ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE FAKULTA STAVEBNÍ KATEDRA GEOMATIKY					
Název předmětu Algoritmy digitální kartografie a GIS					
Úloha U1	Název úlohy: Geometrické vyhledávání bodu				
akademický rok 2024/2025	semestr letní	studijní skupina C102	vypracoval Matyáš Pokorný Tereza Černohousová	datum 15.03.2025	klasifikace

Technická zpráva

1 Zadání

Úloha č. 1: Geometrické vyhledávání bodu
Vstup: Souvislá polygonová mapa n polygonů $\{P_1, \ldots, P_n\}$, analyzovaný bod q.
Výstup: P_i, $q \in P_i$.

Nad polygonovou mapou implementujete Ray Crossing 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.

2 Bonusové úlohy

Z bonusových úloh námi byly zpracovány:

1. Analýza polohy bodu (unitř/vně) metodou Winding NumberAlgorithm
2. Ošetření singulárního případu u Winding NumberAlgorithm: bod leží na hraně polygonu
3. Ošetření singulárního případu u Ray Crossing Algorithm: bod leží na hraně polygonu
4. Ošetření singulárního případu u obou algoritmů: bod je totožný s vrcholem jednoho či více polygonů.
5. Zvýraznění všech polygonů pro oba výše uvedené singulární případy.
6. Rychlé vyhledávání potenciálních polygonů (bod uvnitř min-max boxu).
7. Načtení vstupních dat ze *.shp

3 Pracovní postup a použité algoritmy

Úloha byla vypracována v prostředí QT Creator, jazyk C++. Při tvorbě úlohy bylo vycházeno z probrané látky na cvičení a přednáškách. Pro geometrické vyhledávání bodu byly použity dva algoritmy – *Ray Crossing* a *Winding number*. Následující odstavce popisují jednotlivé soubory projektu a použité funkce, které byly sestaveny pro správné fungování aplikace.

3.1 algorithm.cpp

3.1.1 `analyzePointndPolPosition`

Tento algoritmus používá metodu *Ray Crossing* k určení, zda bod leží uvnitř nebo vně polygonu. Nejprve se kontroluje, zda bod není na hranici polygonu nebo není vrcholem. Pokud bod leží na hranici, algoritmus ihned vrátí tuto informaci.

Pokud bod není na hranici, pokračuje se algoritmem *Ray Crossing*, kde se z bodu kreslí paprsek a počítá se počet průsečíků s hranami polygonu. Pokud je počet průsečíků lichý, bod je uvnitř polygonu; pokud sudý, bod je vně.

Při výpočtu je třeba pečlivě zpracovat hrany polygonu, zejména pokud hrana prochází přes bod. Po provedení algoritmu je výsledek jednoznačný: bod je uvnitř, na hranici, nebo vně polygonu.

Popis algoritmu Ray Crossing [1]

Bodem q vedena polopřímka r (paprsek, tzv. ray)

$$r(q) : y = y_q.$$

Invariance vůči směru r. Počet průsečíků přímky r s oblastí P

$$k\%2 = \begin{cases} 1, & q \in P, \\ 0, & q \notin P. \end{cases}$$

\+ O řád rychlejší než Winding Number.

\- Problémem singularity.

3.1.2 `get2LineAngle`

Tato funkce vypočítá úhel mezi dvěma vektory, které jsou definovány čtyřmi body, tedy dvěma dvojicemi bodů. Pro výpočet úhlu mezi těmito vektory se používá *skalární součin* (dot product), který je definován jako:

$$\vec{a} \cdot \vec{b} = |a||b| \cos(\theta)$$

kde $\vec{a}$ a $\vec{b}$ jsou vektory, $|a|$ a $|b|$ jsou jejich velikosti (délky), a θ je úhel mezi nimi. Funkce tedy nejprve spočítá velikosti obou vektorů pomocí vzorců pro velikost vektoru a jejich skalární součin. Následně pomocí inverzního kosinu (arccos) vypočítá hodnotu úhlu θ mezi těmito dvěma vektory. Tato funkce je užitečná při analýze geometrických vztahů mezi liniemi nebo vektory, například pro určení orientace dvou čar nebo při výpočtu sklonu dvou hran polygonu.

3.1.3 `determinant2x2`

Tato funkce vypočítá determinant matice 2x2, což je základní operace v lineární algebře. Determinant matice 2×2 se vypočítá podle vzorce:

$$\det(A) = a \cdot d - b \cdot c$$

kde matice $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$. Tento výpočet je užitečný při různých geometrických výpočtech, zejména pro testování kolinearity bodů. Determinant 2x2 je součástí metody pro určení, zda tři body leží na stejné přímce, tedy jsou kolineární. Funkce se také může použít při analýze orientace polygonů nebo při hledání průsečíků dvou čar v rovině. Determinant je klíčový při výpočtu oblasti, orientace a dalších geometrických vlastností.

