

**ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE  
FAKULTA STAVEBNÍ, OBOR GEODÉZIE A KARTOGRAFIE  
KATEDRA GEOMATIKY**

název předmětu

**ALGORITMY V DIGITÁLNÍ KARTOGRAFII**

číslo úlohy 3 1. oprava	název úlohy Digitální model terénu – 1. oprava				
školní rok 2020/2021	studijní skup. 60	číslo zadání U3	Zpracoval Usík Svetlana, Vaňková Zuzana	datum 14. 01. 2021	klasifikace

## ***Obsah***

Obsah.....	1
1 Zadání .....	2
2 Popis a rozbor problému .....	3
2.1 Údaje o bonusových úlohách .....	4
3 Popisy algoritmů .....	5
3.1 Delaunay triangulace, polyedrický model terénu.....	5
3.2 Konstrukce vrstevnic.....	9
3.3 Analýza sklonu terénu.....	10
3.4 Analýza orientace terénu (expozice) .....	11
4 Data .....	13
4.1 Vstupní data.....	13
4.2 Výstupní data.....	13
5 Printscreen vytvořené aplikace + zhodnocení .....	14
6 Dokumentace.....	24
6.1 Algorithms.....	24
6.2 Draw .....	25
6.3 Edge.....	26
6.4 QPoint3D.....	26
6.5 sortByX .....	27
6.6 Triangle .....	27
6.7 Widget .....	28
7 Závěr.....	29
8 Přílohy .....	29
9 Seznam literatury.....	29

# 1 Zadání

## Úloha č. 3: Digitální model terénu

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. 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 provedte tyto analýzy:

- S využitím lineární interpolace vygenerujte vrstevnice se *zadaným krokem* a v *zadaném intervalu*, provedte 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.

Zhodnotte 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 provedte 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 zadáné 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: 4 týdny

## 2 Popis a rozbor problému

Nad vstupní množinou bodů měl být v rámci úlohy vytvořen Delaunayovou triangulací polyedrický DMT. Dále měly být zkonztruovány vrstevnice, měla být provedena analýza sklonu terénu a expozice.

Formulaci problému bychom popsali tak, že nad danou množinou bodů ( $P = \{p_1, p_2, \dots, p_n\}$ ;  $P \in \mathbb{R}^2$ ) hledáme triangulaci  $T$ . Tato triangulace je takovým planárním rozdělením, které vytvoří množinu trojúhelníků ( $t = \{t_1, t_2, \dots, t_n\}$ ) a jejich hran tak, aby platily tyto podmínky:

- libovolné dva různé trojúhelníky, mají nejvýše jednu společnou hranu,
- sjednocení všech trojúhelníků tvoří konvexní obálku množiny bodů,
- uvnitř žádného trojúhelníku neleží žádný další bod.

Vzájemný vztah počtu bodů ( $n$ ), počtu hran ( $n_h$ ) a počtu trojúhelníků ( $n_t$ ) lze popsát těmito rovnicemi:

$$n_h = 3n - 3 - k,$$
$$n_t = 2n - 2 - k.$$

kde  $k$  je počet bodů tvořících konvexní obálku.

Využití triangulací je velice široké. V oblastech kartografie, DPZ a GIS, na něž je zaměřeno mnoho předmětů našeho studijního oboru, je triangulace využívána celkem běžně a často. Další možnosti použití jsou například pro zpracování obrazu (segmentace, rozpoznávání vzorků), pro vizualizaci prostorových dat v počítačové grafice. Zajímavá a častá je triangulace v biometrii - telefony, které se odemykají na základě snímání otisku prstů, nebo třeba počítač snímající pro odemčení obličeje jsou dnes běžně využívané záležitosti.



Obr. 1 Delaunay triangulation (left) and its dual Voronoi diagram (right) of the input image [2]

Od algoritmu určeného pro triangulaci požadujeme jednoduchost a snadnou implementaci. Také dostačující rychlosť pro velké množiny bodů (časový odhad by měl být logaritmický). Dále potřebujeme, aby algoritmus nebyl citlivý na singulární případy, kdy řešení není jednoznačné. Měl by být možnost převodu do vyšších dimenzí, schopnost paralelizace a dávat optimální tvar trojúhelníkové sítě.

Pro výběr triangulace jsou důležité tyto parametry: tvar trojúhelníků, povinné hrany, triangulace nekonvexní části. Tvar trojúhelníků by měl být co nejvíce pravidelný a měl by se blížit rovnostranným trojúhelníkům, což nám zaručí dobré přimknutí sítě k terénu. Povinné hrany (v našem případě kosterní čáry) by měli jít vkládat a tím nám dát možnost ovlivnit tvar

terénu. Triangulace by měla být využitelná v nekonvexních oblastech, či oblastech s mezerami, kterém tvoří například vodní plochy, zástavby apod.

[1]

## **2.1 Údaje o bonusových úlohách**

Bonusové úlohy nebyly vypracovány z důvodu dostačujícího počtu bodů za splnění základu úlohy a z důvodu časové zátěže v jiných předmětech.

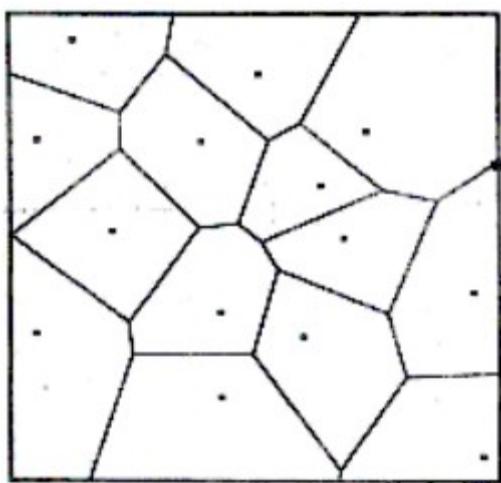
### **3 Popisy algoritmů**

Druhy triangulace podle geometrické konstrukce jsou: Greedy triangulace, Delaunay triangulace, MWT, Constrained triangulace (s povinnými hranami), Datově závislé triangulace. V rámci této úlohy byla metodou inkrementální konstrukce vytvořena Delaunay triangulace nad množinou bodů.

[1]

#### **3.1 Delaunay triangulace, polyedrický model terénu**

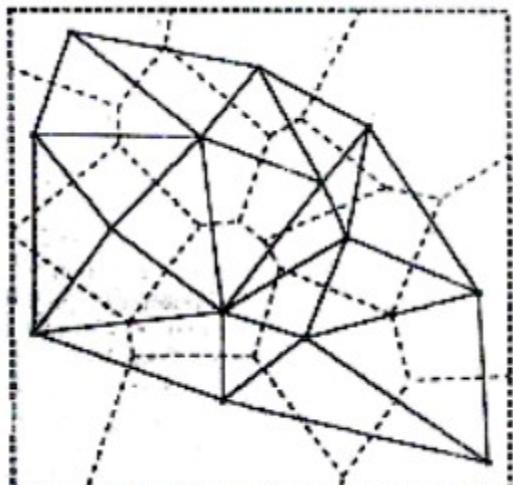
Delaunay triangulace je nejčastěji využívanou triangulací a je standardem v GIS. Je blízce příbuzná k Dirichlet tessellation, která tvoří v množině bodů Thiessonovy (nebo také Voroniovy) polygony. Tyto polygony ohraničí oblast kolem bodu, ve které je vzdálenost k ohrazenému bodu menší než k ostatním bodům.



Obr. 2 Voronoiový polygony [3]

Zjednodušený popis Delaunay triangulace by zněl takto:

1. vezmeme 3 body -> trojúhelník, jemu opíšeme kružnici,
  - 1.a) pokud v kružnici leží jiný bod, vezmeme jiné 3 body
  - 2.b) jinak je trojúhelník zařazen do modelu.



Obr. 3 Delaunay triangulace [3]

Důvodem proč jsou Voronoiový polygony (VP) zmíněny je to, že Delaunay triangulace (DT) je k tomuto grafu grafem duálním. Teda když provedeme následující body, dostaneme DT:

- Do každé stěny VP vložíme jeden vrchol (využijeme již existující body, kdy v každé stěně leží právě jeden) – to jsou vrcholy DT.
- Pro každou hranu VP sestrojíme hranu DT, která spojí ty vrcholy DT, které odpovídají stěnám VP incidentním s danou hranou VP.

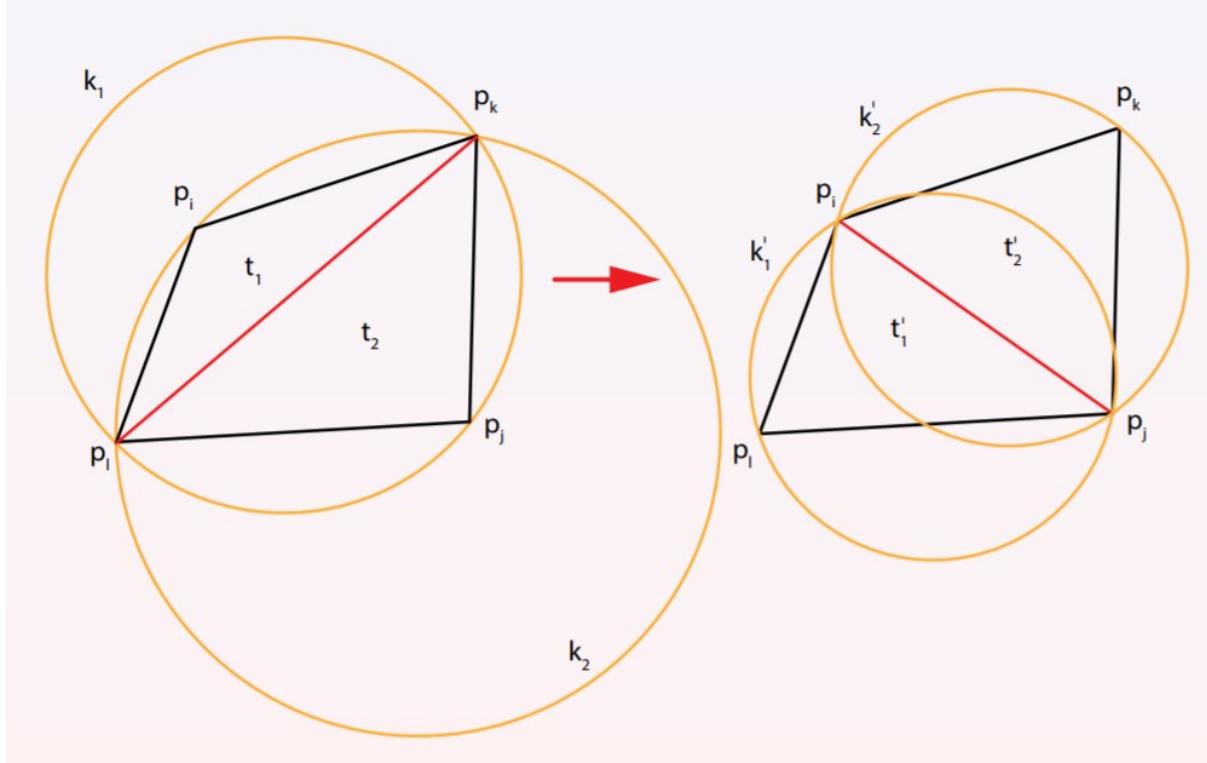
Tato skutečnost je zřejmá z obrázku 3.

