Obsah

[1 Zadání 2](#_Toc58662877)

[1.2 Údaje o bonusových úlohách 2](#_Toc58662878)

[2 Popis a rozbor problému 3](#_Toc58662879)

[3 Popis algoritmů 3](#_Toc58662880)

[3.1 Delaunay triangulace 3](#_Toc58662881)

[3.2 Tvorba vrstevnic 4](#_Toc58662882)

[3.3 Sklon terénu 5](#_Toc58662883)

[3.4 Orientace terénu 5](#_Toc58662884)

[4 Problematické situace, generátory 5](#_Toc58662885)

[4.X Automatický popis vrstevnic 5](#_Toc58662886)

[4.X Generování terénních tvarů 6](#_Toc58662887)

[*4.X.X Náhodné body* 6](#_Toc58662888)

[*4.X.X Kupa* 6](#_Toc58662889)

[*4.X.X XX* 6](#_Toc58662890)

[5 Popis Aplikace 6](#_Toc58662891)

[5.1 Vstupní data 6](#_Toc58662892)

[5.2 Výstupní data 7](#_Toc58662893)

[6 Dokumentace 7](#_Toc58662894)

[6.1 Třída Algorithms 7](#_Toc58662895)

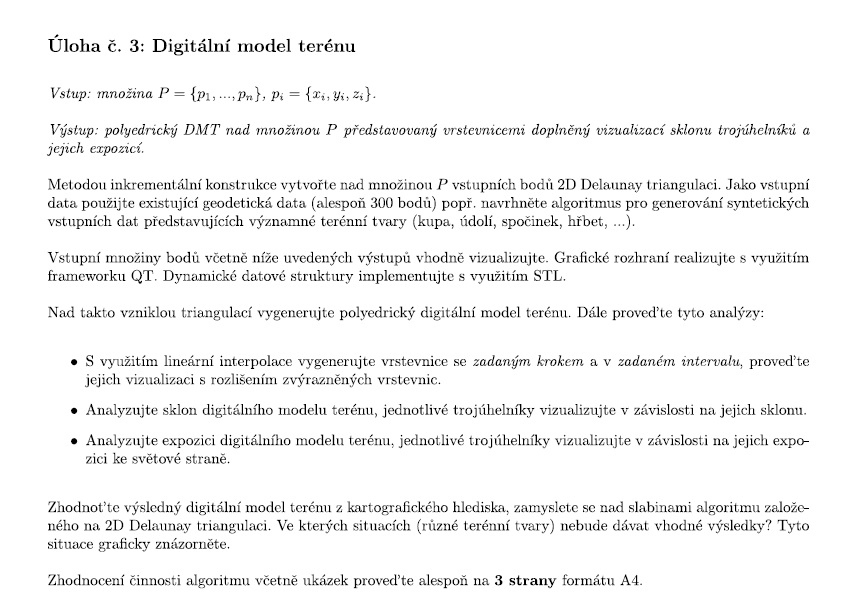
[6.2 Třída Draw 7](#_Toc58662896)

[6.4 Pomocné třídy 7](#_Toc58662897)

[8 Závěr 8](#_Toc58662898)

[Seznam obrázku 9](#_Toc58662899)

### 1 Zadání



## 1.2 Údaje o bonusových úlohách

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

# 2 Popis a rozbor problému

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

# Popis algoritmů

## 3.1 Delaunay triangulace

Delaynay triangulace je metoda pro vytvoření trojúhelníkové sítě. Metoda funguje na principu inkrementální konstrukce, která je založena na postupném přidávání bodů do již vytvořené množiny DT. Nad existující Delaunayovsko hranou *e* je hledán bod p minimalizují poloměr

Tato hraně je orientovaná, bod je hledán pouze nalevo od ní. Pokud se nalevo žádný takový bod nenachází, dojde ke změně orientace hrany.