3.1.4 `isPointOnEdge`

Tato funkce zjistí, zda daný bod leží na jedné z hran polygonu. Využívá se *křížového součinu*, což je operace mezi dvěma vektory, která určuje, zda jsou tyto vektory kolineární, tedy leží na stejné přímce. Pro určení, zda bod q leží na hraně definované dvěma vrcholy A a B, se vypočítá vektor AB (od A k B) a vektor AQ (od A k bodu q). Následně se spočítá křížový součin těchto dvou vektorů. Pokud je křížový součin roven nule, znamená to, že vektory jsou kolineární, což znamená, že bod q leží na přímce mezi vrcholy A a B. Dále se kontroluje, zda bod q leží mezi vrcholy A a B na hranici polygonu. Tato kontrola zahrnuje ověření, že souřadnice bodu q jsou v mezích souřadnic vrcholů A a B. Funkce je užitečná pro detekci, zda bod leží na hraně polygonu, což je důležité pro algoritmy, které potřebují určit, zda bod patří do polygonu nebo jestli je na jeho hranici.

3.1.5 `WindingNumber`

Funkce `WindingNumber` implementuje algoritmus pro určení, zda bod leží uvnitř nebo vně polygonu. Algoritmus vychází z analýzy orientace hran polygonu vůči bodu. Nejprve se kontroluje, zda bod leží na některém vrcholu polygonu, v takovém případě funkce vrátí hodnotu -2. Pokud bod leží na hraně polygonu, vrátí -1.

Pokud bod není ani na vrcholu, ani na hraně, funkce spočítá tzv. winding number. Pro každý okraj polygonu se spočítají vektory mezi vrcholy a bodem, z nichž se vypočítá determinant. Tento determinant určuje, zda bod leží na levé nebo pravé straně hrany. Dále se spočítá úhel mezi vektory, který se přičte nebo odečte od hodnoty ω, v závislosti na orientaci hrany.

Na konci funkce, pokud je hodnota ω blízká 2π, znamená to, že bod leží uvnitř polygonu. Pokud je hodnota menší, bod leží mimo polygon. Tolerance ε určuje přesnost porovnání hodnoty ω. Funkce vrací 1, pokud je bod uvnitř polygonu, 0, pokud je venku, a -1 nebo -2 v případě, že bod leží na hraně nebo vrcholu polygonu.

Popis algoritmu Winding number [1]
Suma Ω všech rotací ω_i (měřená CCW):

$$\Omega(q, P) = \frac{1}{2\pi} \sum_{i=1}^{n} \omega(p_i, q, p_{i+1}),$$

které musí průvodič (q, p_i) opsat nad všemi body $p_i \in P$.
Přímka $P(p_i, p_{i+1})$ dělí σ na σ_l, σ_r (Left/Right Halfplane):

$$\sigma_l = \{q = [x_q, y_q], t \geq 0\},$$

$$\sigma_r = \{q = [x_q, y_q], t < 0\},$$

kde

$$t = \begin{vmatrix} x_{i+1} - x_i & y_{i+1} - y_i \\ x_q - x_i & y_q - y_i \end{vmatrix}$$

Úhly $\omega(p_i, q, p_{i+1})$ orientované:

$$\omega(p_i, q, p_{i+1}) = \begin{cases} +\omega(p_i, q, p_{i+1}), & q \in \sigma_l, \\ -\omega(p_i, q, p_{i+1}), & q \in \sigma_r. \end{cases}$$

Případ 1: Úhel $\omega(p_i, q, p_{i+1})$ má CW orientaci.
Případ 2: Úhel $\omega(p_i, q, p_{i+1})$ má CCW orientaci.
Winding Number

$$\Omega(q, P) = \begin{cases} 1, & q \in P, \\ 0, & q \notin P. \end{cases}$$

3.1.6 `isPointInMinMaxBoxOfPolygon`

Tato funkce určuje, zda bod leží uvnitř minimálně a maximálně ohraničeného obdélníku, který je definován souřadnicemi vrcholů polygonu. Funkce prochází všechny vrcholy polygonu a zjistí nejmenší a největší hodnoty souřadnic x a y, čímž definuje hranice obdélníku, který polygon ohraničuje. Tento obdélník je známý jako *bounding box*. Po vytvoření tohoto obdélníku se zkontroluje, zda bod q leží uvnitř tohoto ohraničujícího obdélníku. Pro bod je bod uvnitř obdélníku, pokud jeho souřadnice x jsou mezi minimální a maximální hodnotou souřadnic x polygonu a jeho souřadnice y jsou mezi minimální a maximální hodnotou souřadnic y. Tato metoda je rychlá, protože se neprovádí komplexní výpočty jako u metod *Ray Crossing* nebo *Winding Number*, ale je užitečná jako předběžný test pro určení, zda je bod v blízkosti polygonu, což může snížit náklady na výpočty v následujících algoritmech.

3.1.7 `normalizePolygons`

Tato funkce je určena pro úpravu souboru polygonů tak, aby všechny polygony byly umístěny do středu obrazovky nebo daného okna. Nejprve je vypočítáno *centroid*, což je průměrná pozice všech bodů napříč všemi polygonálními objekty. Centroid je považován za střed těžiště všech polygonů. Po výpočtu centroidu se každý polygon posune (nebo normalizuje) tak, aby jeho těžiště bylo umístěno ve středu okna nebo obrazovky. Posunutí se provádí pomocí ofsetů, které se získají z rozdílu mezi polohou centroidu a středem okna. Tento krok je užitečný zejména při zobrazení polygonů na obrazovce, protože zajistí, že všechny polygony budou dobře viditelné a vycentrované ve vymezeném prostoru.