Vlastnosti Delaunay triangulace jsou:

- Uvnitř kružnice opsané libovolnému trojúhelníku neleží žádný další bod.
- Maximalizuje minimální úhel avšak neminimalizuje maximální v množině trojúhelníků.
- Je lokálně i globálně optimální vůči kritériu minimálního úhlu.
- Je jednoznačná, pokud žádné 4 body neleží na kružnici.

Ze všech triangulací je výstup Delaunay triangulace nejbližší rovnostranným trojúhelníkům. Při její tvorbě hledáme bod k existující orientované hraně tak, aby opsaná kružnice byla minimální. Zároveň preferujeme řešení, kde kružnice má střed v pravé polorovině (v takovém případě nemusí být poloměr kružnice minimální, pokud kružnice s minimální poloměrem má střed v polorovině levé).

Nad všemi konvexními čtyřúhelníky je opakován prováděna legalizace. Pokud je vhodnější druhé řešení, je prohozena diagonála ve čtyřúhelníku. Nejednoznačné je řešení, pokud leží 4 body na kružnici.



Obr. 4 Edge Flip, legalizace [1]

Pomocí metody inkrementální konstrukce získáme algoritmus využitelný ve 2D i 3D. Metoda je založena na postupném přidávání bodů do již vytvořené DT. Nad existující hranou  $e = (p_1, p_2)$  je hledán  $\underline{p}$ , minimalizující poloměr  $k_i = (e, p_i)$ ,  $p_i \in \sigma_L(e)$ . Hledáme bod  $\underline{p}$  pouze vlevo od orientované hrany. Do DT jsou poté přidány hrany trojúhelníku  $(p_1, p_2, \underline{p})$ :

$$e_1 = (p_2, \underline{p}), \quad e_2 = (\underline{p}, p_1),$$

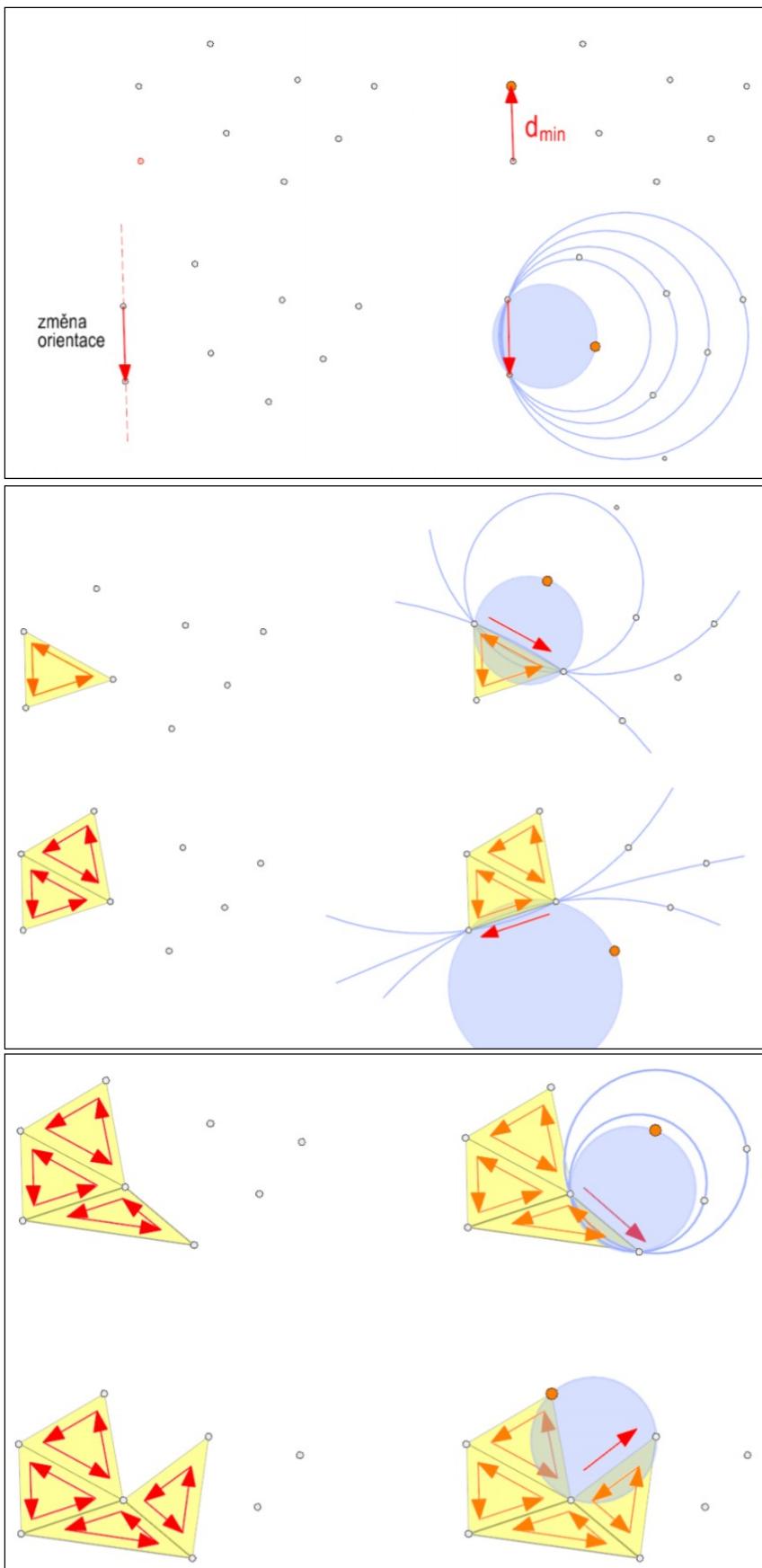
pakliže hrany  $e'_1 = (p_2, \underline{p})$ ,  $e'_2 = (\underline{p}, p_1)$  nejsou v AEL (Active Edge List). Pokud je bod nalezen v pravé polorovině, změníme orientaci hrany  $e$  a hledání opakujeme ( $p_i \in \sigma_r(e)$ ). AEL je modifikovaná datová struktura, která obsahuje hrany  $e$  ke kterým hledáme body  $\underline{p}$ , neukládá se topologický model.

Časový odhad algoritmu je exponenciální ( $O(n^2)$ ). Toto lze ovšem vylepšit. Algoritmus je nestabilní.

Zápis algoritmu inkrementální konstrukce DT:

1.  $p_1 = \text{rand}(P), \|p_2 - p_1\|_2 = \min // Náhodný bod a bod k němu nejbližší.$
2. *Vytvoření první hrany:  $e = (p_1, p_2)$ .*
3.  $\underline{p} = \arg \min_{\forall p_i \in \sigma_L(e)} r'(k_i), k_i = (a, b, p_i), e = (a, b) // Hledání bodu DT$
4. *Pokud není bod nalezen, prohodíme orientaci:  $\underline{p} \gg e \leftarrow (p_2, p_1)$ , jdi na bod 3. .*
5. *Zbývající hrany trojúhelníku:  $e_2 = (p_2, \underline{p}), e_3 = (\underline{p}, p_1)$ .*
6. *Nalezené 3 hrany přidáme do AEL:  $AEL \leftarrow e, AEL \leftarrow e_2, AEL \leftarrow e_3$ .*
7. *Nalezené 3 hrany přidáme do DT:  $DT \leftarrow e, DT \leftarrow e_2, DT \leftarrow e_3$ .*
8. *Dokud AEL není prázdný (while AEL not empty):*
  - a.  *$AEL \rightarrow e, e = (p_1, p_2) // Vezmeme první hrani AEL.$*
  - b.  *$e = (p_2, p_1) // Prohodíme orientaci hrany.$*
  - c.  $\underline{p} = \arg \min_{\forall p_i \in \sigma_L(e)} r'(k_i), k_i = (a, b, p_i), e = (a, b) // Hledáme bod DT i pro opačnou orientaci$
  - d. *Pokud takový bod existuje: If  $\exists \underline{p}$* 
    - i. *Zbývající hrany jsou:  $e_2 = (p_2, \underline{p}), e_3 = (\underline{p}, p_1)$ .*
    - ii. *Přidáme hrani do DT (ne do AEL):  $DT \leftarrow e$ .*
    - iii. *Přidáme ostatní hrany do DT a pokud nejsou v AEL, přidáme je i tam:  $\text{add}(e_2, AEL, DT), \text{add}(e_3, AEL, DT)$ .*

[1][3] [4][5]



Obr. 5, 6, 7 Ilustrace inkrementální konstrukce [1]

## 3.2 Konstrukce vrstevnic

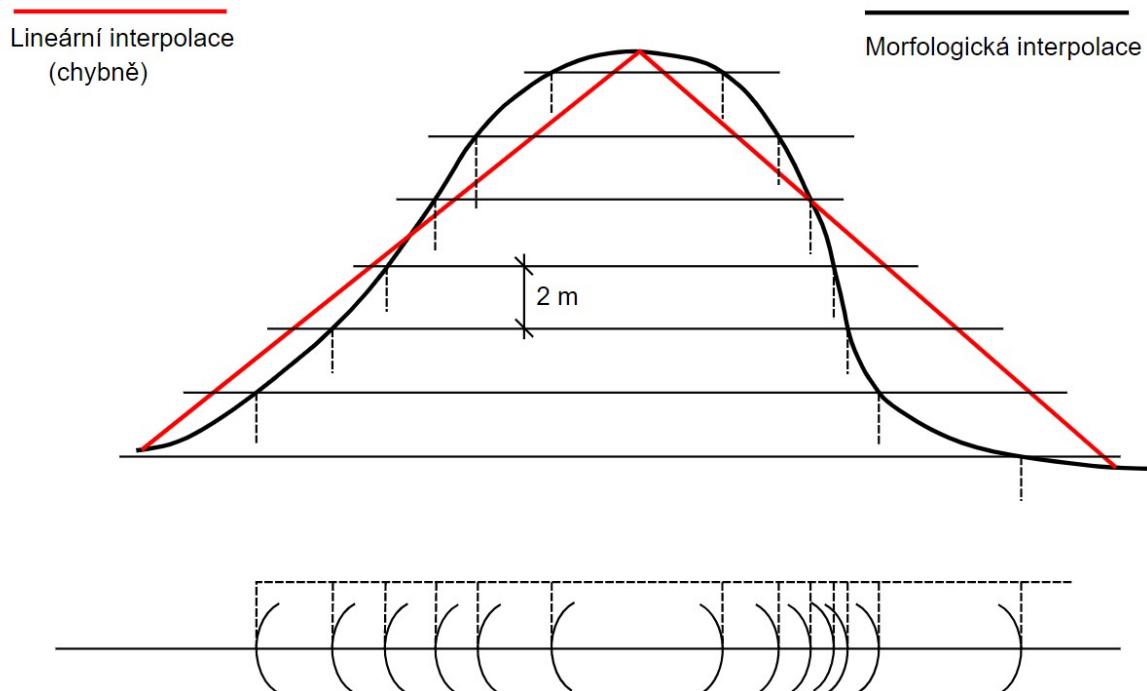
Vrstevnice je linie v mapě, která spojuje body o stejné nadmořské výšce.

