ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

FAKULTA STAVEBNÍ, OBOR GEODÉZIE A KARTOGRAFIE

KATEDRA GEOMATIKY

název předmětu

ALGORITMY DIGITÁLNÍ KARTOGRAFIE A GIS

číslo úlohy	název úlohy:			
3	**Digitální model terénu**			
školní rok:	semestr:	zpracovali:	datum:	klasifikace:
2019/20	zimní	**David Němec, Jan Šartner**	3. 12. 2019	

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. Doplňující úlohy

Byly řešeny tyto doplňující úlohy: *Triangulace nekonvexní oblasti zadané polygonem, Výběr barevných stupnic při vizualizaci sklonu a expozice, Automatický popis vrstevnic, Automatický popis vrstevnic respektující kartografické zásady, Algoritmu pro automatické generování terénních tvarů.*

3. Problematika

Konečným cílem této úlohy bylo vytvoření aplikace, která je schopná vytvořit z množiny vstupních bodů triangulaci, vypočítat vrstevnice v daném intervalu a analyzovat model terénu, přičemž výsledkem by měl být sklon terénu a expozice ke světovým stranám.

3.1 Delaunayova triangulace

Triangulace vytváří trojúhelníky, které se velmi blíží rovnostranným trojúhelníkům a to z toho důvodu, že protáhlé a úzké trojúhelníky nám nedávají spolehlivé výsledky. Triangulace má několik důležitých vlastností: Uvnitř opsané kružnice v libovolném trojúhelníků neleží žádný jiný bod; Maximalizuje minimální úhel, ale neminimalizuje maximální úhel v trojúhelníku; Je lokálně i globálně optimální vůči kritériu minimálního úhlu; Je jednoznačně určena, pokud žádné čtyři body neleží na kružnici.

3.1.1 Princip

Pracujeme s 2D triangulací. Používáme metodu inkrementální konstrukce, která je založena na postupném přidáváním bodů do již vytvořené DT. Nad existující Del hranou hledáme bod p, který minimalizuje poloměr opsané kružnice. Jelikož jsou hrany orientované, hledáme bod p pouze vlevo od ní. Dále používáme modifikovanou datovou strukturu AEL (Active Edge List), která obsahuje hrany, ke kterým hledáme body p. Do DT pak přidáváme hrany trojúhelníků, které prozatím nejsou obsaženy v AEL.

3.1.2 Výpočet

1: Nalezení pivota: náhodný bod, či bod o některé z max/min souřadnic.
2: Nalezení nejbližšího bod k pivotu a vytvoření hrany z těchto dvou bodů
3: Hledání Delaunayova bodu p, jenž splňuje podmínku minimální opsané kružnice. Pokud není nalezen, je vytvořené hraně změněna orientace a hledá se znovu.
4: Vytvoří se zbývající hrany trojúhelníku.
5: Přidání hran trojúhelníku do DT.
6: Přidání hran trojúhelníku do AEL.
7: Cyklus WHILE (AEL není prázdná)
    Vyhledání bodu k hraně z AEL.
    Pokud Delaunayův bod existuje:
        Vytvoří se zbývající hrany trojúhelníku.
        Pokud hrana není v AEL
            Přidat hranu.

3.2 Vrstevnice

Vrstevnice je křivka (vektor hran v našem případě), která spojuje body se stejnou nadmořskou výškou. Zpravidla je mezi jednotlivými vrstevnicemi stejný výškový rozdíl. Dále existují různé typy vrstevnic: Základní vrstevnice, které jsou zobrazeny tenkou linií a nepřerušovaně; Hlavní vrstevnice, většinou každá pátá, která je kreslena tlustší nepřerušovanou čarou. Bývá popsána ve směru stoupání terénu příslušnou výškovou kótou; Doplňkové vrstevnice, které mívají poloviční či čtvrtinový interval vrstevnic základních. Používají se v místech, kde již základní vrstevnice dostatečně nevystihují průběh terénu; Pomocné vrstevnice, které se používají v místech s nestálým terénem např. těžba.

3.2.1 Princip

Vrstevnice se počítají na všech hranách Delaunayovy triangulace. Při průniku vrstevnice z hrany se vychází z podobnosti trojúhelníků, při totožnosti hrany s vrstevnicí se hrana uloží jako vrstevnice, a pokud vrstevnice nijak neprotíná hranu ani s ní není totožná, pak je hrana pod/nad vrstevnicí a nijak se s ní dále nepočítá.

3.2.2 Výpočet

1: Cyklus FOR (velikost vektoru hran DT)
2: Pokud hrana náleží rovině vrstevnice:
3: Hrana se uloží do vektoru vrstevnic.
4: Pokud hrana nenáleží rovině vrstevnice:
5: Hrana se neuloží do vektoru vrstevnic ani se nijak nepočítá průnik.
6: Pokud hrana je průnikem s rovinou vrstevnice:
7: Vypočte se poloha souřadnic průniku:

$$x=\frac{(x_2-x_1)}{(z_2-z_1)}\cdot(z-z_1)+x_1 \qquad y=\frac{(y_2-y_1)}{(z_2-z_1)}\cdot(z-z_1)+y_1$$

8: Hrana se uloží do vektoru vrstevnic.

3.3 Sklon terénu

Jedná se o úhel mezi normálovým vektorem (0,0,1) a normálovým vektorem roviny trojúhelníku.

3.3.1 Výpočet

1: Cyklus FOR (velikost vektoru trojúhelníků DT)
2: Vypočte se normálový vektor roviny trojúhelníku:

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

3: Vypočte se sklon terénu:

$$\phi=\arccos\frac{n_z}{|n_t|}$$

3.4 Expozice terénu

Jedná se o azimut k průmětu normálového vektoru roviny trojúhelníku do roviny x,y.

3.4.1 Výpočet

1: Cyklus FOR (velikost vektoru trojúhelníků DT)
2: Vypočte x, z části normálového vektoru:

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

3: Vypočte se sklon terénu:

$$A = atan2\frac{n_x}{n_y}$$

4. Bonusové úlohy

4.1 Triangulace nekonvexní oblasti zadané polygonem

Úloha byla řešena tak, že byly nejprve vybrány body nacházející se uvnitř předem vykresleného polygonu. Bylo tak učiněno pomocí metody Winding Number, jež byla převzata z první úlohy ADK. Nad těmito body byla následně provedena klasická DT. Posléze byly analyzovány body vrstevnic tak, že pokud se alespoň jeden z jejích bodů nacházel mimo ohraničující polygon, byla tato hrana vrstevnice vymazána.

4.1.1 Doplňkové informace

Triangulace se provádí i mimo konvexní polygon, bylo by třeba napsat funkci pro průsečík s hranami polygonu, aby se tomu zamezilo.