3.1.8 `calculateCentroid`

Tato funkce vypočítá *centroid* (střed těžiště) souboru polygonů, což je průměrná pozice všech bodů, které tvoří všechny polygony. Funkce prochází všechny body každého polygonu a sčítá jejich souřadnice x a y, čímž získává celkový součet pro všechny souřadnice. Poté vyděluje tento součet počtem bodů, čímž získá průměrnou hodnotu pro souřadnice x a y. Výsledkem je bod, který představuje střední hodnotu pozice všech polygonů. Tento střed je užitečný pro různé operace, jako je centrální normalizace polygonů nebo pro výpočty geometrických vlastností. Tato funkce je užitečná například při pokusech o nalezení vyvážené pozice pro zobrazení polygonů nebo pro výpočty těžiště v dalších geometrických algoritmech.

3.2 draw.cpp

Tento kód implementuje třídu `Draw`, která umožňuje vykreslování polygonů a bodu v Qt. Umožňuje interakci s uživatelem pomocí myši pro zadávání bodů a polygonů a poskytuje funkce pro načítání polygonů z textového a SHP souboru, vykreslování polygonů a bodu q, a pro zvýraznění polygonů.

3.2.1 Inicializace třídy `Draw`

Konstruktor `Draw::Draw(QWidget *parent)` inicializuje souřadnice bodu q na $(0, 0)$, nastaví proměnné `add_point` na `false` (režim přidávání polygonu) a `isShapefileLoaded` na `false`.

3.2.2 Zachycení stisku myši

Funkce `mousePressEvent(QMouseEvent *e)` reaguje na stisk myši. Pokud je zapnutý režim přidávání bodu, zaznamená souřadnice jako bod q. Pokud není, přidá bod do posledního polygonu a následně zavolá `repaint()` pro překreslení.

3.2.3 Vymazání polygonů

Funkce `clearPolygons()` odstraní všechny polygony a vykreslí plátno. Funkce `clear()` odstraní všechny polygony a bod q, a také provede překreslení.

3.2.4 Vykreslování objektů

Funkce `paintEvent(QPaintEvent *event)` vykresluje polygony červeně s žlutým vyplněním, zvýrazněné polygony zeleně a bod q jako modrý kruh o poloměru 5.

3.2.5 Přepnutí režimu zadávání bodu nebo polygonu

Funkce `switch_source()` přepíná mezi režimem přidávání bodu a polygonu. Režim se přepíná pomocí proměnné `add_point`.

3.2.6 Zvýraznění polygonů

Funkce `highlightPolygon(const QVector<int> indices)` zvýrazní specifikované polygony a zavolá `repaint()` pro překreslení plátna.

3.2.7 Vymazání zvýrazněných polygonů

Funkce `clearHighlighted()` odstraní všechny zvýrazněné polygony a vykreslí plátno znovu.

3.2.8 Načtení polygonů z textového souboru

Funkce `openFile()` otevře textový soubor s polygonovými daty. Každý řádek obsahuje dvě hodnoty souřadnic bodu polygonu. Prázdný řádek znamená konec polygonu. Funkce načte

souřadnice a vytvoří `QPolygonF` objekty, které uloží do vektoru `polygons`. Po načtení polygonů se plátno překreslí a režim přepne na přidávání bodu.

3.2.9 Načtení polygonů z SHP souboru

Funkce `openSHP()` načítá data z SHP souboru pomocí knihovny GDAL. Po načtení polygonů je normalizuje pomocí `algorithms::normalizePolygons()` a plátno překreslí. Režim přepne na přidávání bodu.

3.3 mainform.cpp

Tento kód představuje implementaci hlavní třídy `MainForm`, která slouží k interakci s uživatelem v prostředí Qt. Obsahuje různé akce pro analýzu polohy bodu vůči polygonu, načítání dat a vykreslování výsledků. Kód používá grafickou knihovnu Qt a knihovnu GDAL pro načítání shapefile souborů.

Inicializace okna aplikace

Funkce `MainForm::MainForm(QWidget *parent)` je konstruktor, který nastavuje uživatelské rozhraní aplikace pomocí `ui->setupUi(this)`. V destruktoru `MainForm:: MainForm()` je pak uvolněna paměť pro uživatelské rozhraní.

Přepínání mezi vstupy: bod nebo polygon

Funkce `on_actionPoint_Polygon_triggered()` umožňuje přepínání mezi vstupy pro analýzu – buď bod q nebo polygon P. Funkce volá metodu `switch_source()` třídy `Canvas`, která přepne režim zadávání dat.

Algoritmus Ray Crossing

Tento algoritmus získá bod q a sadu polygonů. Nejprve vymaže zvýrazněné polygony a ověří, zda bod q leží uvnitř minimálního ohraničujícího boxu polygonu. Pokud ano, spustí algoritmus `analyzePointndPolPosition()`, který analyzuje polohu bodu vůči polygonu. Výsledek je zobrazen v titulku okna: **Inside** – bod je uvnitř polygonu, **Outside** – bod je vně polygonu, **Edges** – bod leží na hraně, **Vertex** – bod je vrcholem polygonu.

