Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra matematiky

Bakalářská práce

GeoTools (the open source Java GIS toolkit) implementace Delaunayho triangulace

Plzeň, 2007 Josef Bezděk

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni 27.8 .2007, Josef Bezděk .....

Poděkování

Rád bych zde poděkoval vedoucímu mé bakalářské práce, panu Ing. Janu Ježkovi, především za podnítění zájmu o danou problematiku a za získané znalosti při tvorbě této práce.

Abstrakt

V první části bakalářské práci je popsán otevřený (*open source*) projekt Geotools. Jsou zde licenční podmínky a aplikace vystavěné na bázi knihovny Geotools. Dále je nastíněn postup při instalaci a několik ukázek při práci s touto knihovnou. V praktické části je rozebírán algoritmus na tvorbu Delaunayovy triangulace, jako je datová struktura a algoritmizace jednotlivých případů, které mohou nastat. Poslední část je věnována popisu implementace třídy pro generování pevných hran v triangulaci. Na závěr této práce se hodnotí dosažené výsledky a budoucí využití implementovaných tříd.

Klíčová slova

Geotools, Delaunay, Delaunayovy triangulace, TIN, pevné hrany, DiMoT.

Abstract

At the first part of this work open source project Geotools is described. You can find here licence conditions and applications based on library Geotools. Below the instructions for installation and several demonstrations with the library are shown. At the practical part of the work the algorithm of Delaunay triangulation is analysed with its data structure and algorithmization of various cases, which could happen. The last part is about implementation of class for generating fixed lines in triangulation. At the end of this work attained results and future usage of implemented classes are rated.

Keywords

Geotools, Delaunay, Delaunay triangulation, TIN, fixed lines, DiMoT.

Obsah

1 Úvod

Význam GIS aplikací v dnešní době stále roste, protože většina dat, které chceme uchovávat, obsahují složky geografického a prostorového charakteru. Uživatel má možnost si vybrat mezi GIS produkty, které jsou komerční nebo volně šiřitelné (*open source*). Existuje celá řada open source projektů, jenž dokáží splnit požadavky uživatelů, ať už se jedná o technickou podporu či vlastní funkcionalitu.

Open source GIS se týká i této práce, ve které je představena knihovna Geotools. Cílem bylo seznámit se blíže s touto knihovnou, popsat její strukturu a ukázat možné využití metod z této knihovny v praktické části bakalářské práce, která se zaměřuje na tvorbu TIN pomocí Delaunayho triangulace.

Přínos práce spočívá v implementaci modulu na tvorbu Delaunayovy triangulace pomocí otevřených nástrojů a v návrhu možných optimalizací, které urychlují výpočet triangulace. Dále byl implementován modul na výpočet TIN s pevnými hranami, který Geotools neobsahují. Bylo vytvořeno i grafické uživatelské prostředí pro lepší vizualizaci zpracovávaných dat.

2 Úvod do Geotools

Na zpracování geoprostorových dat existují programy, kde velkou skupinu tvoří software „open source“, to znamená technickou dostupnost kódu a legální dostupnost licence, která umožňuje uživatelům zdrojový kód využívat při dodržení jistých podmínek, například prohlížet a upravovat. Mezi tento software patří např. programy Jump a OpenJump, což jsou hotové aplikace přímo určené pro koncového uživatele. Geotools je také *open source*, ale nejedná se o hotovou aplikaci, nýbrž o Java knihovnu, která poskytuje sadu nástrojů pro práci s geoprostorovými daty a pro tvorbu nové GIS aplikace. Tyto nástroje jsou implementované třídy a metody psané výhradně v jazyce Java. Instalační balík těchto nástrojů je dostupný na domovských stránkách:

http://geotools.codehaus.org

Rozsah použití je velmi široký od GIS aplikací včetně serverové varianty, které obsahují služby WFS *(Web Feature Services)*, WMS *(Web Map Services)*, až po desktop aplikace.

2.1 Licence Geotools

Knihovna Geotools je vytvářena pod licencí GNU – LGPL, kde GNU je projekt spravovaný nadací Free Software Foundation, která se zaměřuje na svobodný software. Označení GNU je používáno pro souhrn programů, jenž je často označován přímo jako „operační systém GNU“. Organizace GNU je autorem licencí GPL *(General Public License)* a GFDL *(GNU Free Documentation License)*, které mohou používat i jiní autoři softwaru a dokumentace. Licence GPL je určena pro svobodný software a udává pravidla, jak daný software používat. Tímto se podobá licensi GFDL, která je ale určena pro dokumentace, [web12].

Další z licencí GNU je LGPL *(Lesser General Public License)* jenž je také varianta licence pro svobodný software GNU - GPL, která ale na rozdíl od GPL umožňuje spojování s nesvobodným kódem a tak je vhodnější pro knihovny. Pod touto licencí jsou vytvářeny Geotools, [web12].

