

České vysoké učení technické v Praze  
Fakulta stavební  
Katedra geomatiky



Cvičení 3

### Úloha č. 3: Digitální model terénu a jeho analýzy

*Bc. Petr Poskočil a Bc. Marek Fáber*

Vyučující: doc. Ing. Tomáš Bayer, Ph.D.

Studijní program: Geodézie a kartografie, Navazující magisterský

Obor: Geomatika

13. ledna 2020

# Obsah

<b>1</b>	<b>Zadání</b>	<b>1</b>
1.1	Údaje o úlohách . . . . .	2
<b>2</b>	<b>Popis problému</b>	<b>3</b>
<b>3</b>	<b>Popis použitych algoritmů</b>	<b>4</b>
3.1	Delaunay triangulace . . . . .	4
3.1.1	Implementace Delaunay triangulace . . . . .	4
3.2	Vrstevnice . . . . .	5
3.2.1	Implementace metody . . . . .	5
3.3	Sklon terénu . . . . .	5
3.3.1	Implementace metody . . . . .	6
3.4	Expozice terénu . . . . .	6
3.4.1	Implementace metody . . . . .	6
3.5	Problematické situace . . . . .	7
<b>4</b>	<b>Vstup a výstup aplikace</b>	<b>8</b>
4.1	Vstup . . . . .	8
4.2	Výstup . . . . .	8
4.2.1	Zobrazení reálných dat . . . . .	9
4.2.2	Zobrazení sedla . . . . .	12
4.2.3	Zobrazení údolí . . . . .	15
4.2.4	Zobrazení kupy . . . . .	18
<b>5</b>	<b>Dokumentace tříd a jejich metod</b>	<b>21</b>
5.1	Algorithms . . . . .	21
5.2	Draw . . . . .	22
5.3	Edge . . . . .	23
5.4	QPoint3d . . . . .	24
5.5	sortbyx . . . . .	24
5.6	Triangle . . . . .	24
5.7	Widget . . . . .	25
<b>6</b>	<b>Závěr</b>	<b>27</b>
6.1	Neřešené problémy a náměty . . . . .	27

<b>Literatura</b>	<b>28</b>
<b>A Zdrojový kód aplikace</b>	<b>29</b>
<b>B Aplikace - binární soubor</b>	<b>30</b>
<b>C Výstupní data</b>	<b>31</b>

# Kapitola 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. navrhnete algoritmus pro generování syntetických vstupních dat představujících významné terénní tvary (kupa, údolí, spočinek, hrbet, ...).

Vstupní množiny bodů včetně níže uvedených výstupů vhodně vizualizujte. Grafické rozhraní realizujte s využitím framework 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, hrbet, ...).	+10b
3D vizualizace terénu s využitím promítání.	+10b
Barevná hypsometrie.	+5b
<b>Max celkem:</b>	<b>65b</b>

Obrázek 1.1: Zadání cvičení č. 3 [2]

## 1.1 Údaje o úlohách

Povinná část úlohy je rozdělena na dva body:

- *Delaunay triangulace, polyedrický model terénu,*
- *Konstrukce vrstevnic, analýza sklonu a expozice.* [2]

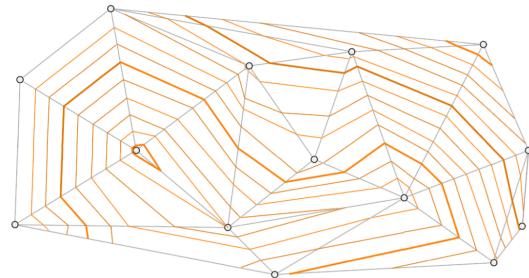
Z bonusové části byly zpracovány úlohy:

- *Výběr barevných stupnic při vizualizaci sklonu a expozice,*
- *Automatický popis vrstevnic,*
- *Automatický popis vrstevnic respektující kartografické zásady (orientace, vhodné rozložení),*
- *Algoritmus pro automatické generování terénních tvarů (kupa, údolí, spočinek, hřbet,...),*
- *Barevná hypsometrie.* [2]

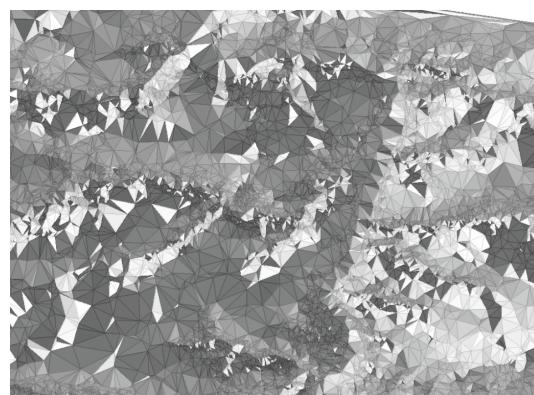
## Kapitola 2

### Popis problému

Cílem úlohy je vytvořit aplikaci, ve které budeme generovat množinu  $P$  s body  $p_i(x_i, y_i, z_i)$ . Úkolem této aplikace bude vytvořit trojúhelníkovou síť nad body této množiny. K vytvoření trojúhelníkové sítě se použije Delaunay triangulace (DT). Dále vytvořit vrstevnice a určit sklon a expozici trojúhelníků DT.



Obrázek 2.1: Vrstevnice nad trojúhelníkovou sítí [1]



Obrázek 2.2: Orientace sklonu jednotlivých trojúhelníků [1]

## Kapitola 3

# Popis použitých algoritmů

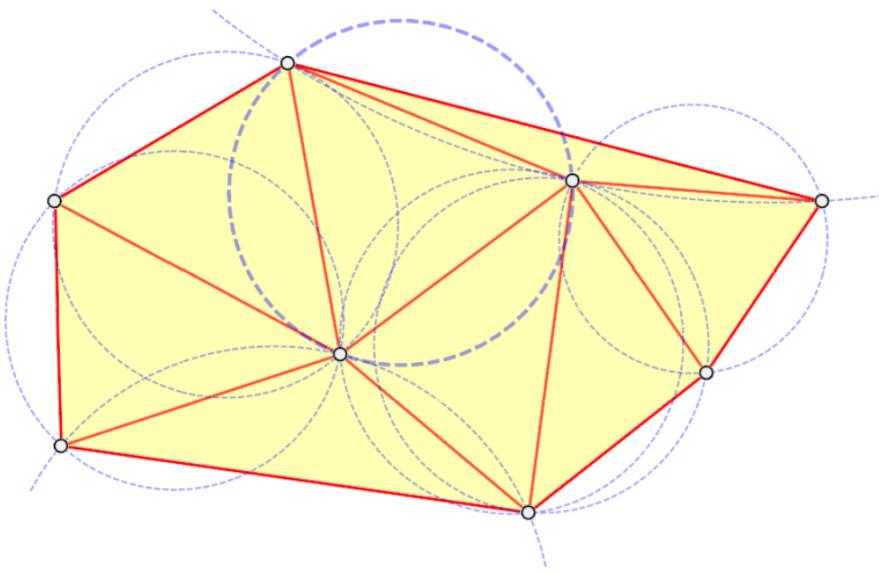
Pro konstrukci trojúhelníkové sítě lze použít mnoho algoritmů, jako jsou například *Greedy triangulace*, *Minimum Weight Triangulation*, *Constrained triangulation*. Pro tuto úlohu byl zvolen nejčastěji používaný algoritmus *Delaunay triangulation*.

