České vysoké učení technické v Praze

Fakulta stavební

Algoritmy v digitální kartografii

Úloha č. 1: Geometrické vyhledávání bodu

Skupina:

Sabina Kličková

Martin Vajner

Zimní semestr 2021/2022

ČVUT
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Obsah

1. Zadání

Vstup: Souvislá polygonová mapa n polygon· {P1, ..., Pn}, analyzovaný bod q.

Výstup: Pi , q ∈ Pi.

Nad polygonovou mapou implementujete Winding Number Algorithm pro geometrické vyhledání incidujícího polygonu obsahujícího zadaný bod q.

Nalezený polygon graficky zvýrazněte vhodným způsobem (např. vyplněním, šrafováním, blikáním). Grafické rozhraní vytvořte s využitím frameworku QT.

Pro generování nekonvexních polygonů můžete navrhnout vlastní algoritmus či použít existující geografická data (např. mapa evropských států).

Polygony budou načítány z textového souboru ve Vámi zvoleném formátu. Pro datovou reprezentaci jednotlivých polygonů použijte špagetový model.

Detekce polohy bodu rozlišující stavy uvnitř, vně, na hranici polygonu.	10b

2. Bonusové úlohy

V této úloze byly zpracovány následující bonusové úlohy:

Krok	Hodnocení
Analýza polohy bodu (uvnitř/vně) metodou Ray Algorithm.	+5b
Ošetření singulárního případu u Ray Algorithm: bod leží na hraně polygonu.	+5b
Ošetření singulárního případu u obou algoritmů: bod je totožný s vrcholem jednoho či více polygonů.	+2b

3. Popis a rozbor problému + vzorce.

Aplikace byla napsána v jazyce C++ pomocí editoru Qt Creator.

Na vstupu se nacházel bod „a“ a vytvořený polygon. Hlavním úkolem bylo zjistit polohu bodu vůči danému polygonu. Tj, zjistit, jestli bod leží uvnitř, vně nebo na linii polygonu. Polygon byl vytvořen interaktivně pomocí prostředí aplikace (Qt Creator), ve které byl současně i zobrazován.

Pro zjištění polohy bodu bylo užito dvou různých algoritmů: Winding number a Ray Crossing.

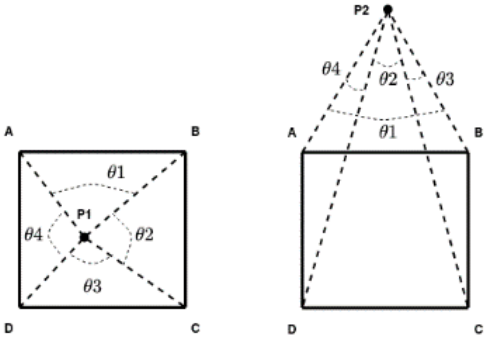
Nejprve bylo vytvořeno grafické prostředí aplikace, ve kterém probíhalo vytvoření polygonu a bodu, volba metody a samotné zhodnocení stavu bodu. Byly vytvořeny třídy Draw a Algorithms, které obsahují definice tříd a samotný kód, který určuje chování aplikace.

4. Popisy algoritmů formálním jazykem

I. Winding Number

Algoritmus Winding Number je definován jako počet rotací křivky, či v našem případě polygonu, v jednom směru okolo daného bodu q. Výpočet je založen na zjištění hodnot úhlů ω mezi daným bodem q a body polygonu a na poloze bodu q vůči segmentu polygonu. Pro všechny segmenty je třeba zjistit, jestli se bod nachází nalevo, či napravo. Pokud jsou nalevo, tak se úhly přičítají, pokud napravo tak se odečítají. Toto platí pro směr ccv (counter clockwise). Pro opačný směr se znaménka prohodí. Pokud se suma všech úhlů nerovná 0, leží bod uvnitř a pokud je suma rovna 0, leží bod mimo polygon. (1)

Princip algoritmu:



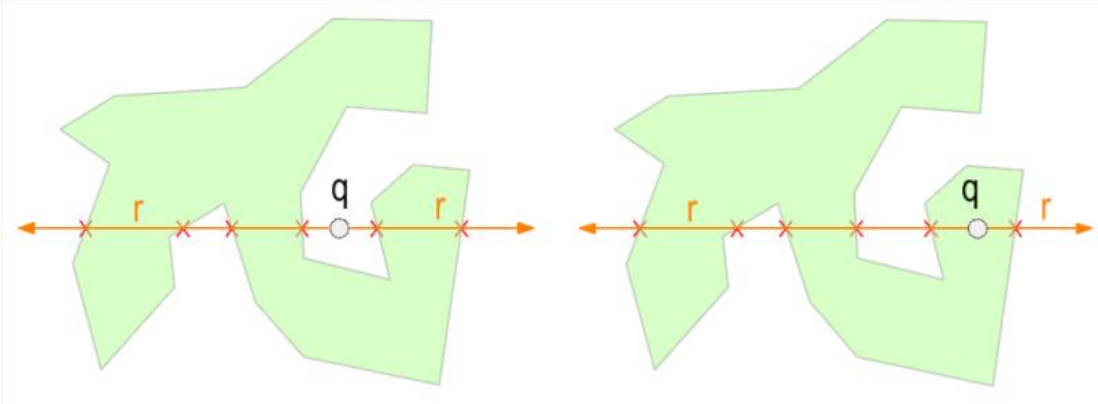
Obrázek 1: Princip Winding Number algoritmu

a. Inicializuj $\Omega=0$, tolerance ε.
b. Opakuj pro Υ trojici (p_i, q_i, p_{i+1})
   Urči polohu q vzhledem k $p = (p_i, p_{i+1})$
   Urči úhel $\omega_i = \angle p_i, q_i, p_{i+1}$
   If $q \in \overline{\sigma_\iota}$, pak $\Omega = \Omega + \omega_i$ //Bod v levé polorovině
   Else $\Omega = \Omega$ - ω_i //Bod v pravé polorovině
c. If $||\omega|-2\pi| < \varepsilon$, pak $q \in P$
d. $Else\ q \notin P$

II. Ray Crossing algorithm

Máme-li danou přímku r procházející zadaným bodem q. Je poloha bodu pomocí Algoritmu Ray Crossing definována počtem průsečíků přímky procházející daným bodem q s hranami polygonu. Pokud je počet sudý, leží bod vně polygonu, pokud lichý, leží uvnitř. Pro eliminaci singularit je uvažována pouze jedna polorovina vzhledem k vedené přímce procházející zadaným bodem. Zároveň vložíme počátek soustavy do bodu q, čímž redukujeme souřadnice vrcholů polygonu vzhledem k bodu. Hledáme tedy jen ty průsečíky, které leží napravo od osy x. Ta po redukci prochází zadaným bodem. Uvažujeme tedy pouze průsečíky prvního kvadrantu lokální soustavy. (2)

Princip algoritmu:



Obrázek 2: Princip upraveného modelu Ray Crossing (2)

Upravená varianta:

Eliminace singularit: $r(q)$ prochází vrcholem P nebo hranou.

Přímka definovaná $r(q)$ dělí σ na σ_u, σ_l

$\sigma_u' = \sigma_u - \{r(q)\}, \sigma_l' = \sigma_l - \{r(q)\}$

σ_u' obsahuje body $p_i = [x_i, y_i]$, kde $y_i > y_q$

σ_l' obsahuje body $p_i = [x_i, y_i]$, kde $y_i < y_q$

Inkrementace $k(q, P)$: jeden z bodů hrany nad $r(q)$ *(v* σ_u'*)* a druhý z bodů na/pod $r(q)$ *(v* σ_l'*)*

Princip upravené varianty s redukcí:

a. Inicializuj k=0
b. Opakuj pro všechny body $p_i \in P$:

$$x_i' = x_i - x_q$$

$$y_i' = y_i - y_q$$

if $((y_i' > 0)$ && $(y_{i-1}' <= 0)$ || $(y_{i-1}' > 0)$ && $(y_i' <= 0))$

$$x_m' = (x_i' * y_{i-1}' - x_{i-1}' * y_i')/(y_i' - y_{i-1}')$$

$if\,(x_m' > 0)$ pak $k = k+1$

c. if $(k\%2) \neq 0$ pak $q \in P$
d. Else $q \notin P$

5. Problematické situace a jejich rozbor (tj. simplexy) + ošetření těchto situací v kódu

Problematické situace, jinak také singularity jsou stavy, kdy se bod nachází buď na linii nebo je totožný s jedním z jejích vrcholů.