Interpolace vrstevnic je kartografická úlohy, která spočívá ve vyhotovení vrstevnic v daném území na základě bodů terénu.

Vrstevnice lze konstruovat lineárně či nelineárně (morfologicky). Při lineární konstrukci předpokládáme konstantní spád mezi 2 body, mezi nimiž je interpolace prováděna. Rozestup vrstevnic je konstantní. Tato metoda je snadná, ale nedává dobrý popis reálného terénu, protože například skutečný kopec nemá v realitě konstantní spád. Lineární interpolace je využívána v mapách velkých měřítek (1 : 500 a 1 : 1 000).

Oproti tomu nelineární interpolační algoritmy předpokládají plynulou změnu sklonu terénu (geomorfologická interpolace). Rozestup vrstevnic mezi dvěma body není konstantní a je zohledňován skutečný tvar terénu (sklon okolních plošek). Sice nám dává reálnější informaci o skutečném tvaru, ale nelze použít jednoduchý postup a těžko se algoritmizuje. Používá se pro menší měřítka (1 : 5 000, 1 : 10 000, příp. menší), protože je redukován počet zaměřených bodů kvůli velikosti oblasti a je využito zákonitostí terénních tvarů k nelineární interpolaci.

### Interpolace vrstevnic v mapách středních měřítek



Obr. 8 Lineární vs morfologická interpolace [6]

V našem řešení byla použita lineární interpolace. Její algoritmus spočívá v hledání průsečnice vodorovné roviny v dané výšce h s rovinou určenou trojúhelníkem z DT. Mohou nastat tyto varianty:

- nemají společný bod = neřešíme,
- průsečnice tvoří 1 bod = neřešíme,
- průsečnice je úsečka.

Musíme dále testovat, zda vodorovná rovina  $\rho : z=h$  neprotíná stranu  $(p_i, p_{i+1})$  trojúhelníku:  

$$(z - z_i)(z - z_{i+1}) < 0.$$

Pokud rovina protíná hranu, bude jedna ze závorek záporná a druhá kladná, protože jeden bod bude nad a druhý pod úrovní roviny a obě závorky budou nenulové. Tedy nerovnost bude platit a vrstevnice bude vykreslena mezi průsečíky. Souřadnice koncových bodů hrany spočteme jako:

$$x_A = \frac{x_2 - x_1}{z_2 - z_1} (z - z_1) + x_1,$$

$$y_A = \frac{y_2 - y_1}{z_2 - z_1} (z - z_1) + y_1.$$

Pokud by byl průsečík v hraně, tak bude výše zmíněná nerovnost rovna nule a vrstevnice se vykreslí v hraně trojúhelníku.

Ještě by mohla nastat situace, kdy bude celý trojúhelník ležet v dané vodorovné rovině. Pak se hrana nevykreslí.

Zápis algoritmu lineární interpolace vrstevnic:

1. Pro všechny trojúhelníky DT.
2. Procházíme roviny v zadáném rozmezí výšek po daném kroku:  

$$\text{for (double } z = z_{\min}; z \leq z_{\max}; z += dz).$$
3. Získáme body trojúhelníku a vypočteme výškové rozdíly  $\Delta z_1 = z - z_1$ ,  $\Delta z_2 = z - z_2$ ,  $\Delta z_3 = z - z_3$  oproti dané rovině:
  - a. Pokud  $\Delta z_1 == 0$  a  $\Delta z_2 == 0$  a  $\Delta z_3 == 0$ , trojúhelník leží v rovině  
 $>>$ vrstevnice se nekreslí.
  - b. Pokud  $\Delta z_1 == 0$  a  $\Delta z_2 == 0$ , nebo  $\Delta z_2 == 0$  a  $\Delta z_3 == 0$ , nebo  $\Delta z_1 == 0$  a  $\Delta z_3 == 0$   $>>$ jedna z hran leží v rovině, vrstevnice je v dané hraně.
  - c. Pokud jsou 2 hrany protnuté rovinou (hrana (1,2) a hrana (2,3), nebo hrana (2,3) a hrana (3,1), nebo hrana (1,2) a hrana (3,1))  $>>$ vypočteme průsečík a vykreslíme hranu. Protnutí hrany posoudíme s využitím nerovnosti:  

$$\Delta z_i \Delta z_j < 0.$$

Souřadnice koncových bodů úseku vrstevnice v daném trojúhelníku určíme z rovnic:

$$x_A = \frac{x_2 - x_1}{z_2 - z_1} (z - z_1) + x_1,$$

$$y_A = \frac{y_2 - y_1}{z_2 - z_1} (z - z_1) + y_1.$$

[1][6]

### 3.3 Analýza sklonu terénu

Jedná se o analytickou úlohu realizovanou nad DMT. Využívá se pro analýzu hydrologických poměrů, sesuvů, lavin, návrhy komunikací, ... . Zprostředkující hodnotou je gradient (vektor maximálního spádu).

Výpočet je prováděn nad každým trojúhelníkem. Algoritmus nejprve vypočte normálu pro rovinu trojúhelníku ( $n_1 = (a, b, c)$ ). Normálový vektor vodorovné roviny je  $n_2 = (0, 0, 1)$ .

Sklon poté určíme z rovnice:

$$\phi = \arccos \left| \frac{(\mathbf{n}_1 \mathbf{n}_2)}{\|\mathbf{n}_1\| \|\mathbf{n}_2\|} \right|, \phi \in \left< 0; \frac{\pi}{2} \right>.$$

Zápis algoritmu výpočtu sklonu (algoritmus pro určení sklonu 1 konkrétního trojúhelníku, výpočet pro všechny trojúhelníky probíhá pomocí volání této metody ve for cyklu v jiné metodě stejné třídy):

1. Vypočteme souřadnicové rozdíly  
(směrové vektory u (z bodu 1 do 2) a v (z bodu 1 do 3)):  

$$u_x = x_2 - x_1$$

$$u_y = y_2 - y_1$$

$$u_z = z_2 - z_1$$

$$v_x = x_3 - x_1$$

$$v_y = y_3 - y_1$$

$$v_z = z_3 - z_1$$
2. Vypočteme normálový vektor trojúhelníku ze směrových vektorů u a v pomocí vektorového součinu:  

$$\mathbf{n}_t = (n_x, n_y, n_z) = ((u_y v_z - u_z v_y), (u_x v_z - u_z v_x), (u_x v_y - u_y v_x))$$
3. Vypočteme euklidovskou normu normálového vektoru:  $\|\mathbf{n}_t\| = \sqrt{n_x^2 + n_y^2 + n_z^2}$ .
4. Úhel mezi oběma rovinami (=sklon) vypočteme jako:  $\phi = \arccos \left| \frac{(\mathbf{n}_t \mathbf{n}_r)}{\|\mathbf{n}_t\| \|\mathbf{n}_r\|} \right|$ , kde  $\mathbf{n}_r$  je normálový vektor vodorovné roviny  $((0, 0, 1)$  s euklidovskou normou rovnou 1). Tedy po úpravě můžeme zapsat:  

$$\phi = \arccos \left| \frac{(n_z)}{\|\mathbf{n}_t\|} \right|.$$

[1]

### **3.4 Analýza orientace terénu (expozice)**

Tato analýza je využívána například ve stavebnictví, zemědělství, hydrologii. Můžeme pomocí ní zjistit, kam dopadá sluneční svít, který ovlivňuje například množství tepla, růst plodin, nebo třeba hydrologické poměry.

Orientace v bodě je azimut průmětu gradientu do roviny. Výpočet je prováděn nad každým trojúhelníkem DMT. Vypočteme normálový vektor  $\mathbf{v} = (a, b, 0)$ , který je hledaným průmětem gradientu do roviny. Azimut vypočteme jako:

$$A = \text{atan} \left( \frac{b}{a} \right), A \in \left< 0; 2\pi \right>.$$

Zápis algoritmu výpočtu expozice (algoritmus pro určení expozice 1 konkrétního trojúhelníku, výpočet pro všechny trojúhelníky probíhá pomocí volání této metody ve for cyklu v jiné metodě stejné třídy):

1. Vypočteme souřadnicové rozdíly

(směrové vektory  $u$  (z bodu 1 do 2) a  $v$  (z bodu 1 do 3)):

$$u_x = x_2 - x_1$$

$$u_y = y_2 - y_1$$

$$u_z = z_2 - z_1$$

$$v_x = x_3 - x_1$$

$$v_y = y_3 - y_1$$

$$v_z = z_3 - z_1$$

2. Vypočteme složky  $n_x, n_y$  normálového vektoru pomocí vektorového součinu:

$$n_x = (u_y v_z - u_z v_y), n_y = (u_x v_z - u_z v_x).$$

// Protože hledáme průměr do roviny, jsou složky vektoru:  $(n_x, n_y, 0)$ .

3. Ze získaných hodnot průměru normálového vektoru do roviny je možné spočítat azimut (úhel od rovnoběžky s osou x ve směru otáčení hodinových ručiček):

$$A = \text{atan} \left( \frac{n_x}{n_y} \right).$$

[1]

## **4 Data**

### **4.1 Vstupní data**

Vstupní data je možno načíst ve formátu .TXT. Načítání je upraveno pro S-JTSK souřadnice.

