ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

ALGORITMY DIGITÁLNÍ KARTOGRAFIE A GIS

KATEDRA GEOMATIKY

Úloha č. 3: Digitální model terénu

Monika KŘÍŽOVÁ
Marek HOFFMANN

4.12.2021

Obsah

1 Zadání **2**

2 Údaje o bonusových úlohách **2**

3 Popis problému **2**
3.1 Formulace problému 2

4 Popisy algoritmů **3**
4.1 Delaunayova trinagulace 3
4.2 Konstrukce vrstevnic lineární interpolací 5
4.2.1 Lineární interpolace 5
4.2.2 Algoritmus 5
4.3 Algoritmus výpočtu sklonu trojúhelníku 6
4.4 Algoritmus výpočtu expozice trojúhelníku 6

5 Problematické situace **7**
5.1 Winding Number algoritmus 7
5.2 Ray Crossing algoritmus 7

6 Vstupní data **9**

7 Výstupní data **10**

8 Printscreen vytvořené aplikace **12**

9 Dokumentace **12**
9.1 Třída Algorithms 12
9.2 Třída Draw 14
9.3 Třída Widget 15

10 Závěr **16**

1 Zadání

Navrhněte aplikaci s grafickým rozhraním, která nad množinou bodů vytvoří polyedrický digitální model terénu.

Jako vstupní data využijte existující data o alespoň 300 bodech. Nad nimi vytvořte Delaunayovu triangulaci metodou inkrementální konstrukce. Triangulaci využijte pro výpočet a vizualizace vrstevnic. Výslednou triangulaci doplňte o vhodnou vizualizaci sklonu jednotlivých trojúhelníků a jejich expozici.

2 Údaje o bonusových úlohách

Zpracovány byly celkem 3 bonusové úlohy ze 7 zadaných.

- Triangulace nekonvexní oblasti zadané polygonem +5b
- Výběr barevných stupnic při vizualizaci sklonu a expozice. +3b
- Automatický popis vrstevnic respektující kartografické zásady +10b - z části

3 Popis problému

Triangulace je jednou ze základních výpočetních úloh nejen v oblasti digitální kartografie. Kromě tvorby DMT své uplatnění najde i při tvorbě jakékoliv vizualizaci prostorových dat ve formě 3D modelů. Dále lze triangulaci využít při detekci otisku prstů či např. modelování přírodních jevů.

3.1 Formulace problému

Mějme množinu bodů P. Triangulací se rozumí převod množiny bodů na trojúhelnikovou síť m trojúhelníků t, tak, aby platilo:

- Libovolné 2 trojúhelníky mají společnou nejvýše jednu hranu.
- sjednovení trojúhelníků je souvislá množina
- unitř žádného trojúhelníku neleží žádný další bod z P

Triangulace by měla tvořit pravidelné trojúhelníky blíží se trojúhelníkům rovnostranným a všechy trojúhelníky by se v každém bodě měly co nejlépe přimykat terénu.

Existuje několik druhů triangulace. Nejpoužívanější je však Delaunayova triangulace, která svými kritérii zajišťuje jednoznačnost, spolehlivé výslekdy a má optimální časovou složitost.

4 Popisy algoritmů

4.1 Delaunayova trinagulace

Vlastnosti

- Uvnitř kružnice k opsanému libovolnému trojúhelníku t_j neleží žádný jiný bod množiny P.
- DT maximalizuje minimální úhel v každém t_j, ale neminimalizuje maximální úhel v t_j.
- DT je lokálně i globální optimální vůči kritériu minimálního úhlu.
- DT je jednoznačná, pokud žádné 4 body neleží na kružnici.

Popis algoritmu Existuje několik metod kontruskce Delaunayho triangulace. V této úloze však byla implementována metoda inkrementální konstrukce, která je založena na postupném přidávání bodů do již vytvořené triangulace.

Pro konstrukci je používána datová struktura AEL (Active Edge List), která obsahuje hrany e, ke kterým hledáme Delaunayovský bod minimalizující poloměr kružnice mezi Delaunayovskou hranou a hledaným bodem. Delaunayovská hrana je orientovaná a bod hledáme pouze vlevo od ní.