4.2 Výběr barevných stupnic při vizualizaci sklonu a expozice

Vpravo od tlačítek pro vizualizaci sklonu a expozice byly umístěny comboboxy obsahující tři barvy. Červenou, zelenou a modrou. Základní barvou po spuštění aplikace je červená. Po kliknutí na barvu je tato barva pomocí indexu přenesena do algoritmu, kde se dále zpracuje podle sklonu či expozice. 0=červená, 1=zelená, 2=modrá.

4.2.1 Doplňkové informace

Pro lepší vizualizaci sklonu terénu by bylo vhodné přidat funkci, která určuje výsledné barvy dle maximálního a minimálního sklonu. Při nepříliš výrazných sklonech není rozdíl patrný.

4.3 Automatický popis vrstevnic

Viz. 4.4.

4.4 Automatický popis vrstevnic respektující kartografické zásady

Pro popis vrstevnic byly vytvořeny dvě speciální třídy. První byla OrCuntours, která obsahuje pouze proměnnou s vektorem hran (Edge). Tato třída byla vytvořena z důvodu špatné nevhodného zacházení se strukturou std::vector<std:vector<Edge>>. OrCuntours tedy obsahuje seřazené hrany vrstevnic. Každá OrContours obsahuje jednu vrstevnici a jedné nadmořské výšce. Seřazení bylo docíleno funkci *std::vector<OrCuntours> orientateContours(std::vector<Edge> &cont);*

Druhou vytvořenou třídou byla QContDescr, jež obsahuje údaje o poloze popisu a jeho natočení.

Popis je počítán tak, že se nejprve seřadí hrany hlavních vrstevnic. Následně se spočítá celková délka vrstevnice. Pokud je její délka menší než 400, bude vrstevnice obsahovat pouze jeden popis a to zhruba v polovině své délky. Pokud je délka vyšší než 400, jsou popisy od sebe vzdálené ve vzdálenosti 400. Natočení textu je prováděno pomocí výpočtu azimutu.

4.4.1 Doplňkové informace

Popis funguje relativně dobře při menším počtu bodů. Při výpočtech s hustými mračny pak občas dochází k nahodilému umístění popisu do prostoru. Občas se stane, že se v prostoru objeví zcela náhodná hodnota některého z bodů. Algoritmus pro výpočet popisu také nezvládá příliš velké počty jednotlivých dílů vrstevnic. Tato úloha by tedy potřebovala nutně vylepšit, aby byla spolehlivá pro jakýkoliv terén.

4.4 Algoritmus pro automatické generování terénních tvarů

Byly vyhotoveny následující automatické terénní tvary: Údolí, kupa, hřbet, sopka a sedlo.

4.4.1 Údolí

Údolí je terénní tvar, který se vyznačuje protáhlým tvarem, jehož okraje dva nebo tři strany jsou lemovány vyvýšeným terénem. V této aplikaci byla použita varianta pro dva vyvýšené okraje. Nejprve byly určeny koncové body údolí ležící na jeho dně (A, B). Mezi těmito body bylo následně vygenerováno několik bodů s podobnou výškou. V dalším cyklu pak byly v pravidelné mřížce generovány body svahů po obou stranách. Jejich výška se počítala pomocí vzorce, jehož vstupním parametrem byla vzdálenost bodu od linie A B. $Z = A.Z + dist \cdot dist/50 + r_z$,kde r_z je náhodná veličina určující základní výšku všech bodů. Všechny tři souřadnice vygenerovaného bodu pak byly upraveny přidáním menší náhodné hodnoty, která vytvoří jakýsi „šum“, aby vrstevnice nebyly úplně pravidelné.

4.4.2 Kopec

Kopec je terénní tvar, který má zřetelně patrný vrchol a vystupuje nad okolní terén. Pro generování kopce byl nejprve určen bod uprostřed Canvasu (A), jemuž byla náhodně přidělena výška. Následně došlo ke generování pravidelném mřížky bodů okolo středu. Body ve větší vzdálenosti od středu mají výšku rozdílnou pouze o malé náhodné číslo. Body v blízkém okolí středu pak mají výšku vypočtenou dle vzorce: $Z = A.Z - dist \cdot dist/70 + r_z$,kde r_z je náhodná veličina určující základní výšku všech bodů. Všechny tři souřadnice vygenerovaného bodu pak byly upraveny přidáním menší náhodné hodnoty, která vytvoří jakýsi „šum“, aby vrstevnice nebyly úplně pravidelné.

4.4.3 Hřbet

Hřbet je terénní tvar, jehož délka přesahuje šířku. Hřbet byl generován stejně jako údolí, pouze inverzně.

4.4.4 Sopka

Zde byl použit tvar stratovulkánu. Jedná se o kuželovitý tvar sopky s příkrými svahy, na jehož vrcholu se nachází kaldera (kotlovitá prohlubeň). Byla vygenerována pravidelná čtvercová síť, na níž pak byly počítány výšky jednotlivých bodů. Vše počítáno od počátečního bodu A(0,0,0). Podle vzdálenosti bodu od středového bodu A pak vypal určena výška dle těchto vzorců:

Pokud dist<PI/2:

$$Z = (\cos(dist^2) + \sin(dist^2) + 1) \cdot r_z$$

else:

$$Z = (\frac{1}{dist} + 2) \cdot r_z$$

r_z je náhodná veličina určující základní výšku všech bodů. Všechny tři souřadnice vygenerovaného bodu pak byly upraveny přidáním menší náhodné hodnoty, která vytvoří jakýsi „šum“, aby vrstevnice nebyly úplně pravidelné.

4.4.5 Sedlo

Sedlo je terénní tvar, jehož dvě strany sestupují do údolí a další dvě strany pak stoupají k vrcholům hor. Byla vygenerována pravidelná čtvercová síť, na níž pak byly počítány výšky jednotlivých bodů. $Z = 50 \cdot (\sin(x^2) + \cos(y^2) + r_z)$,kde r_z je náhodná veličina určující základní výšku všech bodů. Všechny tři souřadnice vygenerovaného bodu pak byly upraveny přidáním menší náhodné hodnoty, která vytvoří jakýsi „šum“, aby vrstevnice nebyly úplně pravidelné.