2.2 Standardy Geotools

Geotools je mladý dynamicky se rozvíjející projekt s velkou základnou programátorů. K tomu, aby byla zaručena kompatibilita s jiným GIS software, Geotools implementují celosvětově uznávané standardy typu ISO norem a specifikací OGC.
OGC *(Open Geospatial Consortium)* – je mezinárodní konsorcium 339 společností, státních agentur a universit, které se podílejí na vývoji veřejně použitelných specifikací, jenž nám dávají rozhraní pro geoprostorové informace. Rozhraní je jakási základní architektura, která dává typy vstupních a výstupních proměnných a metod. To umožňuje přenositelnost dat mezi různými platformami a vzájemnou spolupráci aplikací, [web15]. Mezi významné specifikace od OGC patří:

OGC Reference Model - kompletní sada referenčních modelů

WMS - Web Map Service - komunikační protokol pro přenos dat, kde je mapa v rastrovém formátu typu *.jpg, *.png.

WFS - Web Feature Service - komunikační protokol pro přenos dat, kde je mapa ve vektorovém formátu, např. *.shp.

2.3 Dva přístupy ke Geotools

Ke Geotools můžeme přistupovat dvěma způsoby:

- jako uživatel
- jako vývojář

Na domovských stránkách je k dispozici návod (*turorial*) pro obě tyto skupiny.

2.3.1 Uživatel

Uživatel si stáhne příslušné vydání a po nainstalování může Geotools využívat pro tvorbu vlastních aplikací. Jediné omezení je licence LGPL. Uživatel se nepodílí na rozšiřování knihovny.

2.3.2 Vývojář

Vývojář se podílí na rozšíření Geotools o další implementace. Pro vlastní práci je důležité mít možnost nahlédnout do zdrojového kódu. Ten můžeme získat třemi způsoby:

- stáhnout vydání se zdrojovým kódem (*nové vydání cca. jednou za měsíc*)
- stáhnout zdrojový kód z nového vydání (nightly build)
- použít Subversion

2.4 Aplikace vystavěné s podporou Geotools

2.4.1 GeoServer

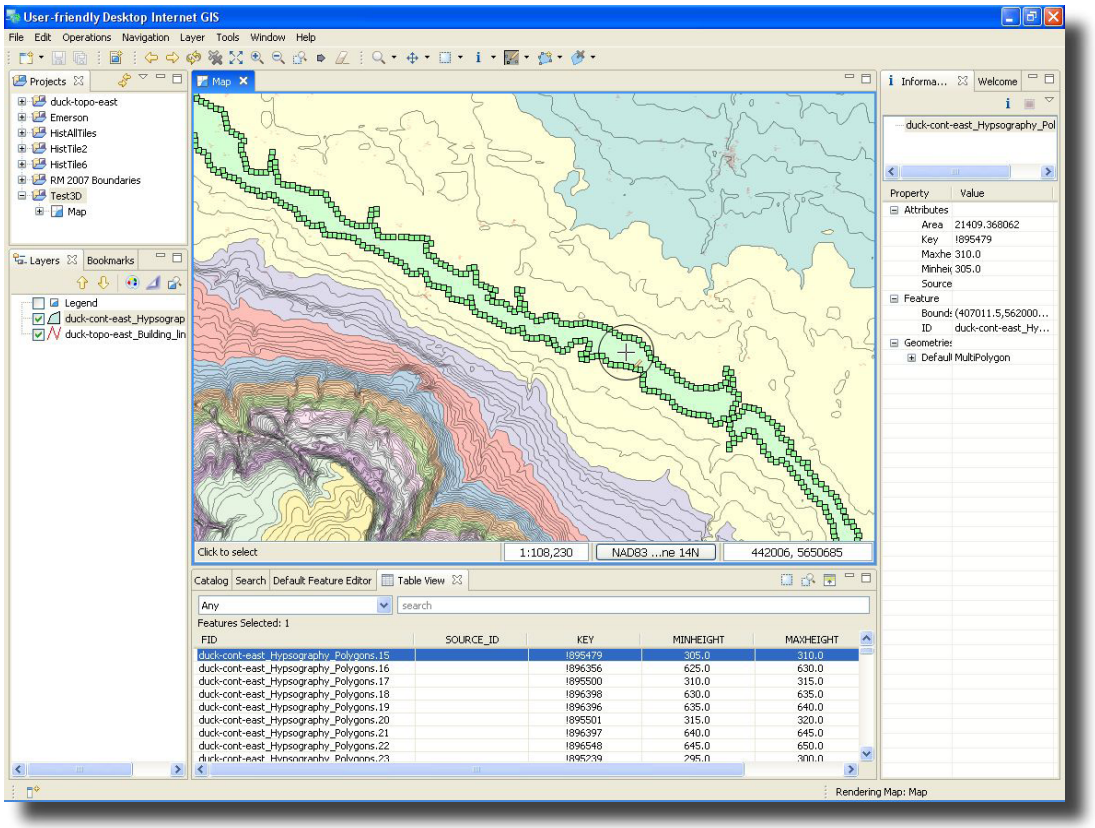
GeoServer je open source aplikace, která má implementované specifikace WFS, WMS, WCS. Je vydávaná organizací The Open Planning Project - TOPP. Umožňuje publikovat geoprostorová data a práci nad těmito daty, jako je mazání, obnovování a vkládání nových objektů. Na základě specifikací OGC umí vytvořit formáty typu *jpg*, *png*, *shapefile* a jiné, [web6].