Popis Algoritmu 2

1. Nalezení bodu q s nejmenší xovou souřadnicí

$$q = min(x_i)$$

2. Nalezení nejbližšího bodu k počátečnímu bodu

$$||p_1 - q||_2 = min$$

3. Vytvoření první hrany
$$e = (q, p_1)$$
4. Hledání Delaunayho bodu
$$\underline{p} = argmin_{\forall p_i \in \sigma_L(e)} r'(k_i); k_i = (a, b, p_i); e = (a, b)$$
5. Při neexistenci Delaunayho bodu změna orientace hrany a opakování předchozího kroku
$$\nexists \underline{p} : e \leftarrow (p_1, q);$$
6. Vytvoření zbývajících hran trojúhelníka
$$e_2 = (p_1, \underline{p}); e_3 = (\underline{p}, q)$$
7. Přidání hran do AEL
$$AEL \leftarrow e; AEL \leftarrow e_2; AEL \leftarrow e_3$$
8. Přidání hran do DT
$$DT \leftarrow e; DT \leftarrow e_2; DT \leftarrow e_3$$
9. Dokud není vektor AEL prázdný:
   Změna orientace první hrany z AEL
$$e = (p_1, q)$$
   Hledání Delaunayho bodu k e
$$\underline{p} = argmin_{\forall p_i \in \sigma_L(e)} r'(k_i); k_i = (a, b, p_i); e = (a, b)$$
   Pokud bod existuje
$$if \exists \underline{p}$$
   Tvorba zbývajících hran trojúhelníku
$$e_2 = (p_2, \underline{p}); e_3 = (\underline{p}, q)$$
   Přidání hran do DT
$$DT \leftarrow e; DT \leftarrow e_2; DT \leftarrow e_3$$
   Přidání hrany do AEL, pokud již neexistuje s opačnou orientací.
$$AELif \ \nexists \leftarrow e_2; AEL \leftarrow e_3$$

4.2 Konstrukce vrstevnic lineární interpolací

Vrstevnice je křivka, která spojuje body se stejnou nadmořskou výškou. Zpravidle bývají vrstevnice vykresllovány s pravidelným intervalem.

Vrstevnice dělíme na základní a hlavní. Základní vrstevnice tvořené s pravidelným základním intervalem bývají vyznačeny tenkou, nepřerušovanou linií a jsou uzavřené. Hlavní vrstevnice je každá pátá základní vrstevnice. V mapách je značena tlustou, nepřerušovanou, uzavřenou linií a bývají doplněny popisem.

Dále existují vrstevnice doplňkové, které doplňují základní vrstevnice v těch místech, kde dostatečně nevystihují zejména sklonitý terén, a vrstevnice pomocné, které se používají pro místa s nestálým reliéfem jako např. v oblastech povrchové těžby. Doplňkové ani pomocné vrstevnice nebyly v této aplikaci implementovány.

4.2.1 Lineární interpolace

Lineární interpolace předpokládá:

1. spád terénu mezi dvěma body, mezi kterými provádíme interpolaci, je konstantní
2. Rozestup vrstevnic mezi dvěma body je konstatní

Na vstupu máme trojúhelník t a vodorovnou rovinu ρ o výšce h. Lineární interpolace je založena na analytické geometrii, kdy hledáme průsečnici roviny T určené trojúhelníkem t a vodorovnou rovinou. Tento výpočet provedeme nad každým trojúhelníkem t

4.2.2 Algoritmus

1. Nalezení průsečíků všech horizontálních rovin s jednotlivými hranami.

   Varianty vzájemné polohy ρ a T:

   (a) Nemamají žádný společný bod - neřešíme

   (b) Průsečnice tvoří jeden bod - neřešíme

   (c) Průsečnice je úsečka:

   - Hrana leží v rovině ρ - vrstevnice je přímo v hraně

- Průsečnice roviny s trojúhelníkem vytvářející nové průsečíky mezi vrcholy trojúhelníka - vrstevnice vykreslena mezi body vypočtenými následujícími rovnicemi:

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

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

(d) Všechny hrany trojúhelníka leží v rovině ρ - neřešíme

4.3 Algoritmus výpočtu sklonu trojúhelníku

Sklon představuje jednu z analytických úloh realizovaných nad DMT. Určuje, zdali terén stoupá, klesá nebo je rovinný.

