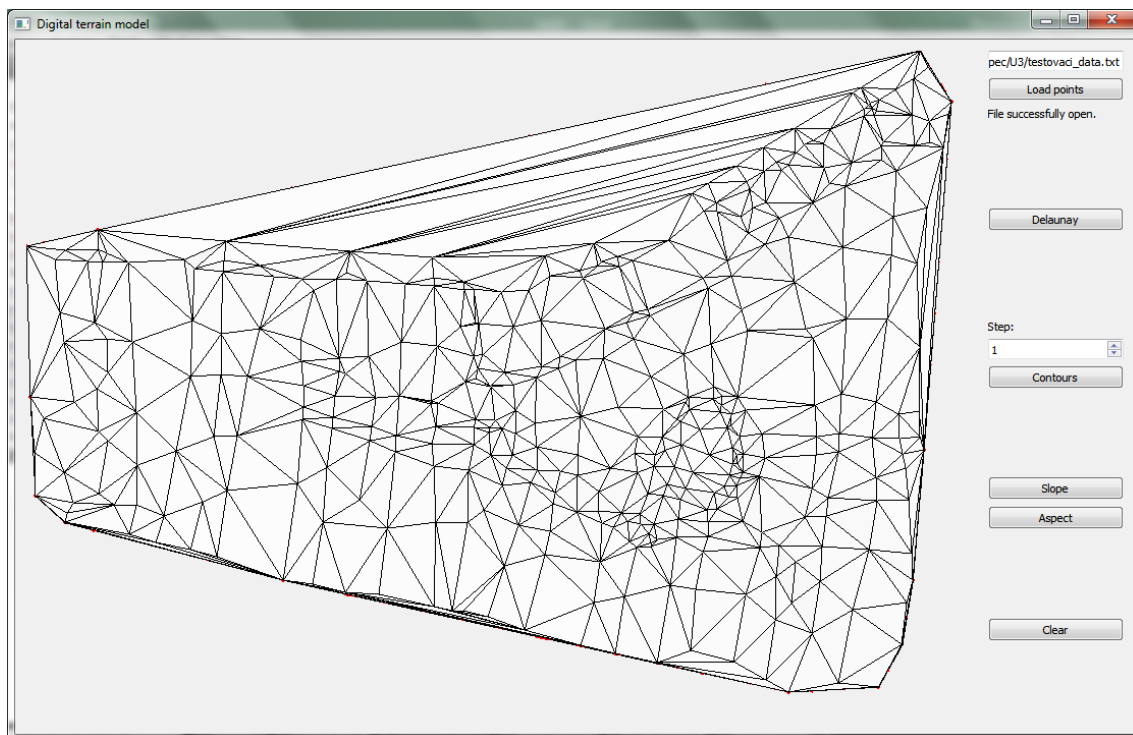




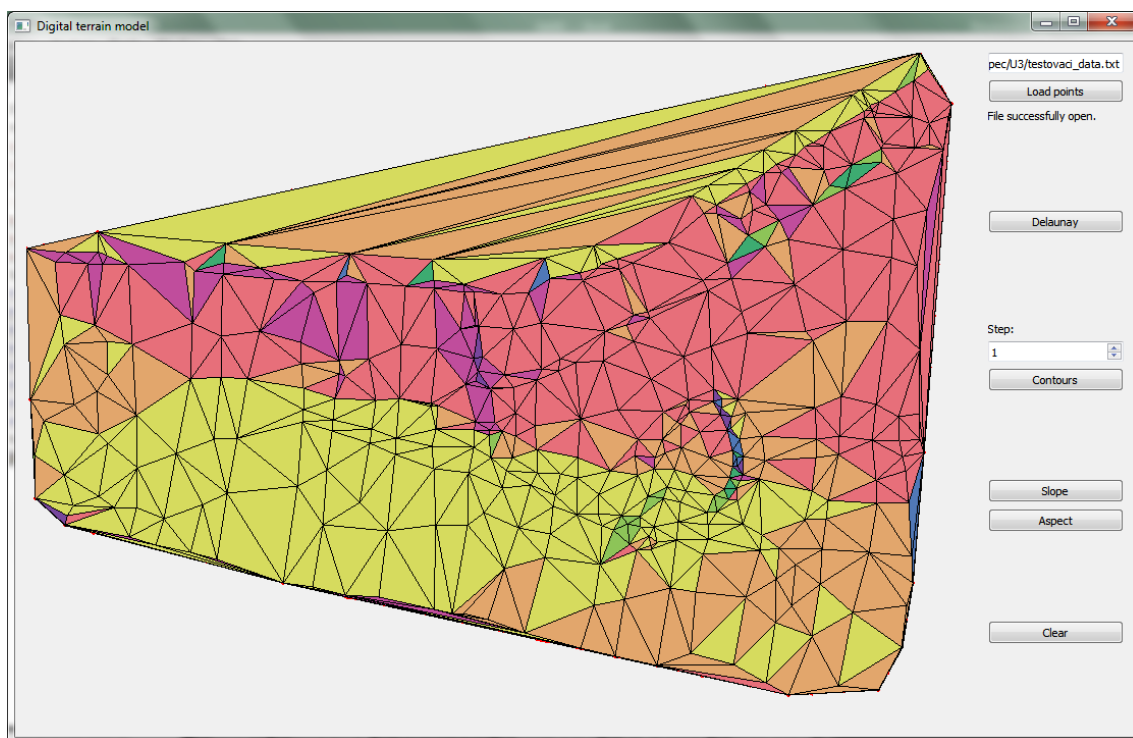
Obrázek 4: Trojúhelníková síť



Obrázek 5: Vykreslení vrstevnic



Obrázek 6: Sklon trojúhelníků