V souboru musí být uvedeny s tímto formátováním:  $YmezeraXmezeraZ$  (*1 bod na jeden řádek*). Oddělovač desetinných míst je desetinná tečka. Tedy například tímto způsobem:

800300.09 1020222.26 245.54  
800345.25 1020290.26 256.49  
800410.16 1020280.26 260.98  
...

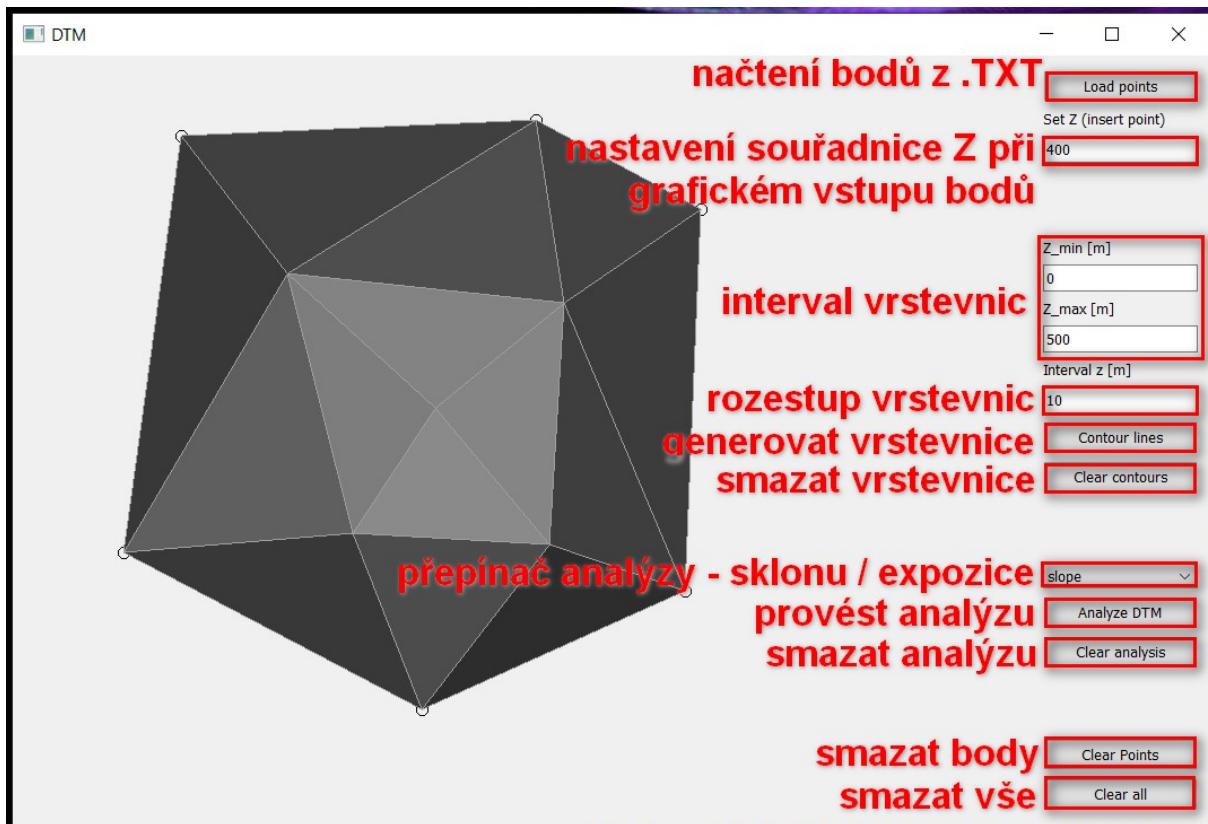
Dále je umožněn grafický vstup bodů kliknutím myši, kde souřadnice x a y jsou přebrány z aktuální pozice kurzoru a výška z je numericky zadána.

### **4.2 Výstupní data**

Výstup této aplikace je realizován grafickým znázorněním triangulační sítě, analýzy (sklonu, nebo expozice) a také vrstevnic se zadaným krokem v zadaném intervalu.

## 5 Printscreen vytvořené aplikace + zhodnocení

Funkce aplikace jsou popsány na následujícím obrázku.

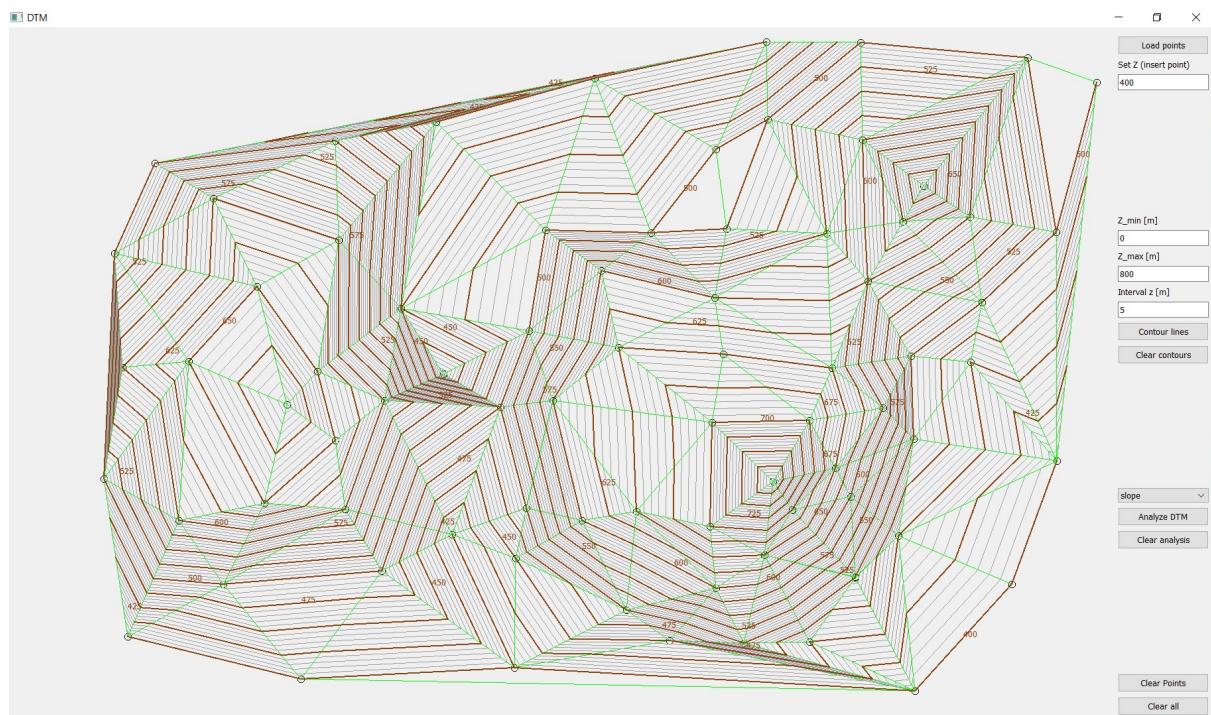


Obr. 9 Popis aplikace

Vrstevnice generujeme tak, že nejprve zadáme interval, poté požadovaný rozestup a následně klikneme na tlačítko Contour lines. Následující obrázky ukazují možnosti výstupu.