2.4.2 GeoVista Studio

GeoVista Studio je otevřené vývojové prostředí uzpůsobené pro geoprostorová data. Uživatelům umožňuje rychle vytvořit aplikaci pro práci či vizualizaci geodat, [web8].

2.4.3 uDig

Je to opět otevřená aplikace, která umožňuje pracovat s geoprostorovými daty. uDig je vyvíjen s důrazem na podporu veřejných standardů OGC a specializuje se na podporu služeb WMS a WFS, [web17]. Na základě implementovaných tříd by se v budoucnu měl vytvořit plugin pro tento software, který bude řešit rekonstrukci digitálního modelu terénu (DMT).



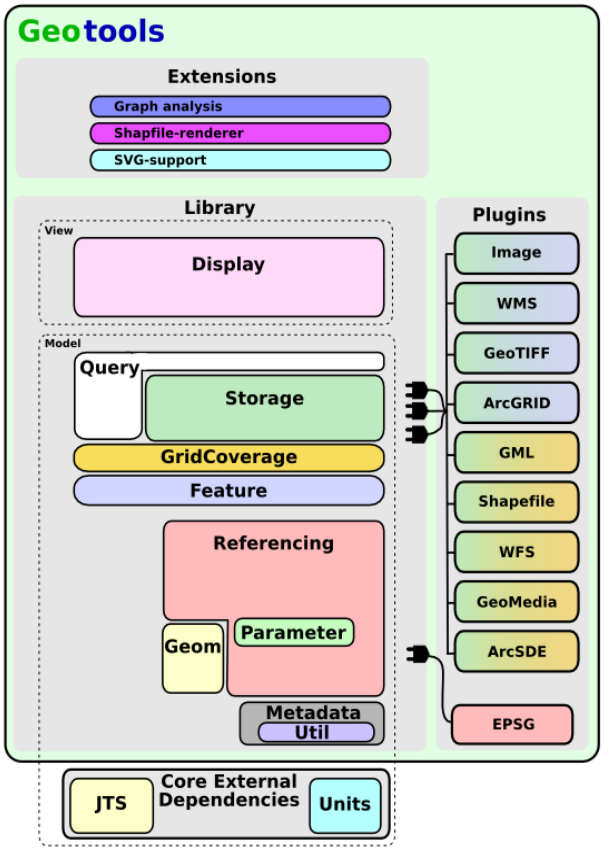
obr. 2.1: program uDig, [web7]

3 Struktura knihovny Geotools

Knihovna Geotools je tedy soubor tříd a metod, které jsou implementovány s podporou jiných knihoven. Dle [Ježe07]: „GeoTools jsou složeny z několika hlavních modulů a dále také obsahují technologii pluginů (zásuvných modulů), které můžeme ke konkrétním modulům připojit. “
Struktura Geotools je následující, [Ježe07]:

- převzaté knihovny
  - GeoApi
  - JTS
- vlastní moduly
  - API
  - cql
  - jdbc
  - metadata
  - render
  - coverage
  - data
  - main
  - referencing
  - xml

Situaci také znázorňuje následují obrázek:



obr. 3.1: struktura Geotools, [web17]

3.1 Převzaté knihovny

Knihovna Geotools vychází z již hotových knihoven, a to z knihoven GeoApi a JTS.

3.1.1 GeoApi

GeoAPI je soubor rozhraní implementovaných v jazyce JAVA, které vycházejí z OGC specifikací. GeoAPI definuje návrh objektů a jejich metod pro základní operace z geografickými daty. Tyto objekty tvoří rozhraní s definovanými metodami a proměnnými, které při implementaci tohoto rozhraní musí vývojář naprogramovat. Cílem GeoAPI je vytvoření standardního systému tak, aby bylo možné propojovat jakékoliv nově vytvořené knihovny s těmi stávajícími, [web14].

3.1.2 Knihovna JTS

Tato knihovna poskytuje jeden ze dvou geometrických modelů, které jsou v Geotools použity. Knihovna JTS (*Java Topology Suite*) je používána přímo pro vlastní implementace objektů *Feature Geometries*. *Feature* je základní jednotka geoprostorové informace, která může být reálná fyzická entita, jako je řeka, budova. *Feature Geometries* jsou tedy geometrické objekty pro reprezentaci zemského povrchu. Protože JTS pracuje striktně v ortogonálním 2D prostoru, Geotools využívají ještě jeden rozšířený souřadnicový systém *Geometry* pro reprezentaci pozice na zemském povrchu dle specifikací projektu GeoAPI. *Geometry* je vlastní implementovaný modul Geotools.
Knihovna JTS byla vyvinuta společností Vivid Solutions a poskytuje silný nástroj pro práci s jednoduchými geometrickými primitivy, jako jsou bod, linie, polygon. Jakmile jsou na vstupu data, které jsou ve 3D, knihovna osu Z zanedbává a veškeré výpočty se provádí pouze v osách x a y. Do budoucna se plánuje, že Geotools přejdou na komplexnější systém, kde se bude počítat i s osou Z podle specifikace OGC, [web9].