Algoritmus Winding Number

Tento algoritmus získá bod q a sadu polygonů. Pro každý polygon ověří, zda bod q leží uvnitř minimálního ohraničujícího boxu. Pokud ano, spustí algoritmus `WindingNumber()`, který určí, zda je bod uvnitř nebo vně polygonu na základě počtu otoček kolem bodu. Pokud bod patří do polygonu, zvýrazní se příslušný polygon a zobrazí se výsledek: **Inside** – bod je uvnitř, **Vertex** – bod je vrcholem, **Edges** – bod je na hraně, jinak **Outside**.

Načtení polygonů z textového souboru

Funkce `on_actionOpen_triggered()` umožňuje načíst polygonální data z textového souboru. Po načtení jsou polygony vykresleny na plátno.

Načtení souboru SHP pomocí GDAL

Funkce `on_actionOpen_SHP_triggered()` načítá shapefile soubor (.shp) pomocí knihovny GDAL. Po načtení souboru jsou polygony vykresleny na plátno.

Ukončení aplikace

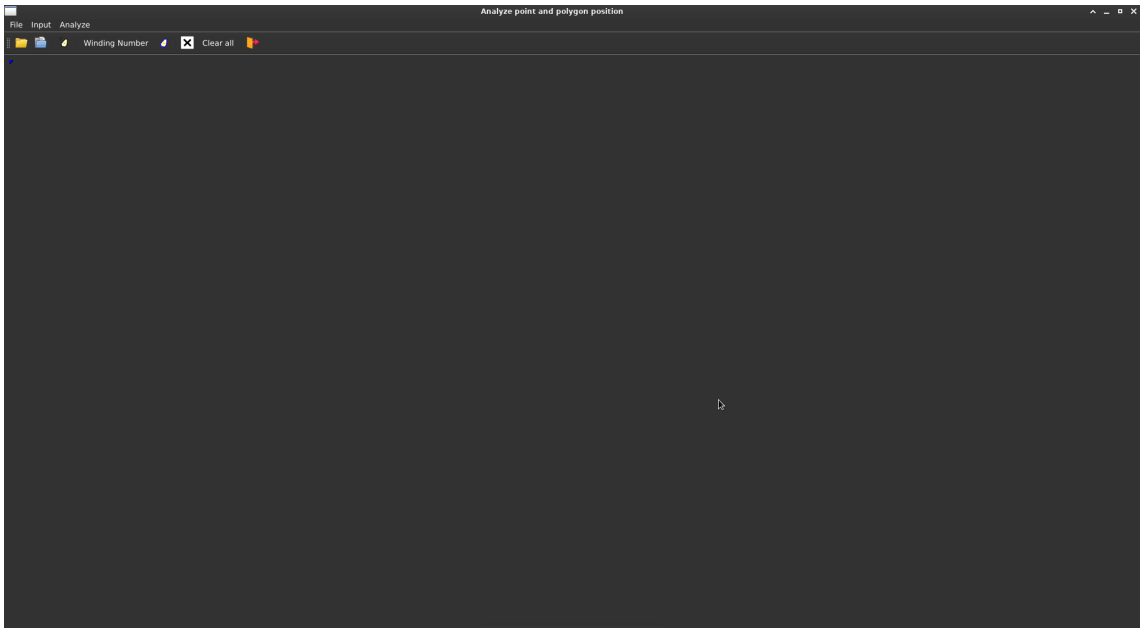
Funkce `on_actionExit_triggered()` ukončuje běh aplikace tím, že volá metodu `QApplication::quit()`.

Vymazání dat

Funkce `on_actionClear_data_triggered()` odstraní všechny vykreslené polygony z plátna. Funkce `on_actionClear_all_triggered()` vymaže všechny polygony, bod q, a provede překreslení plátna.

