ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

GEODEZIE A KARTOGRAFIE

ÚLOHA 3

Algoritmy v digitální kartografii

Digitální model terénu

Katedra geomatiky

Praha 2020

Bc. Eva Frommeltová

Bc. Lucie Hnilicová

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta stavební, Katedra geomatiky
Thákurova 7, 166 29 Praha 6

Obsah

1 Zadání

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.

1.2 Údaje o bonusových úlohách

Krok	**Řešeno/neřešeno**
Delaunay triangulace, polyedrický model terénu.	Ano
Konstrukce vrstevnic, analýza sklonu a expozice.	Ano
Triangulace nekonvexní oblasti zadané polygonem.	Ne
Výběr barevných stupnic při vizualizaci sklonu a expozice.	Ne
Automatický popis vrstevnic.	Ano
Automatický popis vrstevnic respektující kartografické zásady (orientace, vhodné rozložení).	Ne
Algoritmus pro automatické generování terénních tvarů (kupa, údolí, spočinek, hřbet, ...).	Ano
3D vizualizace terénu s využitím promítání.	Ne
Barevná hypsometrie.	Ne

2 Popis a rozbor problému

Cílem úlohy je vytvoření aplikace, která nad množinou bodů vytvoří polyedrický model terénu pomocí Delaunay triangulace. Vstupní body lze do aplikace importovat ze souboru, generátorem nebo zadávat kliknutím myši do prostoru.

3 Popis algoritmů

3.1 Delaunay triangulace

Delaynay triangulace je metoda pro vytvoření trojúhelníkové sítě. Metoda funguje na principu inkrementální konstrukce, která je založena na postupném přidávání bodů do již vytvořené množiny DT. Nad existující Delaunayovskou hranou *e* je hledán bod *p* minimalizují poloměr $k_i = (e, p_i)$. Tato hrana je orientovaná, bod je hledán pouze nalevo od ní. Pokud se nalevo žádný takový bod nenachází, dojde ke změně orientace hrany.

Do množiny *DT* jsou přidávány tři hrany tvořící trojúhelník. Algoritmus obsahuje seznam aktivních hran – *Active Edge List (AEL),* který obsahuje hrany, ke kterým ještě nebyl nalezen třetí bod. Pokud je nalezena nová hrana, je nutno otestovat, jestli se v seznamu již nenachází hrana s opačnou orientací. Pokud ne, je hrana vložena do seznamu. Pokud je k hraně z *AEL* nalezen třetí bod, je ze seznamu odstraněna. Celý algoritmus je ukončen ve chvíli, kdy seznam hran *AEL* je prázdný.

Určení Delaunay bodu *p*:

1) Určení polohy bodu od dané hrany:

$$\vec{u} = (x_2 - x_1, y_2 - y_1),$$
$$\vec{v} = (x_p - x_1, y_p - y_1),$$

$$t = \begin{vmatrix} u_x & u_y \\ v_x & v_y \end{vmatrix} .$$

2) Pokut *t < 0* bod p se nachází v levé polorovině a je vypočten poloměr *r*:

$$m = 0{,}5 \cdot \frac{k_{12} \cdot (-k_4) + k_{11} \cdot k_5 - (k_{10} + k_4 \cdot k_5) \cdot k_6}{x_3 \cdot (-k_4) + x_2 \cdot k_5 + x_1(-k_6)},$$

$$n = 0{,}5 \cdot \frac{k_1 \cdot (-k_9) + k_2 \cdot k_8 + k_3 \cdot (-k_7)}{y_1 \cdot (-k_9) + y_2 \cdot k_8 + y_3 \cdot (-k_7)},$$

$$r = \sqrt{(x_1 - m)^2 + (y_1 - n)^2},$$

kde $k_1 - k_{12}$ jsou koeficienty vypočteny podle vzorců:

$k_1 = x_1^2 + y_1^2,$ $k_2 = x_2^2 + y_2^2,$ $k_3 = x_3^2 + y_3^2,$ $k_4 = y_1 - y_2,$

$k_5 = y_1 - y_3,$ $k_6 = y_2 - y_3,$ $k_7 = x_1 - x_2,$ $k_8 = x_1 - x_3,$

$$k_9 = x_2 - x_3, \qquad k_{10} = x_1^2, \qquad k_{11} = x_2^2, \qquad k_{12} = x_3^2,$$

Algoritmus Delaunay triangulace:

1) Nalezení pivota p_1, $p_1 = min(x)$, nalezení bodu p_2 nejbližšího k pivotu
2) Vytvoření hrany $e = (p_1, p_2)$
3) Nalezení optimálního Delaunay bodu p
4) Pokud p nenalezen, prohoď orientaci e, tj. $e = (p_2, p_1)$, znovu krok 3
5) Vytvoř zbývající hrany trojúhelníku, tj. $e_2 = (p_2, p)$ a $e_3 = (p, p_1)$
6) AEL ← e, AEL ← e_2, AEL ← e_3
7) DT ← e, DT ← e_2, DT ← e_3
8) Dokud AEL není prázdná:
9) AEL → e, $e = (p_1, p_2)$ // vezmi první hranu z AEL
10) Prohození orientace e, tj. $e = (p_2, p_1)$
11) Nalezení optimálního Delaunay bodu p
12) Pokud p existuje
13) $e_2 = (p_2, p)$, $e_3 = (p, p_1)$
14) DT ← e, DT ← e_2, DT ← e_3
15) Prohození orientace e_2 a e_3, tj. $e_2' = (p, p_2)$, $e_3' = (p_1, p)$
16) Pokud e_2' je v AEL
17) AEL → e_2'
18) Jinak AEL ← e_2
19) Pokud e_3' je v AEL
20) AEL → e_3'
21) Jinak AEL ← e_3

3.2 Tvorba vrstevnic

Vrstevnice byly vytvořeny pomocí lineární interpolace, která je založena na analytické geometrii. Je hledána průsečnice roviny tvořená trojúhelníkem a vodorovné roviny s výškou z. Koncové body A, B průsečnice jsou určeny z podobnosti trojúhelníků:

$$x_A = \frac{x_3 - x_1}{z_3 - z_1} \cdot (z - z_1) + x_1,$$

$$y_A = \frac{y_3 - y_1}{z_3 - z_1} \cdot (z - z_1) + y_1,$$

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

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

Z bodů *A, B* je vytvořena hrana určující v daném trojúhelníku vrstevnici o dané výšce *z*. Vrstevnice byly kresleny v případě, že strana trojúhelníku leží ve vodorovné rovině, vodorovná rovina protíná trojúhelník ve vrcholu a protilehlé straně nebo v případě, že vodorovná rovina protíná trojúhelník ve dvou stranách.