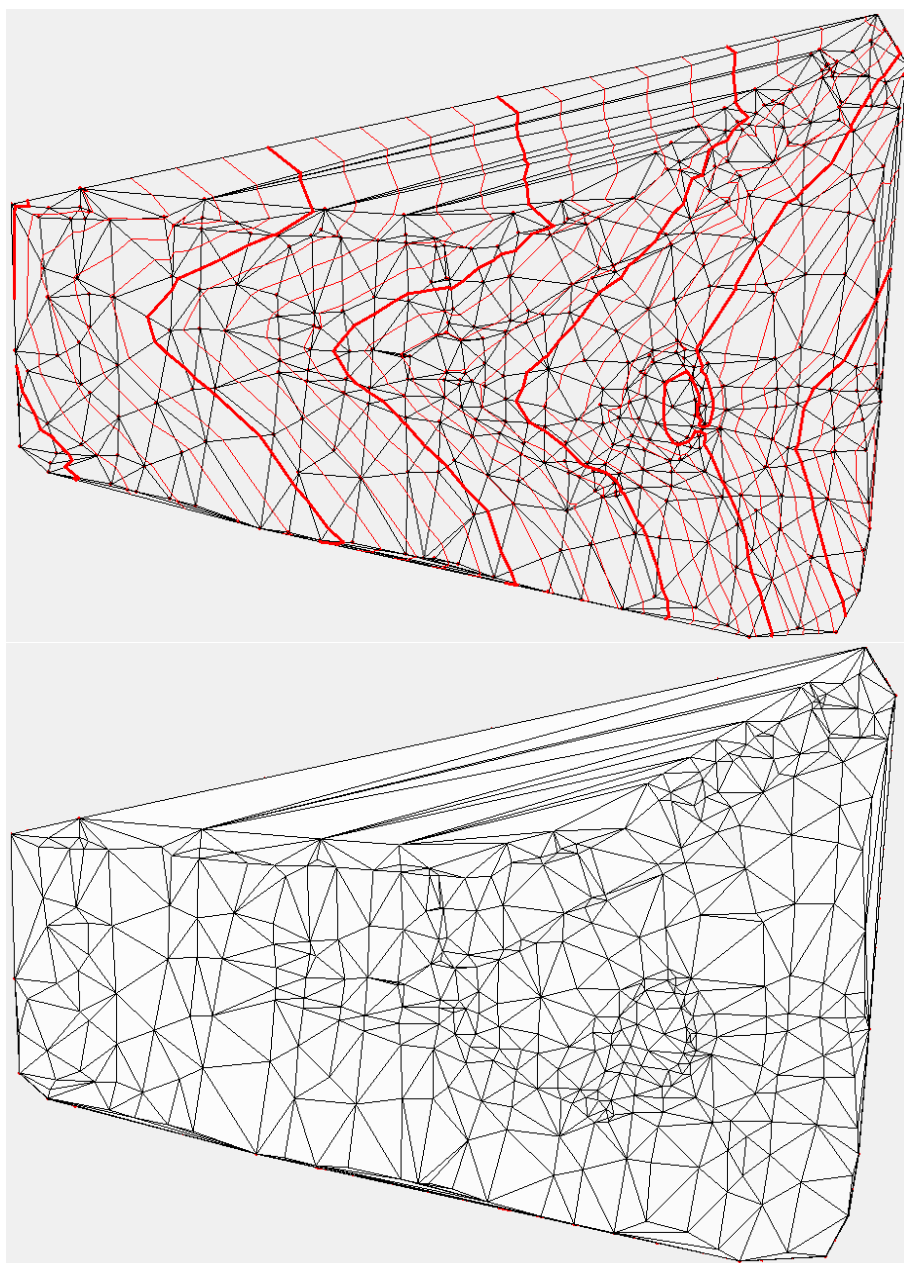


Obrázek 7: Orientace trojúhelníků

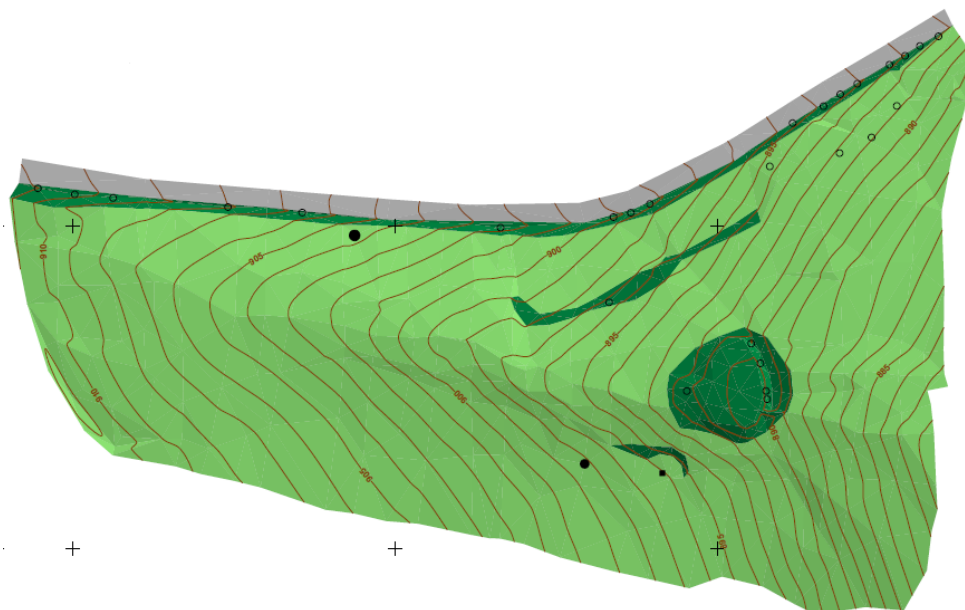
## 7 Zhodnocení činnosti algoritmu

V následující kapitole je porovnán DMT z aplikace a DMT, který byl již dříve vyhotoven v SW *Atlas*. Aplikace nezvládá zobrazovat současně vykreslené vrstevnice a sklon trojúhelníků. Ukázka výstupů je rozdělena do dvou snímků.