5. Vstupní data

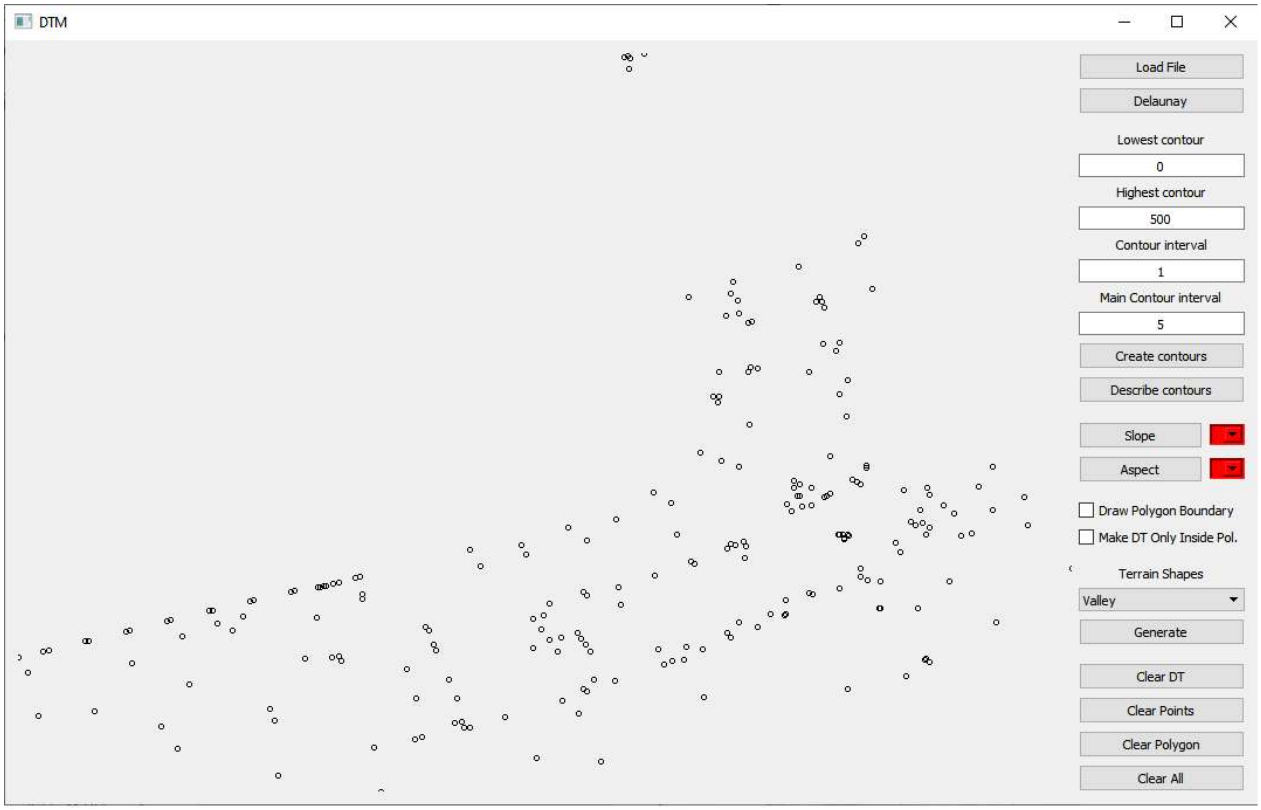
K aplikaci jsou k dispozici data z reálných měření v terénu ve formátu .txt. Dále je možné vygenerovat přednastavené náhodné tvary, které byly popsány v předchozí kapitole. Souřadnice bodů se ukládají do datové struktury: *std::vector<QPoint3D>*.

6. Výstupní data

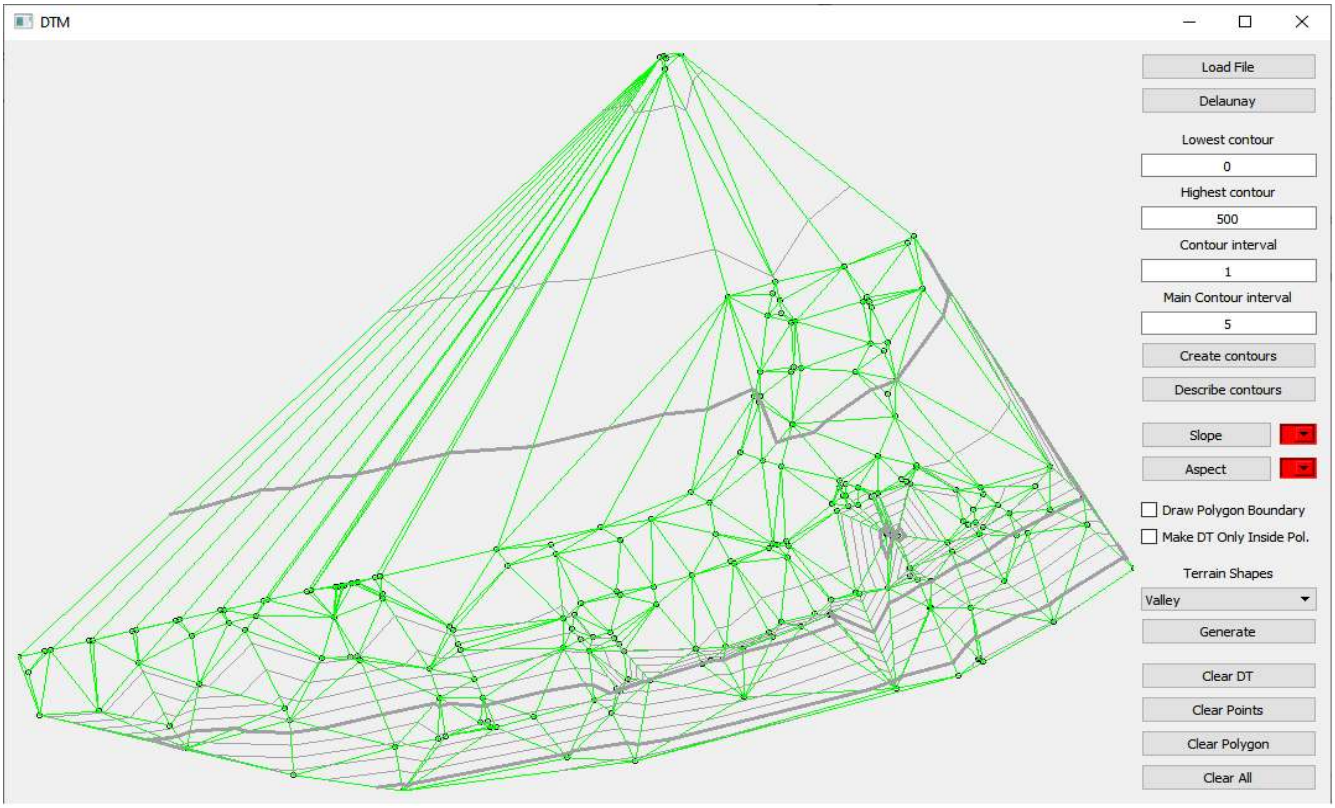
Aplikace neposkytuje žádný výstupní data