Sklon každého trojúhelníku se vypočte následujícím způsobem.

$$u = (u_x, u_y, u_z)$$

$$v = (v_x, v_y, v_z)$$

$$n = \vec{u} \times \vec{v} = (n_x, n_y, n_z)$$

$$\varphi = arccos\frac{n_z}{|n|}$$

4.4 Algoritmus výpočtu expozice trojúhelníku

Expozice představuje orientaci terénu vzhledem ke Slunci.

Orientace v bodě je definován jako azimut průmětu normálového vektoru roviny trojúhelníku do roviny x,y.

$$u = (u_x, u_y, u_z)$$

$$v = (v_x, v_y, v_z)$$

$$n_t = \vec{u} \times \vec{v} = (n_x, n_y, n_z)$$

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

5 Problematické situace

5.1 idk

5.2 idk

6 Vstupní data

Polygonová mapa se do projektu nahrává z textového souboru, který musí splňovat specifický formát, bez něhož nebude zajištěno správné načítání dat.

Vrcholy polygonů jsou načítány postupně řádek po řádku, přičemž musí v nahrávaném souboru platit následující pravidla:

- každý vrchol polygonu je definován na jednotlivém řádku,
- na řádku je pořadí proměnných id » x » y , hodnoty jsou od sebe odděleny jednou mezerou,
- id je identifikátor jednotlivých vrcholů polygonu, x a y jsou souřadnice vrcholů polygonu,
- nový polygon má vždy hodnotu id prvního vrcholu rovnu 1, od této hodnoty se postupně načítají body, a to až do okamžiku, kdy program dojde k dalšímu identifikátoru s označením 1.

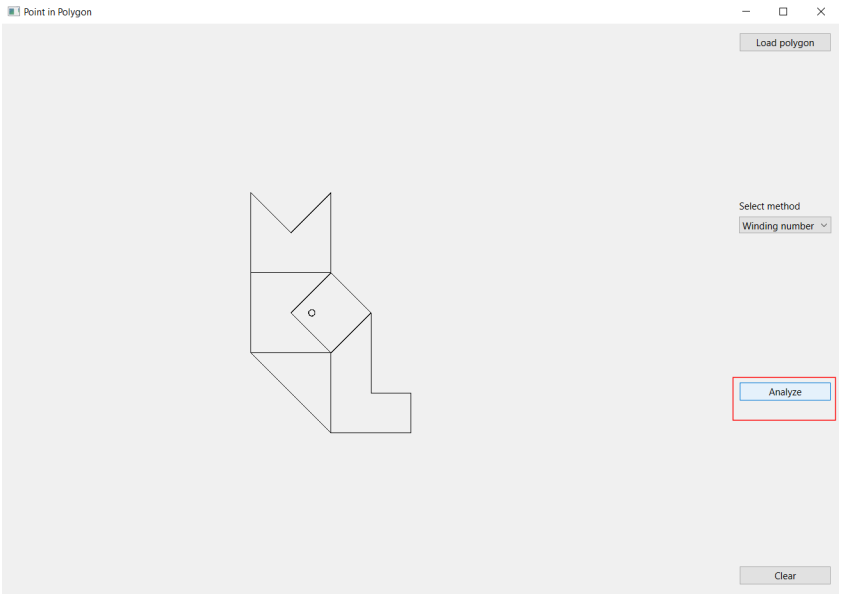
```
1 300 300
2 300 400
3 400 400
4 400 300
1 400 400
2 300 400
3 400 500
```

Obrázek 1: Špagetový model vstupního souboru

Souřadnice bodu q jsou získávány interaktivně odečtením souřadnic kurzoru myši.

