ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

FAKULTA STAVEBNÍ, OBOR GEODÉZIE A KARTOGRAFIE

KATEDRA GEOMATIKY

Název předmětu

ALGORITMY DIGITÁLNÍ KARTOGRAFIE A GIS

Úloha	Název úlohy
3	Digitální model terénu

Akademický rok	Semestr	Studijní skupina	Vypracoval	Datum	Klasifikace
2018/2019	3.	60	Janovský Michal karving47@gmail.com	22. 11. 2018	

1 Obsah

2 Seznam obrázků

1 Zadání úlohy

Úloha č. 3: Digitální model terénu

Vstup: $P = \{p_1, \ldots\ldots, p_n\}, p_i = [x, y_i\ z_i]$
Výstup: Polyedrický DMT nad množinou P představovaný vrstevnicemi doplněný vizualizací sklonu a expozice trojúhelníků

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 představující 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 STI.

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 tři strany formátu A4.

Bonusové úlohy:

- Triangulace nekonvexní oblasti zadané polygonem
- Výběr barevných stupnic při vizualizaci sklonu a expozice
- Automatický popis vrstevnic
- Automatický popis vrstevnic respektujících kartografické zásady
- Algoritmus pro automatické generování terénních útvarů
- 3D vizualizace terénu s využitím promítání
- Barevná hypsometrie

2 Údaje o bonusových úlohách

V rámci úlohy byly implementovány bonusové úlohy:

- Automatický popis vrstevnic
- Algoritmus pro automatické generování terénních útvarů

3 Popis a rozbor problému
Úkolem je vytvoření programu, který nad množinou bodů vytvoří trojúhelníkovou síť, vytvoří vrstevnice a pro jednotlivé trojúhelníky sítě vypočte jejich sklon a expozici. Pro tvorbu trojúhelníkové sítě byl použit algoritmus inkrementální konstrukce Delaunay triangulace ve 2D.

4 Popisy algoritmů formálním jazykem
Metody použité v této úloze, byly implementovány v programovacím jazyce C++ v prostředí Qt Creator.

4.1 Delaunay triangulace
Vlastnosti:
- V opsané kružnici trojúhelníka neleží žádný jiný bod
- Maximalizuje min. úhel
- Vůči kritériu minimálního úhlu je optimální jak lokálně, tak globálně
- Jednoznačná triangulace pokud jsou právě 3 body na kružnici

Postup tvorby trojúhelníkové sítě:
1. Nalezení pivota
2. Nalezení nejbližšího bodu od pivota
3. Vytvoření hrany
4. Hledání Delaunayova bodu
5. Vytvoření zbývajících hran trojúhelníka
6. Vložení hran do trojúhelníka

4.2 Tvorba vrstevnic
Vrstevnice byly vypočteny pomocí lineární interpolace, a to na hranách trojúhelníků vzniklých předešlou triangulací. Z výšek koncových bodů hran bylo nejprve určeno, zda tato hrana obsahuje průsečík s vrstevnicí, a to ze vztahu: $(z - z_i)(z - z_{i+1})$
Pokud je tento vztah roven 0, celá hrana náleží vrstevnici, pokud >0 neobsahuje průsečík a pokud <0 počítáme průsečík.

Průsečík:

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

4.3 Výpočet sklonu
Sklon je úhel mezi normálou trojúhelníka a svislicí.
V programu je sklon vizualizován odstíny šedi.

Svislice:

$$n = (0,0,1)$$

Normála:

$$n_t = \vec{u} \times \vec{v}$$

Sklon:

$$\varphi = acos\left(\frac{n_t n}{|n_t||n|}\right)$$

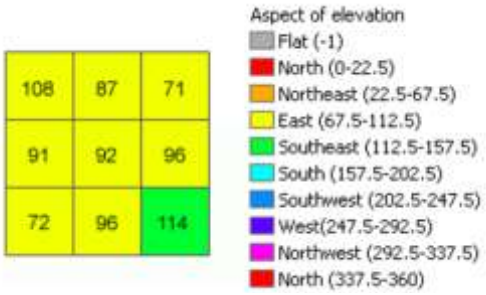
4.4 Výpočet orientace
Orientace (expozice) je brána jako orientace jednotlivých trojúhelníků vůči světovým stranám.
V programu je sklon vizualizován 9 barvami znázorňující směr světových stran.

Normála:

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

Expozice:

$$A = atan2\left(\frac{n_x}{n_y}\right)$$



Obrázek 1 - barevná vizualizace aspect

4.5 Generování bodů

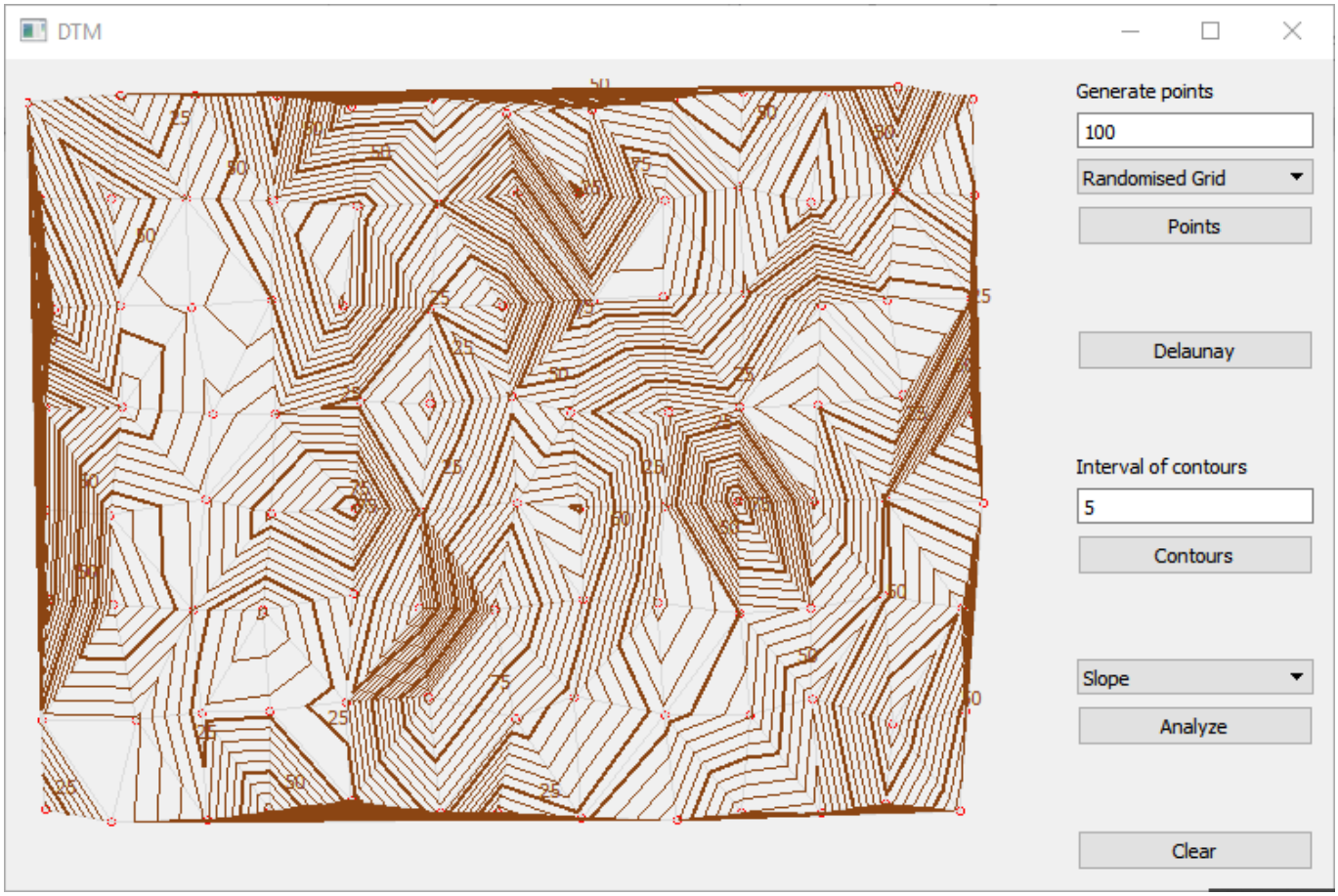
V rámci jedné z bonusových úloh byly vytvořeny algoritmy na generování bodů, pro použití v této úloze. Konkrétně byly navrženy algoritmy na generování kupy, údolí, spočinku, hřbetu a randomizovaného gridu.

