

# ALGORITMY POČÍTAČOVÉ KARTOGRAFIE

## Úloha č. 3: Digitální model terénu

### Zadání

*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, 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 slabiny 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.

### Hodnocení:

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

Čas zpracování: 3 týdny

### Popis a rozbor problému + vzorce

Je dána množina bodů  $P = \{p_1, p_2, \dots, p_n\}$  v  $R^2$ . Úkolem je najít triangulaci  $T$  nad množinou  $P$ .

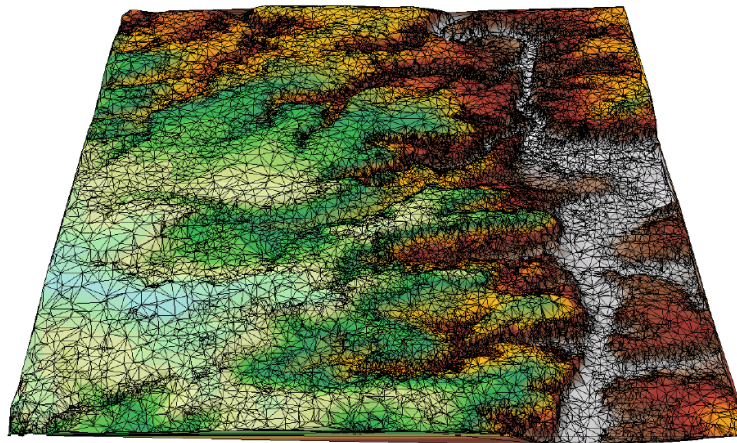
Triangulace  $T$  nad množinou bodů  $P$ , představuje takové planární rozdělení, které tvoří soubor  $m$  trojúhelníků  $t = \{t_1, t_2, \dots, t_m\}$  a hran, aby platilo (Bayer 2018):

- Libovolné dva trojúhelníky  $t_i, t_j \in T$ , ( $i \neq j$ ), mají společnou nejvýše hranu.
- Sjednocení všech trojúhelníků  $t \in T$  tvoří  $H(P)$ .
- Uvnitř žádného trojúhelníku neleží žádný další bod z  $P$ .

### Použití triangulace

Triangulace se nejvíce využívá v oblasti GIS a kartografie, kde se s jejím využitím tvoří digitální modely terénu (DMT). Tyto prostorové modely jsou tvořeny nad množinou bodů o známých souřadnicích  $x$ ,  $y$ ,  $z$ . Pro tvorbu podrobných modelů jsou často využívána data laserového skenování. DMT se v GIS používá na nejrůznější analýzy, například analýza sklonu

reliéfu, orientace reliéfu, viditelnosti apod.. Dále se DMT používá pro vizualizaci. Používá se jako podklad v kartografických mapách nebo třeba ve 3D prostředí. Jedním z příkladů je vizualizace map ve 3D nebo využití DMT ve virtuálních světech.



*Obr. 1 digitální model terénu, zdroj: (Bayer 2018)*

Ve virtuálních světech se používá triangulace také k vizualizaci jiných prostorových dat, jelikož síť trojúhelníků vystihuje velmi dobře strukturu daného objektu. Pro „hladkou“ vizualizaci se množina bodů musí zahustit. Triangulace je po zahuštění detailnější, trojúhelníky jsou mnohem menší a nejsou pouhým okem viditelné, což vytváří efekt „hladkosti“ objektu.

V interpolacích se používá při převodu bodových jevů na plošné. Dále se triangulace používají v biometrii při detekci otisků prstů. Při skenování otisku prstů nad vytvořenými body vznikne triangulace a na základě této triangulace se rozhoduje, zda je otisk validní. V neposlední řadě se využívá Delaunay triangulace k vytváření Voronoi diagramů. Pro tvorbu Voronoi diagramů se používají středy kružnic opsaných jednotlivých trojúhelníků, které reprezentují jednotlivé Voronoi vrcholy. Spojením těchto vrcholů vznikne Voronoi diagram.