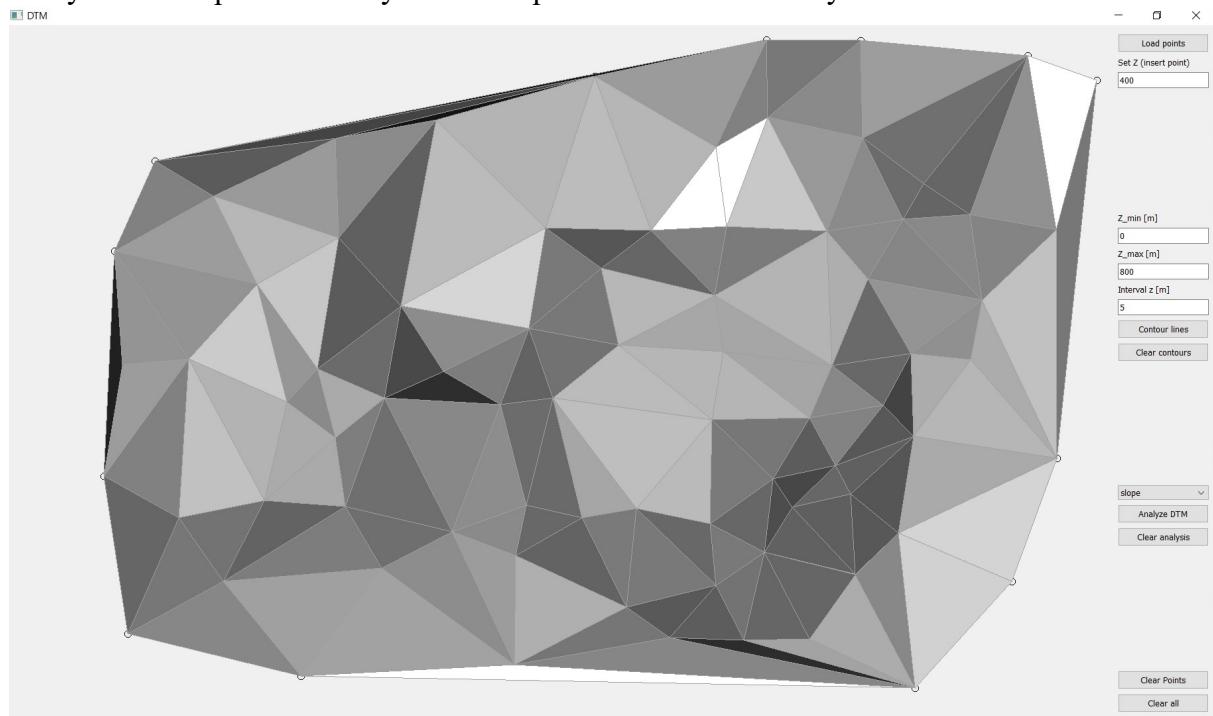


Obr. 10 Ukázka generování vrstevnic nad ručně vloženými body s krokem 10



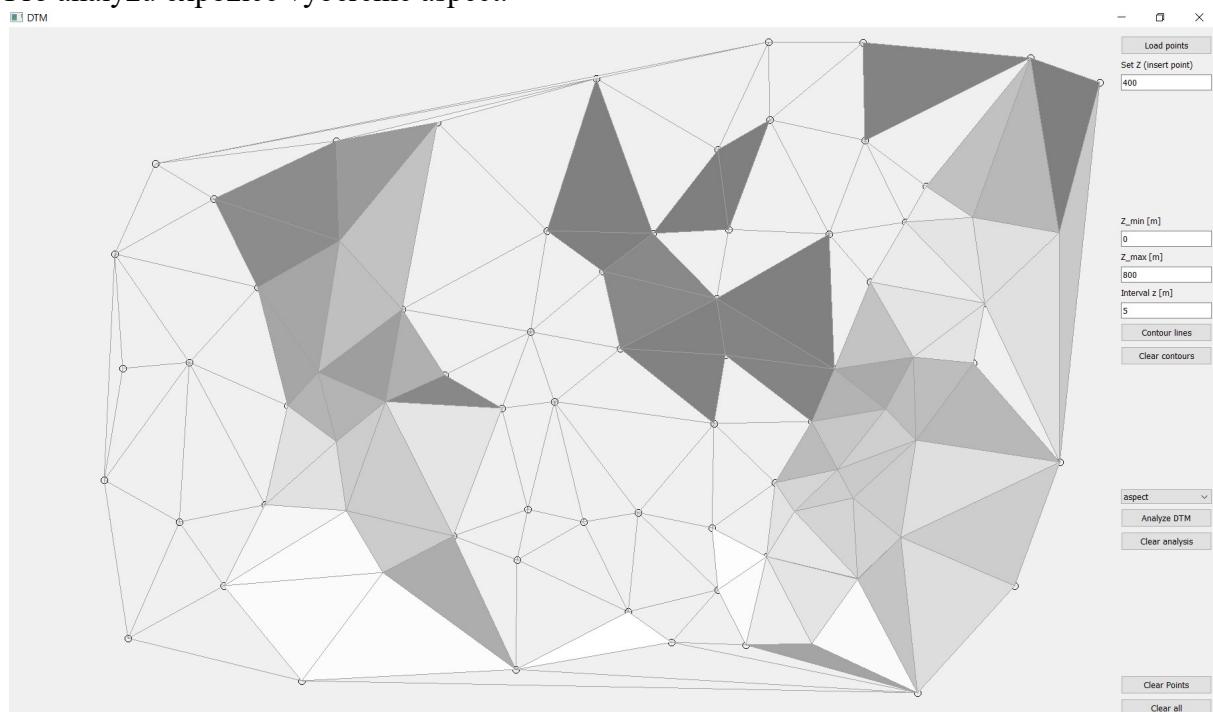
Obr. 11 Ukázka generování vrstevnic nad ručně vloženými body s krokem 5

Analýzu sklonu provedeme vybráním slope a kliknutím na Analyze DMT.



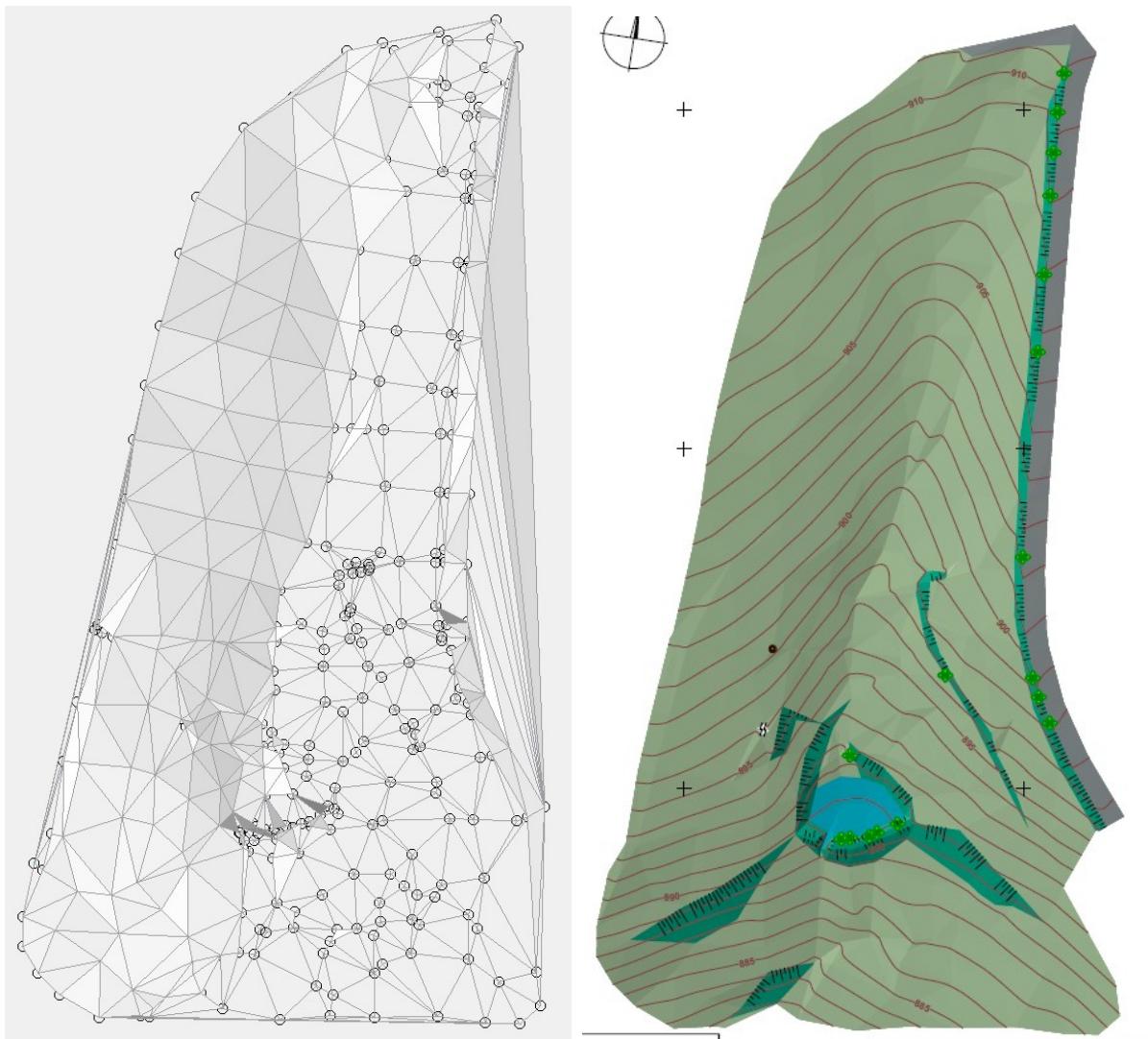
Obr. 12 Ukázka analýzy sklonu

Pro analýzu expozice vybereme aspect.



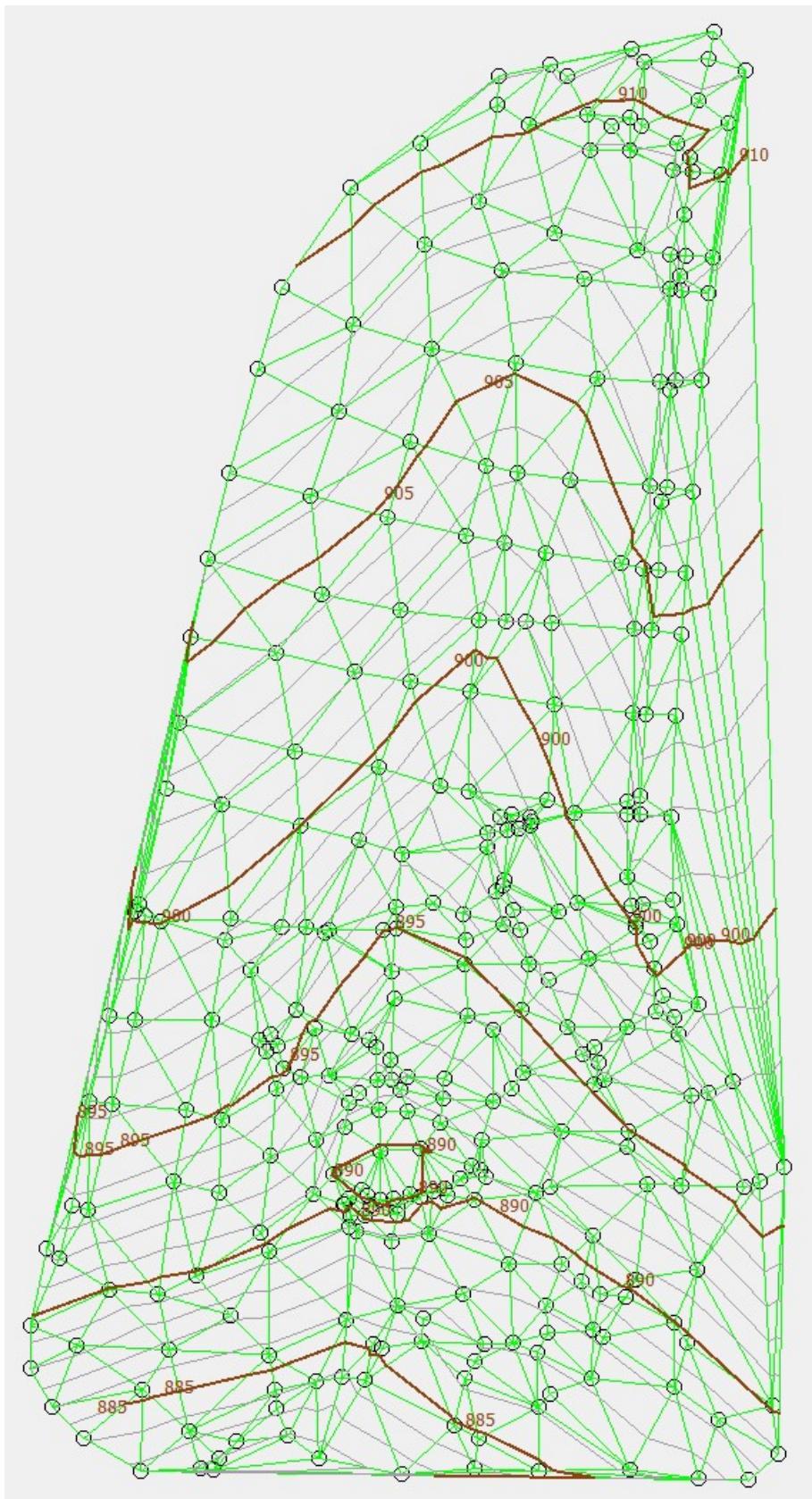
Obr. 13 Ukázka analýzy expozice

Pro nahrání souboru s body vybereme Load points a najdeme cestu k souboru. Jako testovací soubor bodů je přiložen soubor body.TXT, obsahující skutečně měřené body v lokalitě Mariánská u Karlových Varů.