4.5.1 Randomizovaný grid

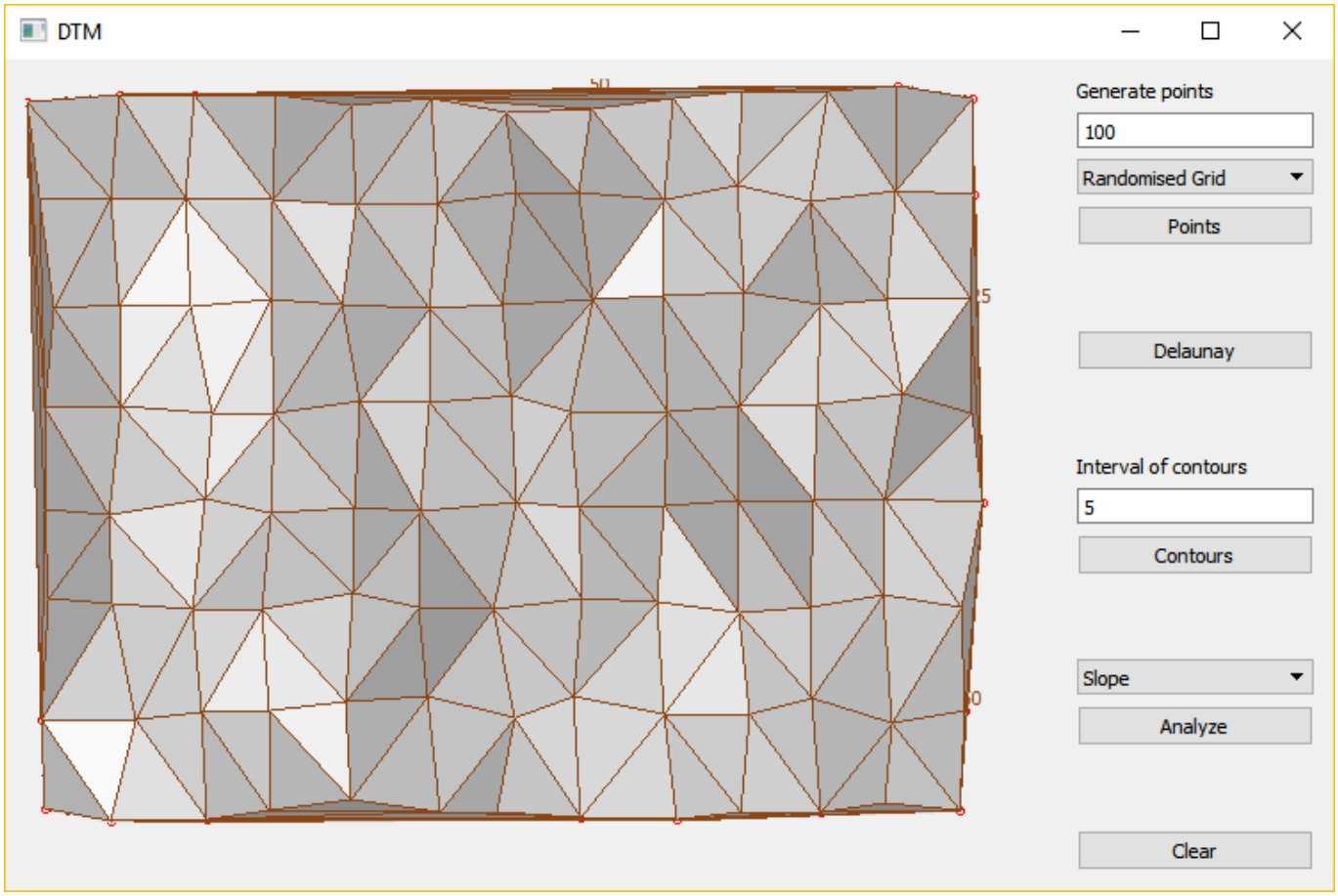
V počátku byl záměr vytvořit grid, ale ukázalo se, že právě na gridu algoritmus pro tvorbu trojúhelníkové sítě kolabuje. Bylo tedy třeba grid upravit, a to tak, aby mezi body nebyla stejná vzdálenost, čehož bylo docíleno připočtením náhodného čísla k souřadnicím bodů.

```
    int h = size.height()-10;
    int w = size.width()-10;
    std::vector<QPoint3D> pts;
    double gapX = h/ceil(std::sqrt(n));
    double gapY = w/ceil(std::sqrt(n));

        for(double x = 10; x < w; x += gapX){
            for(double y = 10; y < h; y += gapY){
                double z = rand()%100;
                pts.push_back(QPoint3D(x+ rand()%10 - rand()%10,y + rand()%10 - rand()%10,z));
            }
        }
    return pts;
}
```

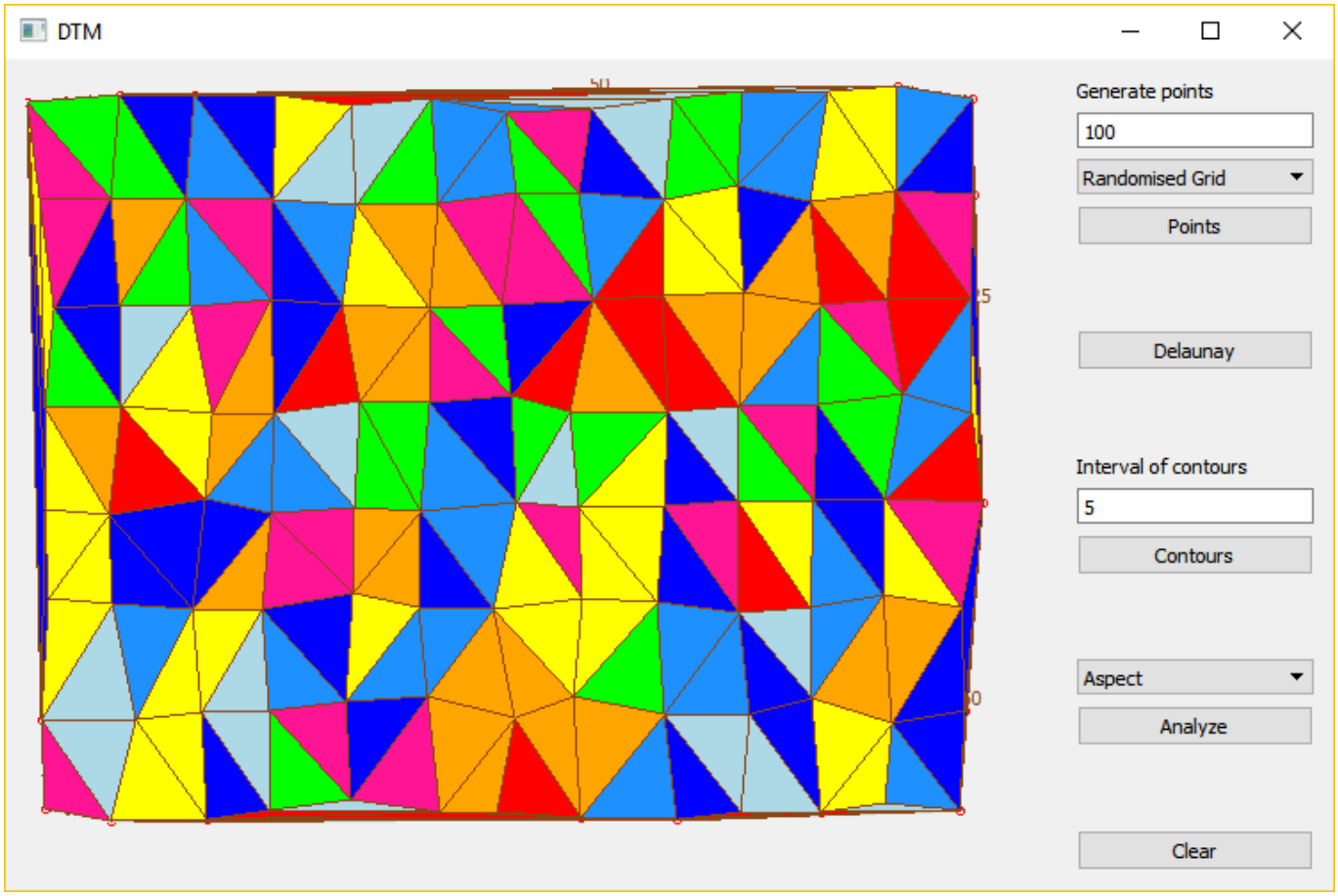
V kódu můžeme vidět použití třídy QSize, která nám může vrátit velikost pracovního okna, čehož se zde využívá ke generování bodů po celém okně, díky čemuž nezáleží na velikosti okna/monitoru, kde pracovní plocha bude vždy rovnoměrně zaplněna body. U vkládání bodů do vektoru dochází k „randomizaci“ gridu, kde se k souřadnicím XY přidává hodnota od -10 do +10. Na tomto principu fungují i další terénní útvary, pro něž bylo implementováno generování bodů.