Triangulace se dají dělit podle geometrické konstrukce a podle použitých kritérií. Podle geometrické konstrukce se dělí na:

- Greedy triangulaci
- **Delaunay triangulaci**
- Minimum weight triangulace
- Triangulace s povinnými hranami

- Datově závislé triangulace

A podle použitých kritérií:

- Lokálně optimální triangulace
- Globálně optimální triangulace
- Multikriteriálně optimalizované triangulace

## Algoritmus

**Delaunay triangulace** se dnes běžně používá a dala by se považovat za průmyslový standart. Delaunay triangulace maximalizuje minimální úhel (max-min kritérium) v trojúhelníku a je lokálně i globálně optimální vůči kritériu minimálního úhlu (Bayer 2018). Výsledné trojúhelníky delaunay triangulace se nejvíce ze všech triangulací blíží k rovnostranným trojúhelníkům, což je jedna z výhod delaunay triangulace, jelikož minimalizuje tvorbu velmi ostrých trojúhelníků. Její další výhodou je jednoduchá implementace, která se dá napsat de-facto během „chvilky“. Delaunay triangulace využívá kružnice opsané, na jejímž obvodu leží dva body, které tvoří hranu Delaunay triangulace k nalezení třetího vrcholu trojúhelníku. Uvnitř libovolné kružnice opsané Delaunay triangulace neleží žádný další bod vstupní množiny.

Nad všemi konvexními čtyřúhelníky Delaunay triangulace je prováděna legalizace, což je operace prohození diagonály ve čtyřúhelníku. Je spočítáno kritérium pro obě varianty, následně je vybrána vhodnější varianta a v důsledku je, či není, diagonála prohozena. Trojúhelníky jsou po legalizaci lokálně optimální k max-min kritériu. Pokud leží čtyři body na kružnici, tak v takovém případě nezáleží na prohození diagonály, jelikož výsledná kritéria budou stejná. Taková triangulace se nazývá nejednoznačná.

Pro konstrukci Delaunay triangulace existuje několik přímých metod:

- Lokální prohazování
- **Inkrementální konstrukce**
- Inkrementální vkládání
- Rozděl a panuj
- Sweep Line

**Vrstevnice** je možné vytvořit dvěma hlavními způsoby, lineární interpolací a nelineární interpolací. Nelineární interpolace bere v úvahu skutečný tvar terénu, jelikož počítá i se sklonem okolních plošek apod. Tyto interpolace jsou velmi obtížné a těžko se algoritmizují. Lineární interpolace jsou na druhou stranu jednodušší, ale nevystihují přesně realitu. Spád a rozestup vrstevnic mezi dvěma body, mezi kterými se provádí interpolace, je konstantní (Bayer 2018). Při konstrukci vrstevnic dochází k hledání průsečnice roviny tvořené trojúhelníkem z Delaunay triangulace a vodorovné roviny o určité výšce. Pokud je průsečnice úsečka, dochází k nalezení souřadnic bodů, které tvoří úsečku. Tento proces se opakuje nad všemi trojúhelníky v Delaunay triangulaci.

**Sklon a orientace** jsou dvě analýzy, které se často provádějí nad digitálním modelem terénu. Analýzy se používají v zemědělství, stavebnictví, hydrologii, geomorfologii apod. Obě analýzy jsou určovány výškovými poměry v trojúhelníku. Sklon je určený hodnotou gradientu, což je vektor maximálního spádu a orientace v bodě. Je dána jako azimut průmětu gradientu do roviny. Tyto analýzy jsou prováděny ve všech trojúhelnících v Delaunay triangulaci.

