Ročníkový projekt – Report (letný semester)

Názov projektu: **Rozdeľovanie komplikovaných elementov na mapách**

Meno študenta:	Denys Martyniuk
Mail:	martyniuk3@uniba.sk
Meno školiteľa:	doc. RNDr. Robert Lukoťka, PhD.
Mail školiteľa:	lukotka@dcs.fmph.uniba.sk
Dátum:	2026-06-02

Cieľ letného semestra

Cieľom letného semestra bolo nadviazať na zimnú časť projektu, v ktorej bol vytvorený základný program na načítanie súboru .omap, identifikáciu plošných objektov a rozlíšenie jednoduchých a komplikovaných tvarov. V letnej časti bol cieľ rozšírený o samotné rozdeľovanie komplikovaných plošných elementov na jednoduchšie geometrické časti.

Za jednoduchú časť sa v aktuálnej verzii považuje trojuholník. Komplikovaný plošný objekt, ktorý je reprezentovaný ako polygon s viac ako tromi vrcholmi, sa program pokúsi rozložiť na množinu trojuholníkov. Takto pripravená reprezentácia môže neskôr slúžiť ako podklad pre algoritmy vyhľadávania najkratších ciest na mapových podkladoch.

Čo bolo implementované

Program bol refaktorovaný z pôvodnej monolitickej verzie do viacerých samostatných tried. Trieda OmapAreaReader zodpovedá za načítanie geometrie zo súboru .omap, GeometryUtils obsahuje pomocné geometrické metódy, EarClippingTriangulator rieši samotné rozdeľovanie polygonov, MapAreaAnalyzer počíta výsledné štatistiky a MapComplexSplitterSummerApp slúži ako konzolový vstupný bod programu.

Načítanie plošných objektov zostalo založené na XML štruktúre formátu .omap. Program prechádza elementy <symbol>, vyberá symboly obsahujúce <area_symbol/> a následne spracúva iba tie elementy <object>, ktorých atribút symbol zodpovedá plošnému symbolu.

Geometria objektu sa číta z elementu <coords>. Oproti zimnej verzii sa už neparsujú iba súradnice x a y, ale aj voliteľný tretí údaj flags. Tým vznikla pomocná reprezentácia RawCoord, ktorá uchováva x, y a príznaky bodu.

Bola doplnená metóda readRawGeometry, ktorá číta surové súradnice vrátane príznakov. Následne metóda buildPolylineGeometry konvertuje tieto súradnice na polygonálnu reprezentáciu, ktorú môže spracovať triangulačný algoritmus.

Bola implementovaná normalizácia polygonu. Program odstraňuje duplicitný záverečný bod, ak sa zhoduje s prvým bodom, ignoruje nulové body a odstraňuje po sebe idúce duplicitné vrcholy. Pri samotnej triangulácii sa odstraňujú aj kolineárne vrcholy, ktoré by mohli zhoršiť stabilitu výpočtu.

Bolo implementované rozdeľovanie polygonov pomocou algoritmu ear clipping. Algoritmus postupne hľadá konvexný vrchol polygonu, overí, že príslušný trojuholník neobsahuje iný vrchol polygonu, uloží tento trojuholník ako jednoduchú časť a odstráni jeho stredný vrchol z pracovného polygonu.

Program podporuje polygony zadané v smere hodinových ručičiek aj proti smeru hodinových ručičiek. Orientácia polygonu sa určuje pomocou podpísanej plochy a podľa nej sa vyhodnocuje konvexnosť vrcholov.

Bolo doplnené ošetrenie okrajových prípadov: neplatné polygony s menej ako tromi bodmi, degenerované polygony s nulovou plochou, duplicitné body, kolineárne vrcholy a situácia, keď sa nepodarí nájsť ďalšie ucho polygonu.

Pre prípravu budúceho spracovania kriviek boli implementované pomocné metódy cubicBezierPoint a approximateCurveSegment. Prvá metóda vypočíta bod kubickej Bezierovej krivky pre parameter t, druhá aproximuje jednu kubickú Bezierovu krivku lomenou čiarou s daným počtom krokov.

Bola doplnená základná detekcia príznakov pre krivky a vnútorné diery. Objekty s vnútornými dierami sa v aktuálnej verzii odmietajú ako nepodporované, pretože ich korektná triangulácia vyžaduje samostatné spracovanie vnútorných kontúr.

Použitý algoritmus rozdeľovania

Na rozdelenie komplikovaných polygonálnych plošných objektov bol použitý algoritmus ear clipping. Tento algoritmus vychádza z vlastnosti jednoduchého

polygonu, že ho možno rozdeliť na trojuholníky postupným odrezávaním takzvaných uší.