Obrázek 2 - vygenerované body a vrstevnice - randomised grid



Obrázek 3 - randomised grid - slope

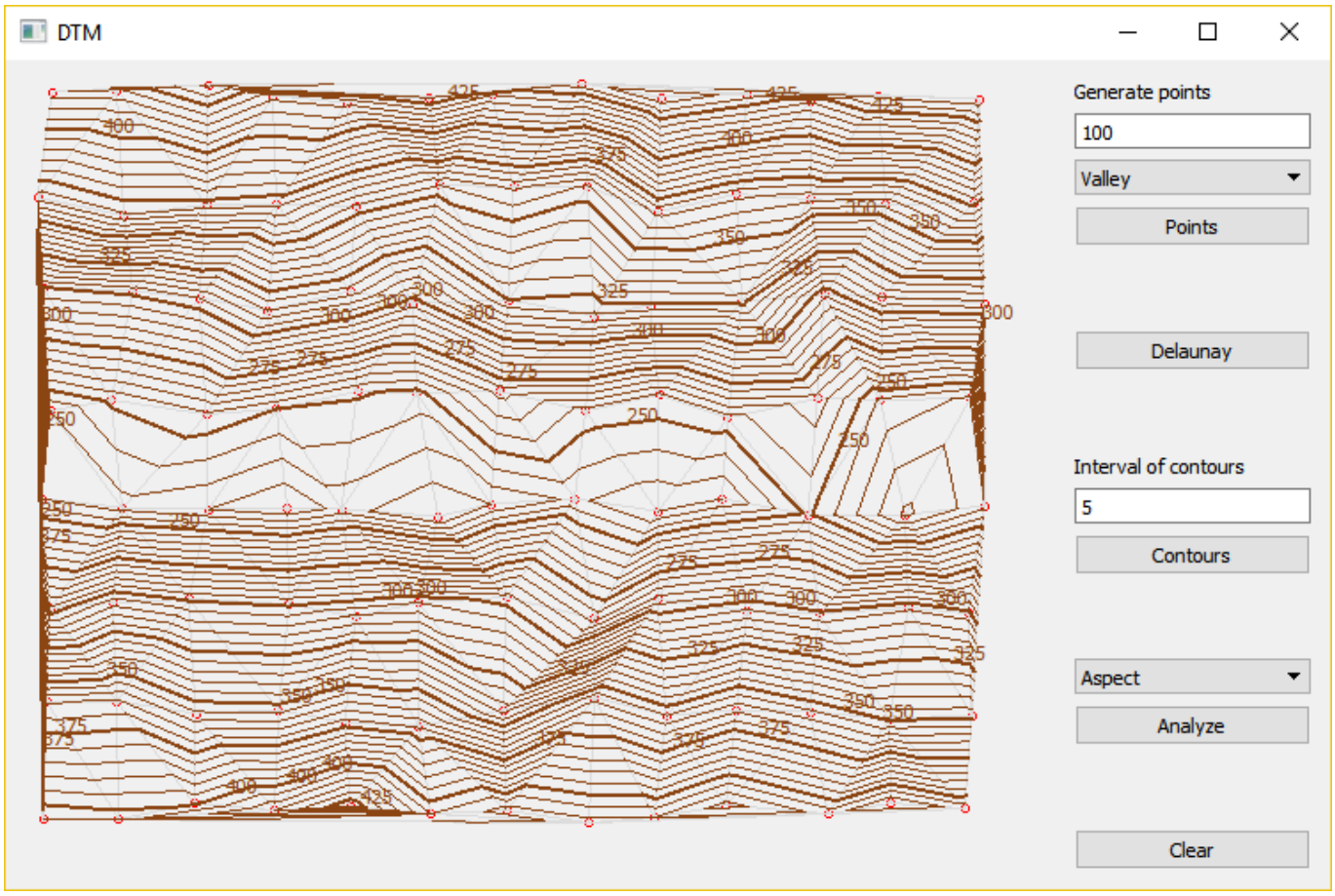


Obrázek 4 - randomised grid - aspect

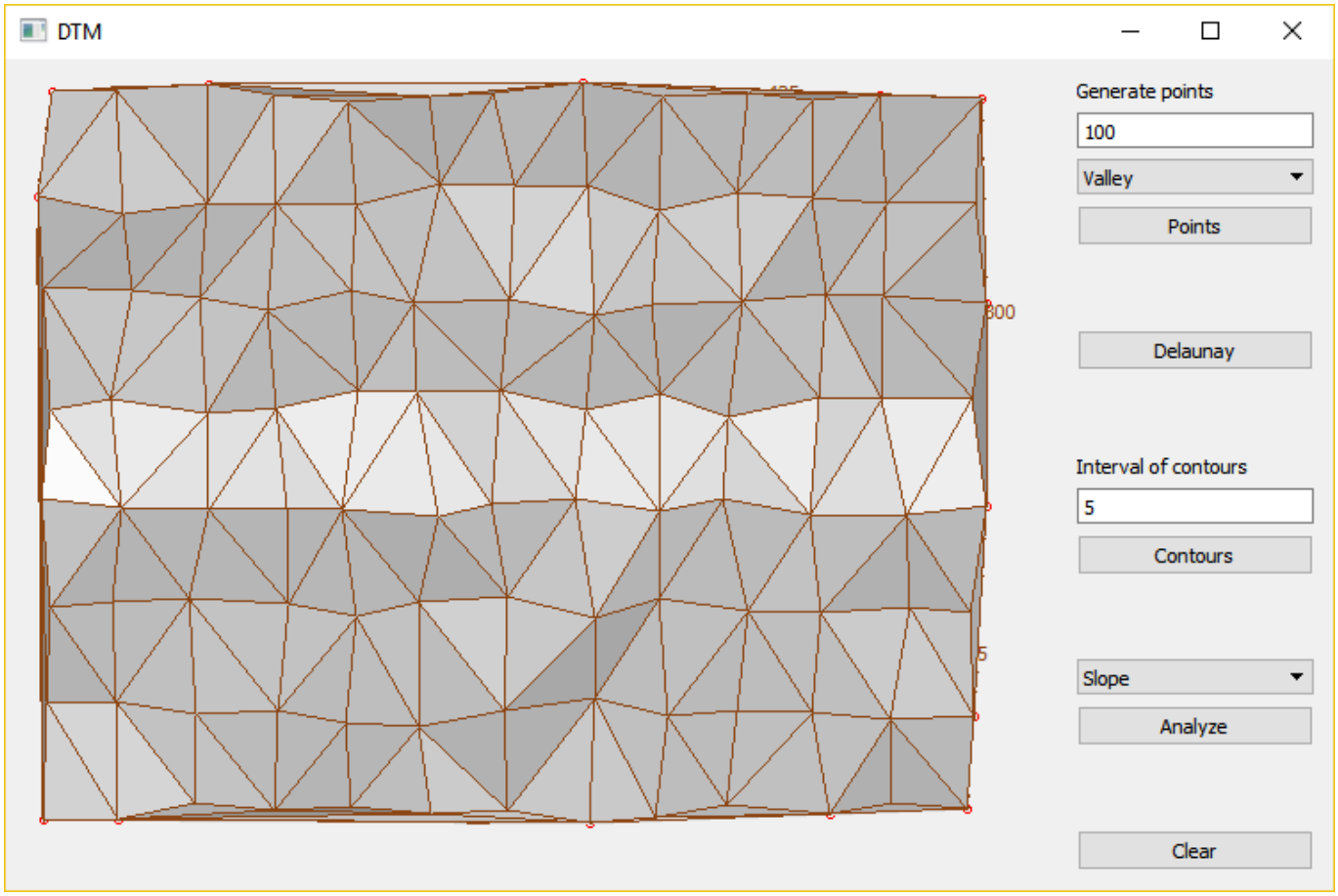
4.5.2 Údolí

Terénní útvar údolí je založen na randomizovaném gridu, jedinou změnou zde jsou Z souřadnice. Body jsou rozděleny na 2 poloviny podle jejich umístění v okně. Horní polovina bodů má spolu s rostoucí Y souřadnicí klesající Z souřadnici, naopak dolní polovina má s rostoucí Y souřadnicí rostoucí Z souřadnici. Výsledkem je jsou 2 plochy, které se protínají horizontálně uprostřed okna a vzniká údolí

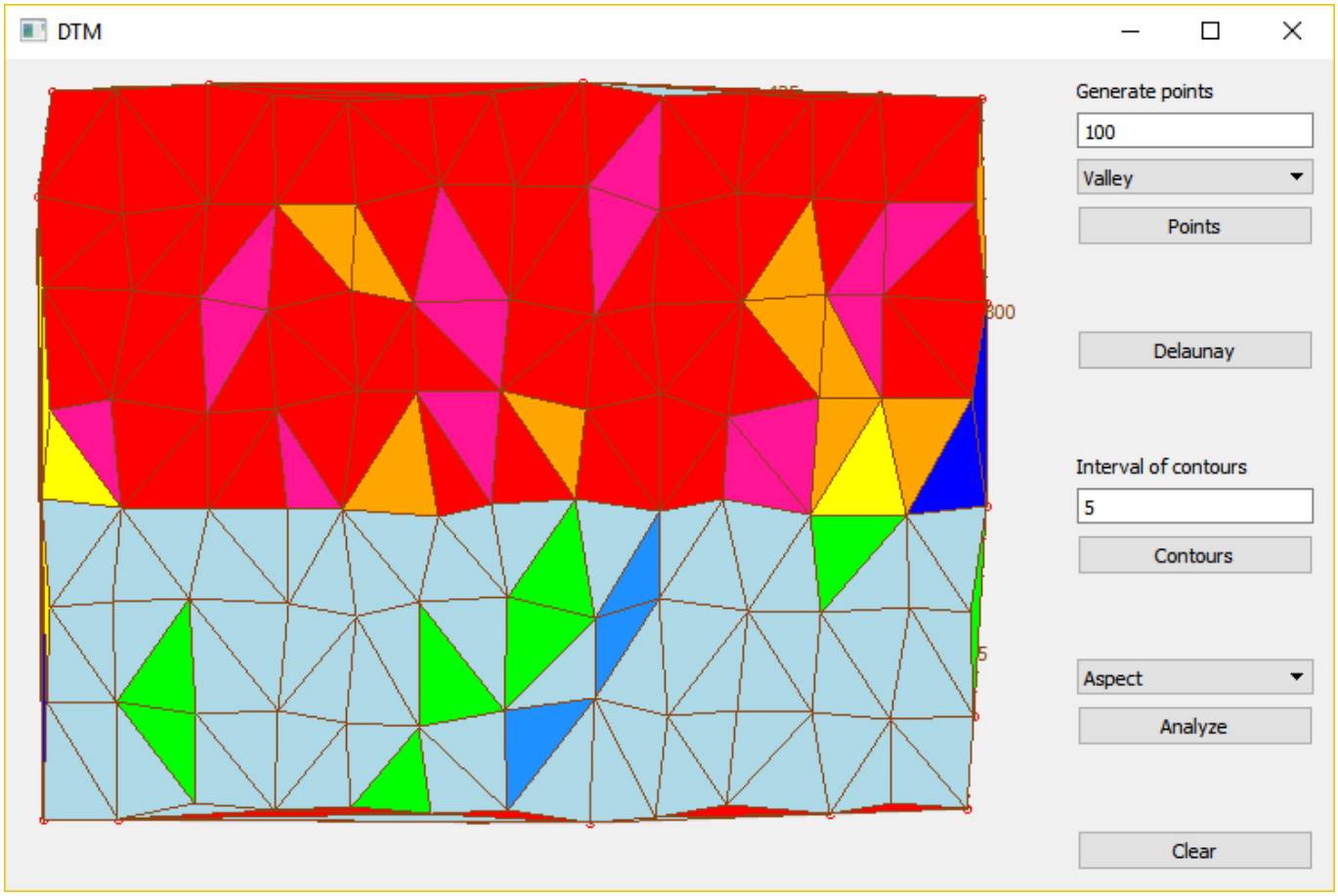
```
if (y < h/2)
        z = round(h-y - rand()%20 + rand()%20);
else
        z = round(y + rand()%20 - rand()%20);
```