### Popis algoritmu Inkrementální konstrukce formálním jazykem

V první kroku je nutné najít náhodný bod  $p_1$  a k němu nejbližší  $p_2$ . Z těchto bodů se vytvoří hrana  $e$ . Dalším krokem (3) je nalezení bodu  $p$  z množiny  $P$ , který tvoří společně s body  $p_1$  a  $p_2$  nejmenší opsanou kružnici. V kroku 4 se zkontroluje, jestli se nehledaly kružnice opsané v polovině, kde nebyl žádný bod. Tedy jestli není výsledný bod *null*. Pokud ano, prohodí se orientace hrany  $e$  a pokračuje se krokem 3. Dalším krokem (5) je vytvoření hran  $e_2$  a  $e_3$  z nového bodu  $p$ . V kroku 6 se přidají tyto hrany do active edge listu, což je list *AEL*, ve kterém se nachází hrany triangulace, ke kterým potencionálně existují další hrany. Poslední krok 7 přípravy algoritmu je přidání vytvořených hran do listu *DT* reprezentující Delaunay triangulaci.

#### Delaunay Triangulation Incremental ( $P, AEL, DT$ )

- 1:  $p_1 = \text{rand}(P), ||p_2 - p_1||_2 = \min.$
- 2:  $e = (p_1 p_2)$
- 3:  $p = \arg \min_{p_i \in \sigma_L(e)} r(k_i), k_i = (a, b, p_i), e = (a, b)$
- 4: Pokud  $\nexists p$ , prohod' orientaci  $e \leftarrow (p_2 p_1)$ . Jdi na 3.
- 5:  $e_2 = (p_2, p), e_3 = (p, p_1)$

6:  $AEL \leftarrow e, AEL \leftarrow e_2, AEL \leftarrow e_3$

7:  $DT \leftarrow e, DT \leftarrow e_2, DT \leftarrow e_3$

Ted' přichází na řadu jádro algoritmu. Vše se děje ve while cyklu, který běží, dokud nebude active edge list prázdný. Na začátku cyklu se vyjme první hrana  $e$  z active edge listu. Dále je prohozena její orientace, aby se hledaly kružnice opsané v polorovině, ve které se ještě nehledalo. V kroku (11) je hledán bod  $p$ , který tvoří nejmenší opsanou kružnici stejně jako v kroku 3, s tím rozdílem, že se používají body, které tvoří hranu  $e$  vyjmutou z  $AEL$  na začátku cyklu. V dalším kroku se testuje, jestli v této polorovině existuje takový bod  $p$ . Pokud ne, tak tato iterace končí. Pokud ano, tak jsou v dalším kroku vytvořeny hrany  $e_2$  a  $e_3$  z nalezeného bodu  $p$ . Dále je hrana  $e$  přidána do listu  $DT$ . V posledním kroku se volá funkce *add*, která rozhodne, zda se hrany  $e_2$  a  $e_3$  přidají jen do listu  $DT$  nebo i do listu  $AEL$ .

8: while  $AEL$  not empty:

9:      $AEL \rightarrow e, e = (p_1 p_2)$

10:     $e = (p_2 p_1)$

11:     $p = \arg \min_{p_i \in \sigma_L(e)} r(k_i), k_i = (a, b, p_i), e = (a, b)$

12:    if  $\exists p$  :

13:          $e_2 = (p_2, p); e_3 = (p, p_1)$

14:          $DT \leftarrow e$

15:          $add(e_2, AEL, DT), add(e_3, AEL, DT)$

Funkce *add* vytvoří novou hranu  $e'$  z bodů hrany vstupní. V druhém kroku kontroluje, zda již taková hrana existuje uvnitř listu  $AEL$ . Pokud ano, tak je hrana  $e'$  odstraněna z  $AEL$ . Pokud ne, tak je hrana  $e$  přidána do  $AEL$ . Na konci funkce je v obou případech hrana přidána do  $DT$ .