### 7.1 Celý model



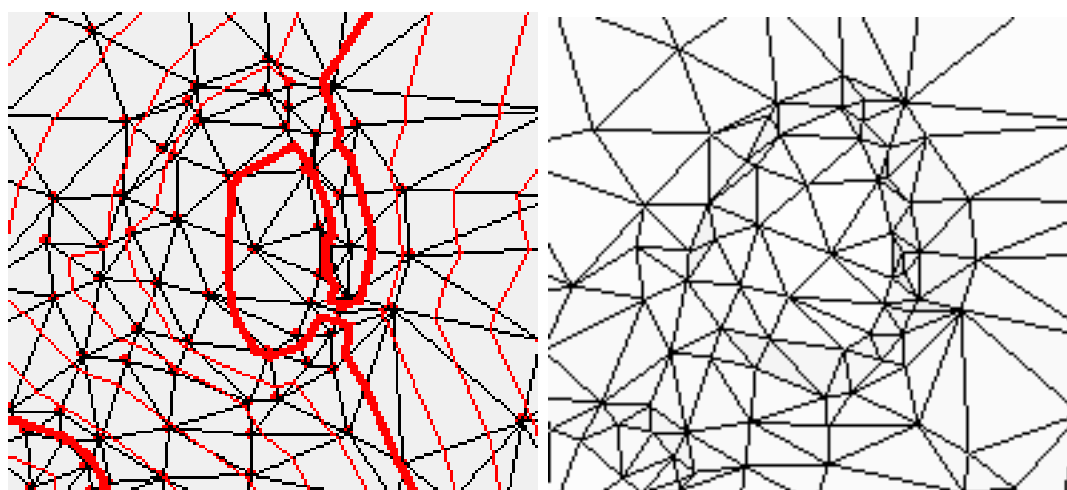
Obrázek 8: DMT vytvořený aplikací



Obrázek 9: DMT ze SW *Atlas*

Na první pohled je patrné, že DMT ze SW *Atlas* je kvalitněji vyhotoven. Problémem Delaunayovy triangulace implementované v aplikaci je zejména chybějící možnost nadefinovat lomové a povinné hrany pro triangulaci, což způsobuje, že se v horní části modelu podél silnice vygenerovaly štíhlé trojúhelníky, které by správně vůbec neměly existovat, jelikož se nachází mimo zájmovou oblast. Vrstevnice z *Atlasu* jsou vyhlazenější, neboť to bylo během generování modelu zajištěno funkcí pro vyhlazení vrstevnic, ale vytvořená aplikace toto neumožňuje. Jelikož sklon terénu byl víceméně rovnoměrný, sklon generovaný v aplikaci není příliš viditelný.