Obrázek 5 - vygenerované body a vrstevnice - valey



Obrázek 6 - valey - slope



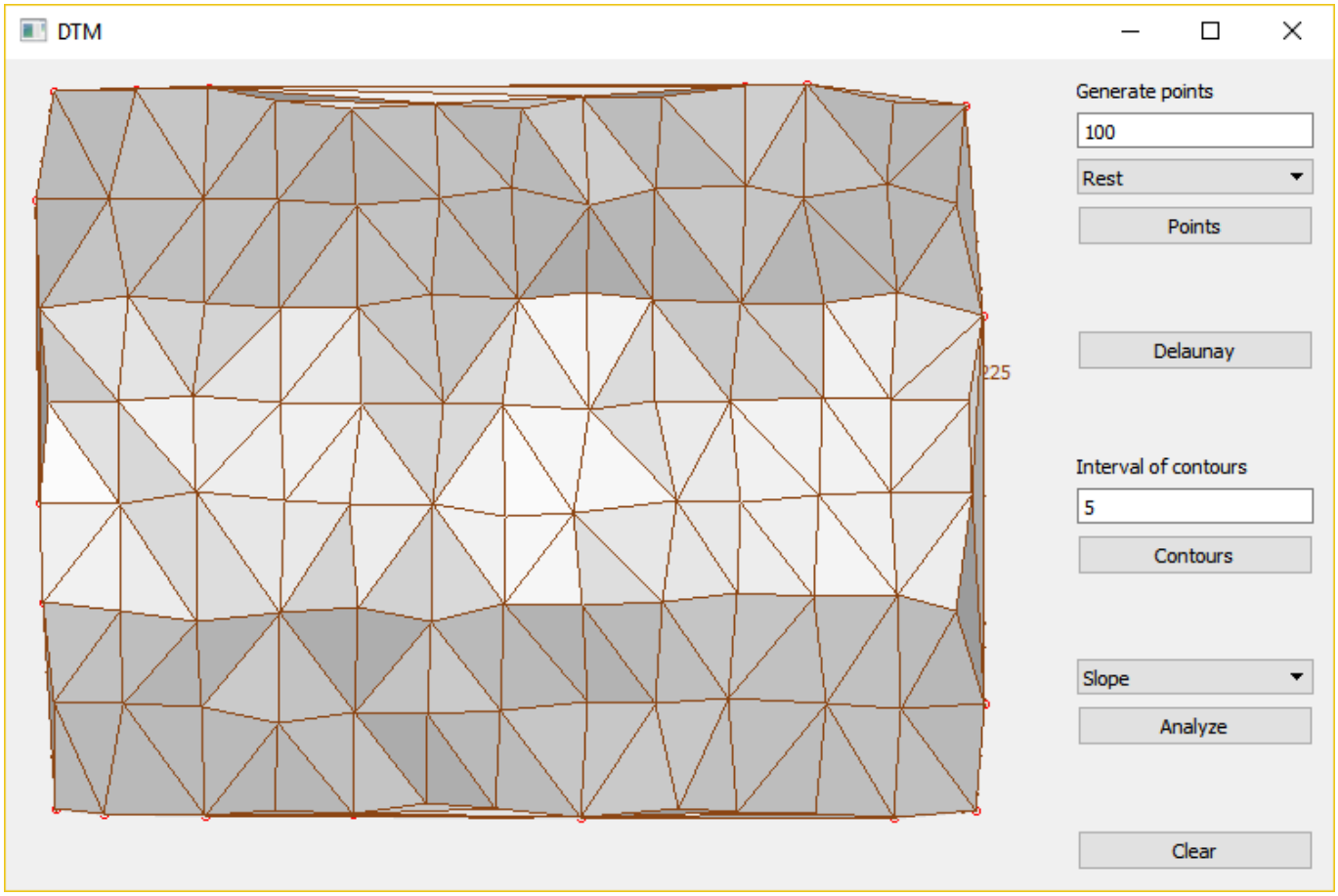
Obrázek 7 - valey - aspect

4.5.3 Spočinek

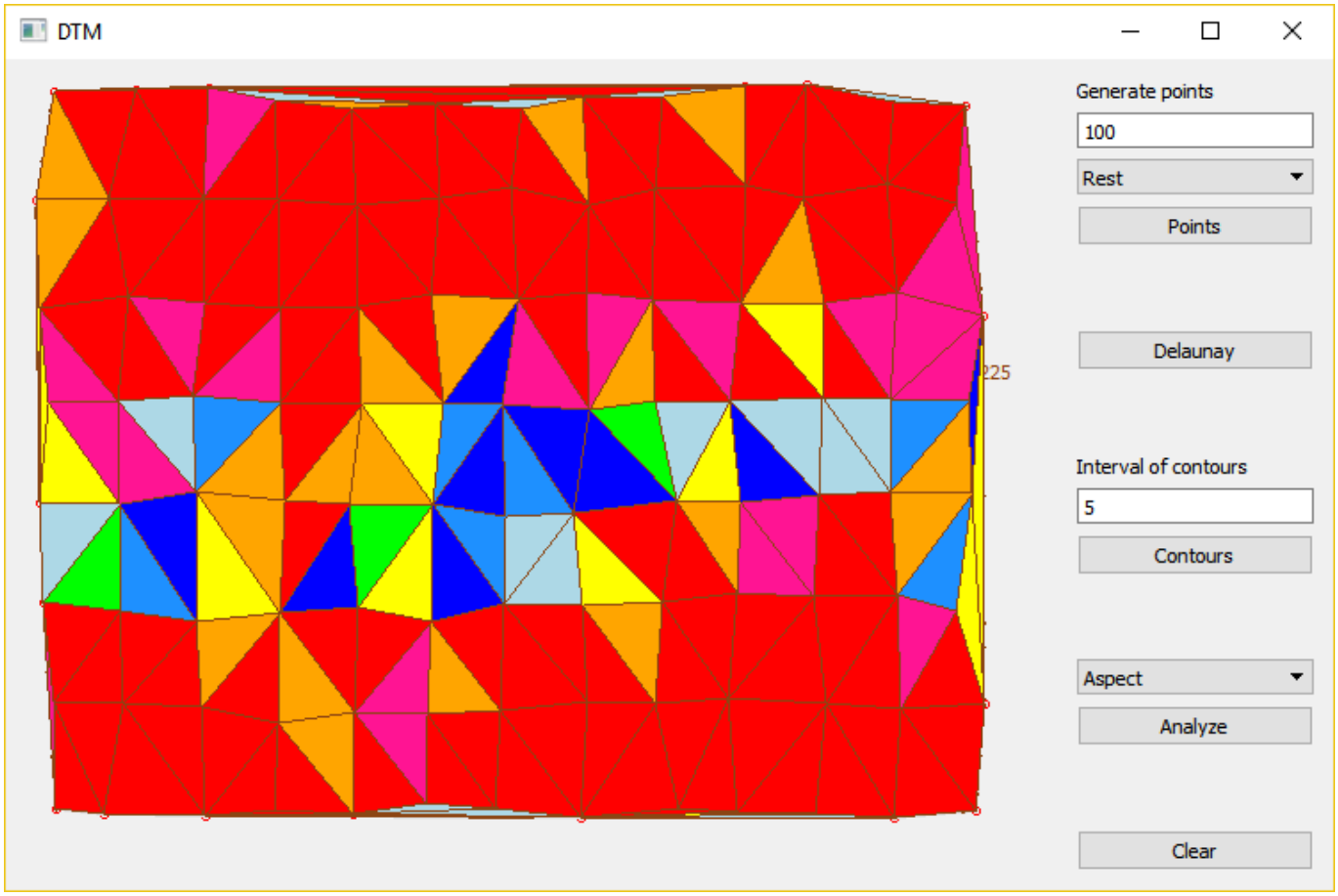
Terénní útvar spočinek je založen na randomizovaném gridu, jedinou změnou zde jsou Z souřadnice. Princip je zde podobný s údolím, kde body jsou rozděleny na 3 části podle jejich umístění v okně. Horní 1/3 bodů s rostoucí Y klesá Z, dolní 1/3 bodů také klesá Z s rostoucí Y. Rozdíl je však u prostřední třetiny, kde Z je přibližně stejná čímž vzniká spočinek. Nevýhodou tohoto postupe je to, že velikost spočinku je poměrně veliká a není vidět část spočinku, kde spočinek přechází do normálního svahu. Další nevýhodou je nutnost volby výšek, které se musí zvolit tak, aby mezi rovnou plochou spočinku a svahy nad/pod ním nebyl svah příliš prudký.

```
if (y < h/3) //upper
    z = (h-y - rand()%20 + rand()%20-75);

else if (y > 2*h/3) //lower
    z = (h-y + rand()%20 - rand()%20+75);

else    //middle
    z = (h/2 - rand()%20 + rand()%20);
```