3.3 Sklon terénu

Sklon terénu je vyjádřen úhlem φ mezi normálovým vektorem *(0,0,1)* a normálovým vektorem roviny trojúhelníku n_t. Pro výpočet je nutné určit směrové vektory v trojúhelníku ($n_x\ n_y, n_z$). Norma normálového vektoru *(0,0,1)* je rovna jedné, tudíž v čitateli argumentu funkce *acos* zůstane pouze n_z.

Směrové vektory v trojúhelníku:

$$n_x = u_y \cdot v_z - v_y \cdot u_z,$$

$$n_y = -(u_x \cdot v_z - v_x \cdot u_z),$$

$$n_z = u_x \cdot v_y - v_x \cdot u_y,$$

kde $u_x = x_2 - x_1; u_x = y_2 - y_1;\ u_z = z_2 - z_1$ *a* $v_x = x_2 - x_3; v_x = y_2 - y_3;\ v_z = z_2 - z_3$.

Normálový vektor roviny trojúhelníku:

$$n_t = \sqrt{n_x^2 + n_y^2 + n_z^2},$$

Úhel φ:

$$\varphi = \operatorname{acos}\left(\frac{n_z}{n_t}\right).$$

3.4 Orientace terénu

Orientace terénu je vyjádřena azimutem α. Azimut je dán průmětem normálového vektoru z roviny trojúhelníku do roviny $\rho_{x,y}$.

Výpočet *x* a *y* části normálového vektoru:

$$n_x = u_y \cdot v_z - v_y \cdot u_z,$$

$$n_y = -(u_x \cdot v_z - v_x \cdot u_z),$$

kde $u_x = x_2 - x_1; u_x = y_2 - y_1;\ u_z = z_2 - z_1$ *a* $v_x = x_2 - x_3; v_x = y_2 - y_3;\ v_z = z_2 - z_3$.

Výpočet orientace terénu:

$$\alpha = \text{atan2}\left(\frac{n_x}{n_y}\right).$$

Pro výpočet azimutu byla použita funkce *atan2*, aby byla směrnice zařazena do správného kvadrantu.

4 Popis vrstevnic a generátory

4.1 Automatický popis vrstevnic

Automatický popis vrstevnic je generován pomocí již vygenerovaných hran. Tyto hrany jsou definovány souřadnicemi počátečního a koncového bodu. Ze souřadnic je vypočtena jejich průměrná hodnota, ke které je pomocí funkce *drawText* přiřazen popis dané výškové kóty. Kóty jsou vykresleny pro každou třetí hranu.

4.2 Generování terénních tvarů

Pro automatické generování terénních tvarů byly vybrány tyto tvary: Náhodné body *(Random), Kupa (Knoll), Hřbet (Ridge) a Údolí (Valley).* Generátory byly vytvořeny v samostatné třídě. Všechny generátory jsou vytvořeny na stejném principu funkcionality.

4.2.1 Náhodné body (Random)

Funkce vygeneruje náhodné body s uživatelem zadaného počtu bodů s náhodným rozmístěním. Náhodné body byly generovány pomocí funkce *rand().* Souřadnice *x,y* jsou děleny výškou a šířkou kreslícího plátna, aby nedocházelo k vykreslení mimo něj. Souřadnice *z* je generována jako náhodné číslo. Tento generátor byl použit pro generování všech ostatních bodů.

4.2.1 Kupa (Knoll)

Tato funkce přebírá souřadnice *x,y* z generátoru náhodných bodů *n*. Z těchto souřadnic je spočteno těžiště, ke kterému je přidělena výška *z = 1200*.

$$x_t = \frac{\sum_{i=1}^{n} x_i}{n},$$

$$y_t = \frac{\sum_{i=1}^{n} y_i}{n},$$

$$z_t = 1200.$$

Pro ostatní náhodně vygenerované body jsou vygenerovány náhodné výšky. Výšky jsou generovány tak, že čím je bod vzdálenější od těžiště, tím je jeho výška nižší oproti těžišti.

4.2.3 Hřbet (Ridge)

Funkce přebírá souřadnice *x,y* z generátoru náhodných bodů. Poté je přiřazena výška prvnímu a poslednímu bodu a ze všech bodů je vypočteno těžiště (*viz vzorce u kupy*). Souřadnice *z* pro těžiště je nastavena jako výška prvního a posledního bodu.

Pro zbylé vygenerované body je pomocí vzdáleností vypočtena jejich poloha vůči počátečnímu a koncovému bodu. Výška takto jednotlivě určených bodů je poté nepřímo úměrně přiřazována vzdálenosti tak, aby byla menší, než jsou výšky počátečního a koncového bodu.

4.2.4 Údolí (Valley)

Tato funkce vychází, z již výše popsané funkce pro generování hřbetu – je jejím opakem. Počáteční a koncový bod mají dánu velmi malou souřadnici *z*. Výška pro ostatní body je přímo úměrně přiřazována vzdálenosti tak, aby byla větší, že jsou výšky počátečního a koncového bodu.