## 7.2 Plošina



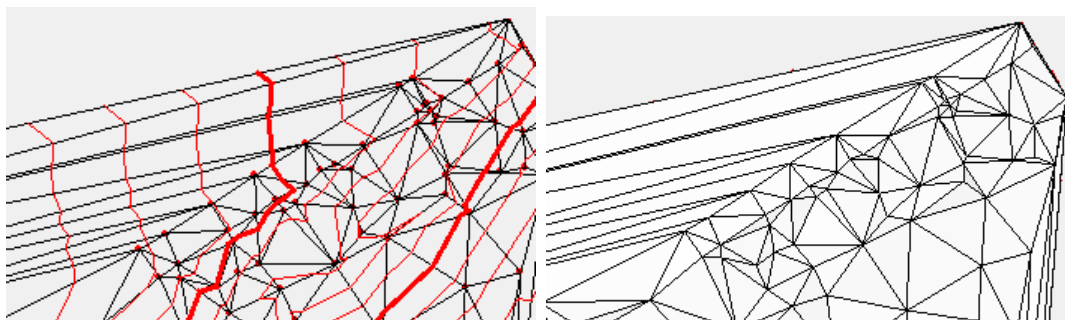
Obrázek 10: Plošina vytvořená aplikací



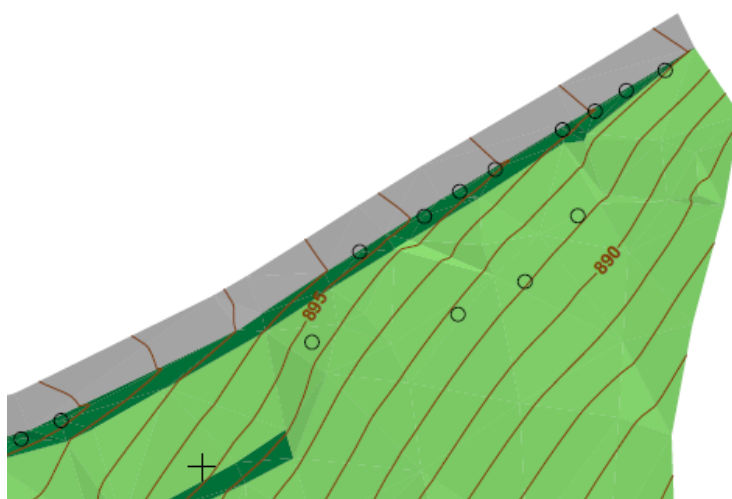
Obrázek 11: Plošina ze SW *Atlas*

Plošina uprostřed území je rozeznatelná zejména z tvarů vrstevnic. Z trojúhelníkové sítě je patrné, že je trochu vyvýšená, ale i tak to chce trochu představivosti.

### 7.3 Hrana



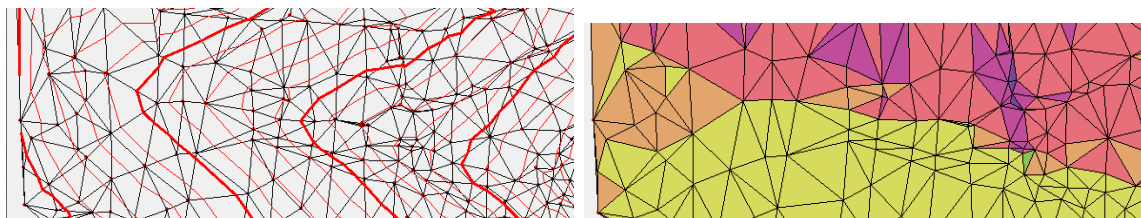
Obrázek 12: Hrana silnice vytvořená aplikací



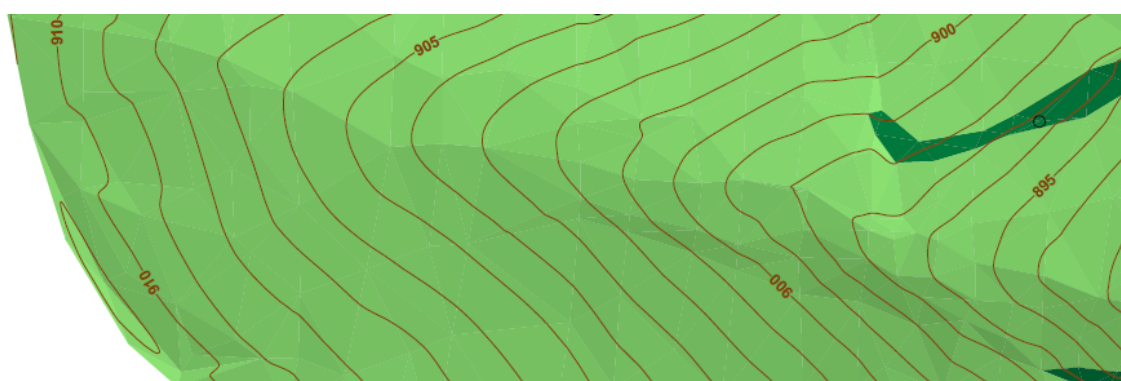
Obrázek 13: Hrana silnice ze SW *Atlas*

Vygenerované vrstevnice v okolí silnice působí silně kóstrbatým dojmem, avšak trojúhelníková síť poměrně věrně kopíruje tvar silnice. Jak již bylo zmíněno výše, velkým problémem jsou štíhlé trojúhelníky v horní části modelu, které by neměly existovat.

