Digitální model terénu
Algoritmy digitální kartografie a GIS

Jakub Šnopl, Dennis Dvořák, Jakub Felenda

28. ledna 2024

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.

Hodnocení:

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

Čas zpracování: 4 týdny

2 Bonusové úlohy

- Automatický popis vrstevnic

3 Použité algoritmy a matematický aparát

Vstupními daty aplikace je množina bodů P o třech souřadnicích (x, y, z) z kterých je následně generována Delaunayho triangulace DT, která je použita k tvorbě vrstevnic, analýze sklonitosti a orientace svahu (expozice).

3.1 Delaunayho triangulace

Delaunayho triangulace je matematická metoda pro rozdělení množiny bodů v rovině do nejvíce "kvalitních"trojúhelníků. Trojúhelníky v této triangulaci mají tu vlastnost, že kružnice opsaná každému z nich neobsahuje žádný bod ze vstupní množiny bodů. Dále pak maximalizuje minimální úhel (avšak neminimalizuje maximální), je lokálně i globálně optimální vůči kritériu minimálního úhlu a je jednoznačná (pokud se nenachází čtyři body na jedné kružnici). Důležitým nástrojem pro Delaunayho triangulaci je Voronoiho diagram, který vytváří oblasti, ve kterých jsou body nejbližší k jednomu konkrétnímu bodu v množině.

Algoritmus 1: DT, inkrementální konstrukce

$AEL = \{\}, DT = \{\}$
$p_1 = rand(P)$
$p_2 = argmin_{p_i \in P} ||p_1 - p_i||$
Vytvoř hrany $e = (p_1, p_2), e' = (p_2, p_1)$
$AEL \leftarrow e, AEL \leftarrow e'$
while AEL *not empty* **do**
    $e_1 = AEL.pop(), e_1 = (p_1, p_2)$
    $e'_1 = (p_2, p_1)$
    $\overline{p} = argmax_{\forall p_i \in \sigma_L(e'_1)} \angle(p_1, p_i, p_2)$
    if $\exists \overline{p}$ **then**
        $e_2 = (p_2, \overline{p}), e_3 = (\overline{p}, p_1)$
        $DT \leftarrow e'_1, DT \leftarrow e_2, DT \leftarrow e_3$
        $updateAEL(e_2, AEL), updateAEL(e_3, AEL)$
    end
end

Algoritmus 2: updateAEL (e=(a,b), AEL)

Vytvoř hranu $e' = (b, a)$
if $e' \in AEL$ **then**
  $AEL \rightarrow e'$
end
else
  $AEL \leftarrow e$
end

3.2 Generování vrstevnic

Vstupem algoritmu je výška vrstevnice Z a Delaunayova triangulace DT, která se skládá z jednotlivých hran e_i, $e_i = (s_i, e_i)$, kde s_i je začáteční bod hrany a e_i je koncový bod hrany, u obou bodů jsou známé souřadnice x, y, z.

Lineární interpolace, výpočet souřadnic průsečíků hrany a vrstevnice (vstup: Hrana(s, e), výška vrstevnice Z).

$$x = \frac{x_e - x_s}{z_e - z_s}(Z - z_s) + x_s \quad y = \frac{y_e - y_s}{z_e - z_s}(Z - z_s) + y_s \tag{1}$$

3.3 Orientace svahu

Orientace svahu je definována jako azimut A průmětu gradientu $\nabla\rho$ do roviny x, y.

$$A = atan2(\frac{a}{b}), \tag{2}$$

kde a, b jsou vektorové součiny pro vektory $\vec{u}$ a $\vec{v}$. Výpočet je prováděn nad každým trojúhelníkem DMT.