7 Výstupní data

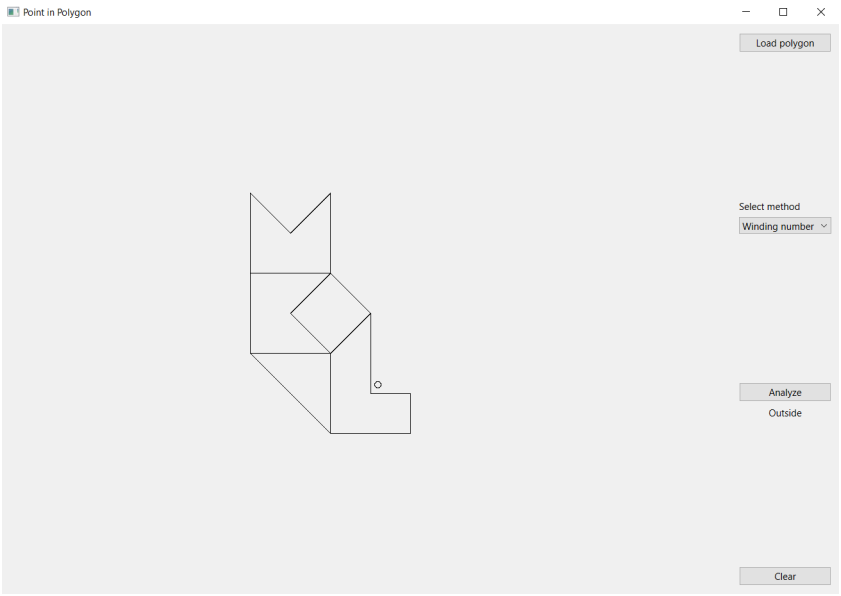
Po načtení souboru s polygonovou mapou, přidání bodu q a stisknutí tlačítka Analyze se v místě vyznačeném na obrázku 2 vytiskne výsledek analýzy polohy bodu.



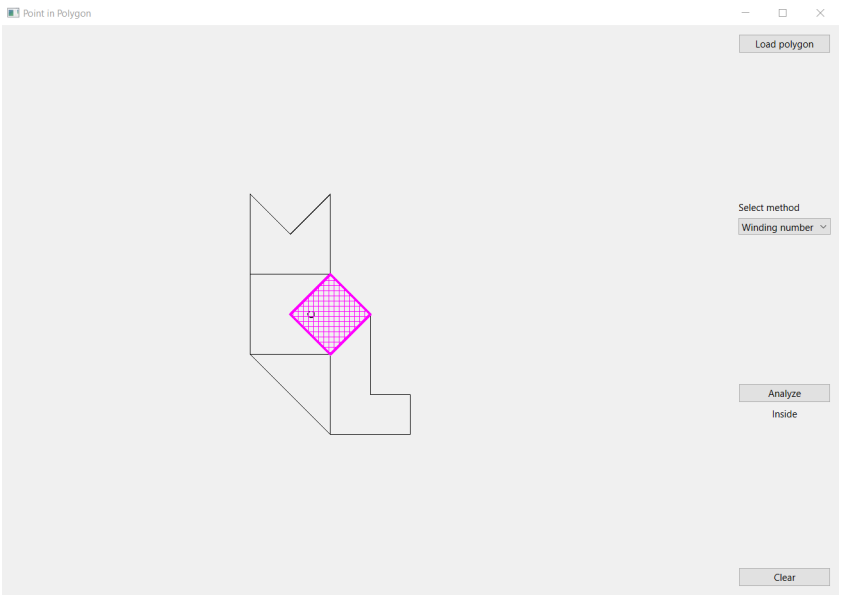
Obrázek 2: Aplikace s načteným polygonem

Výsledek dotazu může mít 3 různé hodnoty:

- je-li bod mimo polygonovou mapu, vypíše se zpráva „Outside“,
- je-li bod uvnitř polygonu, vypíše se zpráva „Inside“ a polygon obsahující zadaný bod se graficky zvýrazní,
- je-li bod na hranici polygonu, vypíše se „Point is on the line“ a graficky se zvýrazní polygon, na jehož linii bod leží.