## 7.4 Údolnice



Obrázek 14: Údolnice vytvořená aplikací



Obrázek 15: Údolnice ze SW *Atlas*

Pro názornost porovnání byla tentokrát použita orientace trojúhelníků ke světovým stranám. Z jejich natočení je patrné, že ve středu dochází ke zlomu v terénu. Vrstevnice tento úkaz hezky vystihují. Ze všech částí DMT vystihla aplikace tento tvar v porovnání s DMT z *Atlasu* nejvěrněji.



## 8 Dokumentace

Tato kapitola obsahuje dokumentaci k jednotlivým třídám.

### 8.1 Algorithms

Třída *Algorithms* obsahuje metody pro výpočet Delaunayovy triangulace a analýzu DTM.

#### delaunayTriangulation

Metoda **delaunayTriangulation** vytváří nad množinou bodů Delaunayovu triangulaci. Na vstupu je vektor bodů typu `QPoint3D`, metoda vrací uspořádaný vektor hran `Edge`, které tvoří jednotlivé trojúhelníky.

**Input:**

- *vector* `<QPoint3D> points`

**Output:**

- *vector* `<Edge>`

#### createContours

Metoda **createContours** vytváří nad vstupní množinou hran vrstevnice dle zadaného kroku, po kterém se vrstevnice budou vykreslovat. Metoda vrací vektor hran, které představují vrstevnice.

**Input:**

- *vector* `<Edge> dt`
- `double z_min` → minimální výška
- `double z_max` → maximální výška
- `double dz` → krok vrstevnic

**Output:**

- *vector* `<Edge>`

#### getSlope

Metoda **getSlope** počítá sklon trojúhelníku, který je tvořen třemi body. Návrátová hodnota typu `double` nabývá hodnot `<0°;90°>` a vrací sklon trojúhelníku.

**Input:**

- `QPoint3D p1`
- `QPoint3D p2`
- `QPoint3D p3`

### **getAspect**

Metoda **getAspect** počítá orientaci trojúhelníku, který je tvořen třemi body, ke světovým stranám. Návrátová hodnota typu **double** vrací orientaci trojúhelníku ve stupních. Orientace je pravotočivá a nabývá hodnot  $\langle -180^\circ; 180^\circ \rangle$ .

#### **Input:**

- **QPoint3D**  $p_1$
- **QPoint3D**  $p_2$
- **QPoint3D**  $p_3$

### **analyzeDTM**

Metoda **analyzeDTM** vytváří z vektoru hran trojúhelníky a počítá pro ně sklon a orientaci. Vypočtené hodnoty ukládá do datového typu **Triangle**. Návrátová hodnota metody je vektor trojúhelníků typu **Triangle**.

#### **Input:**

- *vector*  $\langle \text{Edge} \rangle$   $dt$

#### **Output:**

- *vector*  $\langle \text{Triangle} \rangle$

### **getPointLinePosition**

Metoda **getPointLinePosition** určuje polohu bodu  $q$  vzhledem k přímce tvořené dvěma body. Na vstupu jsou 3 body typu **QPoint3D**, návratová hodnota je nově definovaný typ **TPosition**.

#### **Input:**

- **QPoint3D**  $q$
- **QPoint3D**  $a$
- **QPoint3D**  $b$

#### **Output:**

- **LEFT**  $\rightarrow$  bod se nachází vlevo od přímky
- **RIGHT**  $\rightarrow$  bod se nachází vpravo od přímky
- **ON**  $\rightarrow$  bod se nachází na přímce



### **getCircleRadius**

Metoda **getCircleRadius** počítá poloměr kružnice, která je tvořena 3 body. Na vstupu jsou 4 body typu **QPoint3D**, návratová hodnota typu **double** vrací velikost poloměru kružnice.

#### **Input:**

- **QPoint3D**  $p_1$
- **QPoint3D**  $p_2$
- **QPoint3D**  $p_3$
- **QPoint3D**  $c \rightarrow$  střed kružnice

### **getDistance**

Metoda **getDistance** počítá vzdálenost mezi dvěma body. Na vstupu jsou 2 body typu **QPoint3D**, návratová hodnota typu **double** vrací vzdálenost mezi dvěma body.