3.2 Vlastní moduly

Popis jednotlivých modulů byl převzat z [Ježe07], kde jsou popisovány takto:

3.2.1 API

Modul obsahuje klíčová rozhraní, která jsou používaná dalšími moduly. Pokud je to možné, jsou využívána standardní rozhraní z knihovny GeoAPI.

3.2.2 Coverage

Implementace specifikace Grid Coverage. Modul umožňuje podporu operací s rastry, jako např. aplikace matematických transformací apod.

3.2.3 CQL
Modul, který slouží jako parser pro specifikaci OGC - Common Query Language.

3.2.4 Data
Implementace abstraktních objektů pro práci s daty (Abstract Datastore Implementation).

3.2.5 JDBC
Rozšíření abstraktní implementace pro práci s daty o data uložená v relačních databázích přístupných přes JDBC.

3.2.6 Main
Modul obsahuje klíčová rozhraní, která jsou používaná dalšími moduly.

3.2.7 Metadata
Obsahuje implementace operací s metadaty.

3.2.8 Referencing
Obsahuje implementaci OGC specifikace Coordinate Reference System Implementation. Zde jsou implementovány operace pro konverze a transformace souřadnic.

3.2.9 Render
Modul obsahuje funkcionalitu pro vizualizaci dat postavených na rozhraních z modulů main a api.

3.2.10 XML
V tomto modulu jsou umístěny nástroje pro práci s nejrůznějšími XML schématy potřebných při práci v GIS (GML, SLD atd.).

4 Práce s Geotools

Tato kapitola je stručným návodem, jak s Geotools začít pracovat. Popsána je instalace a následně ukázka několika demonstračních programů. Tyto demonstrační programy využívají moduly, které se používali při implementaci třídy Delaunayho triangulace.

4.1 Instalace Geotools

Instalace Geotools není klasickou instalací, kde spuštěním programu setup či install nainstalujeme aplikaci. Abychom mohli Geotools nainstalovat a používat, potřebujeme několik dalších podpůrných programů. V následujících odstavcích je popsán postup při instalaci a jednotlivé kroky.

1. Java JDK (Java Development Kit)

Abychom mohli knihovnu Geotools využívat, je potřeba nainstalovat JRE (*Java Runtime Enviroment*) pro spuštění aplikací, přičemž toto prostředí obsahuje JVM (*Java Virtual Machine*) a základní sadu knihoven Java Core API. JRE je součástí balíku J2SE - SDK(Standart Develompent Kit), což je soubor základních nástrojů pro vývoj aplikací v jazyce Java od společnosti SUN Microsystems, a který si lze stáhnout na http://java.sun.com/. Od verze Geotools 2.5 a vyšší je požadována Java 1.5 SDK.

2. Java rozšiřující knihovny

Dále musíme nainstalovat dvě rozšiřující knihovny pro práci s obrázky. Tyto knihovny jsou opět od společnosti Sun Microsystems a nejsou standardní součástí balíku SDK. Proto je nutné tyto knihovny stáhnout z domovských stránek http://java.sun.com/ a nainstalovat.
Jedná se o knihovny:

- The Java Advanced Imaging library
- Java Image I/O library

3. Konfigurace Javy

Další krokem v instalaci je nastavení uživatelské proměnné JAVA_HOME na adresář, kde se nachází váš SDK a systémovou proměnnou PATH na „ %JAVA_HOME%\bin; “ Na platformě Windows změna proměnných se provede takto:

Start – Ovládací panely – Systém – Upřesnit – Uživatelské proměnné.

V tomto okně se provedou výše popsané změny.

4. Geotools - stažení distribucí

Geotools jsou distribuovány ve třech oddělených částech:

- jako binární distribuce
- zdrojová distribuce
- javadoc, dokumentace

Každou z těchto částí lze získat na stránce Geotools (http://geotools.codehaus.org/), kde se dají stáhnout příslušné *zip* archívy, které rozbalí v domovském adresáři. Binární distribuce obsahuje archivy typu *jar*, které se následně využijí při tvorbě aplikací. Pokud je požadováno editování a kompilování tříd ze zdrojové distribuce, je potřeba získat další knihovny ze kterých Geotools vycházejí. K tomu je v Geotools uzpůsoben program Maven.

5. Maven

Nyní můžeme sestavit projekt Geotools. K tomu slouží program Maven (http://maven.apache.org/download.html). Jedná se o software, který je uzpůsoben pro řízení a automatické vystavění projektů s programovacím jazykem Java. Maven může dynamicky stahovat moduly z mateřského repositáře (*repository*). V repozitáři jsou uložena zdrojová data a mnoho dalších nových verzí, takzvané buildy Geotools. Maven stáhne příslušné balíky, které rozbalí do lokálního repozitáře.