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

Určení Delaunay bodu p:

1. Určení polohy bodu od dané hrany:
2. Pokut t < 0 bod p se nachází v levé polorovině a je vypočten poloměr r:

kde

Algoritmus Delaunay triangulace:

1. Nalezení pivota *p1*, *p1 = min*(*x*), nalezení bodu *p2* nejbližšího k pivotu
2. Vytvoření hrany *e* = (*p1*, *p2*)
3. Nalezení optimálního Delaunay bodu *p*
4. Pokud *p* nenalezen, prohoď orientaci *e*, tj. *e* = (*p2*, *p1*), znovu krok 3
5. Vytvoř zbývající hrany trojúhelníku, tj. *e2* = (*p2*, *p*) a *e3* = (*p*, *p1*)
6. AEL ← *e*, AEL ← *e2*, AEL ← *e3*
7. DT ← *e*, DT ← *e2*, DT ← *e3*
8. dokud AEL není prázdná:
9. AEL → *e, e* = (*p1*, *p2*) // vezmi první hranu z AEL
10. Prohození orientace *e*, tj. *e* = (*p2*, *p1*)
11. Nalezení optimálního Delaunay bodu *p*
12. Pokud *p* existuje
13. *e2* = (*p2*, *p*), *e3* = (*p*, *p1*)
14. DT ← *e*, DT ← *e2,* DT ← *e3*
15. Prohození orientace *e2* a *e3, tj. e2‘* = (*p*, *p2*), *e3‘* = (*p1*, *p*)
16. Pokud *e2‘* je v AEL
17. AEL → *e2‘*
18. Jinak AEL ← *e2*
19. Pokud *e3‘* je v AEL
20. AEL → *e3‘*
21. Jinak AEL ← *e3*

## 3.2 Tvorba vrstevnic

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

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

## 3.3 Sklon terénu

Sklon terénu je vyjádřen úhlem mezi normálovým vektorem (0,0,1) a normálovým vektorem roviny trojúhelníku. Pro určení úhlu byly použity tyto vzorce:

## 3.4 Orientace terénu

Orientace terénu je vyjádřena azimutem. Azimut je dán průmětem normálového vektoru z roviny trojúhelníku do roviny ρx,y. Pro určení azimutu byly použity tyto vzorce:

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

# 4 Problematické situace, generátory

## 4.X Automatický popis vrstevnic

Pro automatické generování terénních tvarů byly vybrány tyto tvary: Náhodné body *(Random), Kupa (Knoll), XX.* Generátory byly vytvořeny v samostatné třídě.

## 4.X Generování terénních tvarů

Pro automatické generování terénních tvarů byly vybrány tyto tvary: Náhodné body *(Random), Kupa (Knoll), XX.* Generátory byly vytvořeny v samostatné třídě.

## *4.X.X Náhodné body*

Funkce vygeneruje náhodné body s uživatelem zadaného počtu bodů s náhodným rozmístěním. Náhodné body byly generovány pomocí funkce *rand().* Souřadnice *x,y* jsou děleny výškou a šířkou kreslícího plátna, aby nedocházelo k vykreslení mimo něj. Souřadnice z je generována jako desetinné číslo v intervalu <0;1> a vynásobena hodnotou 250. Tento generátor byl použit pro generování všech ostatních bodů.

## *4.X.X Kupa*

Tato funkce přebírá souřadnice *x,y* z generátoru náhodných bodů. Z těchto souřadnic je spočteno těžiště, ke kterému je přidělena výška *z = 500*.

.

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

## *4.X.X XX*

Tato

# Popis Aplikace

V této kapitole je popsána funkcionalita aplikace.