### 3.1 Delaunay triangulace

Jedná se o nejčastěji využívanou metodu triangulace při tvorbě digitálního modelu terénu. Triangulace se provádí hledáním bodu, který bodům již vzniklé hrany vytvoří trojúhelník jehož opsaná kružnice je minimální. Tato metoda se nazývá inkrementální konstrukce. Když se našel takový bod, vytvoří se nové hrany, které utvoří trojúhelník. Hrany, ke kterým dosud nebyl nalezen třetí bod se ukládají do struktury Active Edge List (AEL), pokud tam již není s opačnou orientací. Po nalezení třetího bodu se ze struktury odstraní. Tento algoritmus se provádí dokud není struktura AEL prázdná.

#### 3.1.1 Implementace Delaunay triangulace

1. Nalezení pivota  $q$ :  $q = \min_{\forall p_i \in P} (X_i)$
2. Hledání nejbližšího bodu:  $\|p_1 - q\|_2 = \min$
3. Přidání první hrany:  $e_1 = (q, p_1)$
4. Hledání Delaunay bodu:  $\underline{p} = \operatorname{argmin}_{\forall P_i \in \sigma_L(e_1)} r'(k_i); k_i = (a, b, p_i); e_1 = (a, b)$   
Při nenalezení změnit orientaci hrany:  $\underline{p} : e_1 \leftarrow (p_1, q)$ ; go to 4.
5. Vytvoření zbývajících hran:  $e_2 = (p_1, \underline{p}); e_3 = (\underline{p}, q)$
6. Přidání hran do DT:  $DT \leftarrow e_1; DT \leftarrow e_2; DT \leftarrow e_3$
7. Přidání hran do AEL:  $AEL \leftarrow e_1; AEL \leftarrow e_2; AEL \leftarrow e_3$
8. Dokud není AEL prázdný:  
Hledání Delaunay bodu k hraně z AEL (viz 4)  
Vytvoření zbývajících hran:  $e_2 = (p_1, \underline{p}); e_3 = (\underline{p}, q)$   
Přidej hranu do AEL, pokud tam není.



Obrázek 3.1: Ukázka Delaunay triangulace s opsanými kružnicemi [1]

## 3.2 Vrstevnice

Vrstevnice jsou průsečnice terénu s vodorovnou rovinou, jejichž výška je bezezbytku dělitelná zvoleným rozestupem vrstevnic. Obvykle každá pátá vrstevnice je zobrazena tučnou čárou a je popsána hodnotou její výšky. Popis souřadnic se provádí rovnoměrně po celém poli směrem do kopce.

### 3.2.1 Implementace metody

1. Pro všechny hrany trojúhelníku  $t$ :  $\forall e_i \in t :$
2. Hrana náleží rovině  $z$ :  $(z - z_i) \cdot (z - z_i + 1) = 0 \rightarrow e_i \in \rho$
3. Hrana nenáleží rovině  $z$ :  $(z - z_i) \cdot (z - z_i + 1) < 0 \rightarrow e_i \notin \rho$
4. Hrana je průnikem roviny vrstevnice  $z$ :  $(z - z_i) \cdot (z - z_i + 1) < 0 \rightarrow e_i \cap \rho$   
Výpočet polohových souřadnic:  

$$x = [(x_2 - x_1)/(z_2 - z_1)] \cdot (z - z_1) + x_1$$

$$y = [(y_2 - y_1)/(z_2 - z_1)] \cdot (z - z_1) + y_1$$
Vytvořit hranu tvořící vrstevnici.

## 3.3 Sklon terénu

Sklon terénu je úhel mezi normálovým vektorem roviny trojúhelníku a normálovým vektorem  $(0, 0, 1)$ .

### 3.3.1 Implementace metody

1. Pro všechny trojúhelníky:  $\forall t_i \in DT :$

Výpočet normálového vektoru roviny trojúhelníku:

$$n_t = (u_y \cdot v_z - u_z \cdot v_y)^2 - (u_x \cdot v_z - u_z \cdot v_x)^2 + (u_x \cdot v_y - u_y \cdot v_x)^2$$

Kde:

$$u_x = \Delta x_2, x_1; u_y = \Delta y_2, y_1; u_z = \Delta z_2, z_1$$

$$v_x = \Delta x_2, x_3; v_y = \Delta y_2, y_3; v_z = \Delta z_2, z_3$$

2. Výpočet sklonu:  $\phi = \arccos(n_z / |n_t|)$

## 3.4 Expozice terénu

Expozice terénu je azimut k průmětu normálového vektoru roviny trojúhelníku do roviny x, y.

### 3.4.1 Implementace metody

1. Pro všechny trojúhelníky:  $\forall t_i \in DT :$

Výpočet x a y části normálového vektoru:

$$n_x = (u_y \cdot v_z - u_z \cdot v_y)$$

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

Kde:

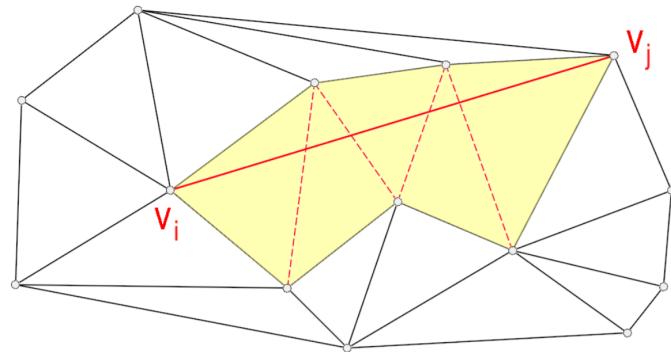
$$u_x = \Delta x_2, x_1; u_y = \Delta y_2, y_1; u_z = \Delta z_2, z_1$$

$$v_x = \Delta x_2, x_3; v_y = \Delta y_2, y_3; v_z = \Delta z_2, z_3$$

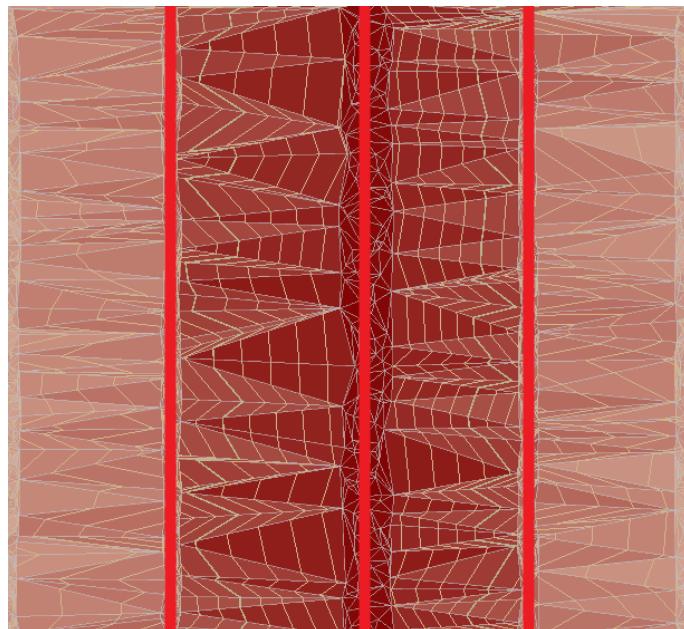
2. Výpočet sklonu:  $A = \text{atan2}(n_z / n_y)$

### 3.5 Problematické situace

Delaunay triangulace není vhodná pro případy terénu, kde jsou ostré hrany (opěrné zdi, silnice, chodníky). Takové situace by bylo potřeba řešit předzpracováním, kdy by se tyto hrany označili jako povinné. Naopak je algoritmus vhodný pro terén, který není příliš členitý.