5 Popis Aplikace

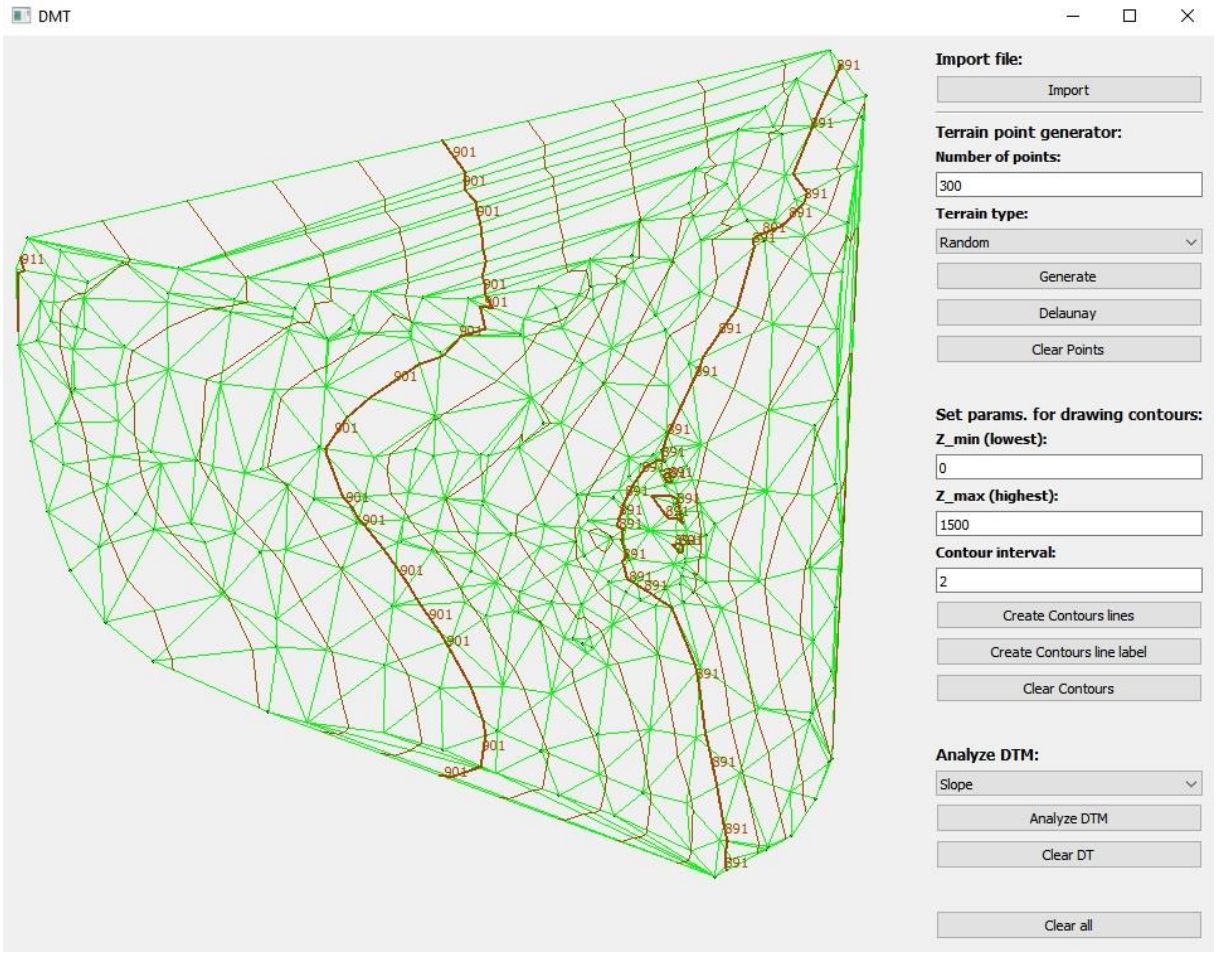
V této kapitole je popsána funkcionalita aplikace.

5.1 Vstupní data

Pro vstup dat je v této aplikace více možností. Jako první možnost je vložení bodů uživatelem pomocí klikání myši na kreslící plátno. Druhou možností vstupu dat (tlačítko *Import*) je nahráním ze souboru **.txt*, kde jsou data pro jednotlivé body uloženy ve tvaru YXZ (*viz ukázka*):

997267.511	*845904.656*	*883.316*
997268.669	*845883.024*	*881.569*
997268.679	*845883.015*	*881.575*
997269.790	*845867.605*	*881.135*

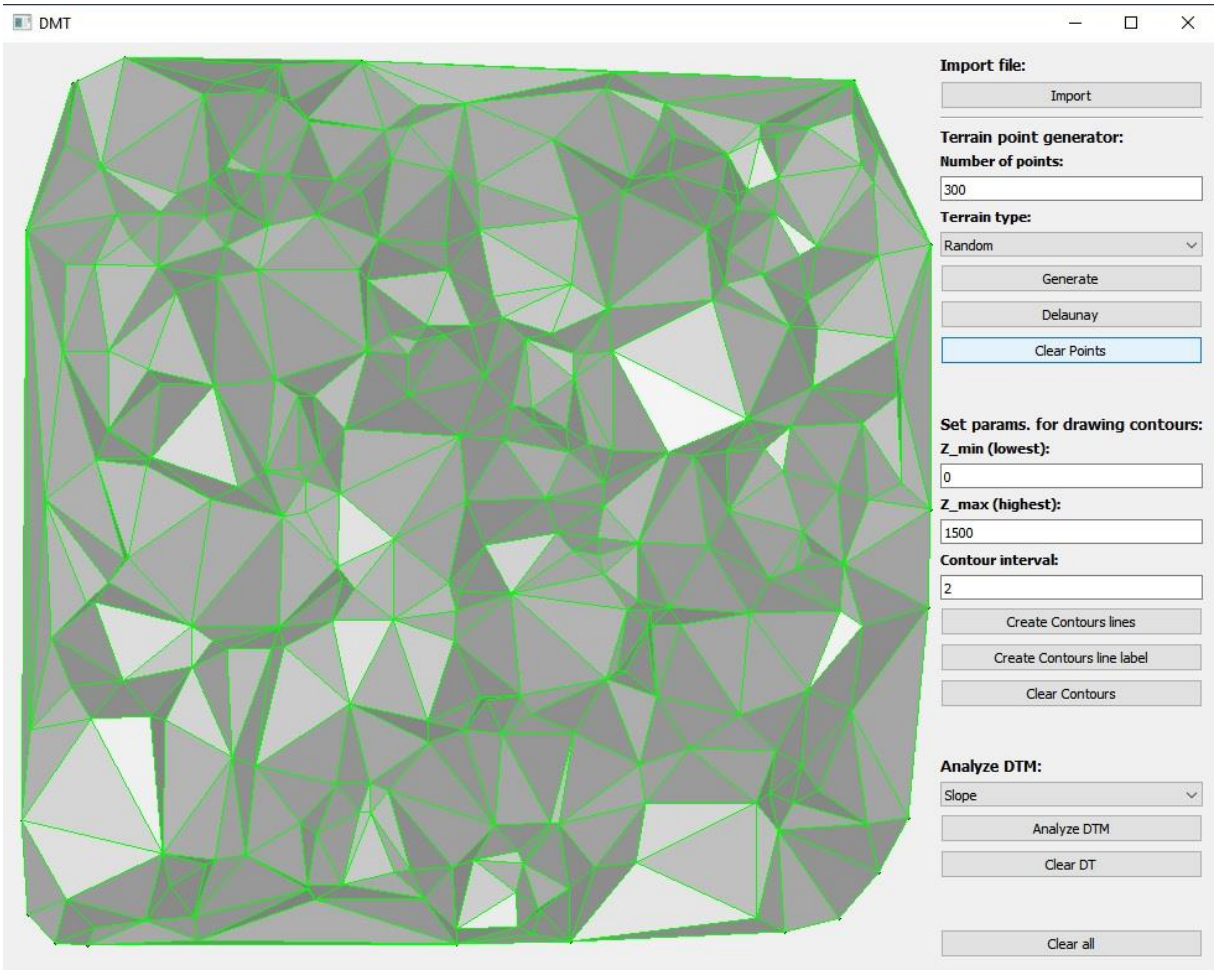
Třetí možností vstupu dat je pomocí generátoru bodů terénu, kde je nutné vložit počet bodů, který se má vygenerovat.



