České vysoké učení technické v Praze

Fakulta stavební



Algoritmy v digitální kartografii

Úloha č. 1: Generalizace budov

Skupina:

Sabina Kličková

Martin Vajner

Zimní semestr 2021/2022

Obsah

[1. Zadání 3](#_Toc86907335)

[2. Bonusové úlohy 3](#_Toc86907336)

[3. Popis a rozbor problémů + vzorce. 3](#_Toc86907337)

[4. Popisy algoritmů formálním jazykem 4](#_Toc86907338)

[5. Problematické situace a jejich rozbor (tj. simplexy) + ošetření těchto situací v kódu 6](#_Toc86907339)

[6. Vstupní data, formát vstupních dat, popis. 6](#_Toc86907340)

[7. Výstupní data, formát výstupních dat, popis 6](#_Toc86907341)

[8. Dokumentaci: popis tříd, datových položek a jednotlivých metod 7](#_Toc86907342)

[9. Závěr, možné či neřešené problémy, náměty na vylepšení 9](#_Toc86907343)

[Citovaná literatura 9](#_Toc86907344)

[Seznam obrázků 9](#_Toc86907345)

# Zadání

Vstup:

Výstup:

Ze souboru načtěte vstupní data představovaná lomovými body budov. Pro tyto účely použijte vhodnou datovou sadu, např. ZABAGED.

Pro každou budovu určete její hlavní směry metodami:

* Minimum Area Enclosing Rectangle
* Wall Average

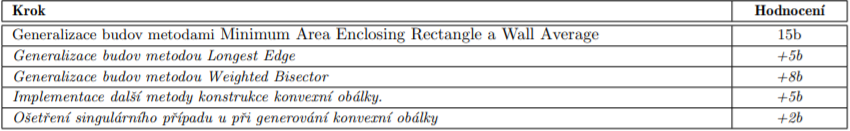
U první metody použijte některý z algoritmů pro konstrukci konvexní obálky. Budovu nahraďte obdélníkem se středem v jejím těžišti orientovaným v obou hlavních směrech, jeho plocha bude stejná jako plocha budovy. Výsledky generalizace vhodně vizualizujte.

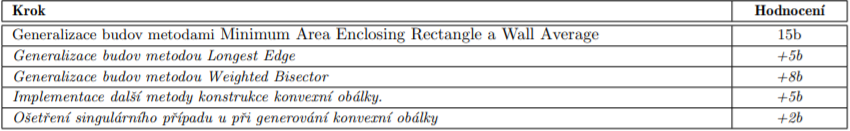
Odhadněte efektivitu obou metod, vzájemně porovnejte a zhodnoťte. Pokuste se identifikovat, pro které tvary budov dávají metody nevhodné výsledky, a pro které naopak poskytují vhodnou aproximaci.



# Bonusové úlohy

V této úloze byly zpracovány následující bonusové úlohy:





1. Popis a rozbor problémů + vzorce.

Definice konvexní obálky: Konvexní obálka konečné bodové množiny S je nejmenší konvexní mnohoúhelník P, který obsahuje množinu S.

Konvexní obálka je hranice spojující body množiny tak, aby každý bod množiny ležel uvnitř nebo na hranici obálky.



Obrázek : Ukázka konvexní obálky nad budovou

Množina S je konvexní, leží-li spojnice libovolných dvou prvků uvnitř této množiny.

1. Popisy metod detekce natočení budov
2. **Minimum Area Enclosing Rectangle**

Metoda je založena na principu minimální plochy obklopující obdélník.

1. **Wall Average**
2. **Longest Edge**
3. **Weighted Bisektor**
4. Problematické situace a popsání bonusových úloh

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;}

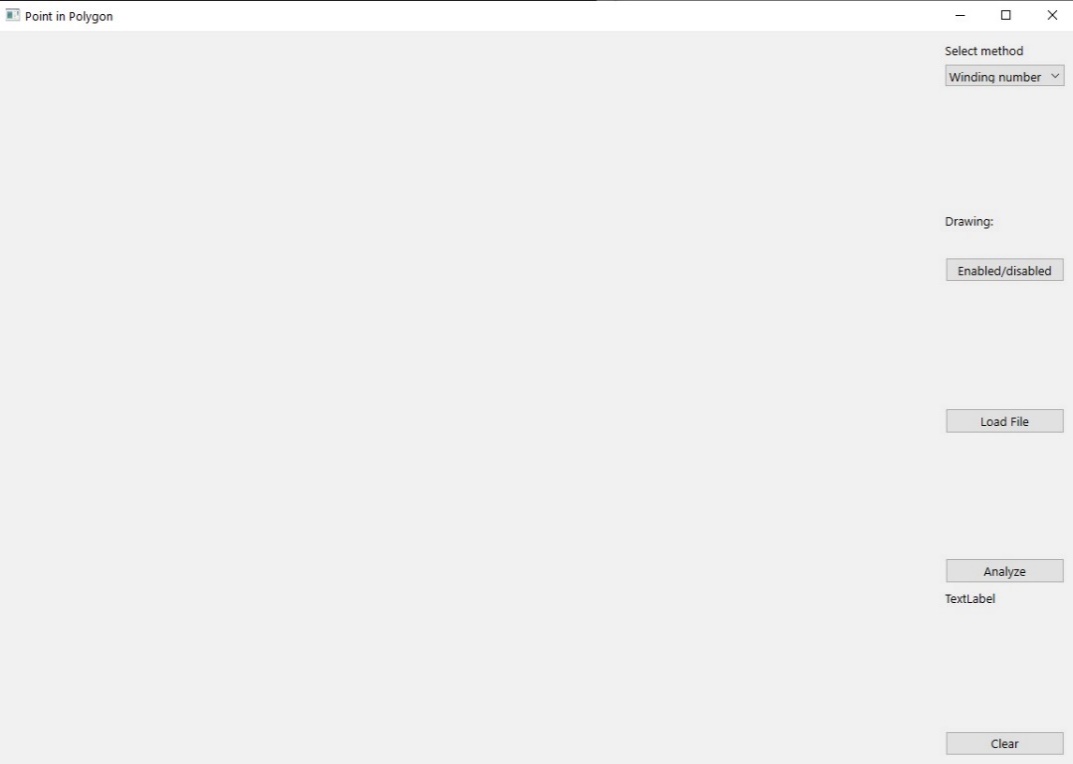
1. Vstupní data, formát vstupních dat, popis.