7. Vzhled aplikace

Obrázek 1: Základní vzhled aplikace



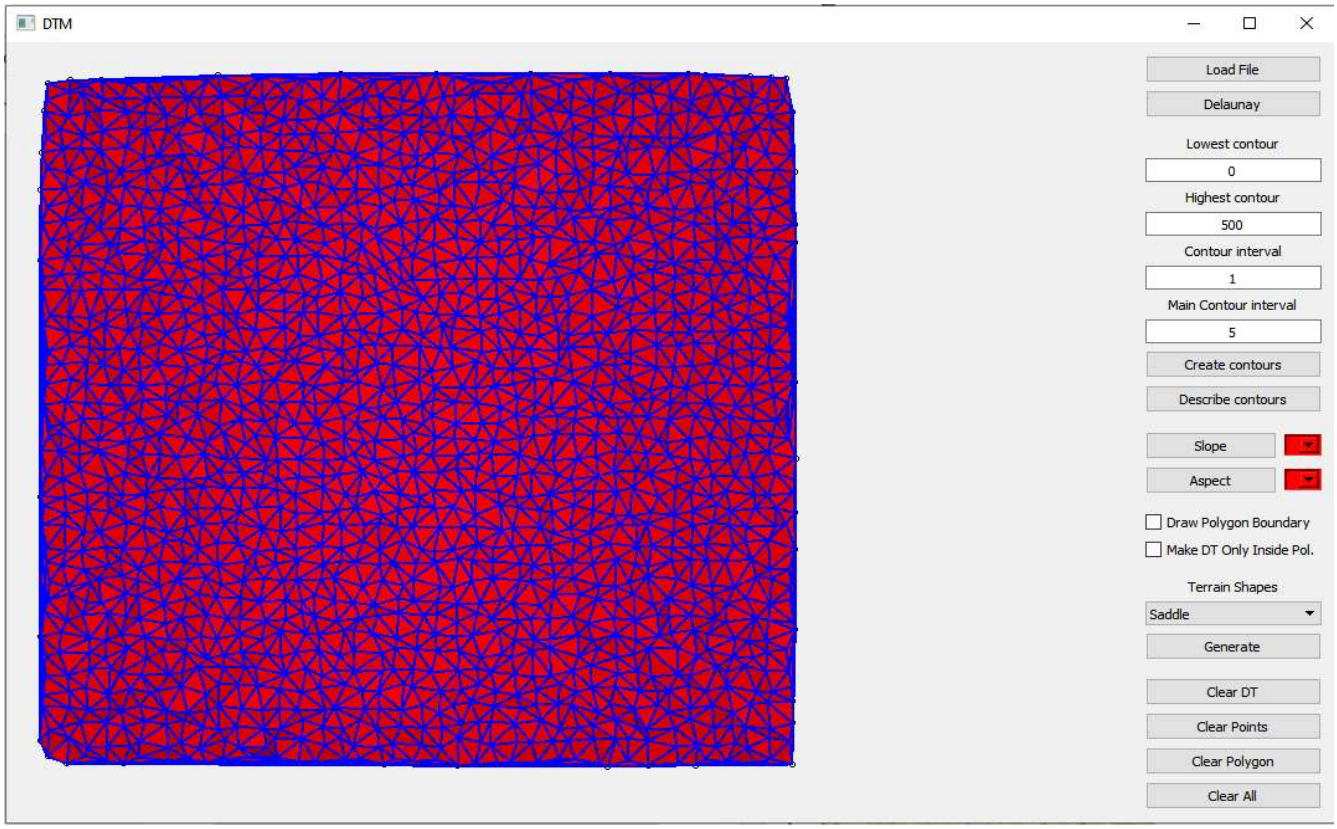
Obrázek 2: Načtení bodů



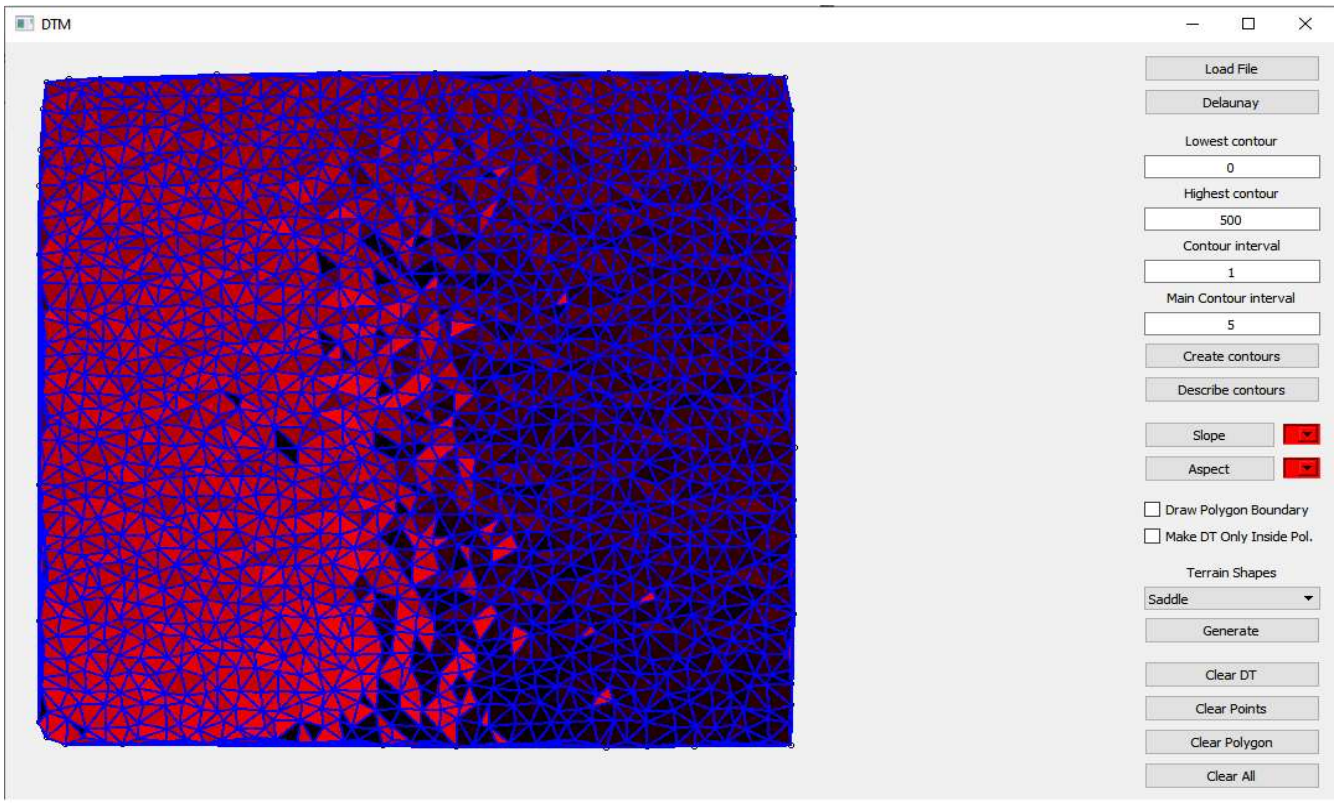
Obrázek 3: Triangulace a vytvoření vrstevnic