Barevná stupnice pro expozici byla zvolena takto($\rightarrow$ v rgb() znamená postupnou změnu):

- $A \in (0, \frac{\pi}{2}) \rightarrow rgb(255, 0 \rightarrow 255, 0)$
- $A \in (\frac{\pi}{2}, \frac{3\pi}{4}) \rightarrow rgb(255 \rightarrow 0, 255, 0)$
- $A \in (\frac{3\pi}{4}, \pi) \rightarrow rgb(0, 255, 0 \rightarrow 255)$
- $A \in (\pi, \frac{3\pi}{2}) \rightarrow rgb(0, 0 \rightarrow 255, 255)$
- $A \in (\frac{3\pi}{2}, \frac{7\pi}{4}) \rightarrow rbg(0 \rightarrow 255, 0, 255)$
- $A \in (\frac{7\pi}{4}, 2\pi) \rightarrow rgb(255, 0, 255 \rightarrow 0)$

Algoritmus 3: Generování vrstevnic

```
CL = {}
for t_j ∈ DT, t_j = (e_{j,i}, e_{j,i+1}, e_{j,i+1}) do
    p_1, p_2, p_3
    for z ∈ Z do
        dz_1 = z − z_{p_1}; dz_2 = z − z_{p_2}, dz_3 = z − z_{p_3}
        if (dz_1 == 0)&(dz_2 == 0)&(dz_3 == 0) then
            continue
        end
        else if (dz_1 == 0)&(dz_2 == 0) then
            CL ← e_{j,i}
        end
        else if (dz_2 == 0)&(dz_3 == 0) then
            CL ← e_{j,i+1}
        end
        else if (dz_3 == 0)&(dz_1 == 0) then
            CL ← e_{j,i+1}
        end
        else if
         (dz_1 ∗ dz_2 < 0)&(dz_2 ∗ dz_3 <= 0)||(dz_1 ∗ dz_2 <= 0)&(dz_2 ∗ dz_3 < 0) then
            a = contourPoint(p_{j_1}, p_{j_2}, z); b = contourPoint(p_{j_2}, p_{j_3}, z)
            CL ← e(a, b)
        end
        else if
         (dz_2 ∗ dz_3 < 0)&(dz_3 ∗ dz_1 <= 0)||(dz_2 ∗ dz_3 <= 0)&(dz_3 ∗ dz_1 < 0) then
            a = contourPoint(p_{j_2}, p_{j_3}, z); b = contourPoint(p_{j_3}, p_{j_1}, z)
            CL ← e(a, b)
        end
        else if
         (dz_3 ∗ dz_1 < 0)&(dz_1 ∗ dz_2 <= 0)||(dz_3 ∗ dz_1 <= 0)&(dz_1 ∗ dz_2 < 0) then
            a = contourPoint(p_{j_3}, p_{j_1}, z); b = contourPoint(p_{j_1}, p_{j_2}, z)
            CL ← e(a, b)
        end
        end
        end
```

3.4 Sklon

Sklon je definován jako úhle mezi normálou trojúhelníku a svislicí. Nabývá hodnot 0° až 90°. Sklon je určen následujícím vztahem.

$$\phi = arccos\frac{\vec{n}_1 \cdot \vec{n}_2}{\|\vec{n}_1\|\|\vec{n}_2\|} = arccos\frac{c}{\|\vec{n}_1\|}, \tag{3}$$

kde $\vec{n}_1 = (a, b, c),\ \vec{n}_2 = (0, 0, 1)$

Jako symbologie byla zvolena spojitá stupnice odstínů šedi, bílá (r=255, g=255, b=255) náleží svahu o sklonitosti 0° a černá (r=0, g=0, b=0) náleží svahu o sklonitosti 90°.

3.5 Popis vrstevnic

Jako symbologie vrstevnic byla zvolena barva odstínu hnědé (r=139, g=69, b=19), kdy každá čára byla nakreslena pomocí metody `drawLine`. Hodnota nadmořské výšky je získána z počátečního bodu (`cl.getS()`) každé vrstevnice. Popis je generován jen pro každou 15. vrstevnici.

Pro zlepšení čitelnosti a dodržení zásad kartografie je vypočítáván úhel sklonu každého popisu vrstevnice (θ) z "x"a "y"souřadnice koncového a počátečního bodu kresby vrstevnice a to pomocí vzorce:

$$\theta = \arctan\left(\frac{\text{getE().y()} - \text{getS().y()}}{\text{getE().x()} - \text{getS().x()}}\right) \tag{4}$$

4 Vstupní data

Vstupními daty je množina bodů se známými souřadnicemi $[X, Y, Z]$. Aplikace tato data dokáže načítat z *.csv* souboru, ve kterém jsou zapsána ve formátu xyz. Vzhledem k tomu, že je použit algoritmus inkrementální konstrukce, o kubické složitosti $O(n^3)$, je aplikace vhodná spíše ke zpracování dat o nižších počtech bodů (v řádech desetitisíců).