#### **Input:**

- **QPoint3D**  $p_1$
- **QPoint3D**  $p_2$

### **getNearestPoint**

Metoda **getNearestPoint** slouží k nalezení nejbližšího bodu z množiny bodů vzhledem k danému bodu  $p$ . Na vstupu je daný bod  $p$  a vektor bodů typu **QPoint3D**. Návratová hodnota typu **int** vrací index nejbližšího bodu.

#### **Input:**

- **QPoint3D**  $p$
- *vector*  $\langle \mathbf{QPoint3D} \rangle$   $points$

### **getDelaunayPoint**

Metoda **getDelaunayPoint** slouží k nalezení třetího bodu trojúhelníku, který splňuje Delaunayovo kritérium nejmenší opsané kružnice. Na vstupu jsou dva body typu **QPoint3D**, které představují orientovanou hranu, a vektor bodů typu **QPoint3D**. Návratová hodnota typu **int** vrací index hledaného bodu.

#### **Input:**

- **QPoint3D**  $s \rightarrow$  počáteční bod hrany
- **QPoint3D**  $e \rightarrow$  koncový bod hrany
- *vector*  $\langle \mathbf{QPoint3D} \rangle$   $points$

## **getContourPoint**

Metoda **getContourPoint** počítá průsečík hrany trojúhelníku tvořené dvěma body typu **QPoint3D** s rovinou o dané výšce  $Z$ . Návrátová hodnota je typu **QPoint3D**.

### **Input:**

- **QPoint3D**  $p_1$
- **QPoint3D**  $p_2$
- **double**  $z$

## **8.2 Draw**

Třída *Draw* obsahuje metody, které nahrávají a vykreslují vstupní množinu bodů. Dále zajišťuje vykreslení a smazání všech operací, kterou jsou nad množinou prováděny.

### **paintEvent**

Metoda **paintEvent** vykresluje vstupní množinu bodů, Delaunayovu triangulaci, vrstevnice a sklon a orientaci trojúhelníků.

### **clearDT**

Metoda **clearDT** slouží k vymazání všech vykreslených dat.

### **getPoints**

Metoda **getPoints** slouží k získání vektoru bodů z kreslicí plochy. Metoda vrací vektor bodů typu **QPoint3D**.

### **getDT**

Metoda **getDT** slouží k získání vektoru hran z kreslicí plochy. Metoda vrací vektor hran typu **Edge**.

### **setDT**

Metoda **setDT** slouží k převedení Delaunayovy triangulace do kreslicího okna.

### **setContours**

Metoda **setContours** slouží k převedení vrstevnic do kreslicího okna.

### **setDTM**

Metoda **setDTM** slouží k převedení digitálního modelu terénu do kreslicího okna.

## **loadDTM**

Metoda **loadDTM** slouží k načtení vstupních dat do aplikace. Součástí metody je i kontrola, zda se soubor úspěšně nahrál. Návrátová hodnota je typu *QString* vrací hlášku, zda byly polygony úspěšně nahrány či nikoli.

## **drawSlope**

Metoda **drawSlope** slouží k vykreslení sklonu trojúhelníků.

## **drawAspect**

Metoda **drawAspect** slouží k vykreslení orientace trojúhelníků.

## **8.3 Edge**

Třída *Edge* slouží k manipulaci s orientovanými hranami. Definuje dva body typu *QPoint3D* jako počáteční a koncový bod hrany.

### **getS**

Metoda **getS** slouží k získání počáteční bodu hrany.

### **getE**

Metoda **getE** slouží k získání koncového bodu hrany.

### **switchOrientation**

Metoda **switchOrientation** prohazuje orientaci hrany.

## **8.4 QPoint3D**

Třída *QPoint3D* slouží k definování nového datového typu *QPoint3D*, který je odvozen od typu *QPointF* a který navíc obsahuje souřadnici Z.