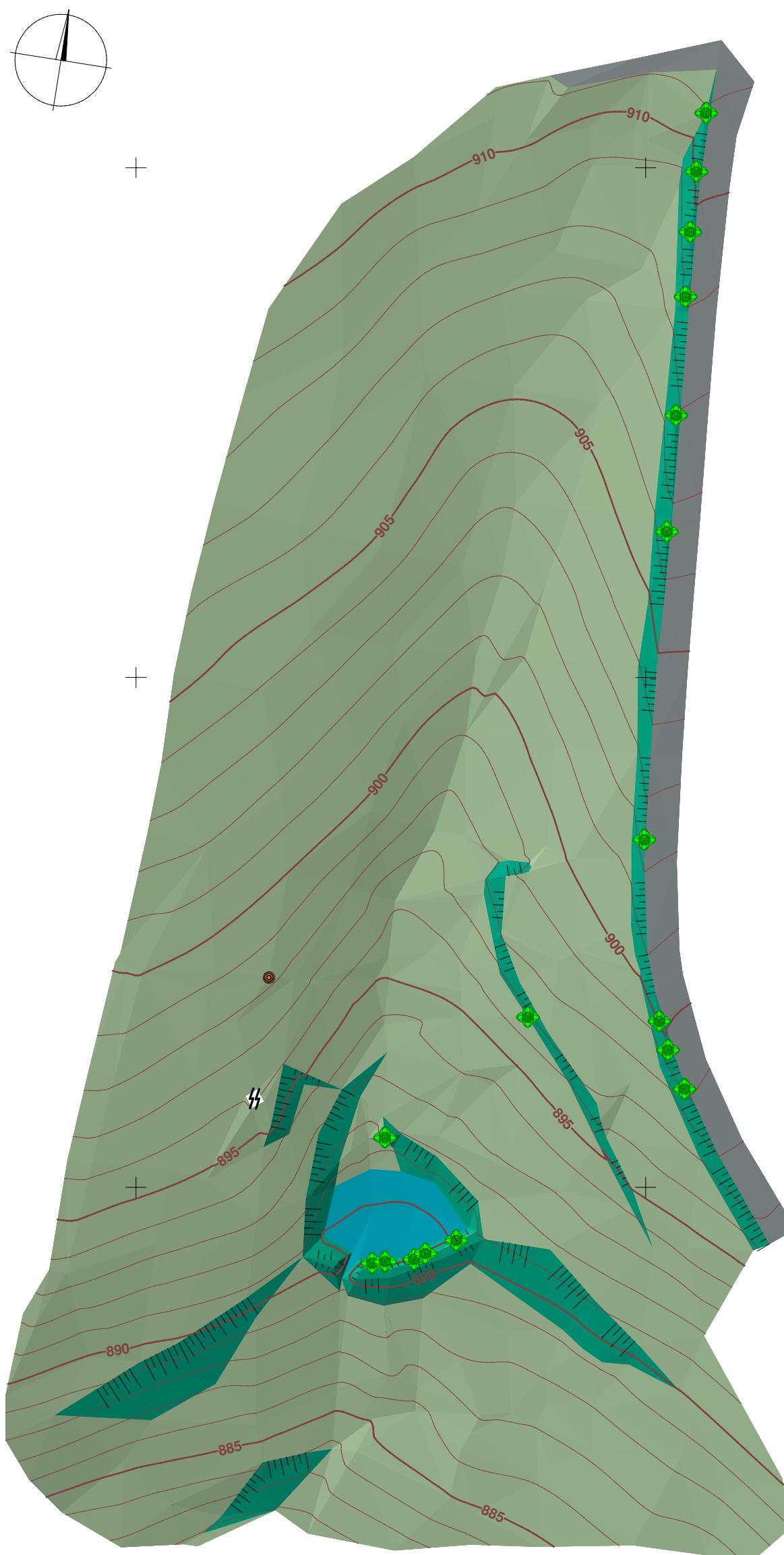


Obr. 14 Porovnání výstupu této úlohy (vlevo) s výstupem z SW Atlas DMT

Vygenerované vrstevnice nad stejnou množinou bodů jsou vizualizovány na následujícím obrázku. Na další stránce je pak přiložen výstup ze SW Atlas DMT.



Obr. 15 Vygenerované vrstevnice



### Legenda

#	Sloup vysokého napětí
◆	Strom
●	Tyč

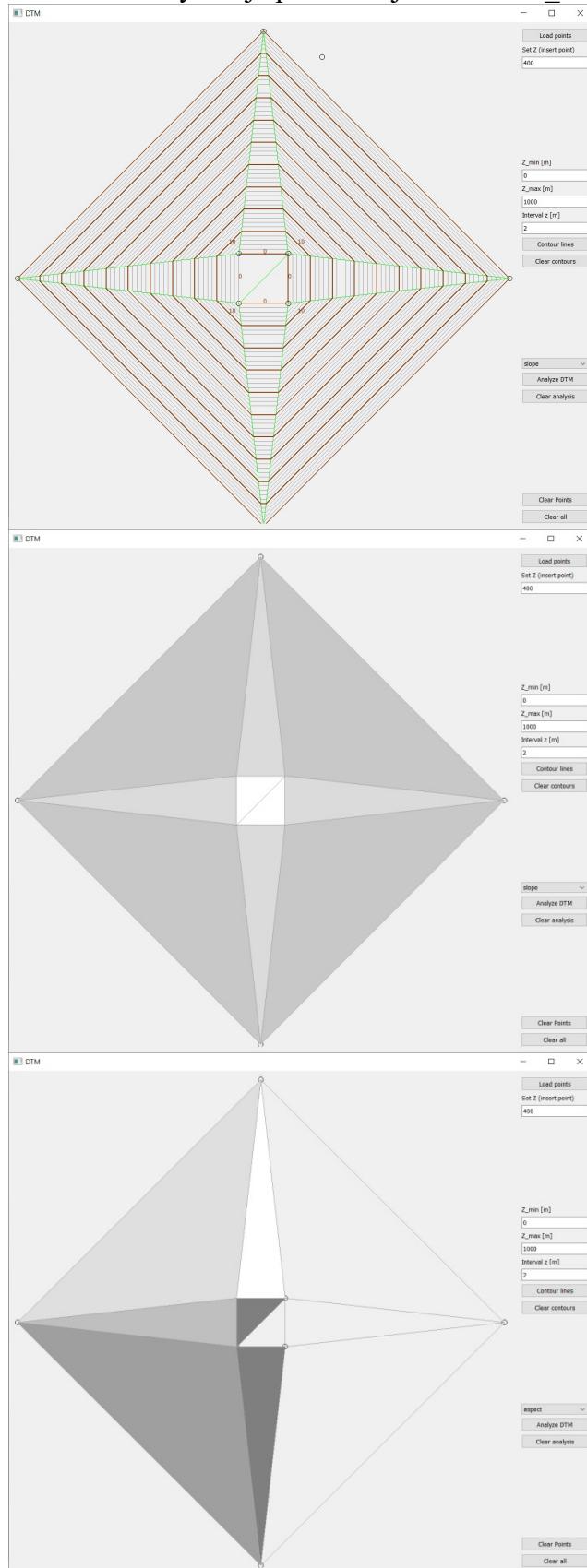
FAKULTA STAVEBNÍ ČVUT V PRAZE  
KATEDRA SPECIÁLNÍ GEODEZIE  
VÝUKA V TERÉNU Z GEODEZIE 3, 4

ÚLOHA: DMT-digitální model terénu

Souřadnicový / výškový systém: S-JTSK / Bpv	Měřítko: 1:1000
--	--------------------

Vypracovala: Vaňková Zuzana	Lokalita: Mariánská u Karlových Varů	Datum: 8.9.2017
		Skupina: 5

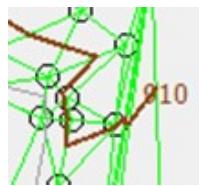
Vzhledem k tomu, že se blíží Vánoce, byla vytvořena vánoční testovací množina bodů ve tvaru hvězdičky. Ta je přiložena jako `vanocni_body.TXT`.



Obr. 16 Vánoční množina bodů (vrstevnice a analýzy)

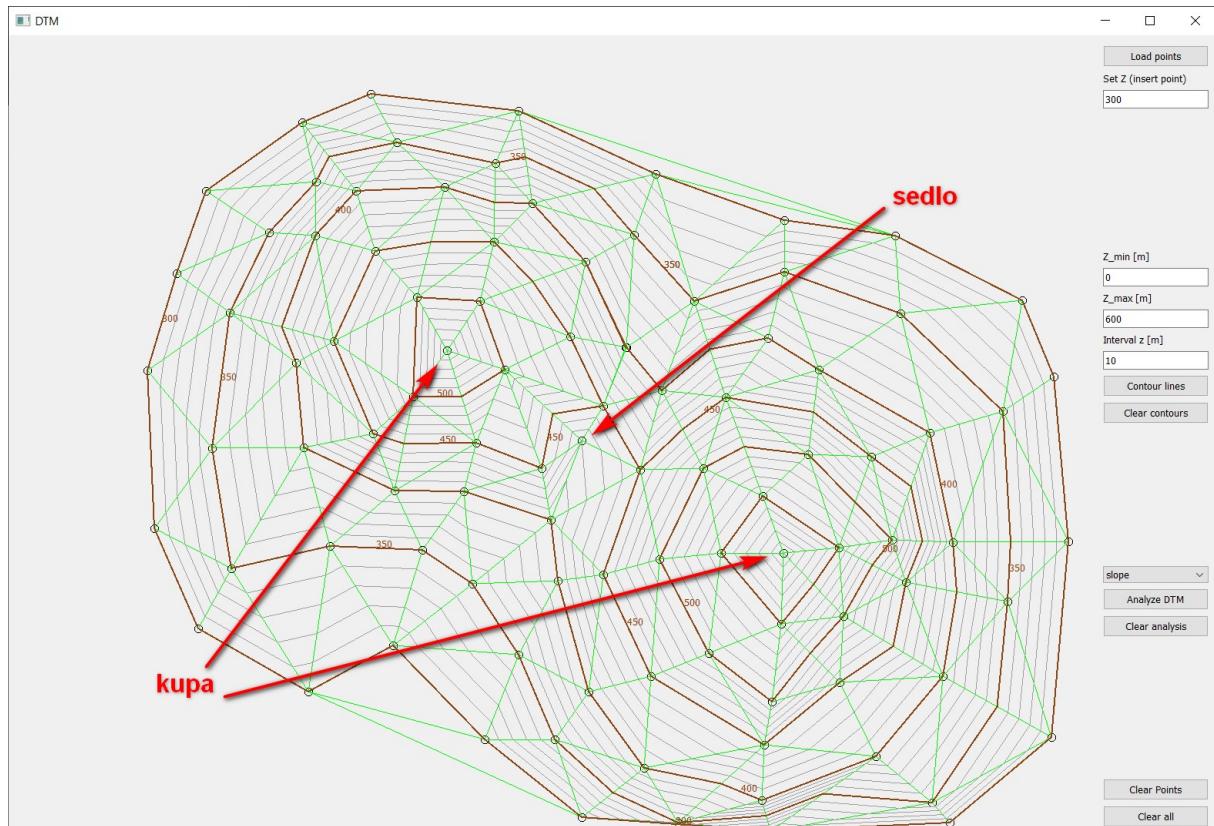
Vzhledem k tomu, že nejsou zohledněny povinné hrany, nelze algoritmus použít pro složitější terén. Například na obrázku 15 lze nedostatky dobře vidět. Když jej porovnáme s výstupem z Atlas DMT (na straně 19), ve kterém byly vybrány povinné hrany, tak se model v místech povinných hran značně liší.