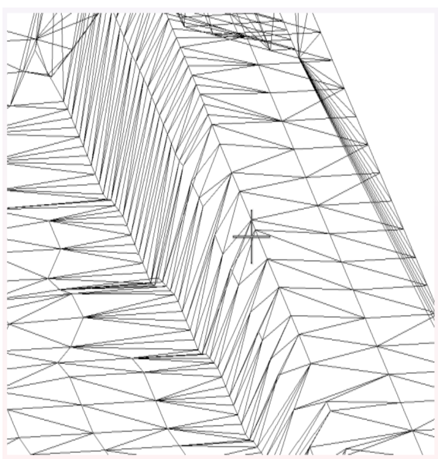
5 Vzhled aplikace

Grafické uživatelské rozhraní bylo vytvořeno pomocí sw. Qt Creator, jako Qt Widget Application. V horní části okna se nachází lišta s ovládacími prvky, zleva se jedná o: načtení dat, DT, vrstevnice, sklon, expozice, vyčištění plátna od výsledků, vyčištění plátna od veškeré kresby, nastavení kresby vrstevnic a konec. Všechny ovládací prvky jsou opatřeny nápovědou, která se zobrazí po krátkém umístění kurzoru na ikonu. Velikost okna aplikace je možné libovolně rozšiřovat.

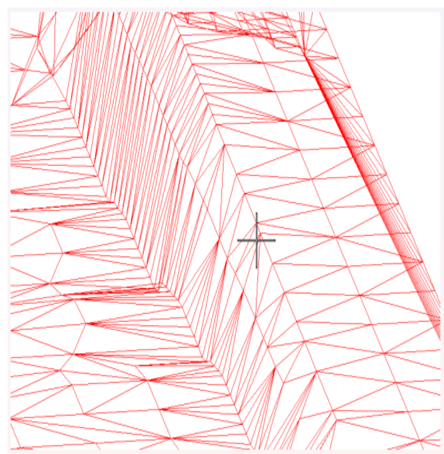
6 Zhodnocení činnosti algoritmu

Polyendrický model terénu bez definice povinných hran či jiných vstupních parametrů je vhodný pro hladké spojité oblasti. Oproti např. gridové (konstantní velikost buňky) reprezentaci terénu disponuje značnou výhodou, možností volby hustoty bodů v závislosti na členitosti terénu, což má za důsledek značné snížení velikosti dat při zachování kvality aproximace terénu. Další výhodou je možné vystižení extrémů.

Hlavním nedostatkem použité 2D triangulace je fakt, že při její tvorbě je brána v potaz pouze poloha bodu v rovině xy, nikoli však výšková informace z. Důsledkem toho, zejména u výškově členitého reliéfu, je nerespektování terénních hran/tvarů (údolnice a hřbetnice). Řešením výše zmíněného může být například tzv. *Data Dependent Triangulation (DDT)* a nebo použití algoritmu respektujícího povinné hrany, které jsou definovány uživatelem (viz. obrázky 1a, 1b a 2). Dalším nedostatkem (oproti gridové reprezentaci) je výpočetní a implementační náročnost.



(a) Nevhodné vystižení terénní hrany (DT)