## 5.1 Vstupní data

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

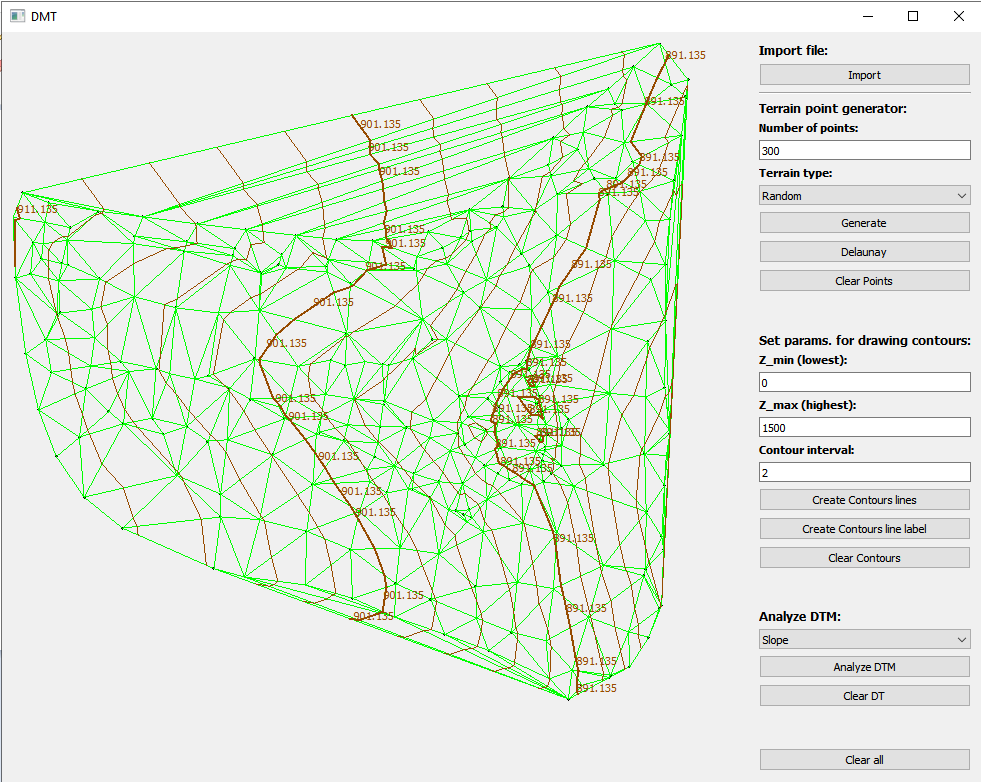
*997267.511 845904.656 883.316*

*997268.669 845883.024 881.569*

*997268.679 845883.015 881.575*

*997269.790 845867.605 881.135*

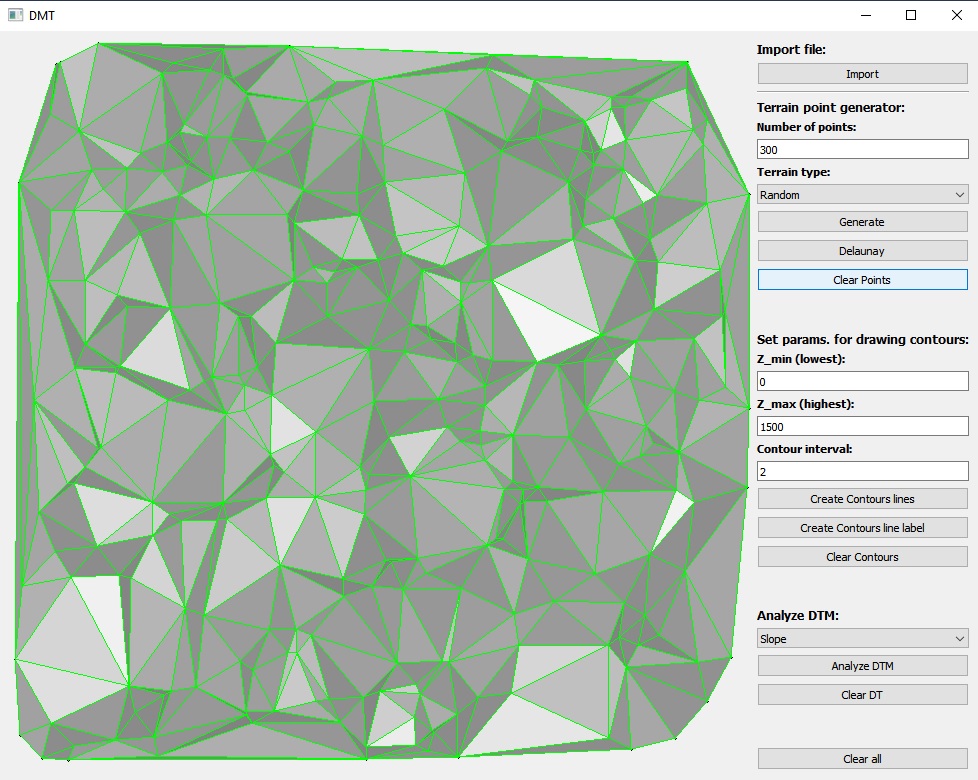
Třetí možností vstupu dat je pomocí generátoru bodů terénu, kde je nutné vložit počet bodů, který se má vygenerovat.