Obrázek 3.2: Mezi body  $v_i$  a  $v_j$  se nachází povinná hrana [1]



Obrázek 3.3: Přibližný návrh pro povinné hrany pro data znázorňující údolí

# Kapitola 4

## Vstup a výstup aplikace

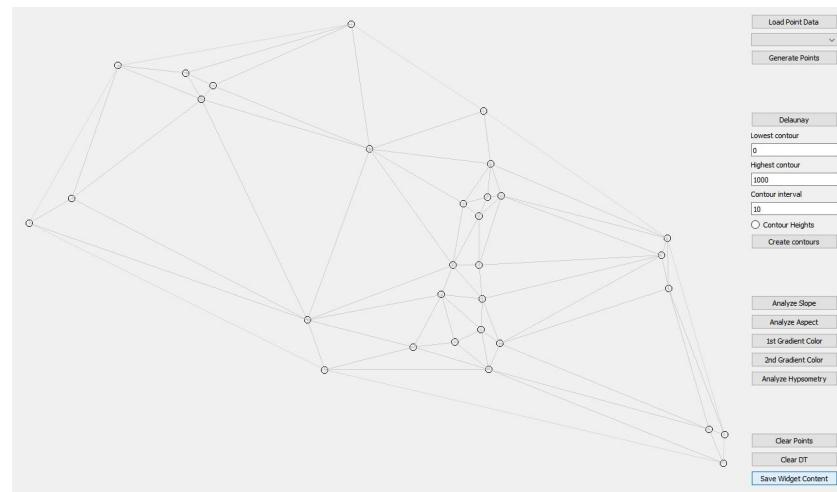
### 4.1 Vstup

Vstupními daty je množina bodů, kterou uživatel může vytvořit kliknutím myši do grafické části aplikace. Druhá možnost jak vytvořit body je automatikou generací, podle zvoleného typu terénního tvaru. Lze zvolit z tvarů kupa, udolí, hřbet a sedlo. Do aplikace lze importovat i skutečná data.

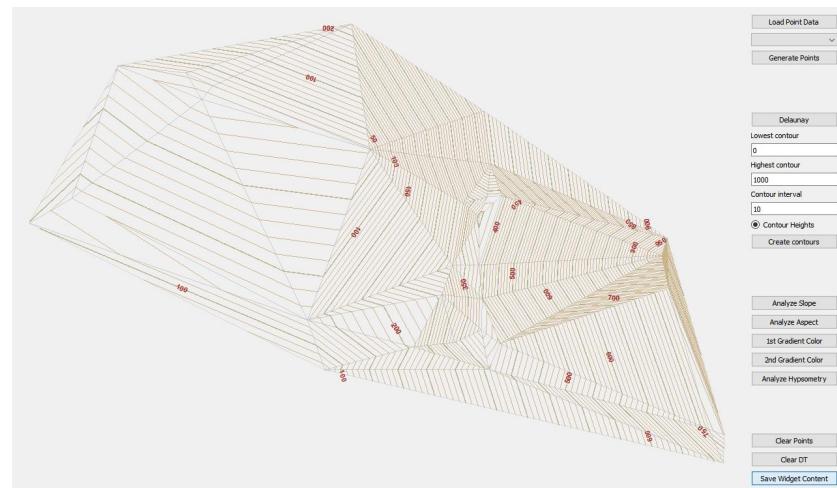
### 4.2 Výstup

Výstupem úlohy je grafická aplikace, která z vygenerovaných bodů vytvoří trojúhelníkovou síť, vrstevnice, analýzu sklonu a expozice a barevnou hypsometrii.