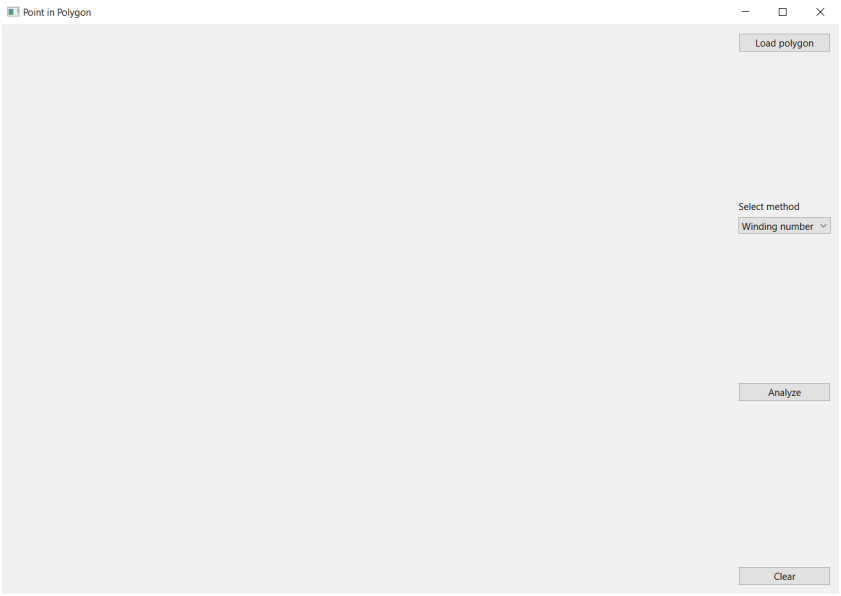


Obrázek 3: Bod vně polygonu



Obrázek 4: Bod uvnitř polygonu

8 Printscreen vytvořené aplikace



Obrázek 5: Úvodní okno aplikace

9 Dokumentace

Kód zahrnuje 3 třídy – Draw, Algorithms a Widget, které budou následně detailněji popsány.

9.1 Třída Algorithms

Třída Algorithms obsahuje 4 funkce :

- int getPointLinePosition(QPoint &a, QPoint &p1, QPoint &p2);
- double get2LinesAngle(QPoint &p1, QPoint &p2, QPoint &p3, QPoint &p4);
- int getPositionWinding(QPoint &q, std::vector<QPoint> &pol);
- int getPositionRayCrossing(QPoint &q, std::vector<QPoint> &pol);

int getPointLinePosition(QPoint &a, QPoint &p1, QPoint &p2);
Analyzuje vzájemnou polohu mezi bodem a linií polygonu, resp. v jaké polorovině vůči linii se bod nachází. Vstupními argumenty funkce jsou souřadnice určovaného bodu (jako QPoint) a souřadnice 2 bodů určujících polohu linie (vrcholy polygonu taktéž jako QPoint). Funkce vrací vždy hodnotu 1, 0 nebo -1 dle následujících pravidel:

- 0 v případě, že bod leží v pravé polorovině,
- 1 v případě, že bod leží v levé polorovině,
- -1 v případě, že bod leží na linii.

double get2LinesAngle(QPoint &p1, QPoint &p2, QPoint &p3, QPoint &p4);
Počítá úhel mezi dvěma liniemi pomocí vztahu **??**. Vstupními argumenty jsou body určující linie, tzn. vrcholy polygonu. Návratovou hodnotou funkce je double – desetinné číslo s velikostí úhlu mezi těmito přímkami.

int getPositionWinding(QPoint &q, std::vector<QPoint> &pol);
Funkce analyzuje polohu bodu metodou Winding Number, jež byla podrobně vysvětlena v kapitole 3. Vstupními argumenty jsou souřadnice bodu q (jako QPoint) a vektor vrcholů polygonu (vektor naplněný prvky QPoint). Funkce vrací integer, jež může nabývat 3 hodnot:

- 1, leží-li analyzovaný bod uvnitř polygonu
- -1 leží-li na linii polygonu
- a 0, leží-li mimo kontrolovaný polygon.

int getPositionRayCrossing(QPoint &q, std::vector<QPoint> &pol);
Funkce analyzuje polohu bodu metodou Ray Crossing, jež byla podrobně vysvětlena v kapitole 3. vstupními argumenty jsou souřadnice bodu q (jako QPoint) a vektor vrcholů polygonu (vektor naplněný prvky QPoint). Funkce vrací, stejně jako funkce předchozí, celé číslo, jež může nabývat následujících hodnot:

- 1, pokud leží analyzovaný bod uvnitř polygonu
- a 0, leží-li mimo kontrolovaný polygon.

Ošetření případu, kdy bod leží na linii nebylo do kódu implementováno, funkce tedy nevrací hodnotu -1, leží-li bod na linii.