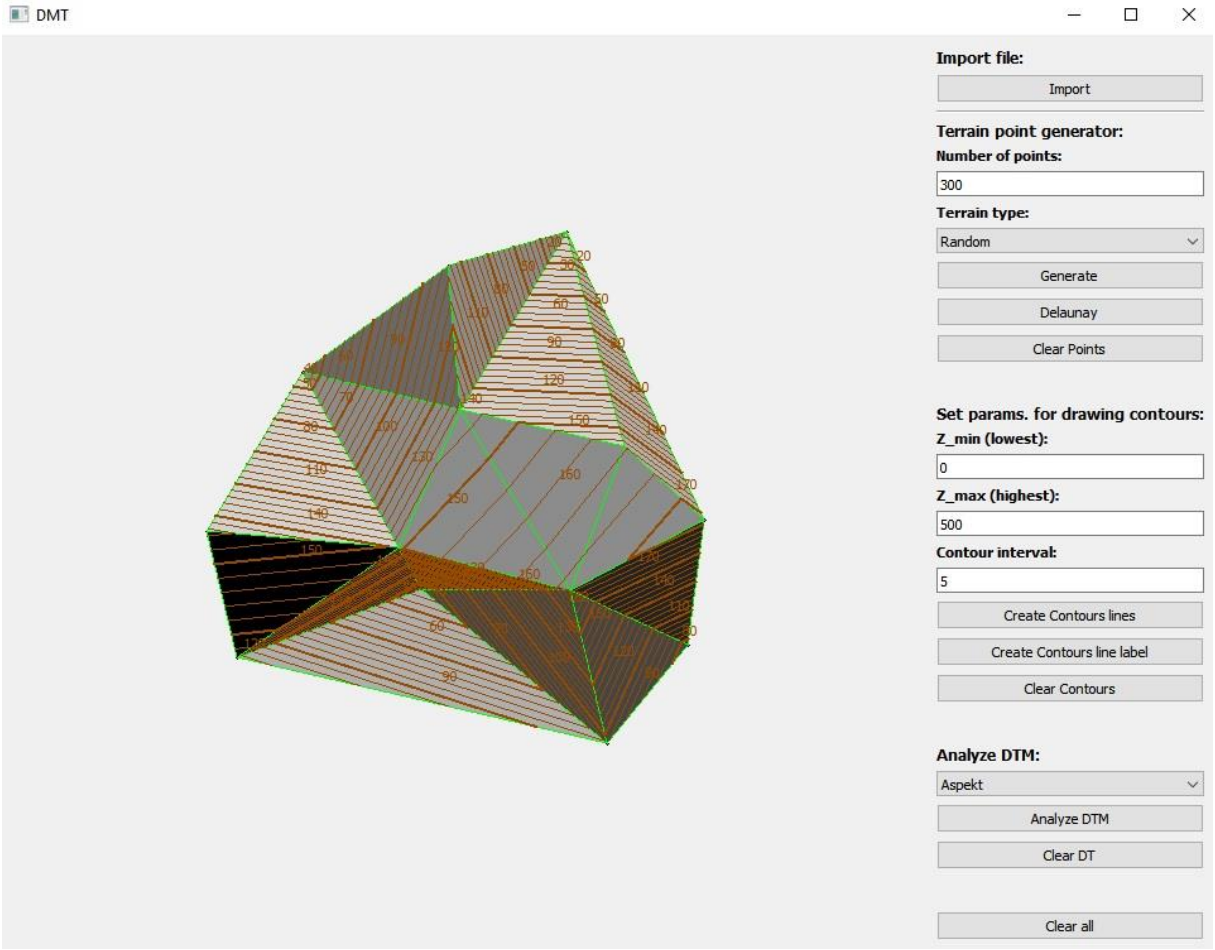
Obrázek 1: Ukázka aplikace – Importované body, vrstevnice + popis

5.2 Výstupní data

Výstupem je grafická vizualizace vstupních dat, nad kterými je možno provádět různé operace pro určení terénu. Je možné vytvořit Delaunayho triangulaci, generovat vrstevnice a vytvořit jim popis nebo analyzovat terén pomocí sklonu či orientace v terénu. Dále jsou zde tlačítka pro mazání, jak jednotlivých částí, tak i celého obsahu kreslícího plátna.



Obrázek 2: Ukázka aplikace – Slope, vygenerované náhodné body



Obrázek 3: Ukázka aplikace – Aspect, body určeny myší

6 Dokumentace

6.1 Třída Algorithms

Třída obsahuje:

```
int getPointLinePosition(QPoint3D &q, QPoint3D &p1, QPoint3D &p2);
```

- Funkce určuje polohu bodu vůči přímce, jestli se bod nachází vlevo od přímky (vrací 1), vpravo od přímky (vrací 0), na přímce (vrací -1)

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

- Funkce na získaní středu kružnice a poloměru

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

- Funkce pro nalezení Delaunayského bodu vhodného pro Delaunay triangulaci

```
double dist(QPoint3D &p1, QPoint3D &p2);
```

- Funkce na získání vzdálenosti mezi dvěma body

```
double getPointLineDistance(QPoint3D &q, QPoint3D &p1, QPoint3D
&p2);
```

- Funkce na získání vzdálenosti bodu od přímky

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

- Funkce pro nalezení nejbližšího bodu

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

- Funkce generující Delaunay triangulaci

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

- Funkce pro práci se seznamem hran

```
QPoint3D getContourPoint(QPoint3D &p1,QPoint3D &p2, double z);
```

- Funkce vrací pro určení bodu na vrstevnici

```
std::vector<Edge> contourLines(std::vector<Edge> &dt,double z_min,
double z_max, double dz);
```

- Funkce generující vrstevnice

```
double calculateSlope(QPoint3D &p1, QPoint3D &p2, QPoint3D &p3);
```

- Funkce pro výpočet sklonu terénu

```
double calculateAspect(QPoint3D &p1, QPoint3D &p2, QPoint3D &p3);
```

- Funkce pro výpočet orientace terénu

```
std::vector<Triangle> analyzeDTM(std::vector<Edge> & dt);
```

- Funkce analyzující terén podle sklonu a orientace

6.2 Třída Draw

Třída obsahuje:

```
private:
 std::vector<QPoint3D> points;
```

- Deklarace proměnné pro množinu bodů

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

- Deklarace proměnné pro množinu hran

```
 std::vector<Edge> contours, label_c, main_c;;
```

- Deklarace proměnné pro množinu hran, které určují vrstevnice, popis vrstevnic a hlavní vrstevnici

```
 std::vector<Triangle> dtm;
```