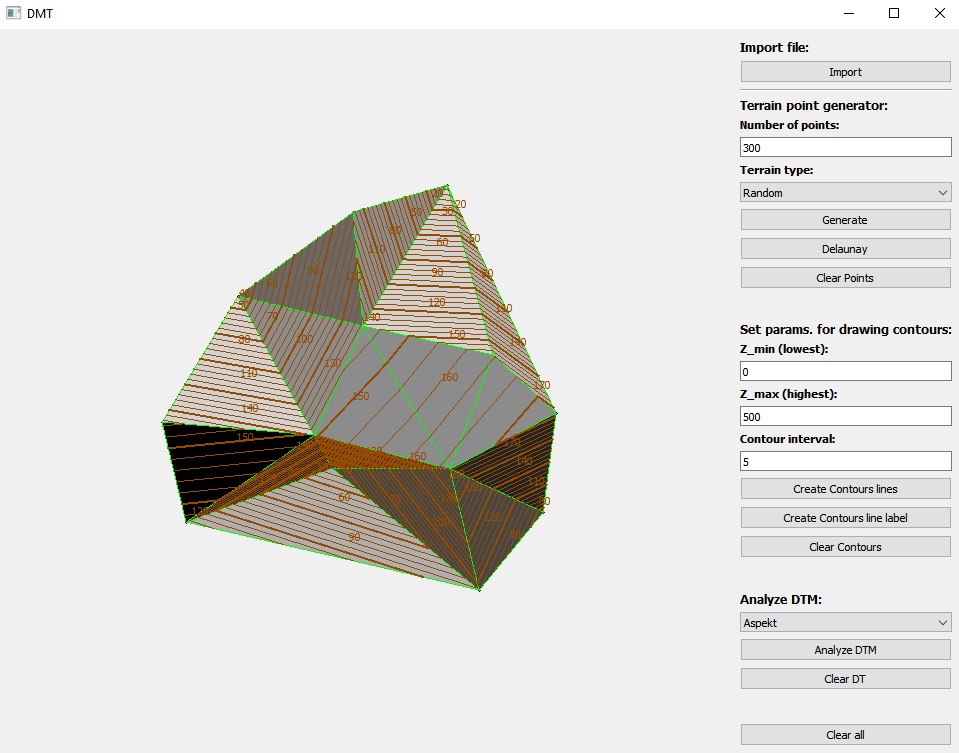
Obrázek : Ukázka aplikace – Importované body, vrstevnice + popis

## 5.2 Výstupní data

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



Obrázek : Ukázka aplikace – Slope, vygenerované Random body



Obrázek : Ukázka aplikace – Aspect, body naklikané myší

# 6 Dokumentace

## 6.1 Třída Algorithms

Třída obsahuje:

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

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

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

* Funkce na získaní středu kružnice a poloměru

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

* Funkce pro nalezení Delaunayského bodu vhodného pro Delaunay triangulaci

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

* Funkce na získání vzdálenosti mezi dvěma body

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

* Funkce na získání vzdálenosti bodu od přímky

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

* Funkce pro nalezení nejbližšího bodu

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

* Funkce generující Delaunay triangulaci

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

* Funkce pro

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

* Funkce vrací pro určení bodu na vrstevnici

std::vector<Edge> **contourLines**(std::vector<Edge> &dt,double z\_min, double z\_max, double dz);