Pre každý vrchol polygonu sa berie trojica bodov: predchádzajúci vrchol, aktuálny vrchol a nasledujúci vrchol. Táto trojica tvorí kandidátny trojuholník. Kandidát je prijatý iba vtedy, ak je aktuálny vrchol konvexný vzhľadom na orientáciu polygonu a ak sa vnútri alebo na hranici kandidátneho trojuholníka nenachádza žiadny iný vrchol polygonu. Po prijatí sa trojuholník uloží medzi výsledné jednoduché časti a aktuálny vrchol sa odstráni z pracovného polygonu.

Postup sa opakuje, kým v pracovnom polygone nezostanú posledné tri vrcholy. Tie vytvoria posledný výsledný trojuholník. Pre polygon s n vrcholmi teda štandardne vznikne (n – 2) trojuholníkov, ak ide o platný jednoduchý polygon bez dier.

Štruktúra programu

Trieda	Úloha
MapComplexSplitterSummerApp	Konzolová aplikácia. Načíta cestu k .omap súboru, spustí analýzu a vypíše výsledné štatistiky.
MapAreaAnalyzer	Spája načítanie geometrie, klasifikáciu objektov a počítanie výsledných štatistík.
OmapAreaReader	Spracúva XML dokument .omap, vyberá plošné objekty a načítava ich súradnice vrátane flags.
GeometryUtils	Obsahuje pomocné geometrické výpočty: plocha polygonu, orientácia, konvexnosť, bod v trojuholníku a pomocné metódy pre Bezierove krivky.
EarClippingTriangulator	Implementuje klasifikáciu jednoduchého tvaru a rozdelenie polygonu na trojuholníky algoritmom ear clipping.

Výstup programu

Program po spracovaní mapy vypíše názov vstupného súboru, počet analyzovaných plošných objektov, počet jednoduchých objektov, počet komplikovaných objektov a počet jednoduchých častí vytvorených rozdelením.

Testovanie

Pre projekt boli vytvorené JUnit 5 testy, ktoré overujú samostatné časti programu aj celý tok aplikácie.

Unit testy pre trianguláciu overujú rozdelenie trojuholníka, štvoruholníka a konkávneho polygonu, správanie pri neplatných a degenerovaných vstupoch, podporu opačnej orientácie vrcholov, spracovanie duplicitných a kolineárnych bodov a zachovanie plochy po triangulácii.

Testy pre prípravu kriviek overujú, že cubicBezierPoint pre t = 0 vracia začiatočný bod krivky a pre t = 1 koncový bod krivky. approximateCurveSegment musí vytvoriť viac ako dva body pri aproximácii krivky.

Testy pre raw geometriu overujú, že readRawGeometry načíta súradnice vrátane flags a že buildPolylineGeometry podporuje obyčajný polygon bez dier, ale odmietne geometriu označenú príznakom vnútornej diery.

Integračný test vytvorí minimálny .omap súbor s jedným jednoduchým trojuholníkom a jedným komplikovaným štvoruholníkom, spustí hlavnú aplikáciu a overí očakávané výstupné štatistiky.

Použité technológie

Java

DOM XML parser na spracovanie .omap súboru

JUnit 5 na unit a integračné testy

Formát máp .omap používaný v OpenOrienteering Mapper

Obmedzenia aktuálnej verzie

Aktuálna verzia je stabilná prvá verzia pre polygonálne plošné objekty. Nepodporuje plnú trianguláciu objektov s vnútornými dierami. Takéto objekty program rozpozná podľa príznakov a odmietne ich, aby nevytvoril nesprávnu trianguláciu.

Pre krivky sú implementované pomocné metódy na výpočet bodu kubickej Bezierovej krivky a jej aproximáciu lomenou čiarou. Plné napojenie na všetky špecifiká kriviek vo formáte .omap je však ponechané ako ďalšie rozšírenie. V aktuálnej verzii sa príznaky kriviek rozpoznajú, ale body sa ďalej spracujú ako polygonálna reprezentácia.

Program tiež neimplementuje špeciálne spracovanie samopretínajúcich sa polygonov. Ak sa pri triangulácii nepodarí nájsť ďalšie platné ucho polygonu a problém nemožno vyriešiť odstránením kolineárneho bodu, vstup sa považuje za netriangulovateľný.

Záver

V letnom semestri bola implementovaná hlavná časť plánovaného rozšírenia projektu: komplikované polygonálne plošné objekty zo súboru .omap sa už nielen identifikujú, ale aj rozdeľujú na jednoduché časti reprezentované trojuholníkmi. Program bol zároveň refaktorovaný do samostatných tried, doplnený o spracovanie raw geometrie a pripravený na budúce rozšírenie o presnejšie spracovanie kriviek a objektov s dierami.

Výsledkom je použiteľný základ pre ďalšiu fázu projektu, v ktorej môžu byť vytvorené jednoduché geometrické časti využité ako podklad pre algoritmy vyhľadávania najkratších ciest v mapovom prostredí.