9.2 Třída Draw

Třída Draw obsahuje následující funkce, které budou dále podrobně popsány:

- void paintEvent(QPaintEvent *event);
- void mousePressEvent(QMouseEvent *event);
- void clear();
- QPoint getPoint()return q;
- std::vector<QPolygon> getPolygon(){return polygons;}
- void fillPolygon(int result);
- void loadData(QString &file_name);

a následující proměnné:

- std::vector<QPolygon> polygons;
- QPoint q; - analyzovaný bod
- int highlighted_polygon=-99; - id polygonu, který obsahuje analyzovaný bod

• void paintEvent(QPaintEvent *event);
Funkce nastavuje grafické atributy vykreslovaných objektů a kreslí na plátno vyhodnocovaný bod a zadané polygony a znázorňuje polygon incidující s bodem.

void mousePressEvent(QMouseEvent *event);
Metoda získává souřadnice myši a ukládá je do proměnné QPoint q, tedy jako souřadnice analyzovaného bodu q.

void clear();
Funkce smaže veškeré objekty, jež jsou vykresleny na canvasu, funkce nemá vstupní argumenty.

QPoint getPoint(){return q;}
Funkce vrací souřadnice bodu q, návratový typ funkce je QPoint, funkce nemá vstupní argumenty.

std::vector<QPolygon> getPolygon(){return polygons;}
Funkce vrací vektor polygonů načtených z textového souboru – návratový typ je std::vector<QPolygon>, funkce nemá vstupní argumenty.

void fillPolygon(int result);
Funkce ukládá identifikátor polygonu obsahujícího analyzovaný bod, pomocí identifikátoru se tento polygon následně ve funkci paintEvent() zvýrazní. Vstupním argumentem je int result, jež stanovuje polohu bodu vůči linii (má hodnoty 0,1,-1,dle definice výše).

void loadData(QString &file_name);
Funkce načítá data z textového souboru a ukládá je do vektoru QPolygon. Vstupní argumentem je cesta k souboru, jejž chceme načíst do aplikace.

9.3 Třída Widget

Třída Widget obsahuje 3 metody:

- void on_pushButtonClear_clicked();
- void on_pushButtonAnalyze_clicked();
- void on_pushButtonLoad_clicked();

void on_pushButtonClear_clicked();
Funkce se volá při stisknutí tlačítka Clear, volá funkci clear(), jež je definovaná v třídě Draw.

void on_pushButtonAnalyze_clicked();
Funkce se volá při stisknutí tlačítka Analyze. Ve funkci nejprve dojde k uložení jednotlivých polygonů do vektoru vrcholů polygonu, následně je zavolána funkce getPositionRayCrossing ze třídy Algorithms nebo getPositionWinding dle výběru v comboboxu.

void on_pushButtonLoad_clicked();
Funkce slouží k inicializaci načítání souboru, spustí se po stisknutí tlačítka Load, čímž zobrazí dialog pro výběr souboru obsahujícího data, jež mají být načtena do aplikace. Po výběru souboru se zavolá funkce loadData ze třídy Draw, která přečte data ze souboru a uloží je do proměnné.

10 Závěr

Vytvořená aplikace vizualizuje polygonovou mapu uloženou ve výše popsaném formátu, po přidání bodu taktéž analyzuje polohu tohoto bodu vůči jednotlivým polygonům mapy. Po analýze bodu se vypíše výsledek – tedy zda bod leží uvnitř nebo vně polygonové mapy, v případě umístění bodu uvnitř polygonu se polygon obsahující tento bod graficky zvýrazní.

V aplikaci jsou pro vyhodnocení polohy bodu implementovány 2 algoritmy – Winding number a Ray Crossing, mezi nimiž je možné přepínat pomocí comboboxu umístěném v pravé části okna. Princip fungování obou algoritmů je popsán v kapitole 3.

Možné zlepšení kódu programu by bylo v řešení singularit polohy bodu q, tedy případu, kdy bod q leží na vrcholu jednoho či více polygonů a dále přidání analýzy případu, kdy bod q leží na linii polygonu pomocí Ray Crossing algoritmu. Dalším možným krokem pro zlepšení funkčnosti by mohlo být přidání bodu q pomocí vepsání souřadnic, nikoliv pouze odečtením polohy myši.