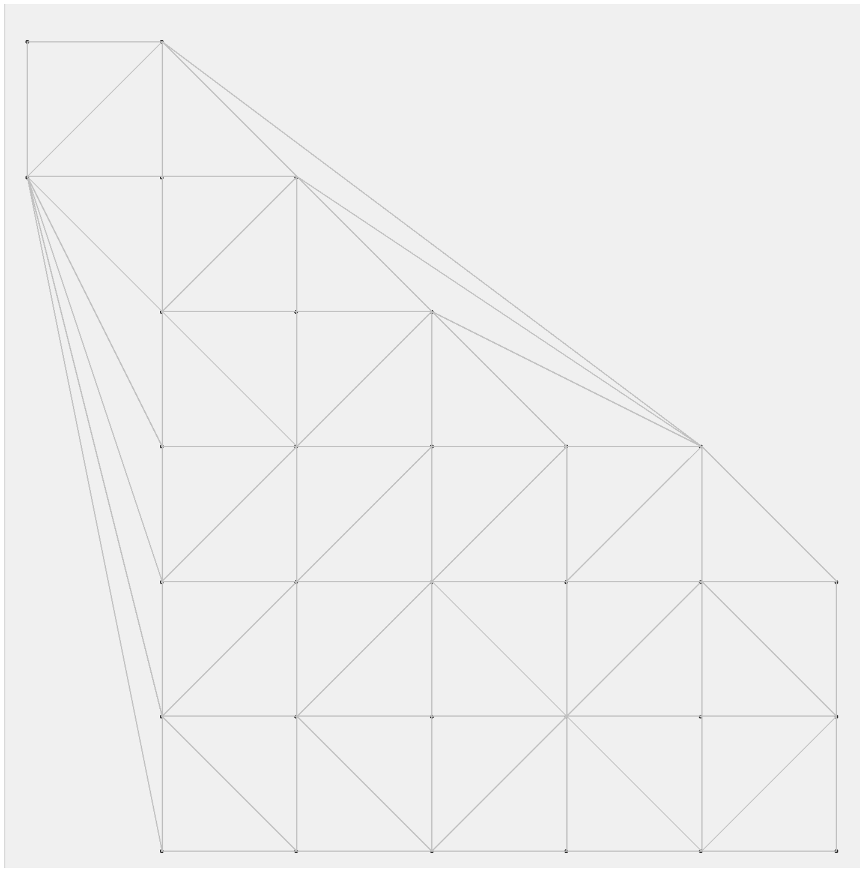
(b) Lepší vystižení terénní hrany (DDT)

Obrázek 1: Příklad různého vystižení terénního útvaru [1]

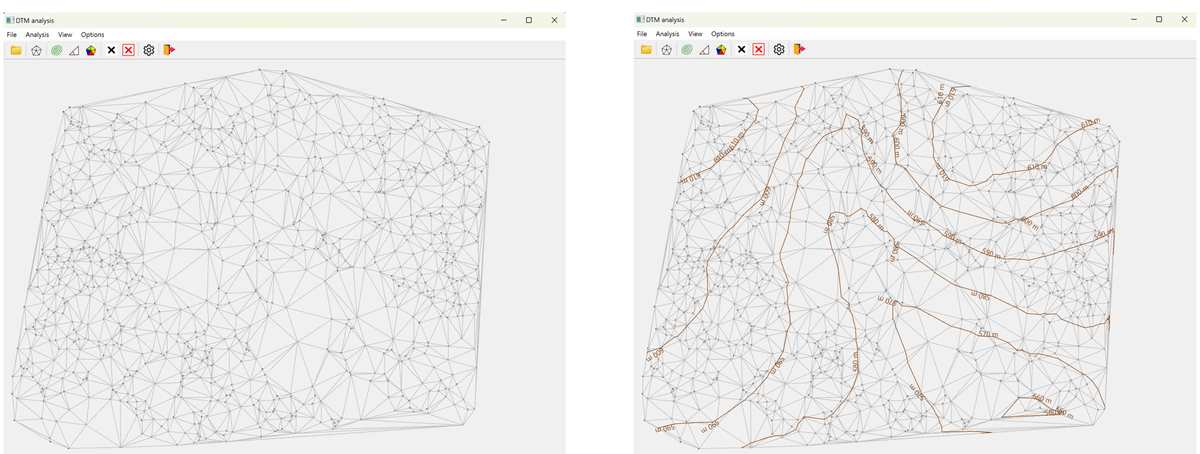
Další záležitostí degradující kartografické hledisko digitálního modelu terénu jsou dlouhé úzké trojúhelníky na okrajích modelu. Tento jev je důsledkem triangulace konvexní oblasti. Jedná se o oblasti, ve kterých interpolovaná výška pravděpodobně neodpovídá skutečnosti, tudíž je zapotřebí tyto oblasti odstranit/nebrat v úvahu.