Je zřejmé, že generování vrstevnic není takto úplně ideální. Vrstevnice nejsou plynulé, ale přesně kopírují lomený tvar modelu. Tedy není respektováno to, že tečna k vrstevnici by měla být kolmá ke kosterním čarám, které model nebere v potaz. A také jsou vytvořené lineární interpolací, která nezohledňuje proměnlivý spád terénu. Takže nahuštění vrstevnic v požadovaných místech, nebo naopak jejich rozředění není realizováno. Pro některé terénní tvary dostáváme také zjevně špatný výsledek, například pro kupu nebo sedlo, jak je patrné z obrázku 10 a 11. Příklad chybného generování vrstevnic z důvodu chybějící povinné spojnice:



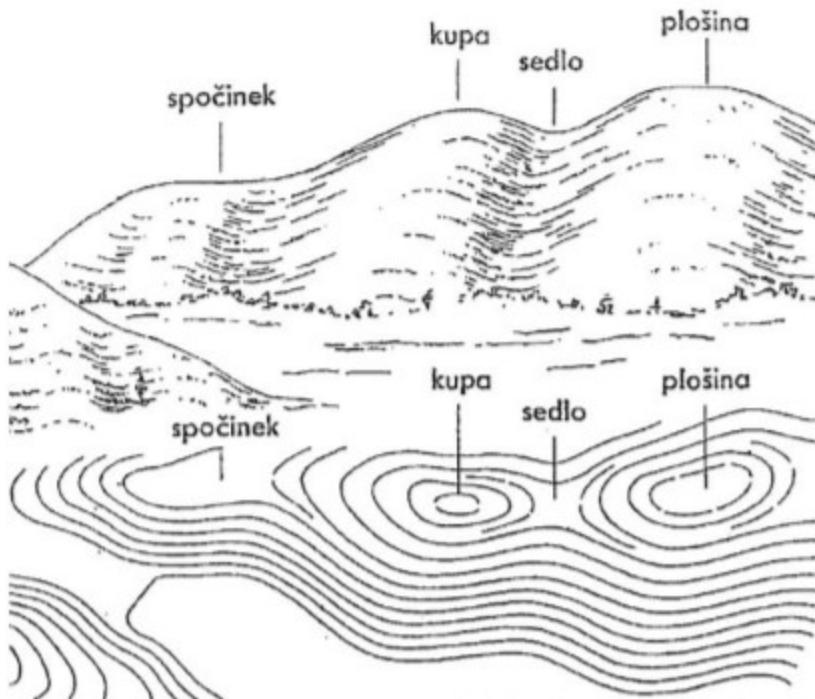
Obr. 17 Chybná vrstevnice

Zjevně chybně vykreslené kupy a sedlo je vidět na následujícím obrázku:



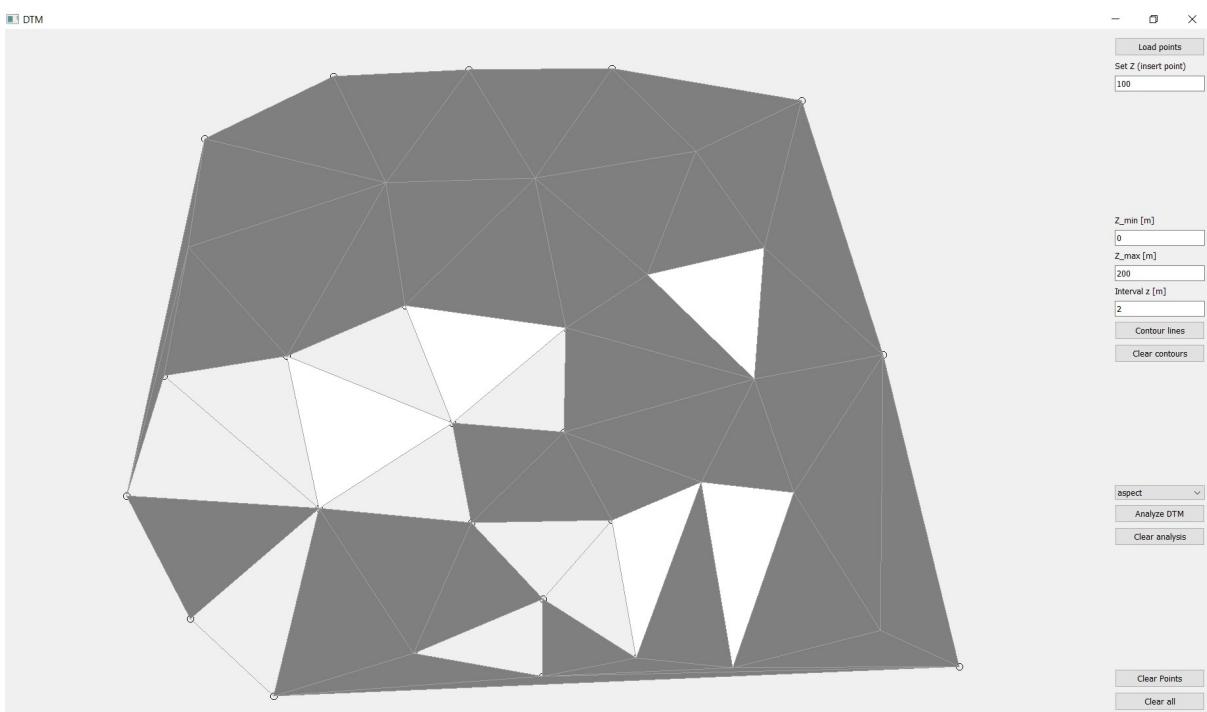
Obr. 18 Chybné kupy a sedlo

Správně vykreslené terénní tvary jsou zobrazeny na následujícím obrázku:



Obr. 19 Tvary na vrcholové části vyvýšeniny [7]

Algoritmus pro určení expozice zjevně selhává na rovných plochách, viz obrázek 16, kde vnitřní 4 body leží ve vodorovné rovině a tudíž by plocha měla být osvětlena stejně, což značí problémy v rovinatém území. Byl proveden pokus – byl analyzován sklon na bodech ve stejné výšce s tímto výsledkem:



Obr. 17 Chybná analýza sklonu v rovině (všechny body mají stejnou výšku)

Co se týče vstupního souboru s body (vstup ve formátu S-JTSK – Y, X, Z), tak ten je pro potřeby zobrazení upraven tak, aby se body zobrazili v mapovém okně. Vhodnější by bylo vyřešit možnost posunu pohledu na dané souřadnice a případně i zoomování v okně a odečet souřadnic jako tomu je v běžných programech pro obdobnou činnost. Takto vlastně data ztrácí reálné souřadnice polohy a je zachován pouze poměr mezi x a y. Tedy souřadnice jsou změněny pomocí měřítka a redukovány tak, aby se všechna data vykreslila.

## **6 Dokumentace**

Dokumentace obsahuje třídy, které obsahují metody a jsou popsány níže.

### **6.1 Algorithms**

**int getPointLinePosition(QPoint3D &q, QPoint3D &p1, QPoint3D &p2)**

Tato metoda určuje polohu bodu q vůči přímce dané 2 body. Principem je výpočet determinantu, který když je  $>0$  tak bod je v levé polorovině a je vrácena hodnota 1, když je  $<0$  tak bod je v pravé polorovině a vracíme hodnotu 0. Pokud jsou body kolineární je vrácena hodnota -1.

**void circleCenterAndRadius(QPoint3D &p1, QPoint3D &p2, QPoint3D &p3, double &r, QPoint3D &s)**

Metoda nastavující poloměr a střed kružnice opsané 3 bodům.

**int findDelaunayPoint(QPoint3D &pi, QPoint3D &pj, std::vector<QPoint3D> &points)**

Metoda sloužící k nalezení pořadí bodu k zařazení do Delaunay triangulace podle stanovených kritérií.

**double dist(QPoint3D &p1, QPoint3D &p2)**

Metoda vracející euklidovskou vzdálenost mezi dvěma body.

**int getNearestpoint(QPoint3D &p, std::vector<QPoint3D> &points)**

Metoda sloužící k nalezení pořadí nejbližšího bodu z points k bodu p.

**std::vector<Edge> DT(std::vector<QPoint3D> &points)**

Metoda vracející triangulaci.

**void updateAEL(Edge &e, std::list<Edge> &ael)**

Metoda sloužící k updatu Active Edge List.

**QPoint3D getContourPoint(QPoint3D &p1, QPoint3D &p2, double z)**

Metoda k získání bodů vrstevnic. Je vypočten průsečík hrany trojúhelníku a dané vodorovné roviny.

**std::vector<Edge> contourLines(std::vector<Edge> &dt, double z\_min, double z\_max, double dz, std::vector<int> &contour\_heights)**

Metoda vracející vrstevnice. Dle zadaných parametrů jsou vytvořeny nad triangulací a vráceny.

**double getSlope(QPoint3D &p1, QPoint3D &p2, QPoint3D &p3)**

Metoda k vypočtení sklonu trojúhelníku.

**double getAspect(QPoint3D &p1, QPoint3D &p2, QPoint3D &p3)**

Metoda k vypočtení orientace trojúhelníku.

### **std::vector<Triangle> analyzeDTM(std::vector<Edge> &dt)**

Metoda sloužící k vypočtení analýzy nad triangulací. Volá metody getSlope a getAspect.

## **6.2 Draw**

Jde o třídu která dědí od třídy Widget a její metody slouží k vykreslení požadovaného výstupu.