- Deklarace proměnné pro množinu trojúhelníků

```
 bool slope, aspect;
```

- Deklarace proměnné, která je využívána pro vykreslení sklonu a orientace terénu

```
public:
```

```
void paintEvent(QPaintEvent *event);
```
- Funkce pro vykreslení

```
void mousePressEvent(QMouseEvent *event);
```
- Funkce pro snímání souřadnic bodů z kreslícího plátna po kliknutí myši

```
static void importPoints(std::string &path, std::vector<QPoint3D>
    &points,  QSizeF &canvas_size, double &min_z, double &max_z);
```
- Funkce pro načtení bodu ze souboru

```
void setPoints(std::vector<QPoint3D> &points_){points = points_;}
```
- Funkce pro nastavení bodů

```
std::vector<QPoint3D> & getPoints(){return points;}
```
- Funkce pro získání bodů

```
void setDT(std::vector<Edge> &dt_){dt = dt_;}
```
- Funkce pro nastavení Delaunay triangulace

```
std::vector<Edge>  & getDT(){return dt;}
```
- Funkce pro získání Delaunay triangulace

```
void setContours(std::vector<Edge> &contours_){contours =
contours_;}
```
- Funkce pro nastavení vrstevnic

```
std::vector<Edge> & getContours(){return contours;}
```
- Funkce pro získání vrstevnic

```
void setMainContours(std::vector<Edge> &main_c_){main_c = main_c_;}
```
- Funkce pro nastavení hlavních vrstevnic

```
std::vector<Edge>& getMainContours() {return main_c;}
```
- Funkce pro získání hlavních vrstevnic

```
void setLabelContours(std::vector<Edge> &label_c_){label_c =
label_c_;}
```
- Funkce pro nastavení popisků vrstevnic

```
void clearContours(){label_c.clear(); main_c.clear();}
```
- Funkce pro mazání popisků vrstevnic a hlavních vrstevnic

```
void setDMT(std::vector<Triangle> &dtm_){dtm = dtm_;}
```
- Funkce pro nastavení DMT

```
std::vector<Triangle> & getDMT(){return dtm;}
```
- Funkce pro získání DMT

```
void setSlope(bool slope_){slope = slope_;}
```
- Funkce pro nastavení sklonu terénu

```
void setAspect(bool aspect_){aspect = aspect_;}
```
- Funkce pro nastavení orientace terénu

6.4 Třída Generatorterrain

Třída obsahuje:

```
std::vector<QPoint3D> generateRandom(int &n, int &w, int &h);
```
- Funkce pro generování náhodných bodů

```
std::vector<QPoint3D> generateKnoll(int &n, int &w, int &h);
```
- Funkce pro generování bodů tvořící terénní tvar kupy

```
std::vector<QPoint3D> generateRidge(int &n, int &w, int &h;
```
- Funkce pro generování bodů tvořící terénní tvar hřbetu

```
std::vector<QPoint3D> generateValley(int &n, int &w, int &h);
```
- Funkce pro generování bodů tvořící terénní tvar údolí

6.4 Pomocné třídy

sortByX

- Třída setřídí body podle souřadnice x

Edge

- Třída pracující s hranami. Umožňuje získat či nastavit počáteční a koncový bod hrany, dále umožňuje změnit hraně orientaci.

QPoint3D

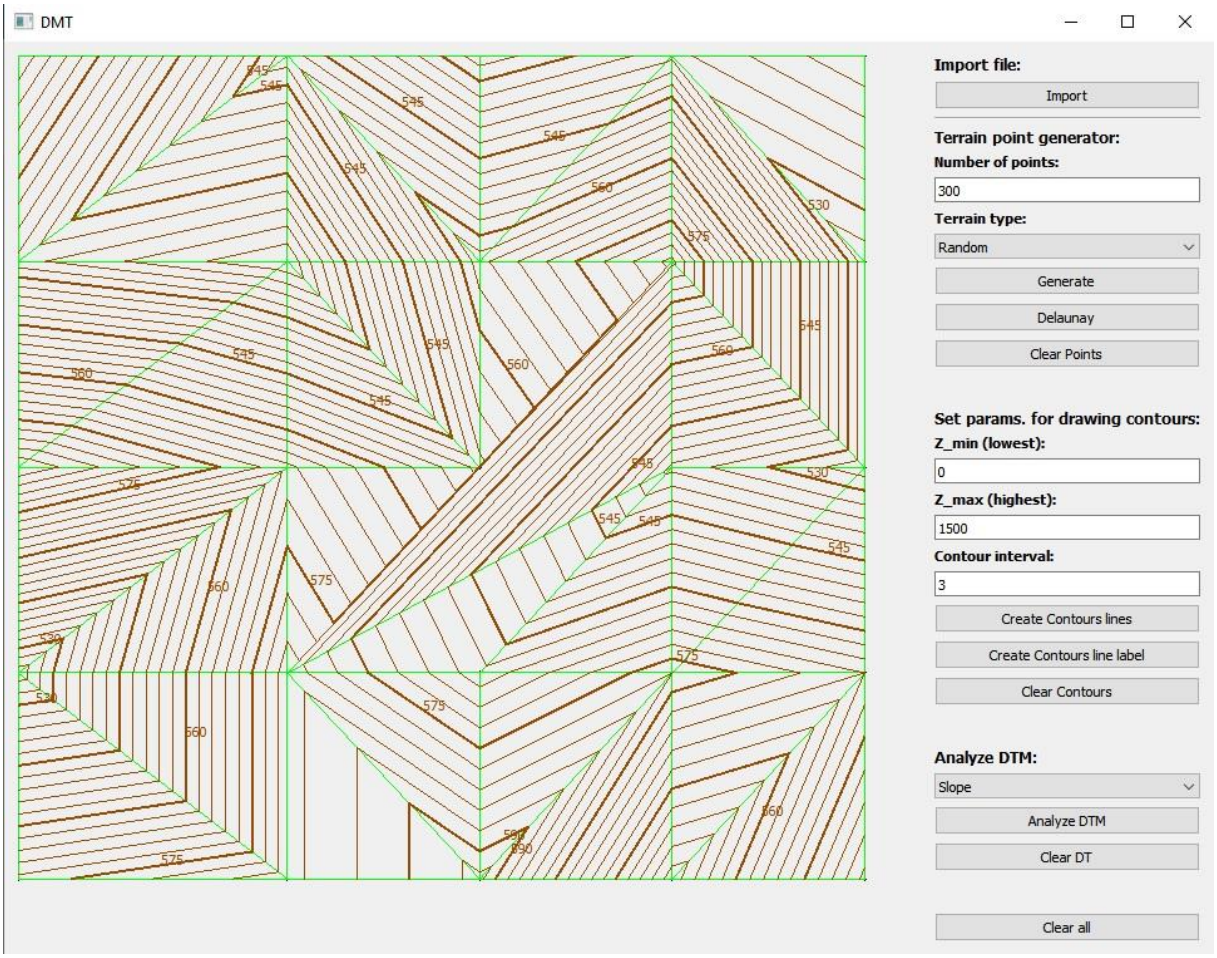
- Třída je odvozená od třídy QPointF, ke které byla přidána souřadnice Z.