### **getZ**

Metoda **getZ** slouží k získání souřadnice Z daného bodu.

### **setZ**

Metoda **setZ** slouží k nastavení souřadnice Z daného bodu.

## 8.5 SortByXAsc

Třída *SortByXAsc* má na vstupu dva body typu `QPoint3D`, návratová hodnota je typu `bool`. Metoda vrací bod s nižší souřadnicí  $X$ . Mají-li oba body shodnou souřadnici  $X$ , vrací bod s nižší souřadnicí  $Y$ .

**Input:**

- `QPoint3D`  $p_1$
- `QPoint3D`  $p_2$

**Output:**

- $0 \rightarrow$  bod  $p_2$  má nižší  $x$  souřadnici
- $1 \rightarrow$  bod  $p_1$  má nižší  $x$  souřadnici

## 8.6 Triangle

Třída *Triangle* slouží k definování nového datového typu **Triangle**, který v sobě uchovává informaci o třech bodech typu `QPoint3D`, které tvoří trojúhelník, a o sklonu a expozici trojúhelníku.

**getP\_i**

Metoda **getP\_i** slouží k získání bodu  $P_i$  daného trojúhelníku.

**getSlope**

Metoda **getSlope** slouží k získání sklonu daného trojúhelníku.

**getAspect**

Metoda **getAspect** slouží k získání orientace daného trojúhelníku.

## 8.7 Widget

Metody třídy *Widget* slouží pro práci uživatele s aplikací. Metody na vstupu nemají žádné parametry a návratové hodnoty jsou typu `void`.

**on\_delaunay\_button\_clicked**

Metoda **on\_delaunay\_button\_clicked** nad vstupní množinou bodů zobrazí Delaunayovu triangulaci.

**on\_clear\_button\_clicked**

Metoda **on\_clear\_button\_clicked** vrací aplikaci do výchozí polohy smazáním všeho, co bylo vykresleno.

### **on\_contours\_button\_clicked**

Metoda **on\_contours\_button\_clicked** nad vygenerovanou trojúhelníkovou sítí z Delaunayovy triangulace vykreslí vrstevnice.

### **on\_slope\_button\_clicked**

Metoda **on\_slope\_button\_clicked** obarví trojúhelníky vygenerované Delaunayovou triangulací do odstínů šedi podle hodnoty sklonu daného trojúhelníku.

### **on\_aspect\_button\_clicked**

Metoda **on\_aspect\_button\_clicked** obarví trojúhelníky vygenerované Delaunayovou triangulací na základě jejich orientace ke světové straně.

### **on\_load\_button\_clicked**

Metoda **on\_load\_button\_clicked** načítá data z textového souboru. Uživatel sám vyhledává cestu k požadovanému souboru.

## 9 Závěr

V rámci úlohy *Digitální model terénu a jeho analýzy* byla vytvořena aplikace, která ze vstupní množiny bodů vytváří digitální model terénu. Implementace některých algoritmů byla náročná, avšak výsledek je obstojný. Z kartografického hlediska by aplikace mohla být vylepšena. Jedná se zejména o přidání možnosti navolení povinných a lomových hran, které by zpřesnily výsledný DMT. Algoritmus generuje přijatelné výsledky pro terén, který neobsahuje příliš výrazné terénní hrany. Dále by bylo vhodné zajistit aspoň zevrubní vyhlazení vrstevnic, jelikož působí kostrbatým dojmem.

Do budoucna by bylo vhodné přidat aspoň popis hlavních vrstevnic, případně barevnou hypsometrii. Autorky jsou s výslednou podobou aplikace spokojené.

## 10 Zdroje

1. *BAYER, Tomáš*. 2D triangulace, DMT [online][cit. 4. 12. 2018].  
Dostupné z: <https://web.natur.cuni.cz>
2. *ArcGIS Blog - New Aspect-Slope Raster Function Now Available* [online] [cit. 5. 12. 2018].  
Dostupné z: <https://www.esri.com/>