6 Test instalace

Na závěr je dobré otestovat instalaci, zda vše funguje správně. K tomuto účelu je v Geotools implementována demonstrační aplikace, která se spouští následovně:

java -jar /path/to/gt2-demo-introduction-VERSION_NUMBER.jar

Je potřeba zadat úplnou cestu k jar archívům, která se doplní místo (*path*) a dále doplnit číslo verze vydání. Po zadání příkazu v příkazové řádce se objeví následují okno:

obr. 4.1: test instalace, [web7]

4.2 Knihovna JTS - použití

Tato externí knihovna byla již představena v kapitole 3.1.2 o převzatých knihovnách. Použití této knihovny je velmi jednoduché. Následující postup ukazuje, jak vytvořit vlastní objekt typu *Feature Geometry – LinearRing* uzavřený polygon o třech stranách:

- importujeme příslušné balíky.

```
import com.vividsolutions.jts.geom.Coordinate;
import com.vividsolutions.jts.geom.GeometryFactory;
import com.vividsolutions.jts.geom.LinearRing;
```

- vytvoříme pole souřadnic bodů

```
Coordinate A = new Coordinate(10,10);
Coordinate B = new Coordinate(20,20);
Coordinate C = new Coordinate(30,10);
Coordinate[] pole = {A,B,C};
```

- instanci pole předáme do konstruktoru pro objekt CoordinateArraySequence

```
CoordinateArraySequence newPointsTriangle =
                                    new CoordinateArraySequence(pole);
```

Krok 2 a 3 lze provést najednou. *CoordinateArraySequence* je rozšířenou třídou *Array*.

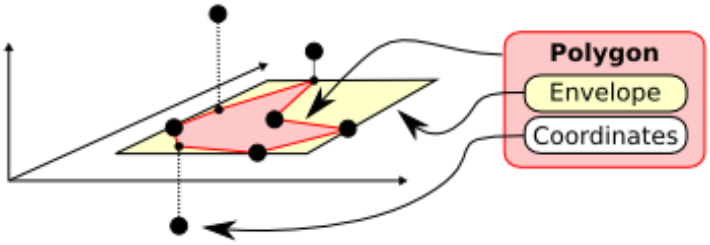
- vytvoříme objekt trojúhelník

```
LinearRing trojúhelník = new LinearRing(newPointsTriangle,new
                                                GeometryFactory());
```

S takto vytvořeným objektem *trojúhelník* můžeme dále pracovat. Knihovna JTS poskytuje řadu metod:

- convexHull() – vyrobí konvexní obal
- coveredBy() – vrátí true, jestli jiná geometrie ji zakrývá
- dále viz. Javadoc : http://www.vividsolutions.com/jts/javadoc/

Následující obrázek ukazuje, jak knihovna JTS pracuje s geometrickými objekty.



obr. 4.2: knihovna JTS a terminologie, [web7]

- Polygon - uzavřený polygon
- Envelope - nejmenší možná obálka polygonu
- Coordinates - souřadnice bodu.

Opět je zde vidět, že JTS je pouze 2D.

4.3 Modul Referencing - použití

Jak již bylo řečeno, tento balík se zabývá souřadnicovými systémy. Implementuje metody, jako jsou konverze a transformace. Souřadnice mohou být i vyšších dimenzí. Každý souřadnicový systém (*CRS - Coordinate Reference System*) má svůj identifikátor, tzv. EPSG kód.

EPSG (European Petroleum Survey Group) byla vědecká společnost s vazbou na Evropský petrolejářský průmysl, která sdružovala specialisty v geodézii, geografy, kartografy. EPSG shromáždila spoustu geoinformací o zemských elipsoidech, zobrazovacích systémech, používaných jednotkách. Role EPSG byla ukončena v roce 2005 a nahradila ji komise OGP. Původní název EPSG se však nadále používá. OGP vydává databázi s architekturou Microsoft Access, [web15].

Transformace souřadnic

Jak tedy pracovat se souřadnicovými systémy:

- importujeme balík CRS

  import org.geotools.referencing.CRS; // import balíku

- nadefinujeme souřadnicové systémy

  CoordinateReferenceSystem sourceCRS = CRS.decode(„EPSG:4326“);
  CoordinateReferenceSystem targetCRS = CRS.decode(„EPSG:23032“);

- získání transformačního klíče

  MathTransform transform = CRS.findMathTransform(sourceCRS, targetCRS);

Tímto způsobem se získá transformační klíč mezi souřadnicovým systém *sourceCRS* a *targetCRS,* který se využije pro převod objektu *Geometry* do cílového souřadnicového systému. Jak tedy postupovat:

- opět se musí importovat potřebné knihovny

  import org.geotools.geometry.jts.JTS;

- tímto příkazem se převede sourceGeometry do targetGeometry v požadovaném souřadnicovém systému.