Triangle

- Třída byla vytvořena pro práci s trojúhelníky. Trojúhelník je tvořen třemi body a dále nese informaci o sklonu a orientaci svahu. Práce s parametry je možná díky get a set funkcím.

7 Zhodnocení funkčnosti aplikace

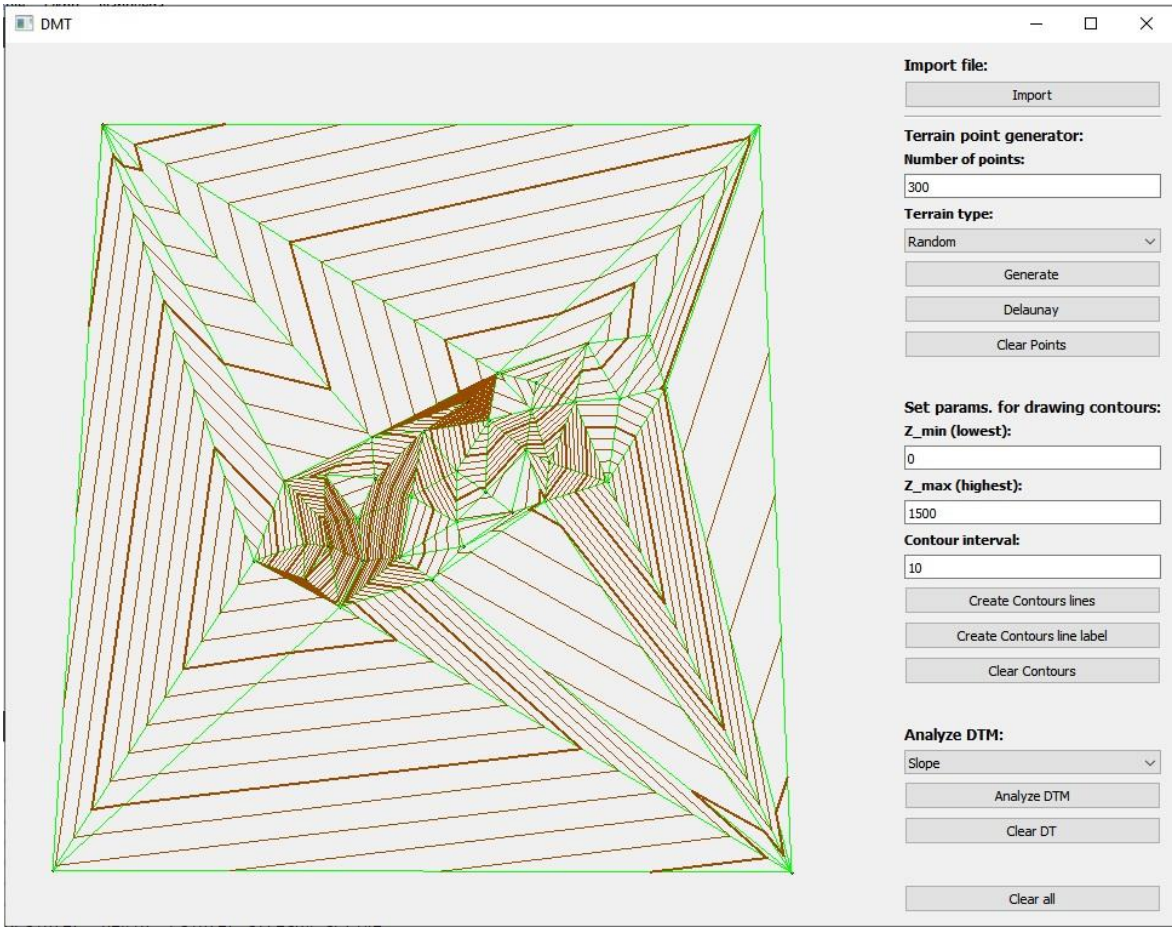
Delaunay triangulace vždy nedává úplně optimální výsledky. Například pro body v gridu, kde výpočet nemá jednoznačné řešení vzhledem ke stejným rozestupům mezi jednotlivými body. Proto není tato triangulace pro takovou množinu vhodná. Vrstevnice působí velmi zvláštně a nereprezentují skutečný terén *(obr.4)*. Simulace bodů v mřížce byla vytvořena pomocí souboru *grid.txt*.



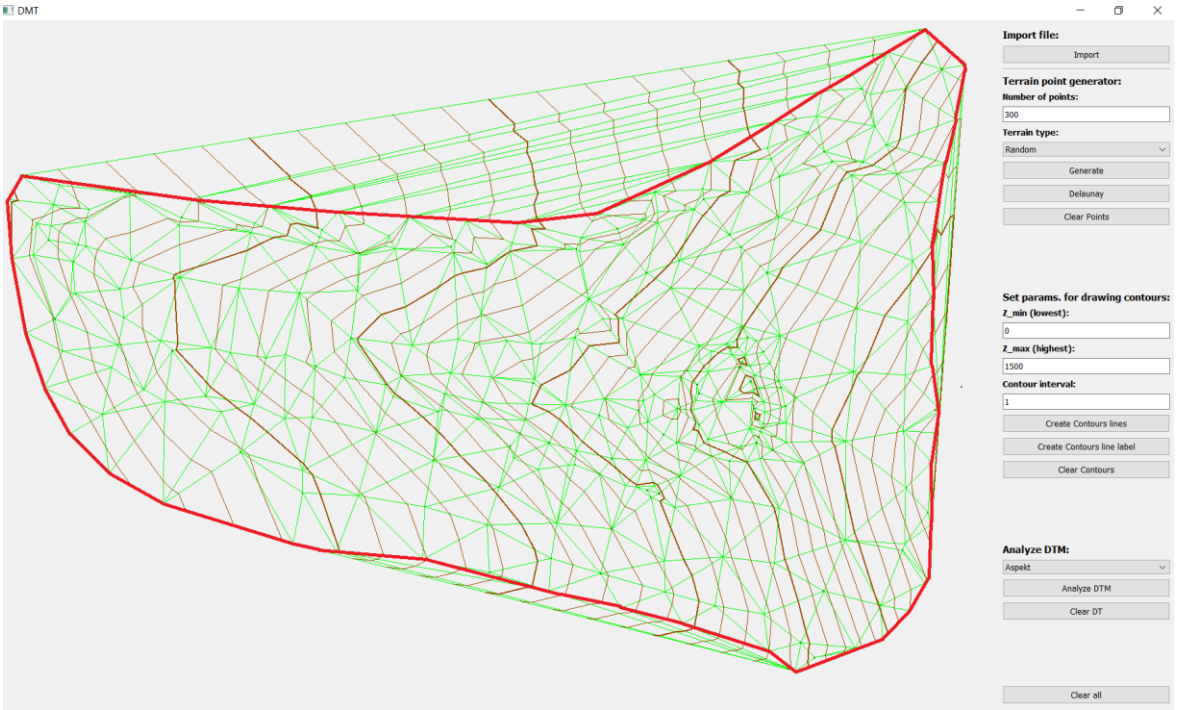
Obrázek 4: Body v mřížce

Dalším případem neoptimálních výsledků je, pokud body nemají alespoň přibližně stejnou hustotu rozmístění v rámci jednoho území. Potom vznikají velké a úzké trojúhelníky, ve kterých vygenerované vrstevnice nemusí plně odpovídat realitě. Vrstevnice mohou být příliš do špičky. Bylo by vhodné vrstevnice na přechodech vhodně zaoblit, aby výsledek více odpovídat skutečnému terénu. Případ byl nasimulován pomocí myši *(obr.5)*.