Obrázek 8 - vygenerované body a vrstevnice - rest



Obrázek 9 - rest - slope



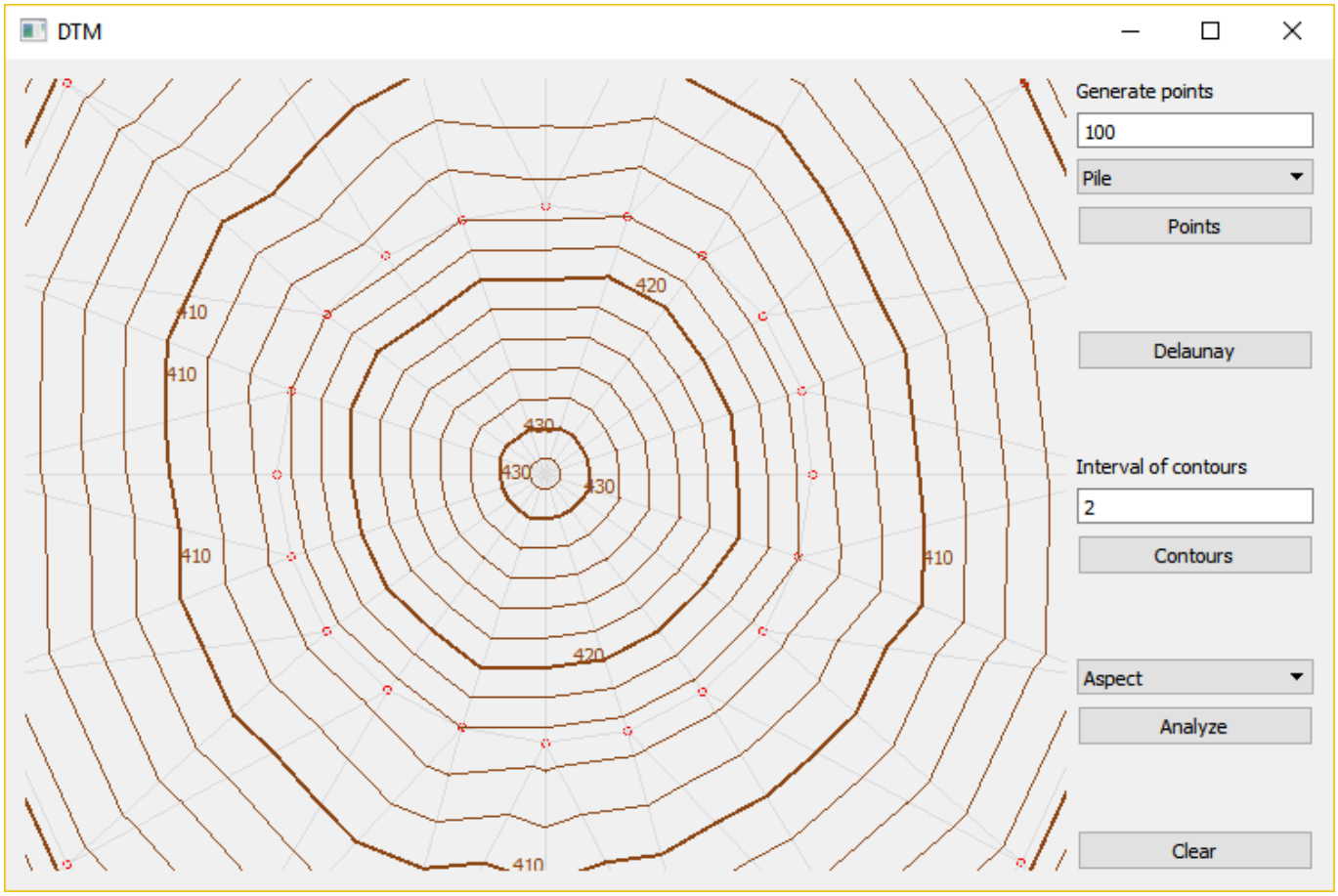
Obrázek 10 - rest - aspect

4.5.4 Kupa

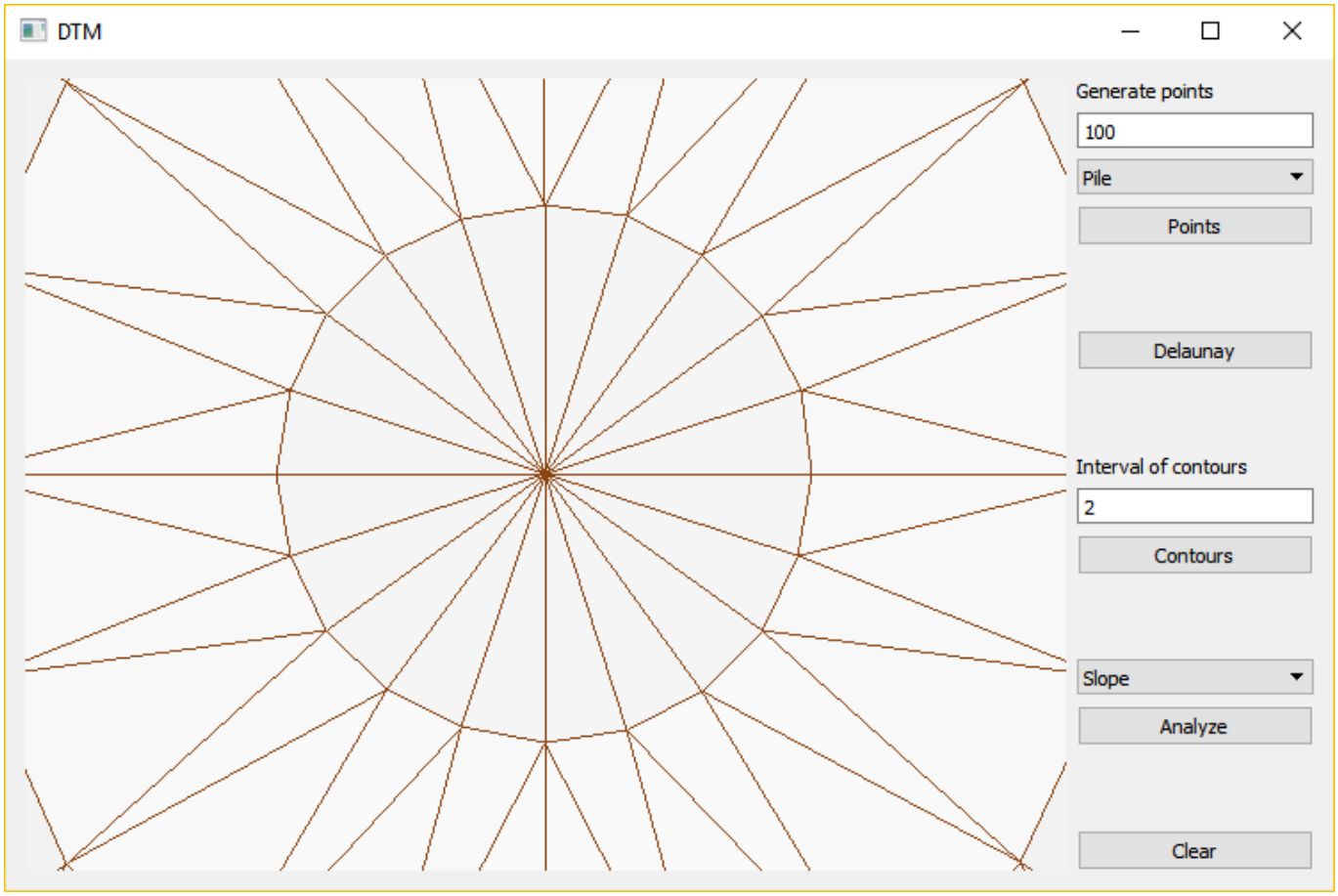
Kupa byla dělána rozdílně než ostatní terénní útvary. Nebylo využito žádného gridu, ale body byly generovány jako kružnice. U standardní kružnice však znovu selhává algoritmus pro sestavení trojúhelníku, kde bylo znova do generování třeba zavést náhodné chyby, aby body od sebe měli různý rozestup. Při generování bodů kružnice se využíval úhel a délka, kde u obou byly zavedeny náhodné hodnoty. Pro znázornění je přiložena část kódu z 1 takové kružnice.

```
for(int i = 0;i<n/5;i
{
    p.setX(center.x() + (150+rand()%3)*cos(i*(fi+rand()%1)));
    p.setY(center.y() + (150+rand()%3)*sin(i*(fi+rand()%1)));
    p.setZ(415+rand()%2);
    pts.push_back(p);
}
```

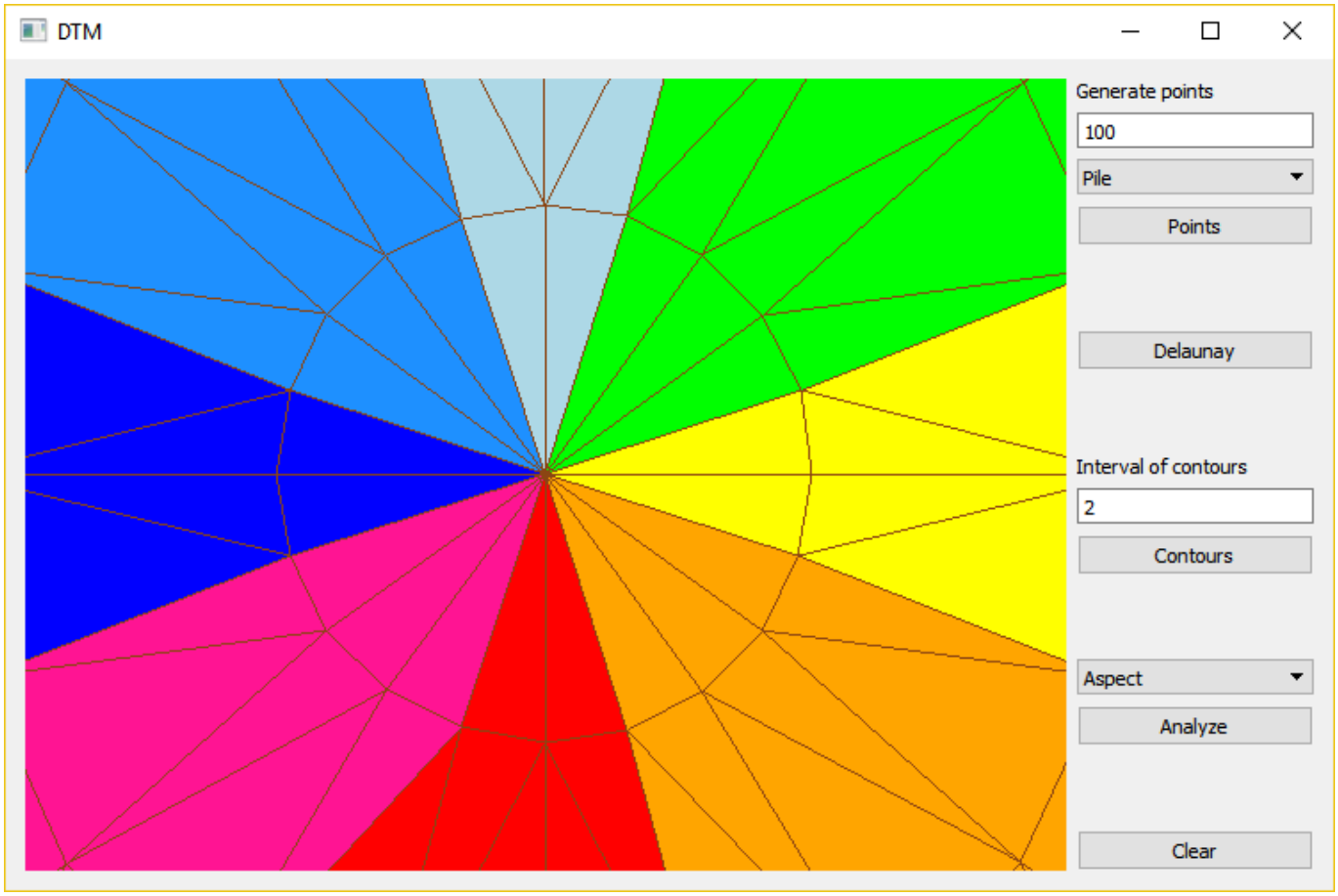
Takovýchto kružnic bylo vytvořeno několik plus středový bod, ze kterého se kružnice generovaly. Problémy s tímto postupem nastávají právě poblíž středu kružnice (při malém poloměru) kde zavedená randomizace je velmi zřetelná a vrstevnice „se zubatí“. Tento problém se však při tomto postupu nedá plně odstranit, jelikož při vyšším vyhlazení bodů (aby se vrstevnice „nezubatily“) začíná kolabovat algoritmus pro tvorbu trojúhelníkové sítě.