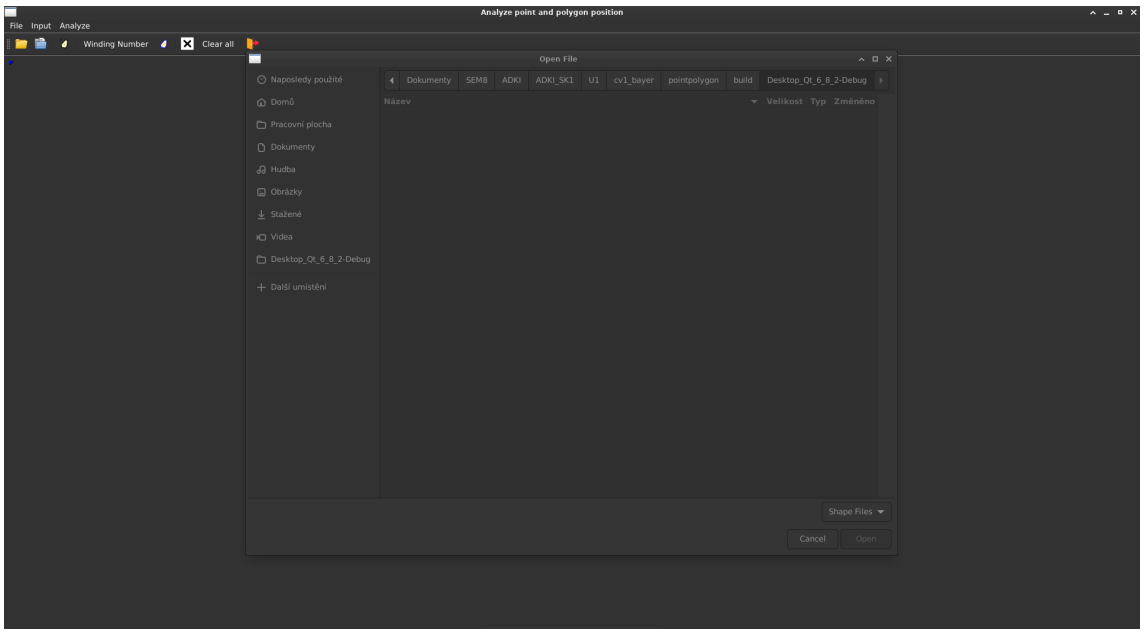
4 Ukázka aplikace

Po spuštění kódu je otevřeno grafické rozhraní, ve kterém je možné naši aplikaci ovládat. Při spuštění se zobrazí prázdná aplikace s několika ikonami. Bohužel pro nás z neznámých důvodů se některé ikony tlačítek nezobrazují a nám se nepodařilo zjistit proč *(Winding number)*. Nicméně na funkčnosti aplikace to nic nemění.



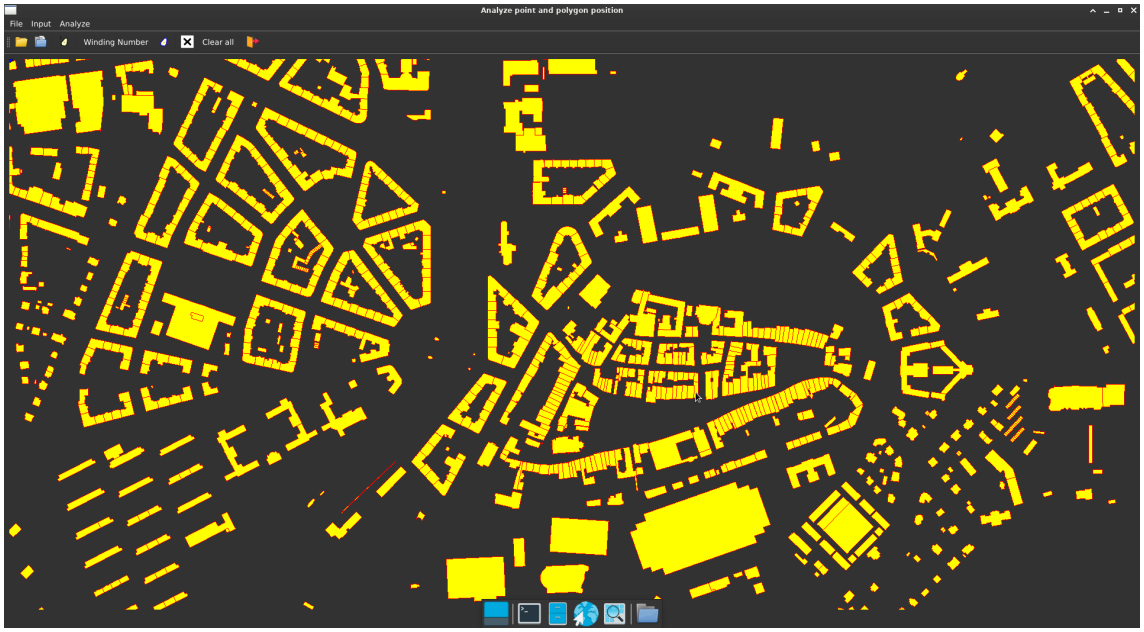
Obrázek 1: Po otevření aplikace

Aplikace načítá SHP soubory s polygony. Pomocí ikony *Open file* je možné vybrat konkrétní soubor z počítače.



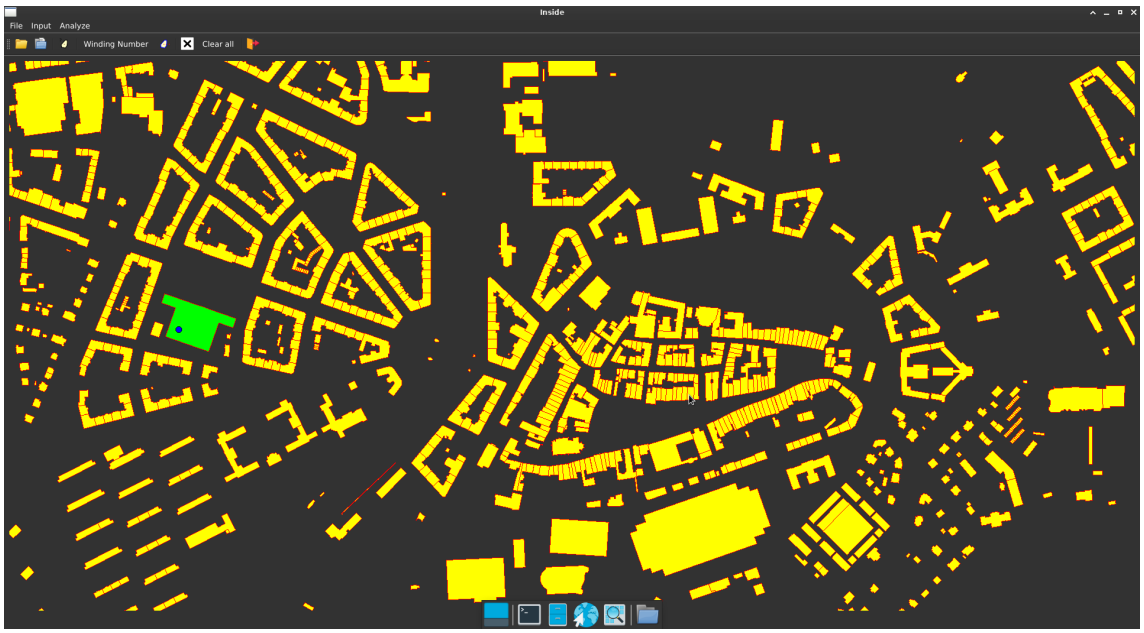
Obrázek 2: Otevření SHP souboru z počítače