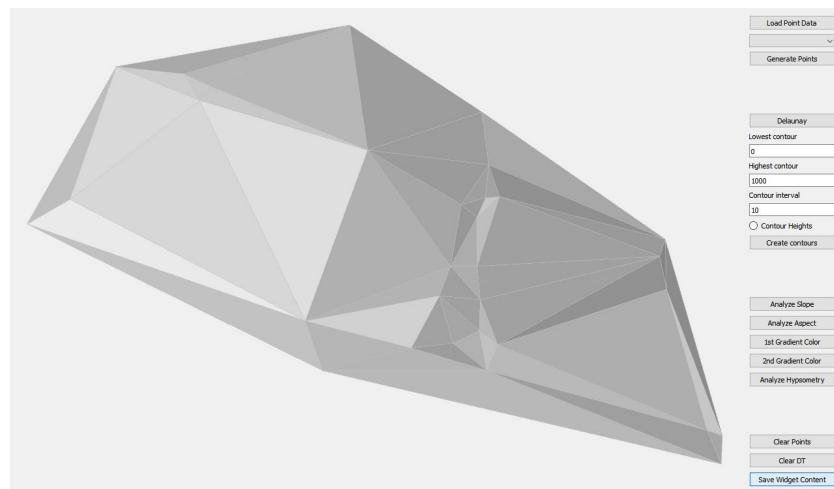
#### 4.2.1 Zobrazení reálných dat



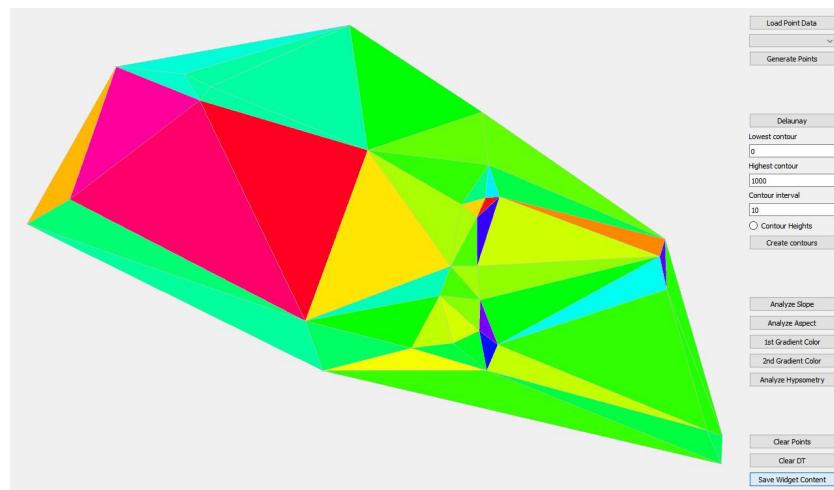
Obrázek 4.1: Body a trojúhelníková síť



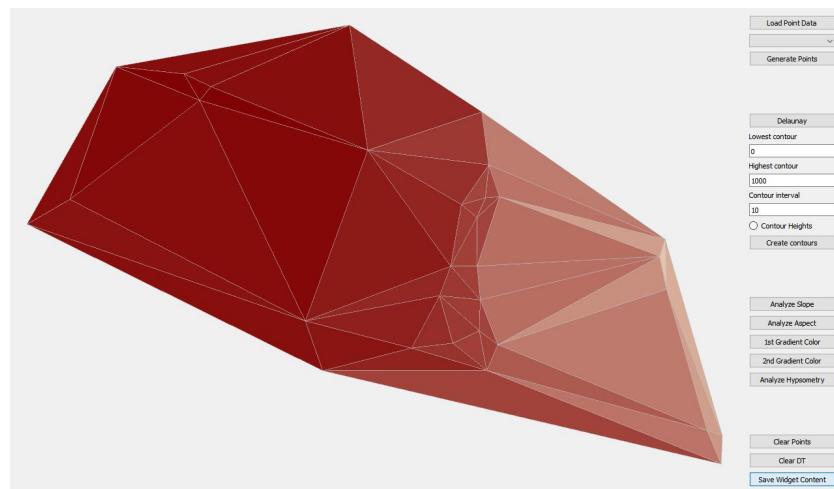
Obrázek 4.2: Vrstevnice



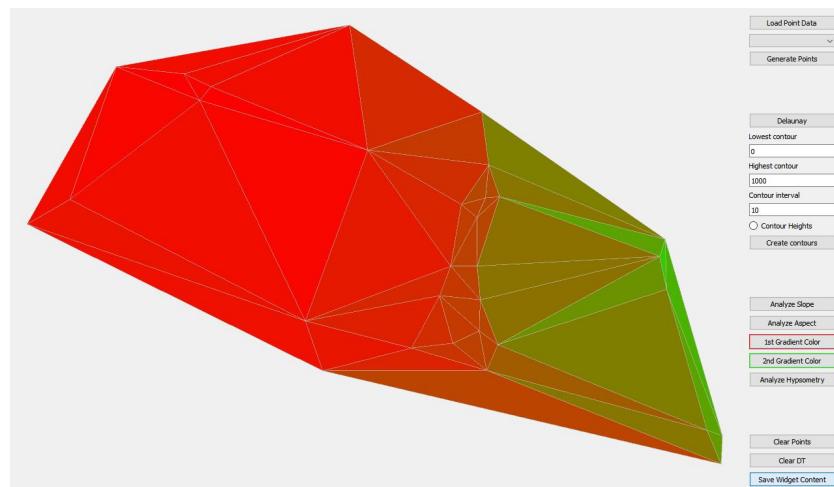
Obrázek 4.3: Analýza sklonu



Obrázek 4.4: Analýza expozice

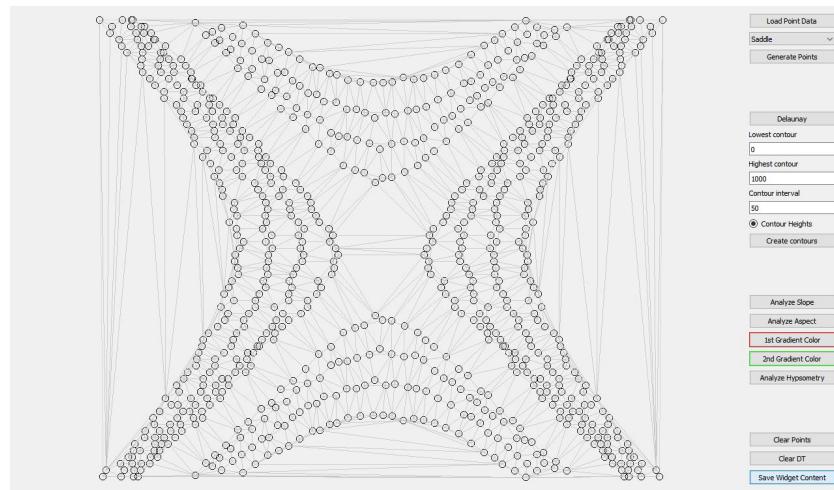


Obrázek 4.5: Hypsometrie pro barvy hnědá a světlehnědá



Obrázek 4.6: Hláška ukočení testu trvání generování a vykreslení algoritmu

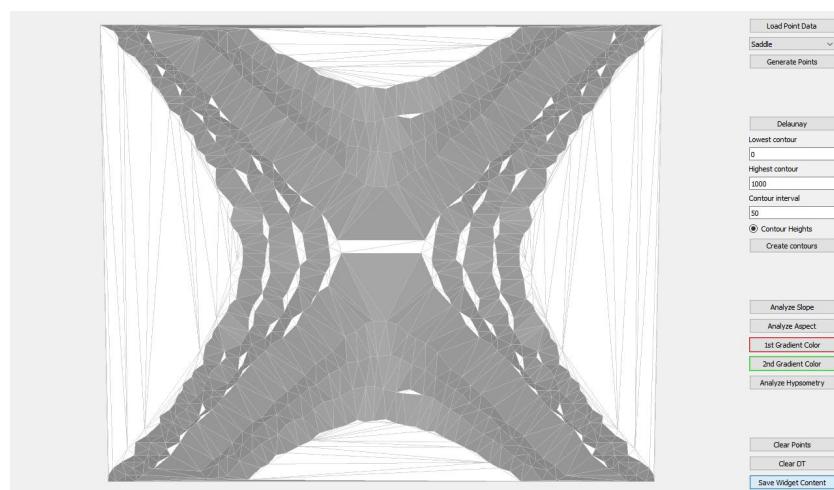
#### 4.2.2 Zobrazení sedla



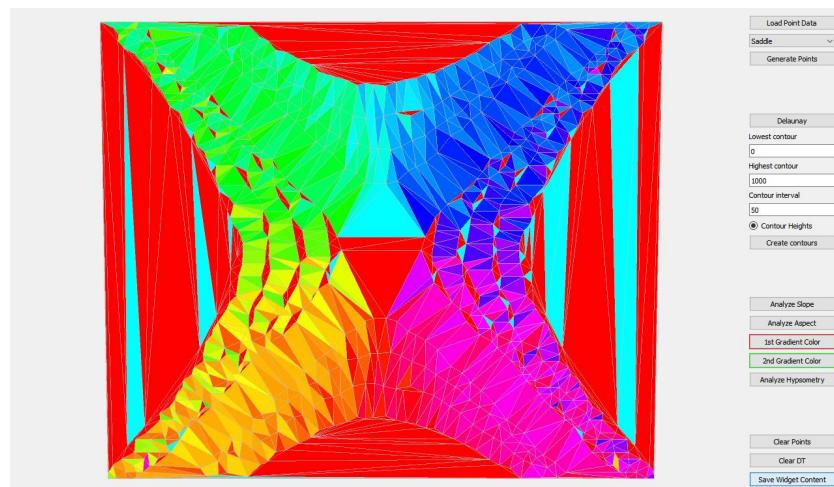
Obrázek 4.7: Body a trojúhelníková síť



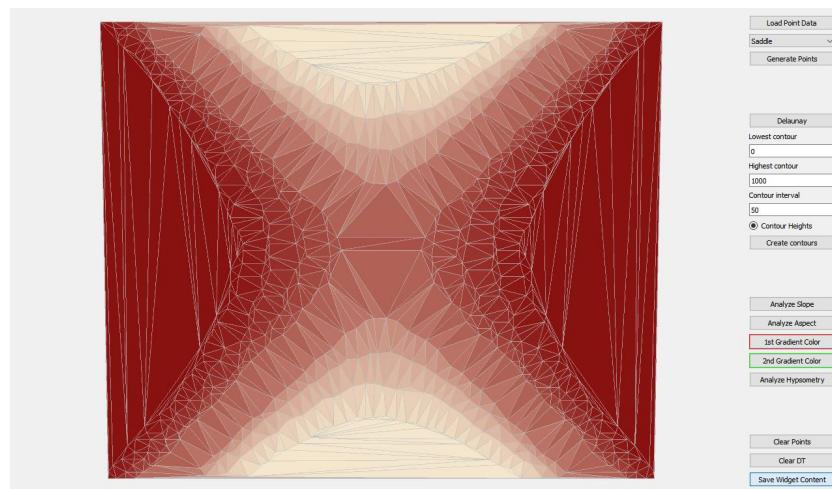
Obrázek 4.8: Vrstevnice



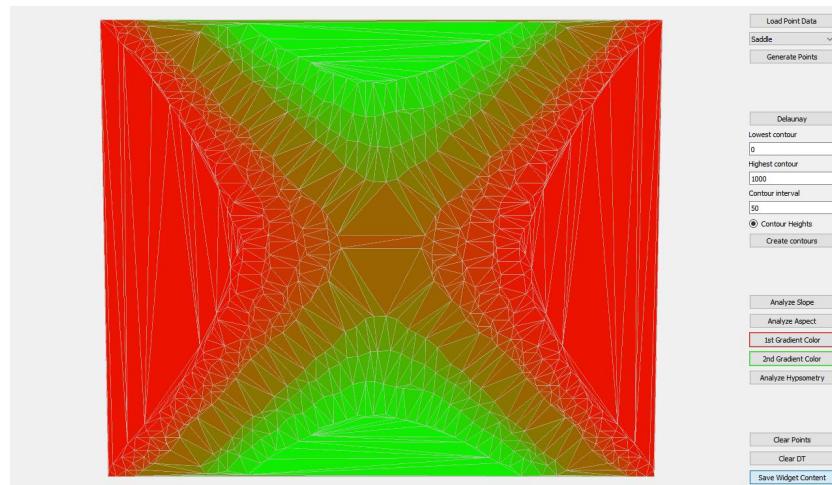
Obrázek 4.9: Analýza sklonu



Obrázek 4.10: Analýza expozice

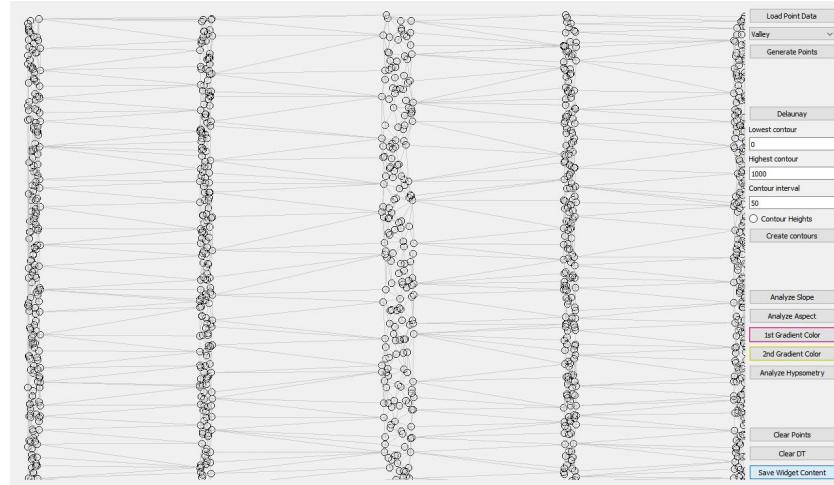


Obrázek 4.11: Hypsometrie pro barvy hnědá a světlehnědá

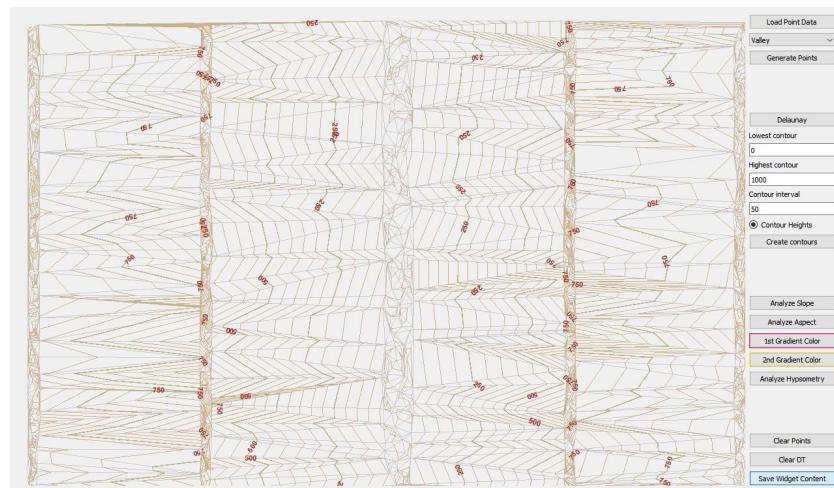


Obrázek 4.12: Hláška ukočení testu trvání generování a vykreslení algoritmu

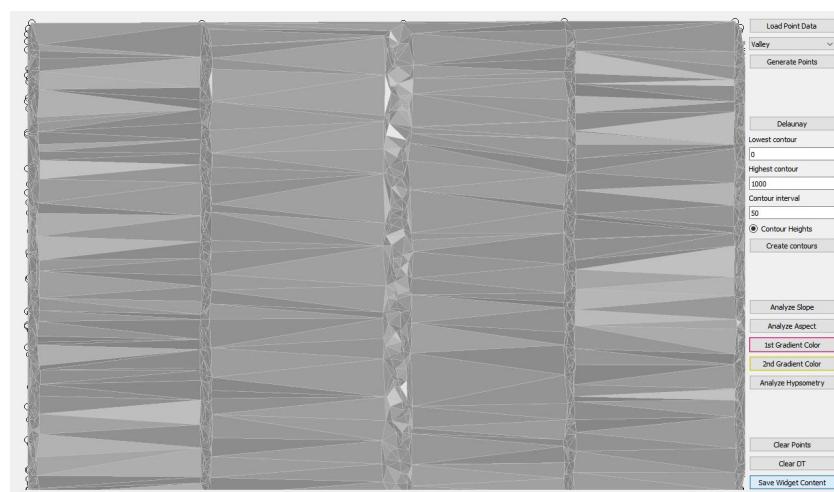
#### 4.2.3 Zobrazení údolí



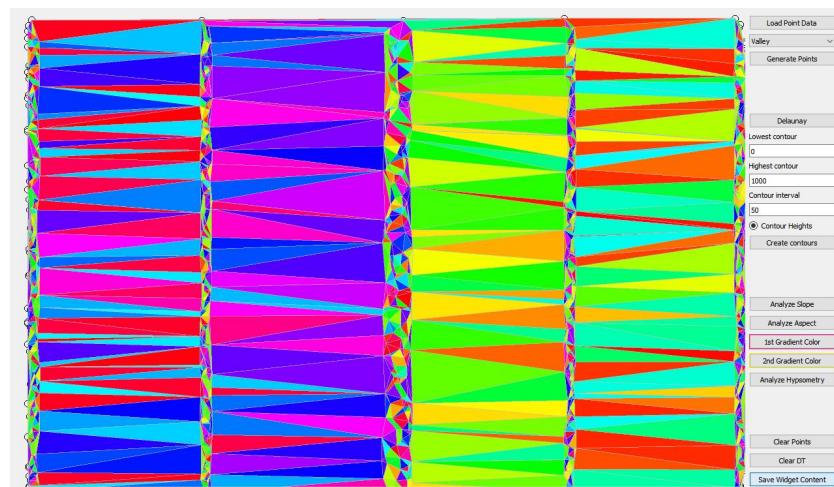
Obrázek 4.13: Body a trojúhelníková síť



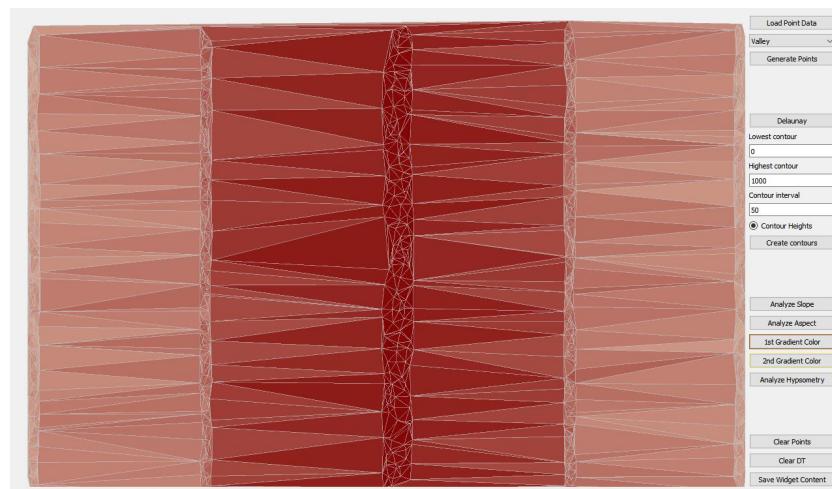
Obrázek 4.14: Vrstevnice



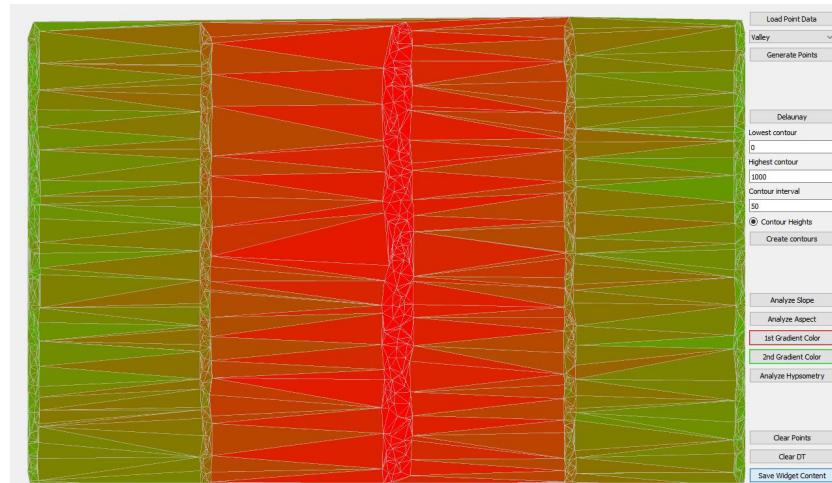
Obrázek 4.15: Analýza sklonu



Obrázek 4.16: Analýza expozice

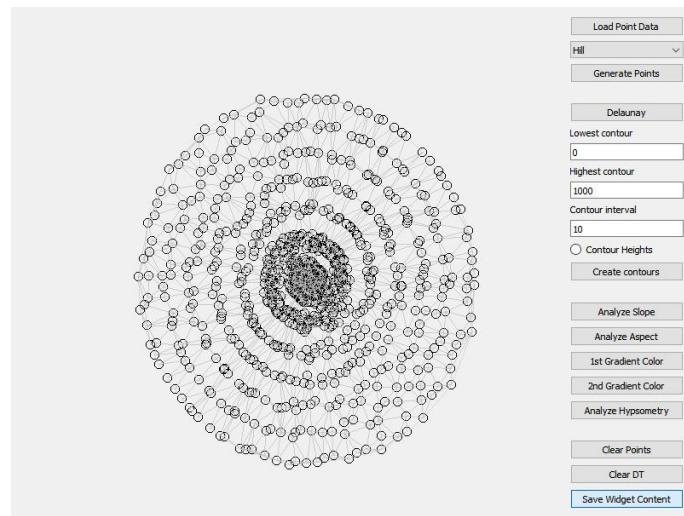


Obrázek 4.17: Hypsometrie pro barvy hnědá a světlehnědá

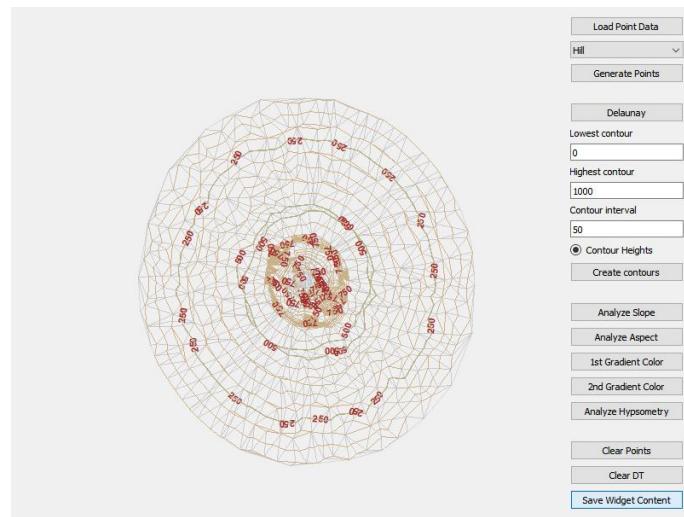


Obrázek 4.18: Hláška ukočení testu trvání generování a vykreslení algoritmu