Obrázek 4: Popis vrstevnic

Obrázek 5: Vytvoření vrstevnic v zadaném polygonu



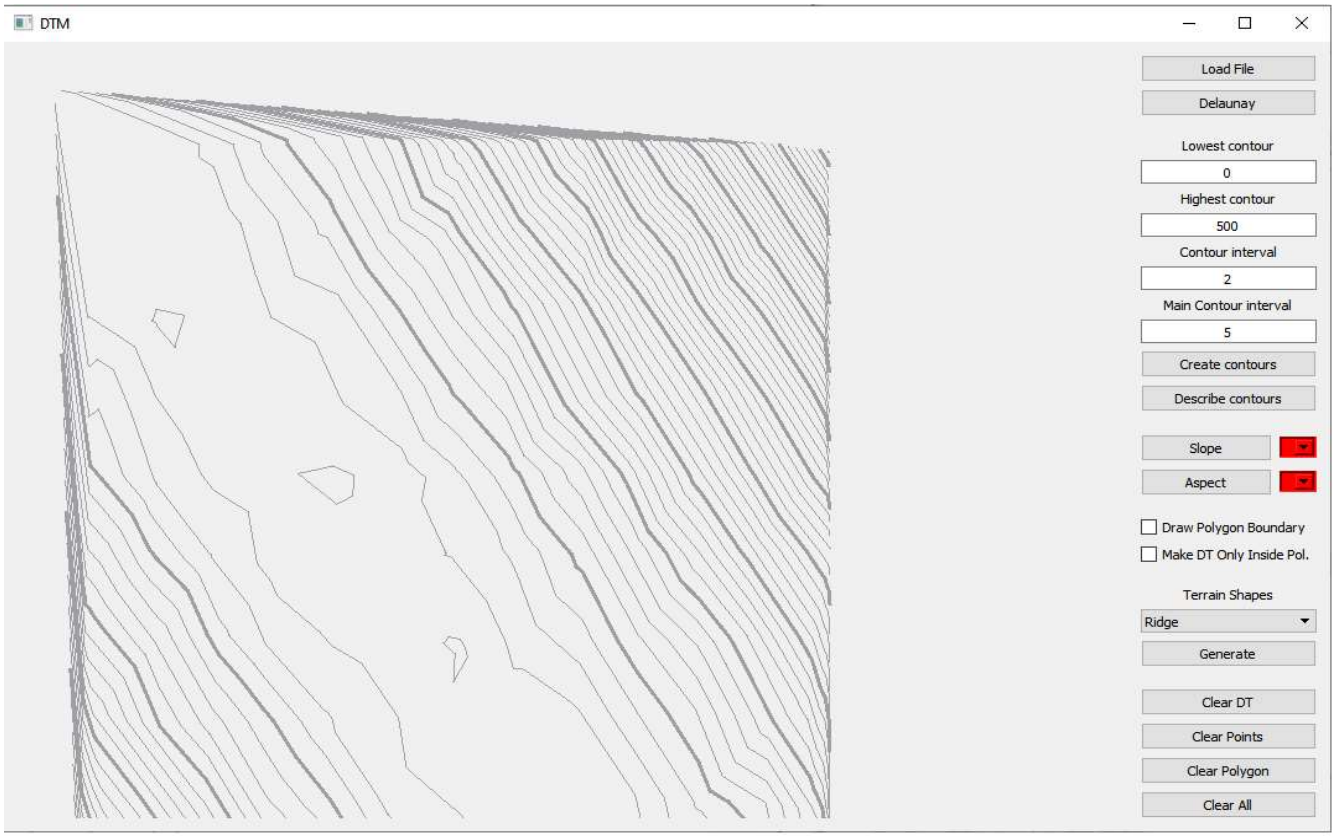
Obrázek 6: Sklon terénu



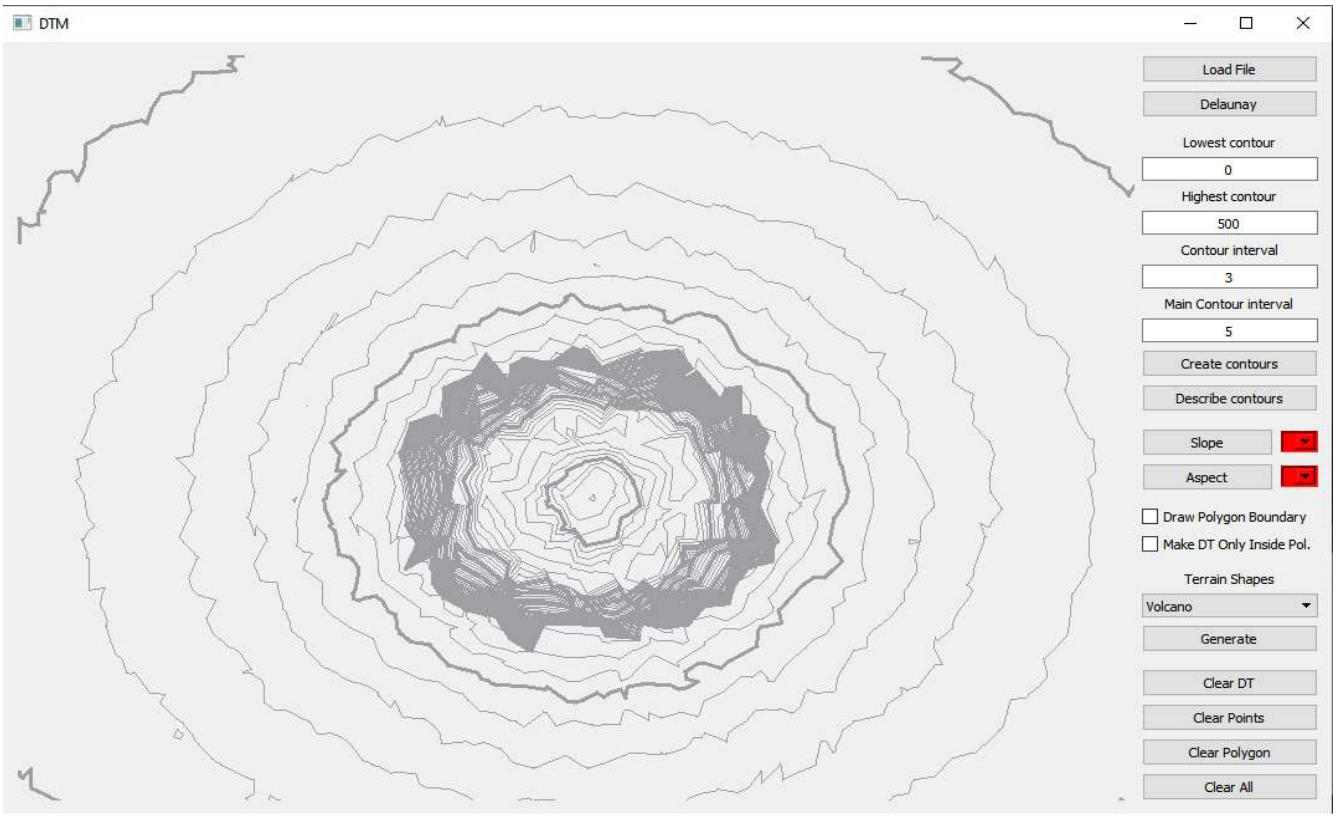
Obrázek 7: Expozice terénu

Obrázek 8: Vygenerované údolí

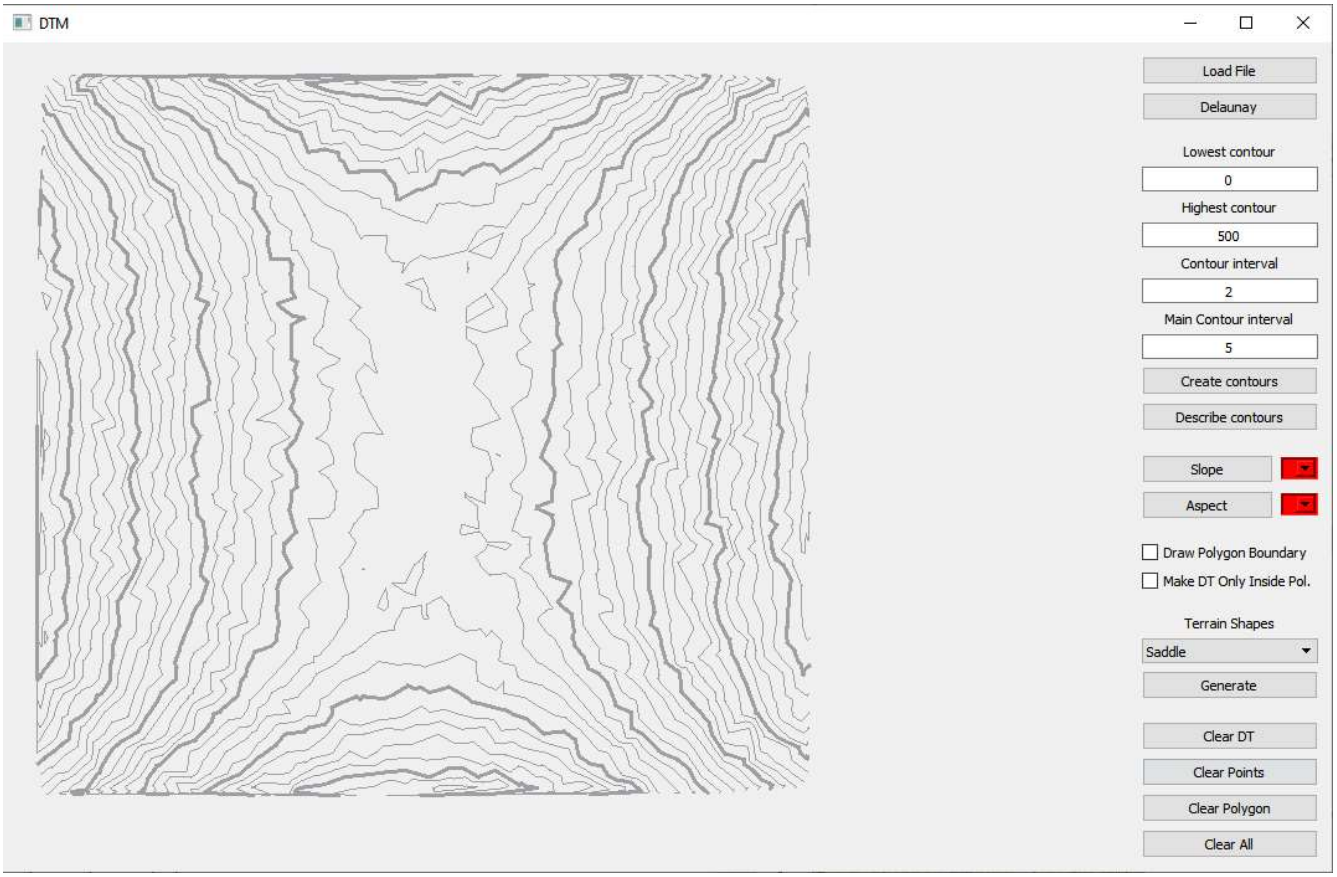
Obrázek 9: Vygenerovaný kopec



Obrázek 10: Vygenerovaný hřbet



Obrázek 11: Vygenerovaná sopka



Obrázek 12: Vygenerované sedlo

8. Dokumentace

Třídy:

Algorithms:

Třída obsahuje 27 funkcí pro výpočet a analýzu digitálního modelu terénu:

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

Funkce zjišťuje, zda bod q leží vlevo či vpravo od vybrané úsečky spojující body p1 a p2. Také určuje, zda neleží přímo na úsečce. Funkce vrací hodnoty *int(1 pokud je bod vlevo, 0 pokud je bod vpravo, -1 pokud bod leží na přímce).*

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

Funkce vrací vzdálenost bodu q od linie tvořené body p1 a p2.

static double getAngle2Vectors(QPoint3D &p1, QPoint3D &p2, QPoint3D &p3, QPoint3D &p4);

Funkce vrací velikost úhlu mezi dvěma vektory, které jsou určeny body p1,p2 a p3,p4. Úhel je vracen v radiánech.

static double get2PointsAzimuth(QPoint3D &p1, QPoint3D &p2);

Funkce vrací velikost azimutu, který svírá linie tvořená body p1 a p2. Úhel je vracen v radiánech.

double getCircleRadius(QPoint3D &p1, QPoint3D &p2, QPoint3D &p3);

Funkce vrací poloměr kružnice tvořené 3 body v typu double.

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

Funkce vrací index bodu z vektoru points, který se nachází nejblíže k bodu p.

static double distance2Points(QPoint3D &p1, QPoint3D &p2);

Funkce vrací vzdálenost mezi dvěma body v typu double.

static int getDelaunayPoint(QPoint3D &s, QPoint3D &e, std::vector<QPoint3D> &points);

Funkce vrací index bodu, jenž splňuje Delaunovy vlastnosti.