  Geometry targetGeometry = JTS.transform(sourceGeometry, transform);

4.4 Modul Datastore - použití

Použití modulu Datastore je velice široké. Tato kapitola se zaměřuje jen na malou část, a to na vytvoření souboru typu shapefile. Především je zde obecně představen shapefile a vlastní vytvoření je pak popsáno v demonstračním programu, který je uložen na přiloženém CD v adresáři */demo/shapefile.java*.

4.4.1 Shapefile

Shapefile je datový formát pro prostorová data. Byl vyvinut firmou ESRI (*Enviromental Systems Research, Inc.*), která zveřejnila strukturu jeho kódu a shapefile se stal standardem pro uchovávání prostorových dat. V souborech shapefile jsou uloženy geometrické objekty a atributové informace. Geometrické objekty nám popisují zobrazovanou skutečnost. Objekty tvoří sada vektorových souřadnic vrcholů geometrie.
Shapefile neobsahuje topologické informace ani nijak nezpracovává topologickou datovou strukturu. Má ale jiné výhody - rychlejší vykreslování a možnost editace.
Shapefile podporuje tyto objekty: body, linie a složené objekty, jako jsou polygonové pořady.
Atributové informace jsou ukládány v datovém souboru typu dBASE. Každý atribut má vztah 1:1 s přidruženým geometrickým objektem.

ESRI shapefile se skládá ze tří souborů:
■ ***main file*** (přípona .shp): tento soubor je přímo přístupný pro zobrazení v GIS softwarech. Data jsou zde organizována takto: první je hlavička souboru (*file header*) , která obsahuje proměnnou *fixed-length file*, ta určuje počet ukládaných objektů v shapefilu. Pak následují jednotlivé záznamy. Záznam se skládá z *Recorcd Header* a *Record Contents.*

- *Record Header* obsahuje proměnnou *record-length*, ta udává počet vrcholů, které tvoří zobrazovaný geometrický útvar.
- *Record Contents* obsahuje data k vlastnímu záznamu, jako jsou souřadnice vrcholů, název objektu (bod, linie, polygon), velikost obálky

■ **index file** (přípona .shx): v indexovém souboru každý záznam obsahuje odkaz do příslušného main file na začátek svých dat. Tento soubor propojuje tabulku a main soubor

■ **dBASE table** (.dbf): obsahuje atributová data geometrických objektů.. Vztahy mezi atributy a geometrickými objekty z main souboru je 1:1, to znamená, že každý geometrický objekt má jeden konkrétní atribut a obráceně. Atributy a geometrické obrazce musí být ukládány ve stejném pořadí, protože jsou vzájemně adresovány podle stejného čísla záznamu.

Je požadováno, aby všechny tvary geometrických obrazců byly stejného typu. Podporovány jsou tyto typy:

- NullShape
- Point
- Polyline
- Polygon
- Multipoint
- PointZ
- PolyLineZ
- PolygonZ
- MultipointZ
- PointM
- PolyLineM
- PolygonM
- MultiPointM
- MultiPatch

Každý typ má svoji konkrétní hodnotu, která může nabývat hodnot 0-33. Některé hodnoty zůstaly volné pro budoucí použití. [web5]

4.4.2 Další důležité objekty pro tvorbu shapefilu

■ **Features** (zobrazované geometrické objekty)
reprezentují geografická data, která se dají analyzovat a řídit v GIS softwarech. Mohou to být budovy, silnice, vodní plochy, administrativní celky a jiné.
Objekt je dán svým polohový určením (*geometry*) a atributovou složkou. Touto svojí strukturou jsou objekty typu *Features* v podstatě jednotlivé záznamy Shapefile. A jsou předem určeny k použití v GIS softwarech.

■ **Pravidla datové struktury:**
Každý objekt *Feature* je složen z pole objektů, které reprezentují jeho atributy. Je třeba se řídit následujícími pravidly:

1) musí obsahovat nejméně jedno polohové určení (*geometry*), které však může být *null*
2) může obsahovat více polohových určení(*geometry*), ale jen jedna je standardní a to ta, která je použita.
3) atributy mohou být cokoliv

Každý objekt typu *Feature* je tvořen třídou, která je zděděna od třídy *FeatureType*. *FeatureType* je interface, který definuje metody a proměnné, které drží sadu atributových složek z třídy *AttributeTypes*.

■ **GeometryAttributeType**
Dále je důležité popsat jeden z atributů. Je to *GeometryAttributeType,* který popisuje geometrické vlastnosti. Atribut obsahuje odkazy na souřadnicový systém crs *CoordinateRefenreceSystem* a *GeometryFactory*. Přidává se k objektu typu *Geometry*

5 Praktická část - Algoritmus pro tvorbu TIN

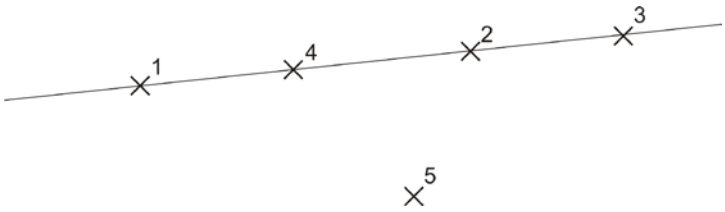
Jak již bylo řečeno, cílem této bakalářské práce je implementace Delaunayovy triangulace inkrementální metodou. To spočívá v postupném vkládání bodů do triangulace a po každém vložení bodu se vypočte TIN. Tato metoda se rozpadá na tři části:

- vytvoření prvního trojúhelníku
- nový bod leží uvnitř konvexního obalu
- nový bod leží mimo konvexní obal

Čerpáno bylo především z [Koho05],[Stro02].