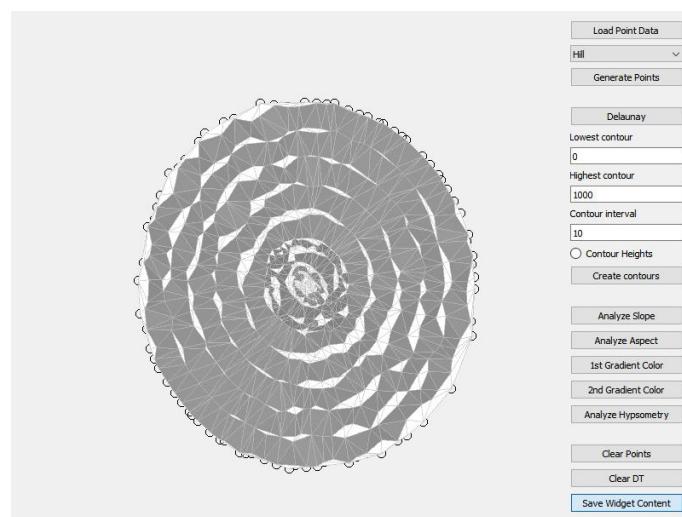
#### 4.2.4 Zobrazení kupy



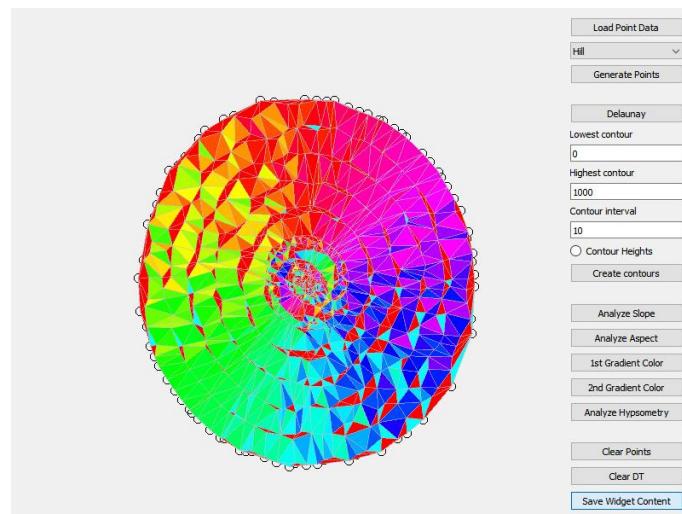
Obrázek 4.19: Body a trojúhelníková síť



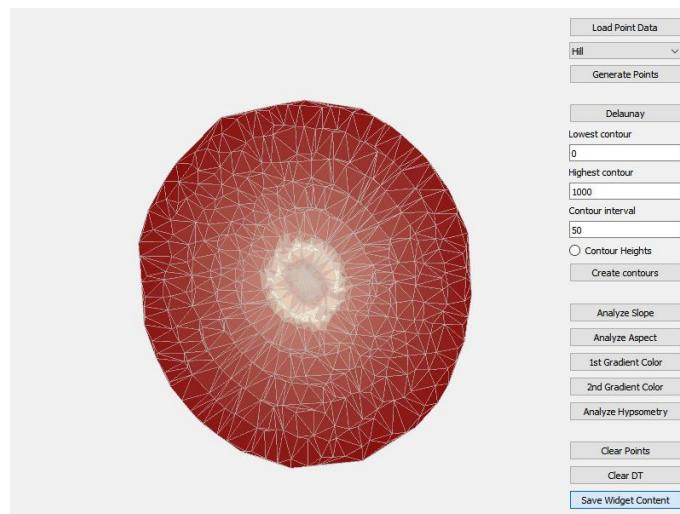
Obrázek 4.20: Vrstevnice



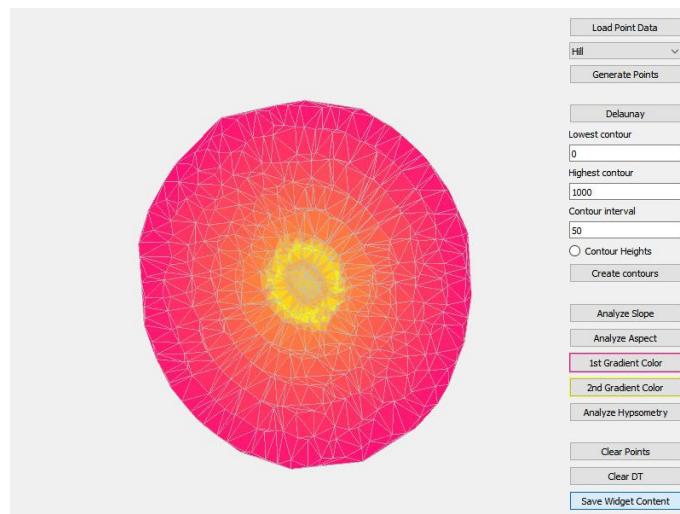
Obrázek 4.21: Analýza sklonu



Obrázek 4.22: Analýza expozice



Obrázek 4.23: Hypsometrie pro barvy hnědá a světlehnědá



Obrázek 4.24: Hláška ukočení testu trvání generování a vykreslení algoritmu

# Kapitola 5

## Dokumentace tříd a jejich metod

### 5.1 Algorithms

- *int getPointLinePosition(QPoint &q, QPoint &p1, QPoint &p2)*

Tato funkce určuje pozici bodu vůči linii. Na vstupu funkce je určovaný bod a dva body přímky "a" a "b". Ze dvou bodů přímky můžeme určit determinant, jehož hodnotu porovnáváme se zvolenou minimální hodnotou "eps". Výstupem funkce je celé číslo, které nabývá hodnot 1 (pod leží v levé polovině), 0 (bod leží v pravé polovině) a -1 (bod leží na hraně).

- *double get2LinesAngle(QPoint &p1, QPoint &p2, QPoint &p3, QPoint &p4)*

Tato funkce počítá úhel mezi dvěma hranami. Na vstupu jsou 4 body, které určují dvě přímky. Výstupem je velikost úhlu ve stupních typu double.

- *double getLength2Points(QPoint q, QPoint p)*

Tato funkce počítá vzdálenost mezi dvěma body. Na vstupu funkce jsou dva body a výstupem je vzdálenost typu double.

- *double getCircleRadius(QPoint3D &p1, QPoint3D &p2, QPoint3D &p3, QPoint3D &c)*

Tato funkce určuje poloměr kružnice opsané trojúhelníku. Na vstupu funkce jsou tři vrcholy trojúhelníku. Výstupem funkce je kladné číslo typu double.

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