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

Funkce vytváří Delaunayovu triangulaci a vrací vektor hran, které tvoří jeho trojúhelníky.

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

Funkce vrací bod na hraně, která se protíná s vrstevnicí. Pro výpočet používá podobnost trojúhelníku.

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

Funkce vytváří vrstevnice ze vstupního souboru hran dt v rozmezí od z_min a z_max a v intervalu dz. Výstupem je pak vektor hran s vrstevnicemi.

static int positionPoiintPolygonWinding(QPoint3D &q, QPolygonF &pol);
Funkce zjišťuje zda bod leží na hraně, uvnitř či vně polygonu. Používá přitom metodu Winding. Vrací tři možné hodnoty: *int(1 pokud bod leží v polygonu, 0 pokud bod leží vně polygonu, -1 pokud bod leží na hraně)*. Vstupní hodnoty jsou: určený bod q a vektor bodů, které prozkoumávaný polygon obsahuje: pol.

static std::vector<QPoint3D> calculatePointsInsidePolygon(std::vector<QPoint3D> &points, QpolygonF &pol);
Funkce vrací vektor bodů, které se nachází uvnitř vytvořeného polygonu.

static std::vector<Edge> calculateContoursInsidePolygon(std::vector<Edge> &conts, QpolygonF &pol);
Funkce vrací vektor hran, které se oběma svýma konci nachází uvnitř vytvořeného polygonu.

static void getLineParameters(QPoint3D &p1, QPoint3D &p2, double &m, double &b, double &dist);
Funkce ze dvou vstupních bodů vrací parametry přímky a vzdálenost mezi těmito body.

static void getLineParameters(QPoint3D &p1, QPoint3D &p2, double &m, double &b, double &dist);
Funkce ze dvou vstupních bodů vrací parametry přímky a vzdálenost mezi těmito body.

static void getLineParameters(QPoint3D &p1, QPoint3D &p2, double &m, double &b);
Funkce ze dvou vstupních bodů vrací parametry přímky.

static std::vector<QPoint3D> generateHill(int &max_x, int &max_y);
Funkce vektor bodů, které tvoří tvar kopce.

static std::vector<QPoint3D> generateValley(int &max_x, int &max_y);
Funkce vektor bodů, které tvoří tvar údolí.

static std::vector<QPoint3D> generateRidge(int &max_x, int &max_y);
Funkce vektor bodů, které tvoří tvar hřbetu.

static std::vector<QPoint3D> generateVolcano(int &max_x, int &max_y);
Funkce vektor bodů, které tvoří tvar stratovulkánu.

static std::vector<QPoint3D> generateSaddle(int &max_x, int &max_y);
Funkce vektor bodů, které tvoří tvar horského sedla.

static std::vector<QPoint3D> generateHill(std::string &path, int &canvas_x, int &canvas_y);
Funkce vytváří vektor bodů ze vstupního textového souboru. Zároveň souřadnice transformuje tak, aby se vešly do Canvasu. Vstupní soubor je ve tvaru: *point_id point_x point_z point_y \n*

static std::vector<QContDescr> describeContours(std::vector<OrCuntours> &cont);
Funkce vrací vektor popisů vrstevnic v typu QContDescr.

static std::vector<OrCuntours> orientateContours(std::vector<Edge> &cont);
Funkce srovnává hrany vrstevnic tak, aby na sebe plynule navazovaly. Vrací pak vektor srovnaných hran v typu OrCuntours.

static double calculateSlope(QPoint3D &p1, QPoint3D &p2, QPoint3D &p3);
Funkce vrací velikost spádu v trojúhelníku tvořeném třemi body.

static double calculateAspect(QPoint3D &p1, QPoint3D &p2, QPoint3D &p3);
Funkce vrací úhel natočení trojúhelníku tvořeném třemi body.

static std::vector<Triangle> analyzeDTM(std::vector<Edge> &dt);
Funkce analyzuje vektor hran D. triangulace a počítá spád a úhel natočení.

Draw:
Třída obsahuje 31 funkcí pro generování a vykreslování proměnných.

void setPoints(std::vector<QPoint3D> &points_);
Funkce zajišťuje převod bodů do Canvasu.

std::vector<QPoint3D> getPoints();
Funkce zajišťuje převod bodů z Canvasu.

void setDt(std::vector<Edge> &dt_);
Funkce zajišťuje převod triangulace do Canvasu.

std::vector<Edge> getDt();
Funkce zajišťuje převod triangulace z Canvasu.

void setContours(std::vector<Edge> &contours_);
Funkce zajišťuje převod vrstevnic do Canvasu.

std::vector<Edge> getContours();
Funkce zajišťuje převod vrstevnic z Canvasu.

void setMainContours(std::vector<Edge> &main_contours_);
Funkce zajišťuje převod hlavních vrstevnic do Canvasu.

std::vector<Edge> getMainContours();
Funkce zajišťuje převod hlavních vrstevnic z Canvasu.

void setPointsInsidePol(std::vector<Points> &points_);
Funkce zajišťuje převod bodů uvnitř polygonu do Canvasu.

std::vector<QPoint3D> getPointsInsidePol();
Funkce zajišťuje převod bodů uvnitř polygonu z Canvasu.

void setPolygon(QPolygonF &polygon_);
Funkce zajišťuje převod polygonu do Canvasu.

QPolygonF getPolygon();
Funkce zajišťuje převod polygonu z Canvasu.

int getDtSize();
Funkce zajišťuje převod velikosti vektoru Dt z Canvasu.

void setAngles(std::vector<double> angles_);
Funkce zajišťuje převod vektoru úhlů do Canvasu.

void setPolygonDraw(bool polygon_draw_);
Funkce zajišťuje převod polygonové kreslící funkce do Canvasu.

void setContoursDesc(std::vector<QContDescr> &cd_);
Funkce zajišťuje převod vektoru popisů vrstevnic do Canvasu.

void setDTM(std::vector<Triangle> & dtm_);
Funkce zajišťuje převod vektoru trojúhelníků dt do Canvasu.

void setSlope(bool slope_);
Funkce zajišťuje převod informace, zda se má vykreslit spád do Canvasu.

void setAspect(bool aspect_);
Funkce zajišťuje převod informace, zda se má vykreslit úhel natočení do Canvasu.

void setColor(int &color_);
Funkce zajišťuje převod barvy vykreslování spádu do Canvasu.

void setColor2(int &color2_);
Funkce zajišťuje převod barvy vykreslování úhlu natočení do Canvasu

*void paintEvent(QPaintEvent *event);*
Funkce ukládá bod po stisknutí levého tlačítka na myši.