Dalším úskalím při generaci TIN a tudíž i polyendrického modelu je čtvercová síť, jedná se totiž o nejednoznačný případ. Uhlopříčné hrany ve čtverci mohou být vždy nahrazený druhou uhlopříčkou čtverce. Proběhnuvší algoritmus značí pouze jeho zdánlivou řešitelnost, která je dána přesností počítače.

Na obrázcích 3 a 4 jsou k nahlédnutí výstupy softwaru. Použitá data byla pořízena v blízkosti kostela Navštívení Pany Marie nedaleko obce Žlutice v Karlovarském kraji.



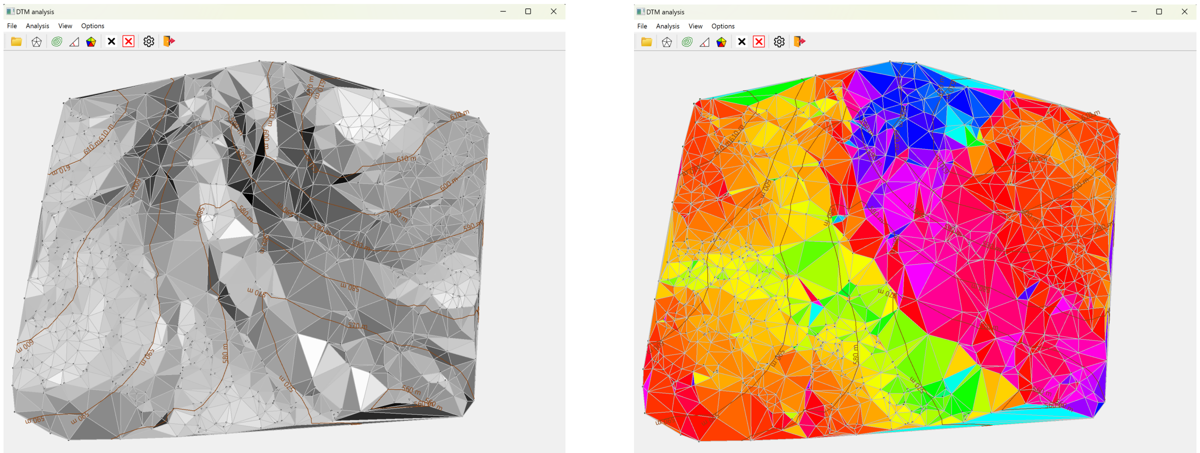
Obrázek 2: Dlouhé úzké trojúhelníky a čtvercová síť



(a) Delaunayova triangulace

(b) Delaunayova triangulace + vrstevnice

Obrázek 3: Výstupy softwaru



(a) Sklonitost + vrstevnice

(b) Expozice + vrstevnice

Obrázek 4: výstupy softwaru

7 Dokumentace

7.1 Třída Algorithms

- **int getPointAndLinePosition**(const QPointF3D &a, const QPointF3D &p1, const QPointF3D &p2);
- **double get2LinesAngle**(const QPointF3D &p1, const QPointF3D &p2, const QPointF3D &p3, const QPointF3D &p4);
- **int getNearestPoint**(const QPointF3D &q, const std::vector(QPointF3D) &points);
- **int getDelaunayPoint**(const QPointF3D &s, const QPointF3D &e, const std::vector (QPointF3D) &points);
- **double getDistance2D**(const QPointF3D &p1, const QPointF3D &p2);
- **std::vector (Edge) createDT**(std::vector (QPointF3D) &points);
- **void updateAEL**(Edge &edge, std::list (Edge) &ael);
- **QPointF3D contourPoint**(const QPointF3D &p1, const QPointF3D &p2, double z);
- **std::vector (Edge) createContourLines**(const std::vector (Edge) &dt, double z_min, double z_max, double dz);
- **std::vector (Triangle) analyzeDTMSlope**(const std::vector (Edge) &dt);
- **std::vector (Triangle) analyzeDTMAspect**(const std::vector (Edge) &dt);