Soubor SHP je načten. Plocha polygonů je vyplněna barvou a odlišnou barvou je znázorněna vnější hranice polygonu.



Obrázek 3: Polygony z SHP

V aplikaci je možné vykreslit bod, poté je na výběr ze dvou algoritmů, které určují geometrickou polohu bodu - *Ray Crossing* nebo *Winding Number*. Algoritmus provede analýzu, zda se bod nachází uvnitř nebo vně polygonu (případně i singulárních případů) a do titulku okna aplikace napíše výsledek. Zároveň je polygon, ve kterém se nachází bod, zvýrazněn.



Obrázek 4: Zvýraznění polygonu a určení polohy zvoleného bodu

5 Závěr

Pomocí QT Creatoru byla vytvořena jednoduchá aplikace pro geometrické vyhledávání bodů. Pomocí aplikace můžete jednoduše vykreslit polygon a bod. Pomocí algoritmů *Winding Number* a *Ray Crossing* lze určit, zda se bod nachází uvnitř nebo vně polygonu. Byly ošetřené i singulářní případy - zda se bod nachází na hraně polygonu nebo v jeho vrcholu. Pro všechny singulární případy bylo provedeno zvýraznění polygonu. Rychlé vyhledávání potenciálních polygonů bylo realizováno pomocí min-max boxu. Polygony je možné do aplikace nahrát hromadně jako SHP soubor.

Aplikace není ošetřená na případy, když má polygon díru. Dalším možným návrhem na zlepšení je například *snipping* kurzoru myši. Přichytávání by zajistilo, že se myš přichytí na vrchol nebo hranu polygonu a bod bude vložen konkrétně tam. Poté by byly zřejmé odpovědi i na singulární případy. U současné verze aplikace není možné se přesně trefit na konkrétní souřadnice vrcholu polygonu, aby bod byl vložen přímo na něj a nastal singulární případ.

Další nápad na zlepšení je, aby se odpověď na geometrickou polohu bodu vůči polygonu, byla zobrazena například pop-upem nebo jinou formou vyskakovacího okna. Současná verze aplikace vypisuje výsledek v titulku aplikace.

V Praze dne: 16.03. 2025 **T. Černohousová**

M. Pokorný

Pseudokód pro algoritmy

algorithms.cpp

Algorithm 1 Analyze Point in Polygon Position

Vstup: Bod q, Polygon *pol*
Výstup: Pozice bodu vůči polygonu: -1 (na hraně), -2 (na vrcholu), 1 (uvnitř), 0 (mimo)
if Bod q je na hraně polygonu *pol* **then**
    vrátit -1
end if
for každý vrchol p_i polygonu *pol* **do**
    if Bod q je na vrcholu p_i **then**
        vrátit -2
    end if
end for
$k \leftarrow 0$
for každá hrana (p_i, p_{i+1}) polygonu *pol* **do**
    Vypočítat zredukované souřadnice: $x_{ir}, y_{ir}, x_{i+1r}, y_{i+1r}$
    if Přetne paprsek z bodu q **then**
        $xm \leftarrow \frac{x_{i+1r} \cdot y_{ir} - x_{ir} \cdot y_{i+1r}}{y_{i+1r} - y_{ir}}$
        if $xm > 0$ **then**
            $k \leftarrow k + 1$
        end if
    end if
end for
if $k\%2 == 1$ **then**
    vrátit 1
else
    vrátit 0
end if

Algorithm 2 Get Angle Between Two Vectors

Vstup: Body $p1, p2, p3, p4$
Výstup: Úhel mezi vektory $\overrightarrow{p1p2}$ a $\overrightarrow{p3p4}$
Vypočítat složky vektoru: $u_x \leftarrow p2.x - p1.x$, $u_y \leftarrow p2.y - p1.y$
Vypočítat složky vektoru: $v_x \leftarrow p4.x - p3.x$, $v_y \leftarrow p4.y - p3.y$
Vypočítat skalární součin: $dot \leftarrow u_x \cdot v_x + u_y \cdot v_y$
Vypočítat velikosti vektorů: $n_u \leftarrow \sqrt{u_x^2 + u_y^2}$, $n_v \leftarrow \sqrt{v_x^2 + v_y^2}$
vrátit $\arccos\left(\frac{dot}{n_u \cdot n_v}\right)$