Tato funkce slouží k nalezení nejbližšího bodu k bodu p.

- *double distance2Points(QPoint3D &p1, QPoint3D &p2)*

Tato funkce počítá vzdálenost mezi dvěma body.

- *int getDelaunayPoint(QPoint3D &s, QPoint3D &e, std::vector<QPoint3D> &points)*

Tato funkce slouží k nalezení bodu, který splňuje Delaunayho podmínu.

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

Metoda vytvářející vektor hran trojúhelníků na triangulaci nad množinou bodů.

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

Metoda počítající průsečíky hran s rovinou definovanou souřadnicí Z.

- `std::vector<Edge> createContourLines(std::vector<Edge> &dt, double z_min, double z_max, double dz)`  
Metoda vytvářející vektor hran tvořících vrstevnice nad trojúhelníkovou sítí.
- `double calculateSlope(QPoint3D &p1, QPoint3D &p2, QPoint3D &p3)`  
Funkce počítající sklon trojúhelníků.
- `int calculateAspect(QPoint3D &p1, QPoint3D &p2, QPoint3D &p3)`  
Funkce počítající expozici trojúhelníků.
- `std::vector<Triangle> analyzeDTM(std::vector<Edge> &dt)`  
Metoda k vytváření trojúhelníků, sklonu a expozici.

## 5.2 Draw

- `void setPoints(std::vector<QPoint> &points_)`  
Metoda, která převádí body z grafického okna.
- `void mousePressEvent(QMouseEvent *event)`  
Metoda, která slouží ke vkládání bodů po kliknutí do grafického okna.
- `std::vector<Edge> getDt()`  
Metoda, která umožní převést Delaunay triangulaci z vykreslovacího okna.
- `void setDt(std::vector<Edge> &dt_)`  
Metoda, která umožní vykreslit Delaunay triangulaci.
- `void setContours(std::vector<Edge> &contours_)`  
Metoda, která umožní vykreslit vrstevnice.