Dalším zlepšením by mohlo být docíleno přidáním povinných hran – patrné z *obr.6* (povinná hrana doplněna ručně v kreslícím editor). Při definici povinných hran by zanikly úzké trojúhelníky mimo měřené území, kde se nenachází žádné měřené body. Díky nedefinování povinných hran je možné, že vykreslení vrstevnic nemusí být v každém případě správné. Také by bylo přínosné, kdyby se v aplikaci daly mazat, které nechceme zahrnovat do vykreslení. Proto není aplikace vhodná pro vykreslování složitějších terénních tvarů.

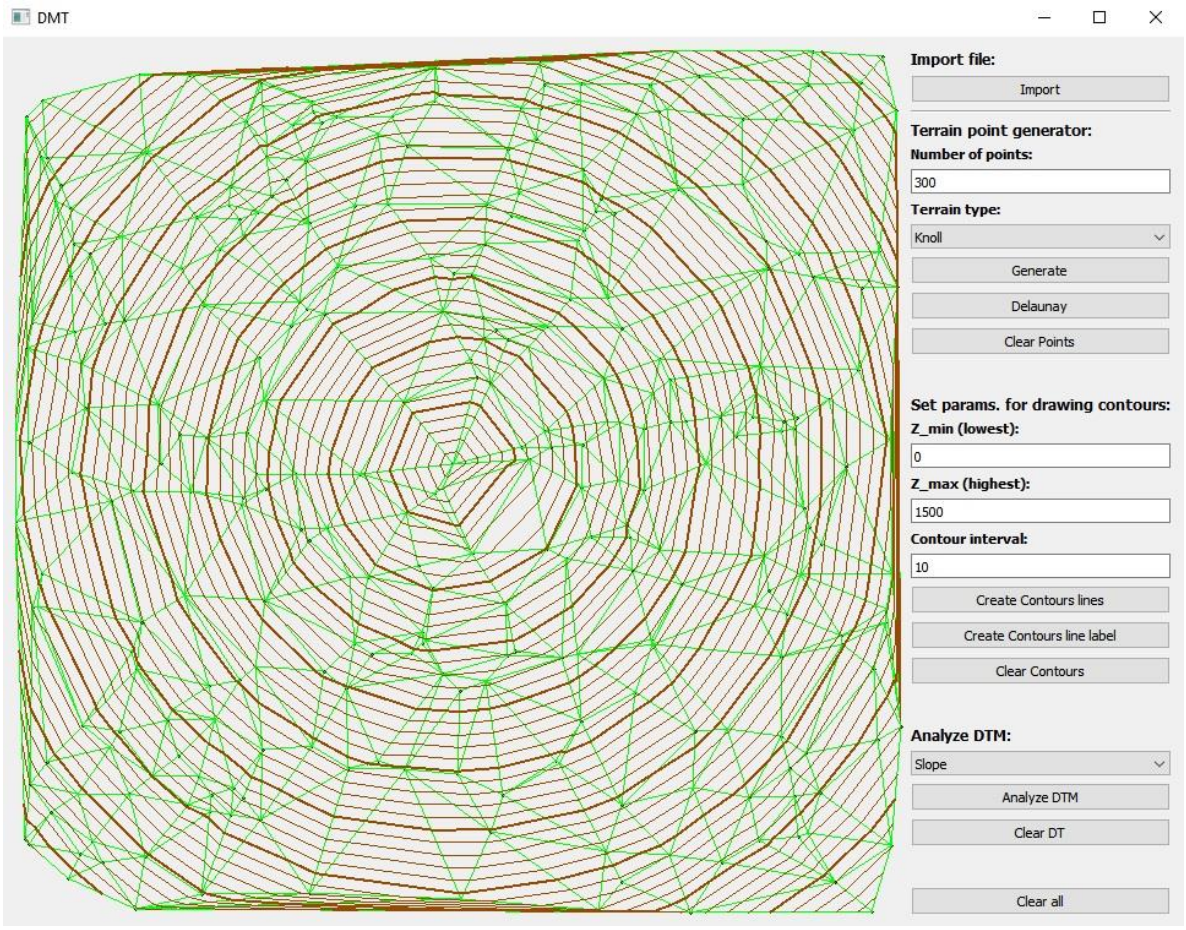


Obrázek 5: Nevhodná konfigurace bodů



Obrázek 6: Terén z měřených dat doplněn o případnou povinnou hranu

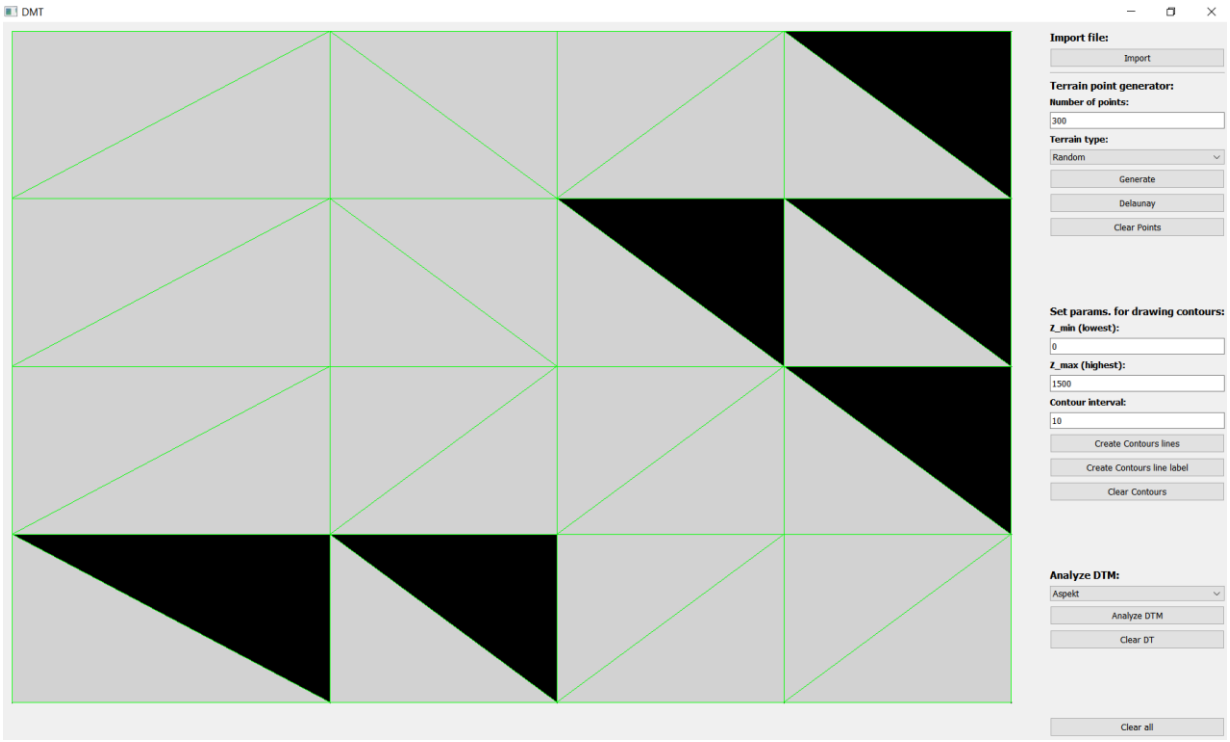
Například optimální řešení dává aplikace v případě, že je vygenerována kupa. V tomto případě vygenerované vrstevnice působí přirozeně a celkem odpovídají skutečnému terénu *(obr.6)*. Na okrajích terénního tvaru je však patrné husté vykreslení vrstevnic. To by bylo vhodné nahradit např. terénními šrafami, pokud je v daném místě velký spád.



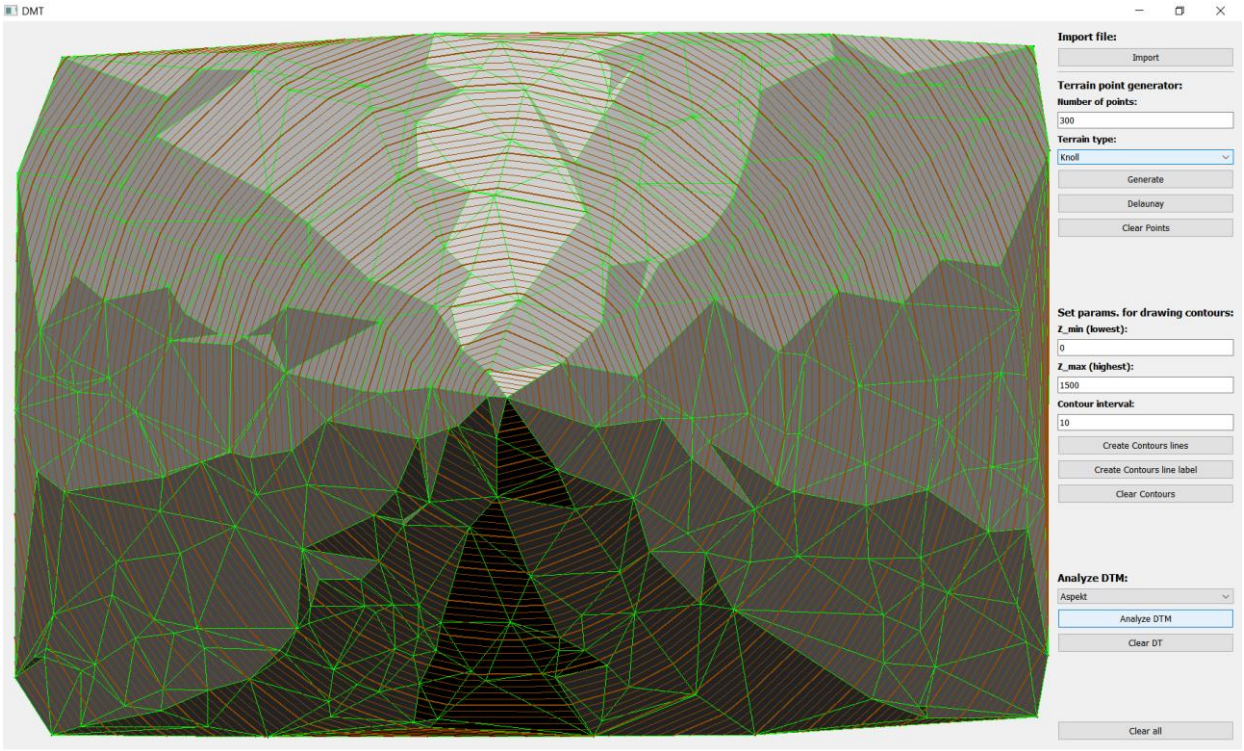
Obrázek 7: Kupa – vrstevnice

Jako další z výčtů pro neoptimální řešení je zde uveden příklad pro určení expozice v rovinném terénu (body na gridu). Podle vykreslené expozice se zde terén jeví jako nerovinný, přestože všechny body na gridu mají přidělenou stejnou výšku *(obr.8)*. Pro příklad je zde uvedena i expozice na kupě, kde výsledek vypadá až na pár detailů vcelku věrohodně *(obr.9)*. K vykreslení expozice by bylo vhodné doplnit i legendu jednotlivých barev.