* Funkce generující vrstevnice

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

* Funkce pro výpočet sklonu terénu

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

* Funkce pro výpočet orientace terénu

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

* Funkce analyzující terén podle sklonu a orientace

## 6.2 Třída Draw

Třída obsahuje:

private:

std::vector<QPoint3D> points;

* Deklarace proměnné pro množinu bodů

std::vector<Edge> dt;

* Deklarace proměnné pro množinu hran

std::vector<Edge> contours;

* Deklarace proměnné pro množinu hran, které určují vrstevnice

std::vector<Triangle> dtm;

* Deklarace proměnné pro množinu trojúhelníků

bool slope, aspect;

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

public:

void ***paintEvent***(QPaintEvent \*event);

* Funkce pro vykreslení

void ***mousePressEvent***(QMouseEvent \*event);

* Funkce pro snímání souřadnic bodů z kreslícího plátna po kliknutí myši

static void **importPoints**(std::string &path, std::vector<QPoint3D> &points, QSizeF &canvas\_size, double &min\_z, double &max\_z);

* Funkce pro načtení bodu ze souboru

void **setPoints**(std::vector<QPoint3D> &points\_){points = points\_;}

* Funkce pro nastavení bodů

std::vector<QPoint3D> & **getPoints**(){return points;}

* Funkce pro získání bodů

void **setDT**(std::vector<Edge> &dt\_){dt = dt\_;}

* Funkce pro nastavení Delaunay triangulace

std::vector<Edge> & **getDT**(){return dt;}

* Funkce pro získání Delaunay triangulace

void **setContours**(std::vector<Edge> &contours\_){contours = contours\_;}

* Funkce pro nastavení vrstevnic

std::vector<Edge> & **getContours**(){return contours;}

* Funkce pro získaní vrstevnic

void **setDMT**(std::vector<Triangle> &dtm\_){dtm = dtm\_;}

* Funkce pro nastavení DMT

std::vector<Triangle> & **getDMT**(){return dtm;}

* Funkce pro získání DMT

void **setSlope**(bool slope\_){slope = slope\_;}

* Funkce pro nastavení sklonu terénu

void **setAspect**(bool aspect\_){aspect = aspect\_;}

* Funkce pro nastavení orientace terénu

## 6.4 Třída Generatorterrain

## 6.4 Pomocné třídy

*sortByX*

- Třída setřídí body podle souřadnice x

*Edge*

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

*qpoint3d*

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

*Triangle*

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

# 7 Zhodnocení funkčnosti aplikace

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |

# 8 Závěr

## Seznam obrázku

[Obrázek 1: Jarvis Scan, zdroj: https://web.natur.cuni.cz/~bayertom/images/courses/Adk/adk4.pdf 4](#_Toc56161707)

[Obrázek 2: Quick Hull, zdroj: https://web.natur.cuni.cz/~bayertom/images/courses/Adk/adk4.pdf 5](#_Toc56161708)

[Obrázek 3: Quick Hull, zdroj: https://web.natur.cuni.cz/~bayertom/images/courses/Adk/adk4.pdf 5](#_Toc56161709)

[Obrázek 4: Sweep Line, zdroj: https://web.natur.cuni.cz/~bayertom/images/courses/Adk/adk4.pdf 6](#_Toc56161710)

[Obrázek 5: Graham Scan, zdroj: https://web.natur.cuni.cz/~bayertom/images/courses/Adk/adk4.pdf 7](#_Toc56161711)

[Obrázek 6: Ukázka aplikace – vložení bodů myší a vykreslení konvexní obálky 9](#_Toc56161712)

[Obrázek 7: Ukázka aplikace – generované body (grid) 10](#_Toc56161713)

[Obrázek 8: Ukázka aplikace – generované body (náhodné) 10](#_Toc56161714)

[Obrázek 9: Ukázka aplikace – generované body (kružnice) 11](#_Toc56161715)

[Obrázek 10:: Ukázka aplikace – generované body (elipsa) 11](#_Toc56161716)

[Obrázek 11: Ukázka aplikace – generované body (čtverec) 12](#_Toc56161717)

[Obrázek 12: Graham Scan – špatné vykreslení konvexní obálky u gridu 23](#_Toc56161718)