Algorithm 3 Check if Point is on Edge of Polygon

Vstup: Bod q, Polygon *pol*
Výstup: Pravda, pokud bod q leží na hraně polygonu *pol*, jinak Nepravda
Nastavit $eps \leftarrow 10^{-6}$ (malá tolerance pro křížový součin)
for každá hrana (p_i, p_{i+1}) polygonu *pol* **do**
    Vypočítat křížový součin: $t \leftarrow$ determinant2x2(a, b, c, d)
    if $|t| < eps$ **then**
        if Bod q je mezi vrcholy hrany **then**
            vrátit Pravda
        end if
    end if
end for
vrátit Nepravda

Algorithm 4 Winding Number Algorithm

Vstup: Bod q, Polygon *pol*
Výstup: 1 (uvnitř), 0 (mimo), -1 (na hraně), -2 (na vrcholu)
for každý vrchol p_i polygonu *pol* **do**
    if Bod q je na vrcholu p_i **then**
        vrátit -2
    end if
end for
if Bod q je na hraně polygonu *pol* **then**
    vrátit -1
end if
Inicializuj $omega \leftarrow 0$
for každá hrana (p_i, p_{i+1}) polygonu *pol* **do**
    Vypočítat determinant t
    Vypočítat úhel θ mezi vektory
    if $t > 0$ **then**
        $omega \leftarrow omega + \theta$ (otáčíme vlevo)
    else
        $omega \leftarrow omega - \theta$ (otáčíme vpravo)
    end if
end for
if $|omega - 2\pi| < \epsilon$ **then**
    vrátit 1
else
    vrátit 0
end if

Algorithm 5 Normalize Polygons

Vstup: Seznam polygonů *polygons*, Šířka okna *width*, Výška okna *height*
Výstup: Normalizované polygonu centrované v okně
Vypočítat centroid polygonů
Vypočítat offset pro centrování polygonů
for každý polygon v seznamu *polygons* **do**
    for každý bod v polygonu **do**
        Přelož bod podle offsetu
        Uprav Y-souřadnici pro výšku okna
    end for
end for

draw.cpp

Algorithm 6 Konstruktor třídy Draw

1: **Nastav** počáteční hodnotu bodu q na $(0, 0)$
2: **Nastav** příznak přidávání bodu (add_point) na `false`
3: **Nastav** příznak načtení shapefile (isShapefileLoaded) na `false`

Algorithm 7 Funkce mousePressEvent

1: Získej souřadnice x a y z události kliknutí myší
2: **if** add_point je `true` **then**
3: Nastav hodnoty souřadnic x a y pro bod q
4: **else**
5: Vytvoř bod p se souřadnicemi (x, y)
6: **if** polygony nejsou prázdné **then**
7: Přidej bod p do posledního polygonu
8: **end if**
9: **end if**
10: Překresli obrazovku

Algorithm 8 Funkce clearPolygons

1: Vymaž všechny polygony
2: Překresli obrazovku

Algorithm 9 Funkce clear

1: Vymaž všechny polygony
2: Nastav bod q na $(0, 0)$
3: Překresli obrazovku

Algorithm 10 Funkce paintEvent

```
1: Začni malovat na widget
2: Nastav pero na červenou a štětec na žlutou pro vykreslení polygonů
3: for každý polygon pol v seznamu polygonů do
4:     if index polygonu není v seznamu zvýrazněných polygonů then
5:         Vykresli polygon
6:     end if
7: end for
8: Nastav pero na červenou a štětec na zelenou pro vykreslení zvýrazněných polygonů
9: for každý zvýrazněný index ind do
10:    if index ind je platný then
11:        Vykresli zvýrazněný polygon
12:    end if
13: end for
14: Nastav pero na černou a štětec na modrou pro vykreslení bodu q
15: Vykresli bod q jako modrý kruh
16: Konec malování
```

Algorithm 11 Funkce switch_source

```
1: Změň režim přidávání bodu nebo přidávání vrcholů polygonu
```

Algorithm 12 Funkce highlightPolygon

```
1: Ulož seznam zvýrazněných polygonů
2: Překresli obrazovku
```

Algorithm 13 Funkce clearHighlighted

```
1: Vymaž seznam zvýrazněných polygonů
2: Překresli obrazovku
```

Algorithm 14 Funkce openFile

```
1: Otevři dialog pro výběr souboru
2: if soubor je vybrán then
3:     Otevři soubor
4:     if soubor je otevřen then
5:         Vyčisti seznam polygonů
6:         Vytvoř dočasný seznam pro polygon
7:         while nebyl dočten celý soubor do
8:             Přečti řádek a ořež bílé znaky
9:             if řádek je prázdný then
10:                if dočasný seznam polygonu není prázdný then
11:                    Přidej polygon do seznamu
12:                    Vyčisti dočasný seznam
13:                end if
14:            end if
15:            Rozděl řádek na souřadnice
16:            if má řádek 2 souřadnice then
17:                Získej hodnoty x a y
18:                if hodnoty jsou platné then
19:                    Přidej bod do dočasného polygonu
20:                end if
21:            end if
22:        end while
23:        if dočasný seznam není prázdný then
24:            Přidej poslední polygon do seznamu
25:        end if
26:        Zavři soubor
27:        Překresli obrazovku
28:    end if
29: end if
```