- **double computeSlope**(const QPointF3D &v1, const QPointF3D &v2, const QPointF3D &v3);
- **double computeAspect**(const QPointF3D &v1, const QPointF3D &v2, const QPointF3D &v3);
- **static std::vector (QPointF3D) transformPoints**(std::vector (QPointF3D) &points_3d, double &x_t, double &y_t, double &scale, int &x_d, int &y_d);

7.2 Třída Draw

- **explicit Draw**(QWidget *parent = nullptr);
- **void mousePressEvent**(QMouseEvent *event);
- **void paintEvent**(QPaintEvent *event);
- **std::vector (QPointF3D) getPoints**();
- **std::vector (Edge) getDT**();
- **void setDT**(std::vector (Edge) &dt_);
- **void setContourLines**(const std::vector (Edge) &contour_lines_);
- **void setTriangles**(std::vector (Triangle) &triangles_);
- **void setAnalysis**(bool analysis_);
- **void clear**();
- **void clearAll**();
- **void clearRes**();
- **void clearContour**();
- **void drawPoints**(std::vector (QPointF3D) &points3d);
- **void setCSVPoints**(std::vector (QPointF3D) &csv);
- **void setScale**(double &scale_);
- **void setTrans**(double &x_t_, double &y_t_);
- **void setOffsets**(int &x_d_, int &y_d_);
- **double getScale**();

- **double getTransX**();
- **double getTransY**();
- **int getDeltaX**();
- **int getDeltaY**();

7.3 Třída Edge

- **Edge**(const QPointF3D &s_, const QPointF3D &e_)
- **Edge changeOrientation**()
- **QPointf3D getS**()
- **QPointf3D getE**()
- **bool operator ==**(const Edge &edge)

7.4 Třída Mainform : public QMainWindow

- **void on_actionCreate_DTM_triggered**();
- **void on_actionContour_lines_triggered**();
- **void on_actionAnalyze_slope_triggered**();
- **void on_actionAnalze_aspect_triggered**();
- **void on_actionClear_results_triggered**();
- **void on_actionClear_all_triggered**();
- **void on_actionExit_triggered**();
- **void on_actionSettings_triggered**();

7.5 Třída QPointF3D : public QPointF

- **QPointF3D(double x, double y, double z_):QPointF**(x,y);
- **QPointF3D():QPointF**(0,0);
- **double getZ**();
- **void setZ**(double z_);

7.6 Třída Settings

- **explicit Settings**(QWidget *parent = nullptr);
- **double getZmin**();
- **double getZmax**();
- **double getDz**();
- **void on_lineEdit_editingFinished**();
- **void on_lineEdit_2_editingFinished**();
- **void on_lineEdit_3_editingFinished**();

7.7 Třída SortPointsByX

- **bool operator()**(QPointF &p1, QPointF &p2);

7.8 Třída Triangle

- **Triangle**(const QPointF3D &v1_, const QPointF3D &v2_, const QPointF3D &v3_, double slope_, double asp_);
- **QPointF3D getV1**();
- **QPointF3D getV2**();
- **QPointF3D getV3**();
- **double getSlope**();
- **double getAspect**();

7.9 Třída CSV

- **static std::vector (QPointF3D) getPointsFromFile**(std::string &filename, double &xmin, double &xmax, double &ymin, double &ymax)

8 Závěr

V rámci třetí úlohy předmětu Algoritmy digitální kartografie a GIS byla vytvořena aplikace nesoucí název DMT. Aplikace byla napsána v jazyce C++ ve vývojovém prostředí Qt Creator.

Aplikace umožňuje načítání vektorových dat (bodových mračen) ve formátu WKT, vytvoření digitálního modelu terénu (ve formě TIN), vrstevnic, analýzu sklonu a analýzu orientace svahu.

Výsledná aplikace je k nalezení na této webové adrese: `https://github.com/jaksno/adk_2023_24/tree/main/DTM`.

Reference

[1] BAYER Tomáš, *2D/2.5D triangulace, DMT*, [online]. [cit. 2024-03-01]. Dostupné z: https://web.natur.cuni.cz/~bayertom/images/courses/Adk/adk5_new.pdf

[2] ISO 19125-1:2004, *Geographic information - Simple feature access*. [cit. 2023-11-13]. Dostupné z: https://www.iso.org/standard/40114.html