U situace, kdy bod leží na vrcholu je potřeba porovnat souřadnice x, y vrcholů polygonu se zadaným bodem. Pokud se souřadnice vrcholů a bodů rovnají jsou totožné.

U algoritmu Winding Number se poloha bodu zjišťuje pomocí námi definované funkce *getPointLinePosition*, která určuje, zdali se bod nachází nalevo či napravo od segmentu polygonu. Pokud se bod nenachází ani na jedné straně, tzn. *(t < eps) V (t > -eps)*, kde eps je definovaná konstanta blízká nule, nachází se bod na linii.

```
//Point in:
 if(poss==1)                    the left halfplane
    {omega_sum +=omega;}
 else if (poss==0)              the right halfplane
    {omega_sum -=omega;}
 else {return -1;}               on the borderline
```

U algoritmu Ray Crossing je singularita řešena pomocí počtu průsečíku na levé a pravé straně od bodu. Pokud je počet na jedné straně sudý, a na druhé lichý, bod se nachází na hraně polygonu.

```
//Border singularity
if (k1%2<k%2 || k1%2>k%2)
   {return -1;}
```

6. Vstupní data, formát vstupních dat, popis.

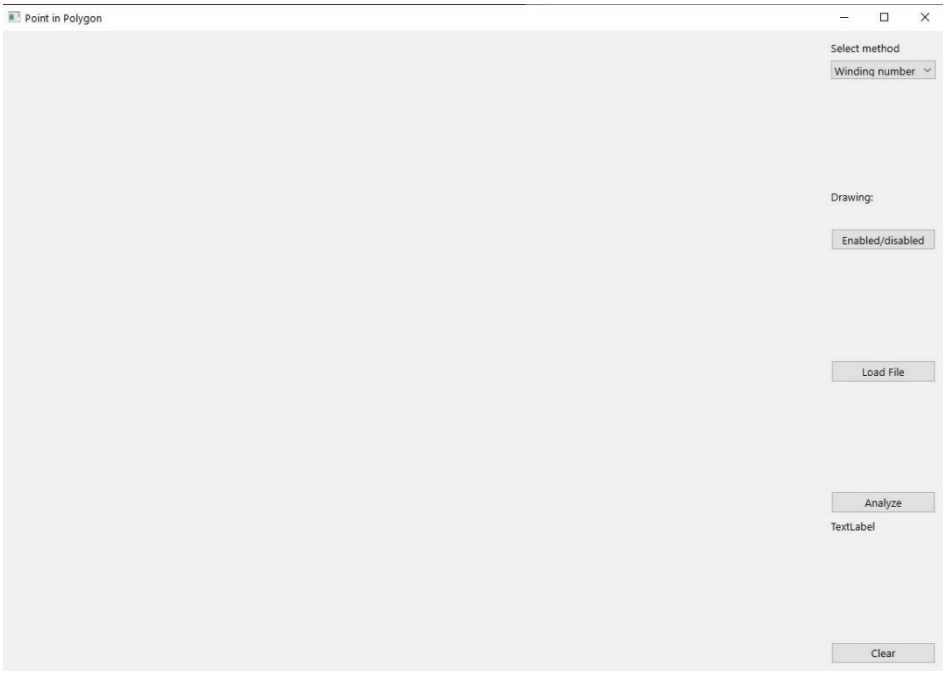
Vstupní data jsou načítána ve formátu txt. Data obsahují id bodu a vrcholy polygonů dané souřadnicemi x a y. Data jsou do programu načítána pomocí tlačítka Load File v grafickém okně.