$Add(e = (a, b), AEL, DT)$

1:  $e' = (b, a)$

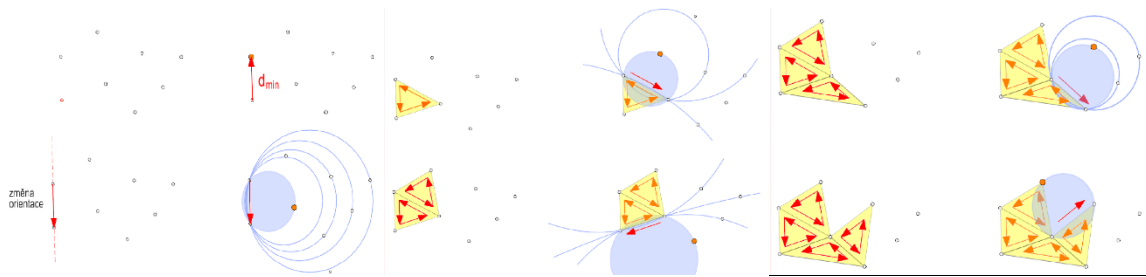
2: if  $(e' \in AEL)$

3:      $AEL \rightarrow e'$

4: else:

5:      $AEL \leftarrow e$

6:  $DT \leftarrow (a, b)$



Obr. 2, 3, 4 Ilustrace inkrementální konstrukce, zdroj: (Bayer 2018)

## Problematické situace

### Vstupní data, formát vstupních dat, popis

Aplikace nabízí dvě možnosti vstupních dat. První možnost je náhodné generování bodů, které umožňuje uživateli rychlé vyzkoušení aplikace bez použití vlastních dat. Druhá možnost je nahrání vlastních bodů v textovém souboru. Textový soubor musí mít zaspaný na každém řádku jeden bod ve formátu souřadnice „x y z“ viz. soubor test.txt. Jednotlivé souřadnice mohou být odděleny téměř „jakkoliv“, jelikož je možné nastavit oddělovač v GUI. Souřadnice x a y bodů jsou při nahrávání přetransformované na stupnici 0 -1, takže nezáleží, v jakém zobrazení se vstupní data nachází.

### Výstupní data, formát výstupních dat, popis

Veškerá výstupní data jsou zobrazena pouze v grafické podobě. Pokud uživatel využije veškeré funkce aplikace, je schopný vykreslit body, polyedrický model terénu (spočítaná triangulace), vrstevnice, sklon a expozici.

Body, ze kterých se tvoří triangulace, se vykreslují jako křížky (body se pro větší přehlednost nevykreslují po spuštění dalších funkcí). Polyedrický model terénu je vykreslován po jednotlivých trojúhelnících. Vrstevnice jak zdůrazněné, tak nezdůrazněné jsou vykreslovány jako liniový prvek. Sklon a expozice jsou znázorněny barevnou výplní trojúhelníku podle určité barevné stupnice.

### Printscreen vytvořené aplikace

## Dokumentace: popis tříd, datových položek a jednotlivých metod

Aplikace se skládá z osmi tříd. Dvě třídy reprezentují vlákna, tři třídy reprezentují datové struktury, jedna algoritmy, jedna grafické rozhraní a jedna vykreslování na plátno. Třída *Point3D* reprezentuje datovou strukturu, která má tři proměnné *double* reprezentující souřadnice *x*, *y*, *z*. Třída *Point3D* se skládá jen z metod, jako jsou getry a setry pro všechny proměnné. Další třídou reprezentující datovou strukturu je třída *Triangle*, reprezentující trojúhelník. Třída má tři proměnné *Point3D*, reprezentující jednotlivé vrcholy trojúhelníku a dvě proměnné *double*, které reprezentují sklon a expozici. Třída má tři metody, první metoda *getNormalVec* počítá normálový vektor trojúhelníku. Další dvě metody tento vektor využívají. Metoda *getSlope* počítá sklon trojúhelníku a metoda *getExposition* počítá expozici trojúhelníku. Poslední třídou, která reprezentuje datovou strukturu, je třída *Edge*. Třída má dvě proměnné *Point3D* reprezentující začátek a konec hrany a jednu proměnnou *boolean*, která značí, jestli má či nemá být hrana zvýrazněna. Třída má tři metody. První metoda *swap* vymění koncový a startovní bod hrany a tím změní její orientaci. Druhá metoda *swappedEdge* je podobná metodě první, ale místo aby měnila orientaci dané hrany, vytváří hranu novou s opačnou orientací. Poslední metoda této třídy *equals* porovnává hranu s objektem. Pokud je porovnáváný objekt totožný s hranou (tzn. je to stejná hrana), metoda vrací *true*, jinak vrací *false*.