Obrázek 11 - vygenerované body a vrstevnice - pile



Obrázek 12 - pile - slope

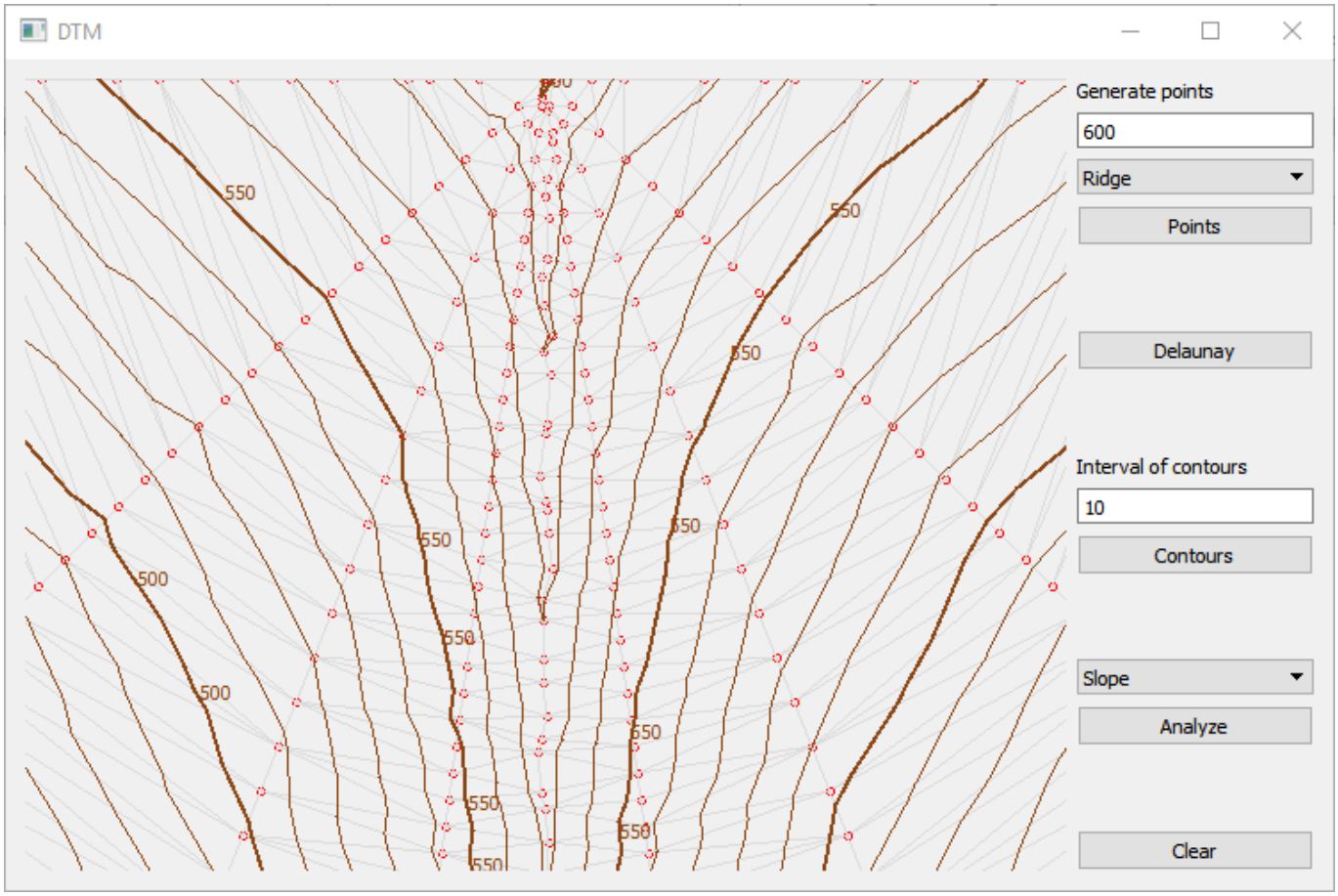


Obrázek 13 - pile - aspect

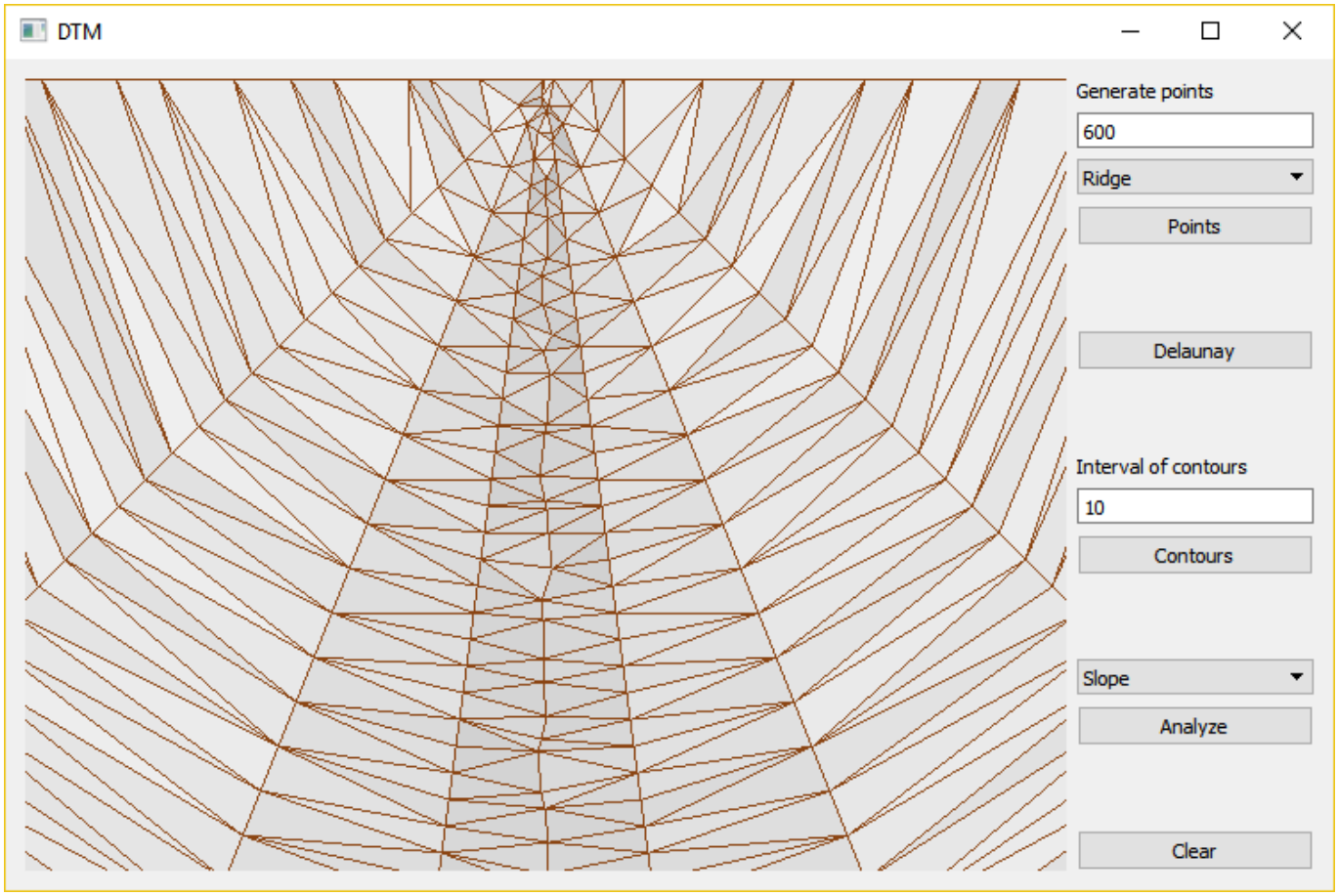
4.5.5 Hřbet

U hřbety byl zpočátku zvolen postup podobný jako u Kupy, tedy generování pomocí kružnic, nicméně kvůli problémům bylo od tohoto řešení opuštěno. Byla zvolena metoda přírůstků XY k hlavnímu bodu ($Y = 0$ a $X = w/2$), tedy bodu uprostřed horní hrany okna. Od tohoto bodu byla vedena linie hřbetu směrem dolů, která rovnoměrně klesá v Z. Následně byly vedeny další linie zrcadlově vlevo a vpravo od linie hřbetu, které klesaly v Z prudčeji než samotný hřbet. Pro ukázku je přiložen kus kódu pro vygenerování jedné této linie.