- `void setMainContours(std::vector<Edge> &mainContours_)`  
Metoda, která umožní vykreslit vrstevnice.
- `void paintEvent(QPaintEvent *event)`  
Metoda, která vykresluje vygenerované body, trojúhelníkovou síť a vrstevnice.
- `void mousePressEvent(QMouseEvent *event)`  
Metoda, která umožní vytvoření bodu po kliknutí do aplikace. K bodu vygeneruje náhodnou souřadnici Z.
- `std::vector<QPoint3D> getPoints()`  
Metoda, která umožní převést body z vykreslovacího okna.
- `void clearPoints()`  
Slouží k vyčištění grafického okna od všech bodů.
- `void clearDT()`  
Slouží k vyčištění grafického okna od trojúhelníkové sítě.

- *void clearDTM()*  
Slouží k vyčištění grafického okna od trojúhelníků sklonu a expozice.
- *void clearContours()*  
Slouží k vyčištění grafického okna od vrstevnic.
- *int getDtSize()*  
Metoda, která určuje počet vrstevnic.
- *void setDTM(std::vector<Triangle> & dtm\_)*  
Metoda, která převádí trojúhelníkovou síť do grafického okna.
- *static void importPoints(std::string &path, std::vector<QPoint3D> &points, QSizeF &canvas\_size, double &z\_min, double &z\_max)*  
Metoda, která umožní importovat vlastní body.
- *void drawRotatedText(QPainter\* painter, float degrees, double x, double y, QString &text)*  
Metoda, které vykresluje popis vrstevnic podle kartografických pravidel.
- *std::vector<int> generateColor(int &coefficient)*  
Metoda, která generuje barevnost hypsometrie modelu.
- *static std::vector<QPoint3D> generateTerrainHill(int n\_points, QSizeF &canvas\_size, double &z\_min, double &z\_max)*  
Metoda, která generuje množinu bodů ve tvaru kupy.