Dvě třídy rozšiřující třídu *SwingWorker* reprezentující vlákna jsou *GenerateWorker* a *CalculateWorker*. Obě třídy mají metody *doInBackground* a *done*. Metoda *doInBackground* pracuje v pozadí v novém vláknu a v případě třídy *GenerateWorker* generuje buďto náhodné body nebo vytváří body z nahraného textového souboru. Na konci metody dochází k volání metody pro výpočet výškového rozdílu mezi minimem a maximem generovaných bodů. Pokud jsou body nahrané z textového souboru, tak je ještě volána metoda, která transformuje souřadnice *x* a *y* na stupnici 0 – 1. Po skončení této metody se automaticky volá metoda *done*, která vytvoří zprávu o výškovém rozdílu pro uživatele a přepíše interval vrstevnic, aby uživatel nemusel interval vymýšlet. Následně metoda *done* volá metodu pro překreslení plátna. Třída *GenerateWorker* má ještě metodu *parse*, která rozdělí řádek na jednotlivé elementy a převede je na *double*. Tato metoda se volá při načítání dat z textového souboru.



Ve třídě *CalculateWorker* počítá metoda *doInBackground* jeden z procesů: Delaunay triangulace, vytváření vrstevnic, sklon nebo expozici. V případě expozice dochází k převádění „z“ souřadnic na stupnici 0 - 1 a následně k výpočtu expozice. Převod souřadnic probíhá také při počítání sklonu, pokud již převod neproběhl. Po výpočtu sklonu dochází ještě k nalezení minimálního a maximálního sklonu. Pokud jsou počítány vrstevnice a již proběhl převod „z“ souřadnic, tak jsou „z“ souřadnice převedeny zpátky na původní hodnoty a dochází k vytvoření vrstevnic. Metoda *done* volá metodu pro překreslení plátna. Ve třídě *CalculateWorker* se nachází ještě dvě další metody. Metoda *scalingZ*, která převede „z“ souřadnice na stupnici 0 – 1 a metoda *scalingZBack*, která převede „z“ souřadnice zpátky na původní hodnoty.

Nejobsáhlejší třída *Algorithms* obsahuje téměř veškeré metody, ve kterých probíhají výpočty. Třída má dvě proměnné, *double* EPSILON a *enum* OrientationEnum. Proměnná EPSILON je velmi malá hodnota, která se používá při porovnání místo nuly. Proměnná OrientationEnum určuje orientaci bodu vůči úsečce. Jedna z hlavních metod je metoda *delaunay*, která vypočítává Delaunay triangulaci. Tato metoda volá několik dalších metod, které provádějí dílčí výpočty. Metoda *minimalBoundingCircle* je jednou z těchto metod a volá další dvě metody. Metoda hledá body v jedné polovině od hrany z množiny bodů pomocí metody *getOrientation*, která vypočítá determinant a na základě hodnoty determinantu vrátí orientaci bodu vůči hraně. Jakmile metoda *minimalBoundingCircle* má bod se správnou orientací, volá metodu *circleRadius*, která spočítá poloměr opsané kružnice za pomoci metody *dist*, jenž počítá vzdálenost bodů v rovině. Následně metoda *minimalBoundingCircle* vrátí bod, který tvoří nejmenší opsanou kružnici se vstupní hranou. Poslední metodu, kterou metoda *delaunay* volá je metoda *addToAel*. Tato metoda testuje, zdali vstupní hrana již existuje ve vstupním listu či nikoliv. Pokud ano, tak ji smaže.