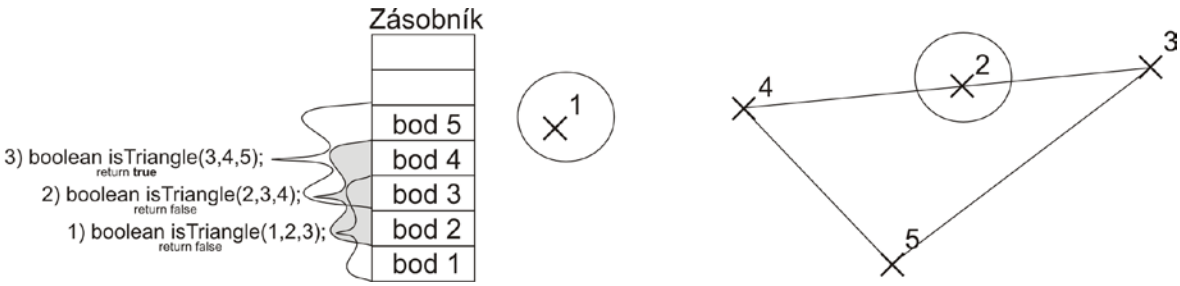
5.1 Vytvoření prvního trojúhelníku:

Tato metoda v sobě ukrývá jedno nebezpečí. Je to vytvoření prvního trojúhelníku. První trojúhelník nemusí vždy vzniknout po zadání tří bodů. Tento případ nastane pokud první tři nebo více bodů leží v přímce.



obr. 5.1: vstup bodů ležících v přímce

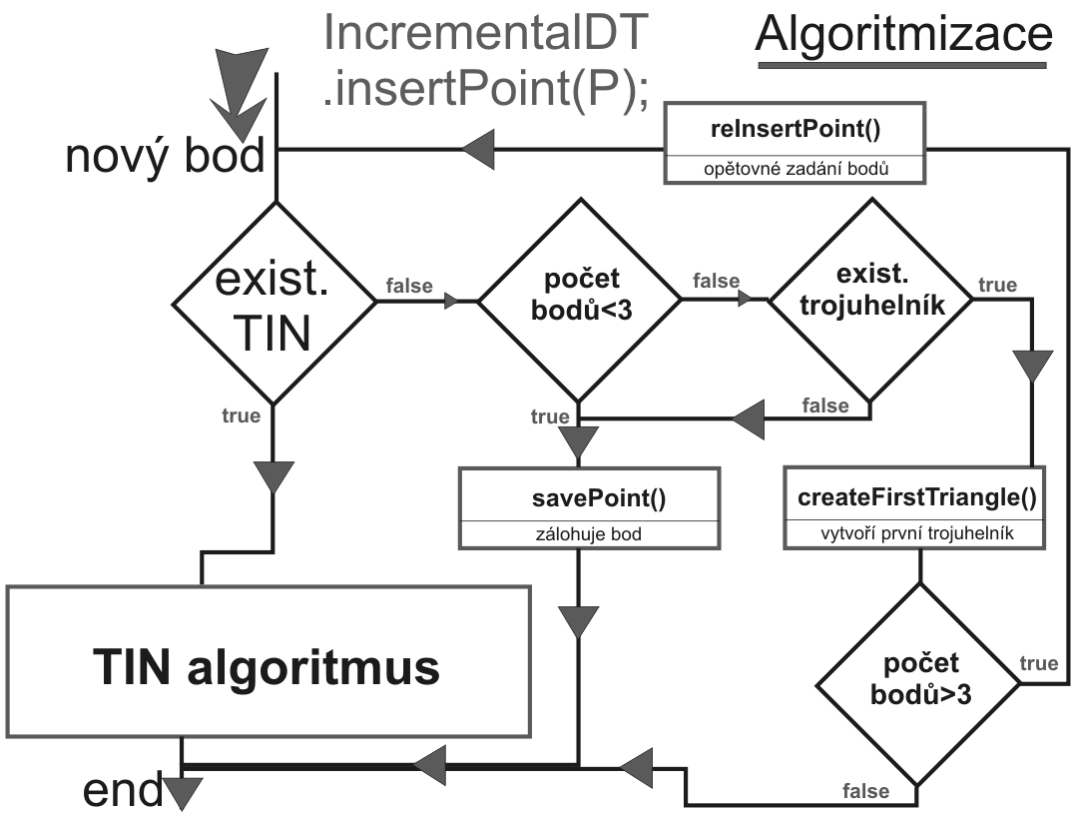
Na obrázku 5.1 je znázorněn tento případ. Je proto nutné vytvořit zásobník, kde se budou ukládat body, než bude možné vytvořit první trojúhelník.