### 5.3 Edge

- *Edge(QPoint3D &start, QPoint3D &end)*  
Třída odvozená od třídy QPointF, která slouží k uložení bodu s výškou.
- *QPoint3D &getStart()*  
Metoda, která vrátí počáteční bod hrany.
- *void setStart(QPoint3D &s)*  
Metoda, která nastaví počáteční bod hrany.
- *QPoint3D &getEnd()*  
Metoda, která vrátí koncový bod hrany.
- *void setEnd(QPoint3D &e)*  
Metoda, která nastaví koncový bod hrany.
- *void changeOrientation()*  
Metoda, která změní orientaci hrany.

## 5.4 QPoint3d

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

Třída odvozená od třídy QPointF, která slouží k uložení bodu s výškou.

- *double getZ()*

Metoda, která vrátí výšku bodu.

- *void setZ(double z)*

Metoda, která nastaví výšku bodu.

## 5.5 sortbyx

Třída sortbyx slouží k setřídění bodů podle X-ové souřadnice.

- *bool operator() (QPoint &p1, QPoint &p2)*

Z dvojice bodů vrátí ten s větší X-ovou souřadnicí.

## 5.6 Triangle

- *Triangle(QPoint3D p1\_, QPoint3D p2\_, QPoint3D p3\_, double slope\_, int aspect\_)*

Třída odvozená od třídy QPointF, která slouží k uložení bodu s výškou.

- *QPoint3D getP1()*

Metoda, která vrátí první bod trojúhelníku.

- *QPoint3D getP2()*

Metoda, která vrátí druhý bod trojúhelníku.

- *QPoint3D getP3()*

Metoda, která vrátí třetí bod trojúhelníku.

- *QPoint3D getSlope()*

Metoda, která vrátí sklon trojúhelníku.

- *QPoint3D getAspect()*

Metoda, která vrátí expozici trojúhelníku.

- *void setP1(QPoint3D &p1\_)*

Metoda, která nastaví první bod trojúhelníku.

- *void setP2(QPoint3D &p2\_)*

Metoda, která nastaví druhý bod trojúhelníku.

- *void setP3(QPoint3D &p3\_)*

Metoda, která nastaví třetí bod trojúhelníku.

- *void setSlope(double slope\_)*

Metoda, která nastaví sklon trojúhelníku.

- *void setAspect(int aspect\_)*  
Metoda, která nastaví expozici trojúhelníku.

## 5.7 Widget

- *void on\_pushButton\_clicked*  
Spustí výpočet algoritmu pro Delaunay triangulaci. V aplikaci se jedná o tlačítko Delaunay.
- *void on\_pushButton\_2\_clicked*  
Smaže množinu bodů. V aplikaci se jedná o tlačítko Clear points.
- *void on\_pushButton\_3\_clicked*  
Smaže trojúhelníkovou síť. V aplikaci se jedná o tlačítko Clear DT.
- *void on\_pushButton\_4\_clicked*  
Vygeneruje vrstevnice. V aplikaci se jedná o tlačítko Create contours.
- *void on\_pushButton\_5\_clicked*  
Analyzuje sklon. V aplikaci se jedná o tlačítko Analyze Slope.
- *void on\_pushButton\_6\_clicked*  
Analyzuje expozici. V aplikaci se jedná o tlačítko Analyze Aspect.
- *void on\_pushButton\_7\_clicked*  
Vytváří hypsometrii. V aplikaci se jedná o tlačítko Analyze Hypsometry.
- *void on\_pushButton\_8\_clicked*  
Vygeneruje množinu bodů. V aplikaci se jedná o tlačítko Generate points.
- *void on\_lineEdit\_editingFinished*  
Umožňuje nastavit vrstevnici s minimální výškou. V aplikaci se jedná o okno Lowest contour. Implicitně je nastaveno na 0.
- *void on\_lineEdit\_2\_editingFinished*  
Umožňuje nastavit vrstevnici s maximální výškou. V aplikaci se jedná o okno Highest contour. Implicitně je nastaveno na 1000.
- *void on\_lineEdit\_3\_editingFinished*  
Umožňuje určit interval vrstevnic. V aplikaci se jedná o okno Contour interval. Implicitně je nastaveno na 10.
- *void on\_load\_clicked*  
Umožňuje exportovat do aplikace vlastní data. V aplikaci se jedná o tlačítko Load Point Data.
- *void on\_saveImage\_clicked*  
Umožní uložit aktuální vzhled aplikace. V aplikaci se jedná o tlačítko Save Widget Content.

- *void onRadioButton\_clicked(bool checked)*

Umožňuje volit popis vrstevnic. V aplikaci se jedná o volbu Contour Heights.

# Kapitola 6

## Závěr

Ve vytvořené aplikaci lze vytvořit ručně, vygenerovat třemi různými způsoby nebo importovat množinu bodů a vytvořit Delaunay triangulaci, vrstevnice a hypsometrický model. Vrstevnice lze popsat podle kartografických pravidel.

### 6.1 Neřešené problémy a náměty

- Z časové tísně nebyly v úloze vyřešeny některé bonusové úlohy. Nebyly řešeny triangulace nekonvexní oblasti zadáné polygonem a 3D vizualizace terénu s využitím promítání.
- Data jsou předem velikostně definována, jelikož program má předdefinovanou velikost vykreslovacího okna. V ideálním případě by bylo dobré, kdyby se načtená data sama přizpůsobila velikosti okna.
- Delaunay triangulace nedává dobré výsledky pro případy, kde se nachází ostré hrany. Tato problematika je více popsána v kapitole 3.5.

# Literatura

- [1] T. Bayer. 2d triangulace, dmt, přednáška č. 5 předmětu Algoritmy v digitální kartografii, 2016. <https://web.natur.cuni.cz>.
- [2] T. Bayer. Digitální model terénu a jeho analýzy, cvičení č. 3 předmětu Algoritmy v digitální kartografii, 2016. <https://web.natur.cuni.cz>.

## Příloha A

### Zdrojový kód aplikace

Zdrojový kód aplikace je dostupný z veřejného profilu *petrposkocil* na [github.com](https://github.com)

Jméno repozitáře: *ADK\_poskocil\_faber*

Podsložka: *U3*

[https://github.com/petrposkocil/ADK\\_poskocil\\_faber/U3/DTM](https://github.com/petrposkocil/ADK_poskocil_faber/U3/DTM)

#### **Obsažené soubory:**

*DTM.pro* //Qt creator project file  
*algorithms.h* //Header file  
*draw.h* //Header file  
*edge.h* //Header file  
*qpoint3d.h* //Header file  
*sortbyx.h* //Header file  
*triangle.h* //Header file  
*widget.h* //Header file  
*main.cpp* //Source code file  
*algorithms.cpp* //Source code file  
*draw.cpp* //Source code file  
*edge.cpp* //Source code file  
*qpoint3d.cpp* //Source code file  
*sortbyx.cpp* //Source code file  
*triangle.cpp* //Source code file  
*widget.cpp* //Source code file  
*widget.ui* //User interface file

## Příloha B

### Aplikace - binární soubor

Aplikace je dostupná z veřejného profilu *petrposkocil* na [github.com](https://github.com)

Jméno repozitáře: *ADK\_poskocil\_faber*

Soubor: *DTM.exe*

[https://github.com/petrposkocil/ADK\\_poskocil\\_faber/U3/DTM.exe](https://github.com/petrposkocil/ADK_poskocil_faber/U3/DTM.exe)

## Příloha C

### Výstupní data

Výstupní data jsou ve formě grafického znázornění digitálního modelu terénu nad množinou bodů.