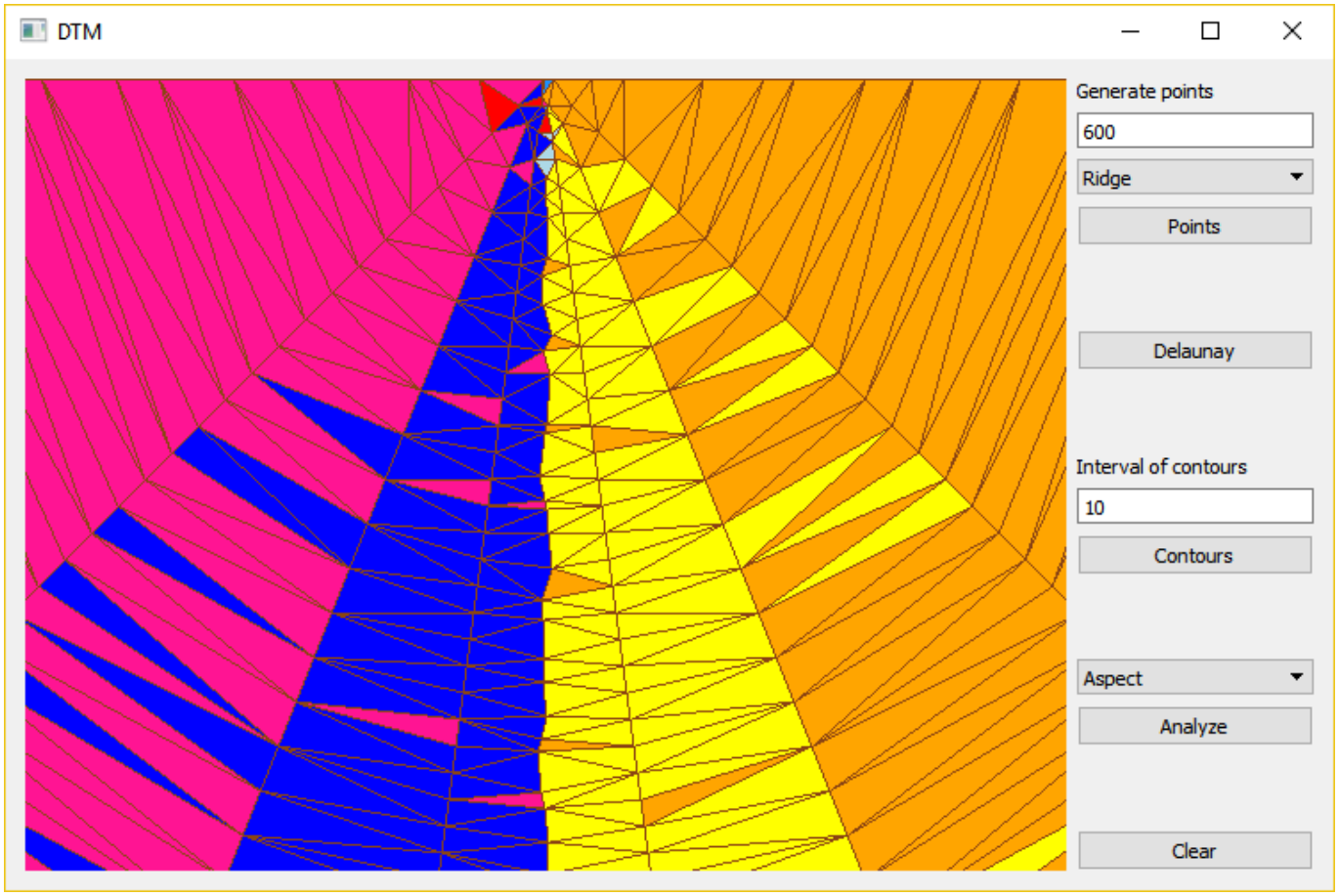
```
fůr(int i = 0;i<n/7;i++)
{
    p.setX(center.x() + 2*i );
    p.setY(center.y() + (i*15));
    p.setZ(n-2*i-rand()%3);
    pts.push_back(p);
}
```



Obrázek 14 - vygenerované body a vrstevnice - ridge



Obrázek 15 - ridge - slope



Obrázek 16 - ridge - aspect

5 Problematické situace a jejich rozbor + ošetření těchto situací v kódu

Jediným nalezeným problémem je nemožnost použití Delaunay triangulace na takové množiny bodů, kde jsou body pravidelně rozmístěny (např. grid). Tento problém je způsoben tím, jak tato metoda triangulace funguje. Hledáme totiž bod, který spolu s 2 body úsečky tvoří nejmenší opsanou kružnici. Problém nastává, když takových bodů je více a program se zasekne (a spadne). Jiné řešení, než je použití jiné metody konstrukce trojúhelníkové sítě není autorovi známo. Z tohoto důvodu při generování bodů byla do polohy bodů zanesena náhodná hodnota v souřadnicích, aby se možnost těchto případů minimalizovala.

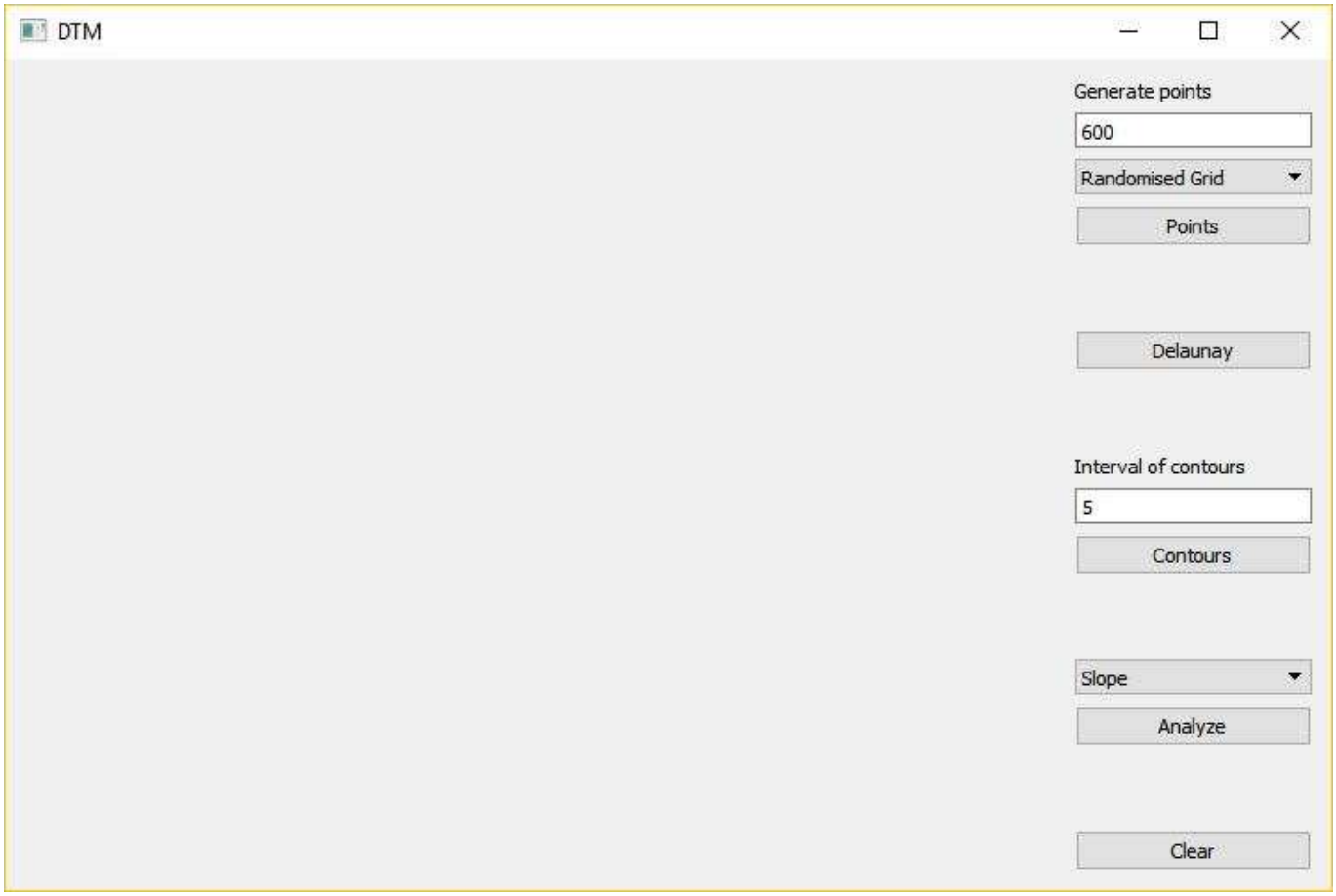
6 Vstupní data, formát vstupních dat, popis

Vstupními daty je vygenerovaná množina bodů, která představuje významné terénní útvary. Množina je tvořena body o souřadnicích x,y,z. Druhým možným vstupem je

7 Výstupní data, formát výstupních dat, popis

Výstupem z programu je trojúhelníková síť z množiny vygenerovaných bodů, nad kterou jsou zkonstruovány a vizualizovány vrstevnice a sklonu a orientace jednotlivých trojúhelníků.