Algorithm 15 Funkce openSHP

```
1: Otevři dialog pro výběr souboru typu SHP
2: if soubor je vybrán then
3:     Registrovat všechny formáty GDAL
4:     Otevři SHP soubor
5:     if soubor je otevřen then
6:         Načti první vrstvu shapefile
7:         Vyčisti seznam polygonů
8:         while nejsou přečteny všechny prvky do
9:             Načti geometrický prvek
10:            if geometrie je polygon then
11:                Získej exteriérový prstenec polygonu
12:                for každý bod v prstenci do
13:                    Získej souřadnice x a y
14:                    Přidej bod do polygonu
15:                end for
16:                Přidej polygon do seznamu
17:            end if
18:        end while
19:        Zavři soubor
20:        Normalizuj polygony
21:        Překresli obrazovku
22:    end if
23: end if
```

mainforma.cpp

Algorithm 16 Konstruktor MainForm

1: Inicializuj `ui` pomocí `setupUi(this)`

Algorithm 17 Destruktor MainForm

1: Uvolni paměť pro `ui`

Algorithm 18 Funkce on_actionPoint_Polygon_triggered

1: Zavolej funkci `switch_source` na `Canvas`, aby změnila vstup na bod nebo polygon

Algorithm 19 Funkce on_actionRay_Crossing_triggered

Získej bod q a seznam polygonů z `Canvas`
Vymaž zvýrazněné polygony pomocí `clearHighlighted`
Vytvoř prázdný seznam `indices`
Inicializuj proměnné `res, isInside, isInVertex, isOnEdge` na 0 a `false`
for každý polygon *pol* v seznamu polygonů **do**
    if bod q je v minimální a maximální oblasti polygonu **then**
        Zavolej `analyzePointndPolPosition` a ulož výsledek do `res`
        if `res` je 1 **then**
            Přidej index polygonu do seznamu `indices`
            Nastav `isInside` na `true`
        else if `res` je -2 **then**
            Přidej index polygonu do seznamu `indices`
            Nastav `isInVertex` na `true`
        else if `res` je -1 **then**
            Nastav `isOnEdge` na `true`
            Přidej index polygonu do seznamu `indices`
        end if
    end if
end for
Zvýrazni polygony podle seznamu `indices`
if isInside **then**
    Nastav název okna na "Inside"
else if isInVertex **then**
    Nastav název okna na "Vertex"
else if isOnEdge **then**
    Nastav název okna na "Edge"
else
    Nastav název okna na "Outside"
end if

Algorithm 20 Funkce on_actionWinding_Number_triggered

Získej bod q a seznam polygonů z `Canvas`
Vymaž zvýrazněné polygony pomocí `clearHighlighted`
Vytvoř prázdný seznam `indices`
Inicializuj proměnné `res, isInside, isInVertex, isOnEdge` na 0 a `false`
for každý polygon *pol* v seznamu polygonů **do**
    if bod q je v minimální a maximální oblasti polygonu **then**
        Zavolej `WindingNumber` a ulož výsledek do `res`
        if `res` je 1 **then**
            Přidej index polygonu do seznamu `indices`
            Nastav `isInside` na `true`
        else if `res` je -2 **then**
            Přidej index polygonu do seznamu `indices`
            Nastav `isInVertex` na `true`
        else if `res` je -1 **then**
            Nastav `isOnEdge` na `true`
            Přidej index polygonu do seznamu `indices`
        end if
    end if
end for
Zvýrazni polygony podle seznamu `indices`
if isInside **then**
    Nastav název okna na "Inside"
else if isInVertex **then**
    Nastav název okna na "Vertex"
else if isOnEdge **then**
    Nastav název okna na "Edge"
else
    Nastav název okna na "Outside"
end if

Algorithm 21 Funkce on_actionOpen_triggered

Zavolej funkci `openFile` na `Canvas` pro otevření souboru s polygonem

Algorithm 22 Funkce on_actionOpen_SHP_triggered

Zavolej funkci `openSHP` na `Canvas` pro načtení shapefile souboru

Algorithm 23 Funkce on_actionExit_triggered

Ukonči aplikaci pomocí `QApplication::quit()`

Algorithm 24 Funkce on_actionClear_data_triggered

1: Vymaž všechny polygony na `Canvas` pomocí `clearPolygons`

2: Překresli obrazovku pomocí `repaint()`

Algorithm 25 Funkce on_actionClear_all_triggered

1: Vymaž všechny data na `Canvas` pomocí `clear()`

2: Překresli obrazovku pomocí `repaint()`

Odkazy

1. BAYER, Tomáš. *Advanced Data Structures and Algorithms for Computational Geometry* [Online]. 2025. Dostupné také z: `https://web.natur.cuni.cz/~bayertom/images/courses/Adk/adk3_new.pdf`. Accessed: 2025-03-16.