### **void paintEvent(QPaintEvent \*event)**

Metoda sloužící k vykreslení všech požadovaných výstupů.

### **void mousePressEvent(QMouseEvent \*event)**

Metoda sloužící ke grafickému vstupu bodů. Souřadnice x a y jsou převzaty z pozice kurzoru a z pak dle zadané hodnoty.

### **void setPoints(std::vector<QPoint3D> &points\_)**

Setter pro nastavení hodnoty bodů (proměnná points).

### **std::vector<QPoint3D> & getPoints()**

Getter vracející hodnotu bodů (proměnná points).

### **void setDT(std::vector<Edge> &dt\_)**

Setter pro nastavení hodnoty proměnné dt.

### **std::vector<Edge> & getDT()**

Getter vracející hodnotu proměnné dt.

### **void setContours(std::vector<Edge> &contours\_)**

Setter pro nastavení hodnoty proměnné vrstevnic - contours.

### **void setBoldContours(std::vector<Edge> &contours\_, std::vector<int> &contours\_z\_)**

Setter pro nastavení proměnných pro hlavní vrstevnice.

### **std::vector<Edge> & getContours()**

Getter vracející proměnnou contours – vrstevnice.

### **std::vector<Edge> & getBoldContours()**

Getter vracející proměnnou bold\_contours – hlavní vrstevnice.

### **void setDTM(std::vector<Triangle> &dtm\_)**

Setter pro nastavení trojúhelníků (uložení informací o sklonu, expozici, pozici).

### **std::vector<Triangle> & getDTM()**

Getter vracející informace o trojúhelnících.

**void importData(std::string &path, std::vector<QPoint3D> &points, QSizeF &canvas\_size, double &min\_z, double &max\_z)**

Metoda sloužící k importu bodů z .TXT souboru. Body jsou nahrány do vektoru points se souřadnicemi upravenými do vhodného měřítka a vhodně redukovanými tak, aby se zobrazili v mapovém okně.

**void setSlope(bool &slope\_){slope = slope\_}// Flags to indicate type of analysis**

Metoda sloužící k nastavení TRUE/FALSE podle toho, zda bylo nebo nebylo v aplikaci zvoleno vykreslení sklonu.

**void setAspect(bool &aspect\_)**

Metoda sloužící k nastavení TRUE/FALSE podle toho, zda bylo nebo nebylo v aplikaci zvoleno vykreslení expozice.

**void setZClicked(double z)**

Metoda, pro získání souřadnice z z grafického vstupu (použita ve třídě Widget), která nastaví hodnotu privátní proměnné z\_clicked třídy Draw.

**double getZClicked()**

Metoda vrací hodnotu z pro grafický vstup.

## 6.3 Edge

**Edge(QPoint3D &s\_, QPoint3D &e\_)**

Třída, která slouží k uložení hrany a obsahuje 2 body – počáteční a koncový bod hrany.

**void setStart(QPoint3D &s\_)**

Setter počátečního bodu hrany.

**void setEnd(QPoint3D &e\_)**

Setter koncového bodu hrany.

**QPoint3D & getStart()**

Getter k vrácení počátečního bodu hrany.

**QPoint3D & getEnd()**

Getter k vrácení koncového bodu hrany.

**void changeOrientation()**

Metoda, která složí ke změně orientace hrany (prohodí počáteční a koncový bod).

## 6.4 QPoint3D

**QPoint3D(double x, double y, double z\_)**

Třída, odvozená od QPointF. Umožňuje ukládat výškovou souřadnici.

### **double getZ()**

Getter třídy, který vrací souřadnici z.

### **void setZ(double z\_)**

Setter třídy, který nastavuje souřadnici z.

## **6.5 sortByX**

Třída, která je metodami volána k setřídění bodů podle souřadnice x.

## **6.6 Triangle**

### **Triangle(QPoint3D &p1\_, QPoint3D &p2\_, QPoint3D &p3\_, double slope\_, double aspect\_)**

Třída sloužící k uložení trojúhelníku, jeho sklonu a orientace.

### **QPoint3D getP1()**

Getter, který vrací hodnotu prvního bodu trojúhelníku.

### **QPoint3D getP2()**

Getter, který vrací hodnotu druhého bodu trojúhelníku.

### **QPoint3D getP3()**

Getter, který vrací hodnotu třetího bodu trojúhelníku.

### **double getSlope()**

Getter, který vrací hodnotu sklonu trojúhelníku.

### **double getAspect()**

Getter, který vrací hodnotu orientace trojúhelníku.

### **void setP1(QPoint3D &P1)**

Setter, který nastavuje hodnotu prvního bodu trojúhelníku.

### **void setP2(QPoint3D &P2)**

Setter, který nastavuje hodnotu druhého bodu trojúhelníku.

### **void setP3(QPoint3D &P3)**

Setter, který nastavuje hodnotu třetího bodu trojúhelníku.

### **void setSlope(double &slope\_)**

Setter, který nastavuje hodnotu sklonu trojúhelníku.

### **void setAspect(double &aspect\_)**

Setter, který nastavuje hodnotu orientace trojúhelníku.

## 6.7 Widget

### **void on\_lineEdit\_editingFinished()**

Metoda, která po ukončení editace v grafickém okně nastaví minimální hodnotu pro vykreslení vrstevnic.

### **void on\_lineEdit\_2\_editingFinished()**

Metoda, která po ukončení editace v grafickém okně nastaví maximální hodnotu pro vykreslení vrstevnic.

### **void on\_lineEdit\_3\_editingFinished()**

Metoda, která po ukončení editace v grafickém okně nastaví krok pro vykreslení vrstevnic.

### **void on\_loadPoints\_clicked()**

Metoda sloužící k importu polygonů.

### **void on\_analyzeDTM\_clicked()**

Po stisknutí tlačítka tato metoda volá metodu třídy Algorithms – analyzeDMT a je provedena analýza podle zadaného požadavku v combo boxu.

### **void on\_createContour\_clicked()**

Metoda volá metodu contourLines třídy Algorithms. Tím jsou vykresleny vrstevnice se stanovenými parametry.

### **void on\_clearContour\_clicked()**

Metoda slouží k vymazání stávajících vrstevnic.

### **void on\_analysisClear\_clicked()**

Metoda slouží k vymazání stávající analýzy.

### **void on\_clearPoints\_clicked()**

Metoda slouží k vymazání stávajících bodů.

### **void on\_clearAll\_clicked()**

Metoda slouží k vymazání všech výstupů.

### **void on\_lineEdit\_4CursorPositionChanged(int arg1, int arg2)**

Metoda, sloužící k zadání hodnoty souřadnice z při grafickém vstupu. Metoda volá metodu setZClicked třídy Draw, čímž předává hodnotu třídě Draw pro další použití. Hodnota je aktualizována po posunutí kurzoru myši, protože jak bylo zjištěno tak editingFinished hodnotu načte až po kliknutí na jinou funkci, proto bylo zvoleno toto řešení, aby bylo jisté, že hodnota je při vstupu bodu aktuální.

## **7 Závěr**

Byla vytvořena grafická aplikace, kde metodou inkrementální konstrukce byla naprogramována Delaunay triangulace. Nad triangulací je generován polyedrický model terénu a provedeny analýzy sklonu a expozice. Také bylo naprogramováno generování vrstevnic. Aplikace umožňuje vstup bodů z textového souboru, ale i grafický vstup s možností nastavení výšky. Bonusové úlohy nebyly vypracovány.

## **8 Přílohy**

Elektronicky:

body.TXT – Vstupní testovací množina bodů  
vanocni\_body.TXT – vánoční testovací množina bodů  
Soubory aplikace

## **9 Seznam literatury**

[1] BAYER, Tomáš. *Rovinné triangulace a jejich využití: Greedy Triangulation. Delaunay Triangulation. Constrained Delaunay Triangulation. Data Dependent Triangulation. DMT.* [online]. [cit. 2020-12-20]. Dostupné z: <https://web.natur.cuni.cz/~bayertom/images/courses/Adk/adk5.pdf>

[2] MOENNIG, Carsten. *Computational Geometry for Computer Vision* [online]. Dec 20, 2018 [cit. 2020-12-20]. Dostupné z: <https://medium.com/datadriveninvestor/computational-geometry-for-computer-vision-f140fab91c76>

[3] 1. *Delaunayho triangulace: Kapitola 8. Triangulace* [online]. [cit. 2020-12-20]. Dostupné z: <https://kgm.zcu.cz/studium/ugi/cviceni/ch08s01.html>

[4] *Fortunův algoritmus a jeho implementace* [online]. 8. 3. 2011 [cit. 2020-12-20]. Dostupné z: [https://ivankuckir.blogspot.com/2011/03/fortunuv-algoritmus-jeho-implementace.html?fbclid=IwAR1fEBzRvVKWgAJdSU19GBMFKccIxt\\_6OM3ygJMeHprYLCH7wnsfwKwnNT4](https://ivankuckir.blogspot.com/2011/03/fortunuv-algoritmus-jeho-implementace.html?fbclid=IwAR1fEBzRvVKWgAJdSU19GBMFKccIxt_6OM3ygJMeHprYLCH7wnsfwKwnNT4)

[5] DEMEL, Jiří. *Grafy a jejich aplikace* [online]. Vyd. 2., (Vlastním nákladem 1.). Libčice nad Vltavou: J. Demel, 2015 [cit. 2020-12-20]. ISBN 978-80-260-7684-1. Dostupné z: [https://kix.fsv.cvut.cz/~demel/grafy/gr.pdf?fbclid=IwAR2czX3GiyBiWpKutHap3vj\\_Jl6fcOK0BFvbljIBYTjZ6Ytmu8ytl7ndqwc](https://kix.fsv.cvut.cz/~demel/grafy/gr.pdf?fbclid=IwAR2czX3GiyBiWpKutHap3vj_Jl6fcOK0BFvbljIBYTjZ6Ytmu8ytl7ndqwc)

[6] Interpolace vrstevnic. *Mapování* [online]. Praha: Katedra geomatiky, Fakulta stavební ČVUT v Praze., © 2015 [cit. 2020-12-20]. Dostupné z: <https://pepa.fsv.cvut.cz/~mapovani/web/vyskopis/interpolace.html>

[7] VICHROVÁ, Martina. *Topografické mapování: Morfologie terénních tvarů* [online]. [cit. 2020-12-20]. Dostupné z: [http://old.gis.zcu.cz/projekty/Geomatika\\_multimedialne/TOMA/Morfologie%20terennich%20tvaru\\_T.pdf](http://old.gis.zcu.cz/projekty/Geomatika_multimedialne/TOMA/Morfologie%20terennich%20tvaru_T.pdf)