Samostatná metoda *scaling* ve třídě *Algorithms* transformuje souřadnice x a y bodu na stupnici 0 - 1. Další samostatnou metodou je metoda *heightDifference*, která spočítá výškový rozdíl minima a maxima vytvořených bodů. Třída *Algorithms* obsahuje dalších sedm metod. Čtyři z těchto metod spolu souvisí, jelikož jsou volány při počítání sklonu a expozice. Metoda *det* počítá determinant 3x3, metoda *dotProd* počítá skalární součin, metoda *len* počítá velikost vektoru a metoda *angle* počítá úhel mezi dvěma vektory. Poslední tři metody vytvářejí vrstevnice. Metoda *calcContourPoints* počítá průsečíky vrstevnic se stranou

trojúhelníku. Metoda *calcContours(Triangle t, double interval)* vytváří list vrstevnic v daném trojúhelníku a poslední metoda *calcContours(List <Triangle> t, double interval)* vytváří list všech linií reprezentující vrstevnice.

*DrawPanel* je třída, která zajišťuje veškeré vykreslování na plátno. Tato třída má několik proměnných, jako jsou *listy*, *pole* a *Path2D*. Tyto datové struktury umožňují jednoduché vykreslování. Kromě konstruktoru a předdefinované metody *paintComponent* má třída dvě metody. Metodu *affineTransform*, která transformuje a škáluje datovou strukturu *Path2D*, aby byla správně vykreslena a metodu *colorScale*, která převádí hodnotu na stupnici 0 – 255. Metoda *paintComponent* se stará o veškeré vykreslování. Vykresluje body, linie a plochy. Veškeré vykreslované elementy jsou škálované podle velikosti plátna ze stupnice 0 - 1.

Poslední třídou je třída *GUI*, která reprezentuje grafické rozhraní. V této třídě se nachází obě již zmíněné třídy reprezentující vlákna. Třída má několik proměnných s různými datovými strukturami. Obsahuje dvě proměnné reprezentující vlákna, dále pak dvě proměnné reprezentující *Int*, *boolean*, tři reprezentující *double* a jednu reprezentující *JFileChooser*. Dále také obsahuje veškeré proměnné reprezentující tlačítka apod. v GUI. Nejdůležitější metoda třídy *GUI* je metoda *main*, která tvoří celé grafické rozhraní a nastavuje jméno aplikace. Dále pak třída obsahuje metody, které se volají po stisknutí některého z tlačítek grafického rozhraní. Tyto metody jsou vesměs podobné. Metoda nastaví jeden z parametrů a spustí jedno z vláken. Podle nastaveného parametru v daném vlákně proběhne proces, který dané tlačítko grafického rozhraní reprezentuje.

## **Závěr, možné či neřešené problémy, náměty na vylepšení**

Vylepšení, které mě napadlo, bylo již implementováno. Vylepšení spočívá v počítání výškového rozdílu mezi minimem a maximem vytvořených bodů. Následně tento rozdíl zobrazím v grafickém rozhraní a přepíši nabízený interval vrstevnic. Tím dám uživateli informaci, jaký základní interval vrstevnic by měl zvolit, aby vypadal výsledek esteticky příjemně. Dále jsem přidal do GUI možnost nastavit oddělovač při načítání dat ze souboru.

## **Seznam literatury**

BAYER, T. 2018. *Rovinné triangulace a jejich využití: Greedy Triangulation. Delaunay Triangulation. Constrained Delaunay Triangulation. Data Dependent Triangulation. DMT.*

[elektronický zdroj]. Praha. Dostupné také z:

<https://web.natur.cuni.cz/~bayertom/index.php/teaching/algoritmy-pro-tvorbu-digit-map>