Formát vstupních dat: id >> x >> y

7. Výstupní data, formát výstupních dat, popis

Výstupem a výsledkem je grafické okno, které obsahuje ovládací prvky aplikace, okno pro grafické zobrazení dat a samotný popis polohy bodu v podobě psaného textu.

Ovládacími prvky jsou combobox pro výběr algoritmu, tlačítka pro povolení kresby bodu a analýzu bodu a tlačítko clear, které smaže okno.



Obrázek 3: Okno aplikace po spuštění

Obrázek 4: Okno aplikace - analýza polohy bodu

8. Dokumentaci: popis tříd, datových položek a jednotlivých metod

```
class Algorithms
public:
  int getPointLinePosition(QPointF &a, QPointF &p1, QPointF &p2);
```

- zjištění vzájemné polohy přímky a bodu (vlevo, vpravo, hrana)
- vstup: souřadnice bodu q a lomových bodů polygonu

```
double get2LinesAngle(QPointF &p1, QPointF &p2, QPointF &p3,QPointF &p4);
```

- zjištění hodnoty úhlu mezi dvěma hranami polygonu
- vstup: souřadnice bodů vektorů

```
int getPositionWinding(QPointF &q, std::vector<QPointF> &pol);
```

- zjištění polohy bodu vzhledem k polygonu
- vstup: souřadnice bodu a polygonu

```
int getPositionRayCrossing(QPointF &q, std::vector<QPointF>&pol);
```

- zjištění polohy bodu vzhledem k polygonu
- vstup: souřadnice bodu a polygonu

```
class Draw : public QWidget
private:
QPoint q;
```

- definice proměnné pro načtení bodu

```
boolean enabledraw;
```

- vypnutí/zapnutí možnosti kreslit bod na plátno

```
double x,y;
```

- definice proměnné souřadnic

```
std::vector<QPolygonF> pol;
```

- definice proměnné pro načtení polygonu

```
public:
void paintEvent(QPaintEvent *event);
```

- vykreslení polygonu a bodu

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

- definice souřadnic bodu

```
void clear();
```

- mazaní vykreslených proměnných

```
void changeStatus(){enabledraw=!enabledraw;};
```

- změna povolení pro kreslení bodu

```
QPointF getPoint(){return q;};
```

- vrací souřadnice kresleného bodu

```
std::vector<QPolygonF> getPolygon(){return pol;};
```

- vrací souřadnice polygonu

```
void loadFile(std::string &path);
```

- načtení dat formátu txt

```
void setX(double x_){x=x_;};
```

- přiřazuje hodnotu x

```
void setY(double y_){y=y_;};
```

- přiřazuje hodnotu y

9. Závěr, možné či neřešené problémy, náměty na vylepšení

Námi vytvořená aplikace dokáže z načteného souboru a námi definovaného bodu určit, nachází-li se bod uvnitř, vně nebo na hraně polygonu.

Z bonusových úloh byly vyřešeny následující: Analýza polohy bodu (uvnitř/vně) metodou Ray Algorithm, ošetření singulárního případu u Ray Algorithm a ošetření singulárního případu u obou algoritmů kdy bod je totožný s vrcholem jednoho či více polygonů.

Aplikace nefunguje při určení polohy bodu pro multipolygony z důvodu nedostatečných zkušeností s programováním.

Aplikace by se dala vylepšit zvýrazněním řešeného polygonu pomocí barev či šraf. Bohužel tento problém nebyl řešen vzhledem k malým zkušenostem s programováním.

Citovaná literatura

1. **Topiwala, Anirudh.** *towards data science.* [Online] 2020. [Citace: 17. 10 2021.] https://towardsdatascience.com/is-the-point-inside-the-polygon-574b86472119.

2. **Tomáš, Bayer.** Perslonal page of Bayer Tomas. *Charles University of Prague.* [Online] [Citace: 17. 10 2021.] https://web.natur.cuni.cz/~bayertom/images/courses/Adk/adk3.pdf.

Seznam obrázků

ČVUT
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

V Praze dne 2.11.2021