*void void mousePressEvent(QMouseEvent *event);*
Funkce vykresluje vytvořené a vygenerované proměnné.

void clearPoints();
Funkce maže vektor bodů

void clearDT();
Funkce maže D. triangulaci.

void clearContours();
Funkce maže vektor vrstevnic.

void clearMainContours();

Funkce maže vektor hlavních vrstevnic.

void clearAngles();
Funkce maže vektor úhlů.

void clearPolygon();
Funkce maže ohraničující polygon.

void clearPointsInsidePol();
Funkce maže body uvnitř ohraničujícího polygonu.

void clearContDesc();
Funkce maže vektor s popisy vrstevnic.

void clearDtm();
Funkce maže analyzovaný model terénu.

Edge:

Třída obsahuje 9 funkcí. Byla vytvořena ze dvou 3D bodů a vytváří tak hranu.

QPoint3D & getStart();
Funkce vrací bod počátku hrany.

void setStart(QPoint3D &s);
Funkce nastavuje bod počátku hrany.

QPoint3D & getEnd();
Funkce nastavuje bod konce hrany.

void setEnd(QPoint3D &e);
Funkce vrací bod konce hrany.

QPointF getFStart();
Funkce vrací QPointF bod počátku hrany.

QPointF getFEnd();
Funkce vrací QPointF bod konce hrany.

double getLength();
Funkce vrací vzdálenost mezi body hrany.

void changeOrientation();
Funkce obrátí orientaci hrany.

bool operator == (const Edge &h);
Operátor, zda se hrany rovnají.

OrCuntours:

Třída obsahuje 2 funkce. Byla vytvořena z vektoru orientovaných hran z důvodu problematiky při vkládání vektoru hran do dalšího vektoru.

std::vector<Edge> getOrientCont();

Funkce vrací vektor orientovaných hran.

void setOrientCont(std::vector<Edge> orient_cont_);

Funkce nastavuje vektor orientovaných hran.

QContDescr:

Třída obsahuje 6 funkcí. Třída byla vytvořena pro popis vrstevnic. Je tvořena bodem p(souřadncie popisu), angle (úhlem natočení textu) a z (výška, která má být popsána).

QPoint3D getP();

Funkce vrací bod p.

double getAngle();

Funkce vrací úhel natočení.

double getZ();

Funkce vrací souřadnici z.

void setP(QPoint3D &p_);

Funkce nastavuje bod p.

void setAngle(double &angle_);

Funkce nastavuje úhel natočení.

void setZ(double &z_);

Funkce nastavuje souřadnici z.

QPoint3D:

Třída obsahuje 2 funkce. Třída byla odvozena z třídy QPointF. Navíc obsahuje souřadnici Z.

double getZ();

Funkce vrací souřadnici z.

void setZ(double z);

Funkce nastavuje souřadnici z.

SortbyX:

Třída obsahuje 1 operátor. Slouží k porovnání souřadnic x QPoint3D bodů.

bool operator ()(QPoint3D &p1, QPoint3D &p2);

Operátor porovnává, zda je p1.x menší než p2.x-

SortbyZ:

Třída obsahuje 1 operátor. Slouží k porovnávání souřadnic z u OrCuntours (orientované hrany)

bool operator()(OrCuntours &e1, OrCuntours &e2);

Operátor porovnává, zda je zetová souřadnic e1 menší nebo rovno než zetová souřadnice e2.

Triangle:

Třída obsahuje 10 funkcí. Třída vytváří trojúhelník a jsou zde vytvořeny funkce pro jeho popis a analýzu.

QPoint3D getP1();

Funkce vrací bod p1.

QPoint3D getP2();

Funkce vrací bod p2.

QPoint3D getP3();

Funkce vrací bod p3.

void setP1(QPoint3D &p1_);

Funkce nastavuje bod p1.

void setP2(QPoint3D &p2_);

Funkce nastavuje bod p2.

void setP3(QPoint3D &p3_);

Funkce nastavuje bod p3.

double getSlope();

Funkce vrací spád.

double getAspect();

Funkce vrací úhel natočení.

void setSlope(double slope_);

Funkce nastavuje spád.

void setAspect(double aspect_);

Funkce nastavuje úhel natočení.

Widget:

void on_pushButton_clicked();

Po stisknutí tlačítka Delaunay se vytvoří Delaunayova triangulace.

void on_pushButton_2_clicked();

Po stisknutí tlačítka Clear Points se smažou body pro výpočet.

void on_pushButton_3_clicked();
Po stisknutí tlačítka Clear All se vymaže vše v Canvasu.

void on_pushButton_4_clicked();
Po stisknutí tlačítka Create contours se vytvoří vrstevnice.

void on_pushButton_5_clicked();
Po stisknutí tlačítka Generate se vytvoří automatický terénní tvar dle vybraného profilu nad tlačítkem.

void on_pushButton_6_clicked();
Po stisknutí tlačítka Clear DT se smaže triangulace.

void on_pushButton_7_clicked();
Po stisknutí tlačítka Describe contours se vytvořené vrstevnice popíší.

void on_pushButton_cl_polygon_clicked();
Po stisknutí tlačítka Clear Polygon se smaže ohraničující polygon.

void on_pushButton_load_file_clicked();
Po stisknutí tlačítka Load File se otevře vyhledávací okénko sloužící k načtení textového souboru se souřadnicemi pro výpočet.

void on_lineEdit_editingFinished();
Po dokončení editace se nastaví z_min.

void on_lineEdit_2_editingFinished();
Po dokončení editace se nastaví z_max.

void on_lineEdit_3_editingFinished();
Po dokončení editace se nastaví interval vrstevnic.

void on_lineEdit_4_editingFinished();
Po dokončení editace se nastaví interval hlavních vrstevnic.

`void on_checkBox_clicked(bool checked);`
Po zaškrtnutí lze kreslit ohraničující polygon.

void on_pushButton_slope_clicked();
Po stisknutí tlačítka Slope se vytvoří barevné znázornění spádu.

void on_pushButton_aspect_clicked();
Po stisknutí tlačítka Aspect se vytvoří barevné znázornění úhlu natočení trojúhelníků.

void on_comboBox_2_currentIndexChanged(int index);
Po změně indexu se nastaví příslušná barva pro vykreslení spádu.

void on_comboBox_3_currentIndexChanged(int index);

Po změně indexu se nastaví příslušná barva pro vykreslení úhlu natočení trojúhelníků.

void on_pushButton_aspect_clicked();

Plonková funkce, nenalezeno, jak bez následných problémů odstranit.

9. Závěr

Omlouváme se, ale z důvodu časové tísně dodáme závěr a zhodnocení algoritmu do konce tohoto týdne...