8 Printscreen vytvořené aplikace



Obrázek 17 – idle program po spuštění

9 Dokumentaci: popis tříd, datových položek a jednotlivých metod

9.1 Třída Algorithms

Třída obsahující veškeré početní výkony pro vygenerování trojúhelníků, vytváření vrstevnic a výpočtu sklonů a orientací trojúhelníků

9.1.1 Metody třídy Algorithms

```
static TPosition getPointLinePosition(QPoint3D &q, QPoint3D &a, QPoint3D &b);
```
metoda určující vztah bodu q k úsečce tvořené body a,b

```
static double getCircleRadius(QPoint3D &p1, QPoint3D &p2, QPoint3D &p3, QPoint3D &c);
```
vypočte střed kružnice c, která prochází body p1, p2, p3

```
static int getNearestPoint(QPoint3D &p, std::vector<QPoint3D> &points);
```
vrací index nejbližšího z bodů ve vektoru points k bodu p

```
static double distance(QPoint3D &p1, QPoint3D &p2);
```
vrací hodnotu vzdálenosti mezi body p1 a p2

```
static int getDelaunayPoint(QPoint3D &s, QPoint3D &e, std::vector<QPoint3D> &points);
```
vrací index bodu kde má spojnice a bod minimální poloměr opsané kružnice

```
static std::vector<Edge> DT(std::vector<QPoint3D> &points);
```
provádí triangulaci a vrací seznam hran

```
static QPoint3D getContourPoint(QPoint3D &p1, QPoint3D &p2, double z);
```
nalezne bod o zadané výšce z mezi body p1 a p2

```
static std::vector<Edge> createContours(std::vector<Edge> &dt, double z_min, double z_max, double dz);
```
vytvoří vektor hran vrstevnic s rozestupem dz

```
static double getSlope(QPoint3D &p1, QPoint3D &p2, QPoint3D &p3);
```
vypočítá sklon zadaného trojúhelníka

```
static double getAspect(QPoint3D &p1, QPoint3D &p2, QPoint3D &p3);
```
vypočítá expozici zadaného trojúhelníka

```
static std::vector<Triangle> analyzeDTM(std::vector<Edge> &dt);
```
vypočítá sklon a expozici pro všechny trojúhelníky

9.2 Třída Draw

Slouží k vykreslení bodů, trojúhelníků, vrstevnic, a vybarvení trojúhelníků.

9.2.1 Datové položky třídy Draw

```
std::vector<QPoint3D> points;
```
vektor bodů

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

```
std::vector<Edge> contours;
```
vektor vrstevnic

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

9.2.2 Metody třídy Draw

```
void paintEvent(QPaintEvent *e);
```
metoda volaná při každém kreslení

```
void mousePressEvent(QMouseEvent *e);
```
slouží k vykreslení manuálně „naklikaných" bodů

```
void clearPoints()
smazání množiny bodů

void clearDT();
smazání trojúhelníků

std::vector<QPoint3D> & getPoints()
slouží k předání hodnoty points

std::vector<Edge> & getDT()
slouží k předání trojúhelníků

void setDT(std::vector<Edge> &dt_)
„setr" nastavení hran triangulace

void setContours(std::vector<Edge> &contours_)
„setr" nastavení vrstevnic

void setDTM(std::vector<Triangle> &dtm_)
„setr" nastavení trojúhelníků
```

9.3 Třída Widget

Třída pro práci s grafickým prostředím.

9.3.1 Sloty třídy Widget

```
void on_Points_clicked();
po kliknutí se vygeneruje množina bodů P

void on_Delaunay_clicked();
po kliknutí se nad množinou P sestaví trojúhelníková síť

void on_Contours_clicked();
po kliknutí se nad trojúhelníkovou sítí vygenerují vrstevnice

void on_Clear_clicked();
po kliknutí smaže vše a navrátí aplikaci do počátečního stavu

void on_Analyze_clicked();
po kliknutí se vypočte sklon a expozice a graficky se znázorní
```

9.4 Třída GeneratePoints

Slouží ke generování množin bodů P.

9.4.1 Metody třídy GeneratePoints

```
static std::vector<QPoint3D> generateGrid(int &n,QSize &size);
static std::vector<QPoint3D> generateKupa(int &n,QSize &size);
static std::vector<QPoint3D> generateUdoli(int &n,QSize &size);
static std::vector<QPoint3D> generateSpocinek(int &n,QSize &size);
static std::vector<QPoint3D> generateHrbet(int &n,QSize &size);
metody pro generování množin bodů P o n bodech představující zvolený terénní útvar
```

9.5 Třída QPoint3D

Třída odvozená od třídy QPoint rozšířená o souřadnici Z

9.5.1 Datové položky třídy QPoint3D

```
double z
hodnota výšky bodu
```

9.5.2 Metody třídy QPoint3D

```
double getZ()
„getr“ vrátí výšku bodu

void setZ(double z_)
„setr“ nastaví výšku bodu
```

9.6 Třída Edge

Třída Edge slouží k ukládání hran trojúhelníků

9.6.1 Datové položky třídy Edge

```
QPoint3D s, e;
Počáteční a koncové body hrany
```

9.6.2 Metody třídy Edge

```
QPoint3D & getS()
QPoint3D & getE()
„getr“ navrátí počáteční/koncový bod

void switchOr()
obrácení orientace hrany

bool operator == (const Edge &e_)
přetížení operátoru == k porovnávání „hrana == hrana“
```

9.7 Třída Triangle

Třída složená pro ukládání a práci s trojúhelníky

9.7.1 Datové položky třídy Triangle

```
QPoint3D p1, p2, p3;
Body definující trojúhelník

double slope, aspect;
hodnoty sklonu a expozice trojúhelníků
```

9.7.2 Metody třídy Triangle

```
QPoint3D getP1()
QPoint3D getP2()
QPoint3D getP3()
Vrací jednotlivé body trojúhelníka
double getSlope()
double getAspect()
Vrací hodnoty sklonu a expozice trojúhelníka
```

9.8 Třída SortByXAsc

Slouží k seřazení bodů podle X souřadnice vzestupně

10 Závěr, možné či neřešené problémy, náměty na vylepšení

10.1 Závěr

Pomocí Delaunay triangulace byly sestrojeny trojúhelníky, z nichž se dále generovaly vrstevnice a byly vypočteny hodnoty sklonu a expozice. Jako data pro úlohu byla vytvořena třída GeneratePoints, ve které jsou metody pro generování umělých dat představující terénní útvary kupa, spočinek, údolí a hřbet. Ukázka vytvořených teréních prvků i s popisem je k nalezení v kapitole 4.5.

Při vytváření třídy GeneratePoints byl zjištěn největší nedostatek Delaunay triangulace, a to že tato triangulace nefunguje, pokud mají body stejný rozestup. Toto bylo zjištěno při pokusu použít tuto metodu na gridu a dále na kružnici s body o stejném rozestupu, kdy při pokusu použít tuto triangulaci program automaticky spadne. Tento nedostatek byl eliminován přidáním náhodných hodnot k souřadnicím generovaných bodů, čímž tyto body přestávají mít stejný rozestup a při Delaunay triangulaci dokáže algoritmus říci, který bod je blíže (odstraní se situace, kdy 2 body jsou stejně daleko). Dále u generování trojúhelníkové sítě nejsou ošetřeny možnost „děr“ v datech, tedy oblastí, kde trojúhelníky vytvářet nechceme a okrajů dat, kde vznikají trojúhelníky mimo zájmové oblasti.

Dalším z nedostatků je generování vrstevnic, kde vygenerované vrstevnice jsou velmi kostrbaté a nejsou vhodné pro další využití. Toto je způsobeno kombinací faktorů jako je například počet a rozmístění bodů, tak i metoda použité interpolace. Dalšími problémy s vrstevnicemi je popis vrstevnic, kde při vstupu náhodně vygenerovaných dat se jen těžko dá odhadnout, jak by měl být popis rozmístěn, navíc způsob uložení vrstevnic jako vektor hran všech vrstevnic (a nikoliv jako jednotlivé vrstevnice) ztěžuje implementaci popisu vrstevnic. Problém nastává především u umístění popisu, kde by bylo zapotřebí ošetřit vzdálenosti mezi jednotlivými kótami, kde by bylo třeba zohlednit délku celé vrstevnice a podle ní určit počet kót a jejich umístění řešit např. bufferem, nicméně tyto implementace jsou již na autora kódu příliš časově náročné.

Jako poslední jsou vypočteny a vizualizovány sklon a/nebo expozice. Pro vizualizaci sklonu byly použity barvy stupně šedi a pro expozici byly zvoleny barvy, které jsou použity v software ArcMap. Nevýhodou barevného zobrazení je, že barvy nejsou zcela asociativní (člověk si pod barvou nepředstaví/nedomyslí co je barvou myšleno) a v některých případech přechody mezi barvami mohou být dosti nepřirozené (náhlá změna zobrazených hodnot).

10.2 Náměty na vylepšení

Náměty na vylepšení se týkají především estetické části programu. Při generování jsou mnohdy vygenerované body rozložené nerovnoměrně, což ovlivňuje vzhled vrstevnic. Toto by se dalo vyřešit vyhlazením vrstevnic, nebo rovnoměrným rozmístěním bodů. Dále u popisu vrstevnic je problém s překrýváním kót a vrstevnic, a kóty nejsou natočené ve směru svahu a natočené. Řešením by mohl být buffer bílé barvy kolem kóty, nicméně by se muselo ošetřit, aby buffer nezasahoval i do sousedních vrstevnic. Natočení vrstevnic by se dalo řešit výpočtem úhlu mezi souřadnicovou osou a spojnicí počátečního a koncového bodu a o úhel mezi těmito přímkami otočit kótu vrstevnice.

10.3 Neřešené problémy

Při popisu vrstevnic dochází k tomu, že popis je překrýván vrstevnicí. Dále u popisu vrstevnic se popis generuje u každé hrany, to bylo eliminováno popisem pouze každé X-té hrany, nicméně takto je popis nepravidelný a stává se, že popis je v některých částech plochy častý a na jiných částech chybí.