Lomové body budov.

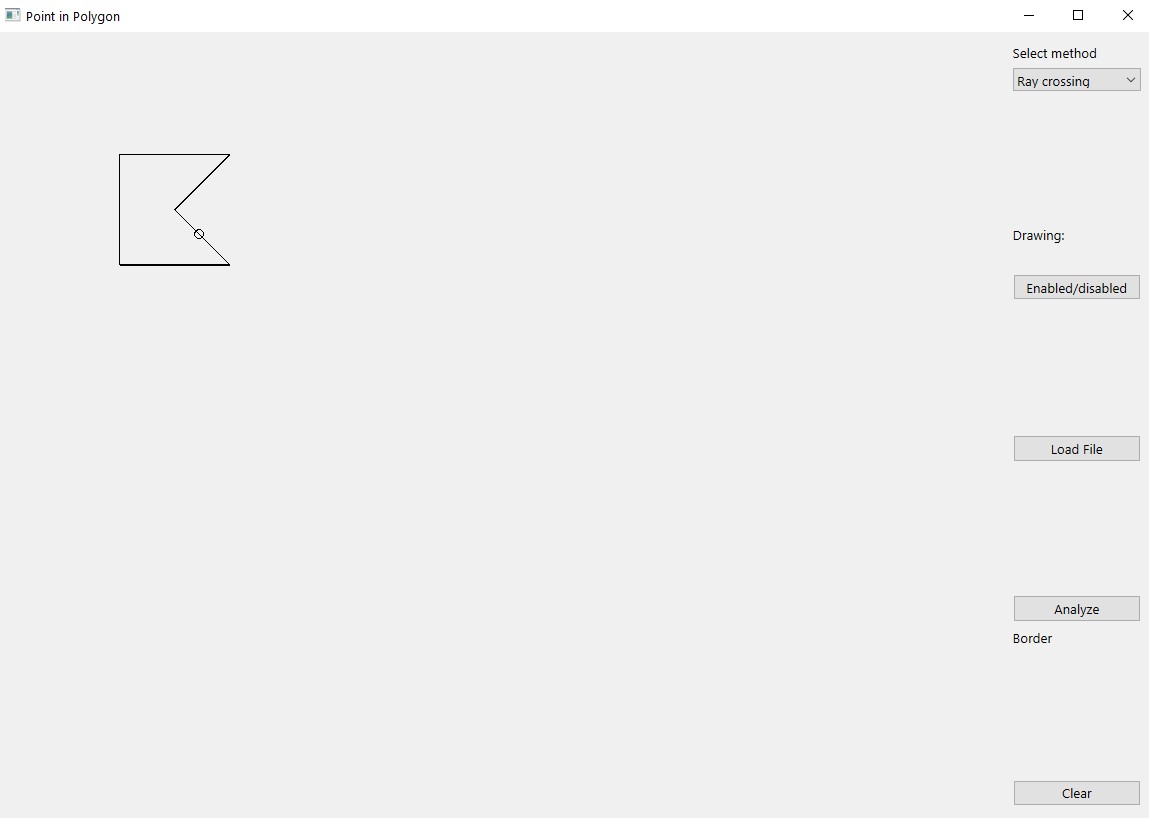
1. Výstupní data, formát výstupních dat, popis

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 : Okno aplikace po spuštění



Obrázek : Okno aplikace - analýza polohy bodu

1. 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

1. 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

[Obrázek 1: Princip Winding Number algoritmu 4](file:///C:\Users\Sabča\Downloads\Úloha_1%20(1).docx#_Toc86750476)

[Obrázek 2: Princip upraveného modelu Ray Crossing (2) 5](#_Toc86750477)

[Obrázek 3: Okno aplikace po spuštění 7](#_Toc86750478)

[Obrázek 4: Okno aplikace - analýza polohy bodu 7](#_Toc86750479)

V Praze dne 2.11.2021