Dalším nedostatkem je neznalost souřadnicového systému, ve kterém jsou body vykresleny. Body importované ze souboru *points.txt* jsou v S – JTSK, což ale není z vykreslení patrné. Jako řešení se nabízí transformace bodů do vhodného souřadnicového systému vhodného pro vykreslení a přizpůsobení velikosti kreslícího plátna. Toto však nebyla náplň této úlohy, proto se řešením této problematiky nezabýváme.



Obrázek 8: Body v mřížce se stejnou výškou při funkci Aspect



Obrázek 9: Kupa – expozice

Pro vykreslení vrstevnic byla použita lineární interpolace, ale vhodnější by bylo použít morfologickou interpolaci. Ta totiž předpokládá plynulou změnu spádu terénu. Bohužel pro ni není definován přesný postup práce algoritmu, proto není možné ji implementovat.

8 Závěr

Byla vytvořena aplikace, která umožňuje vygenerovat Delaunay triangulaci a vrstevnice. K vrstevnicím je možné vygenerovat popis, který však není kartograficky korektní. Aplikace dále umožňuje analyzovat terén pomocí sklonu a orientace svahu. Uživatel si může generovat množiny bodů, kde rozložení bodů reprezentují různé terénní tvary (kupa, hřbet, údolí), také je možné vygenerovat body náhodně. Další možností je vstupní množinu bodů nahrán ze souboru *.txt či určit myší.

V Praze 18.12. 2020

Oprava 5.1.2021

Frommeltová

Hnilicová

Seznam obrázku