obr. 5.2: ukládání bodů do zásobníku

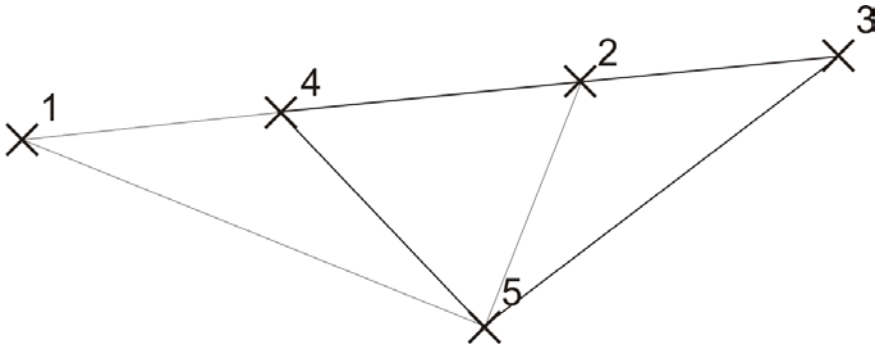
Do zásobníku se tedy ukládají data vstupních bodů. Jakmile jsou zadány tři body, začne testování, zda je možné vytvořit trojúhelník. Vždy se testují poslední tři vložené body. Z obrázku 5.2 je vidět, že první trojúhelník může vzniknout až po zadání bodu číslo 5 a to z bodů 4,3,5. Tento nově vzniklý trojúhelník se uloží do datové struktury. V zásobníku zůstávají body 2 a 1. Tyto body je nutno ještě jednou zadat do triangulace.

Tento algoritmus nastiňuje vývojový diagram viz obr. 5.3.



obr. 5.3: metody pro vytvoření prvního trojúhelníku

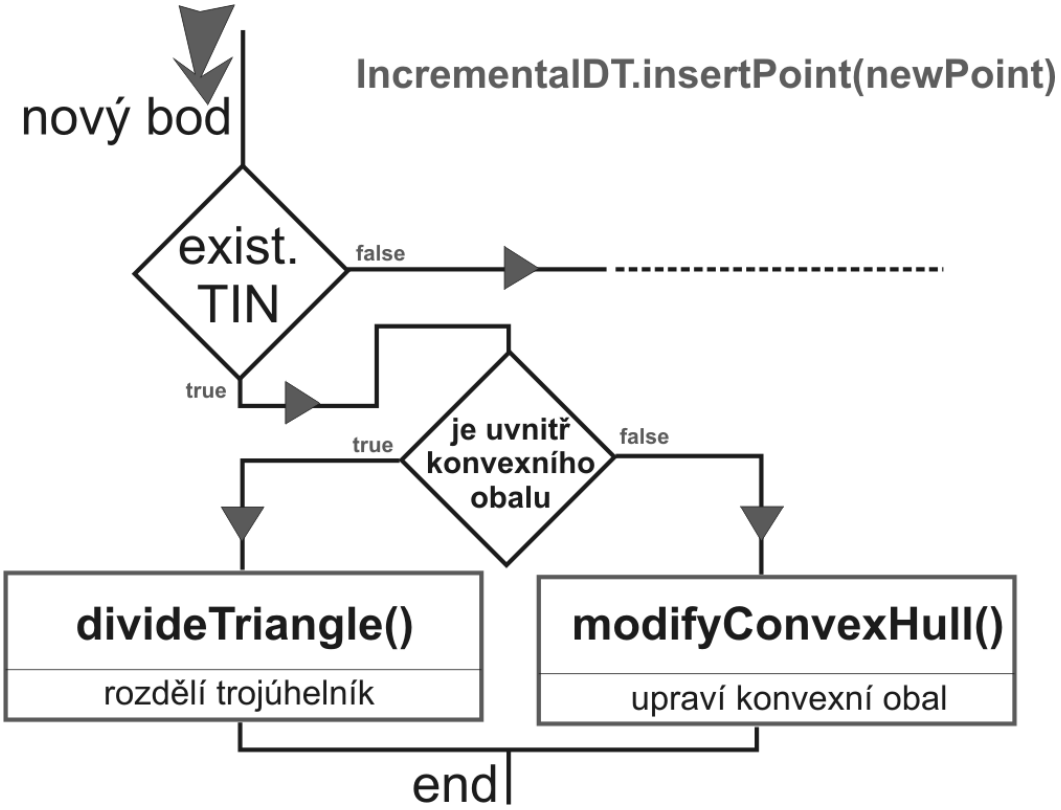
Zavolá se metoda insertPoint(P), kde parametr P je nový bod. Metoda otestuje, zda existuje TIN. V případě, že TIN ještě neexistuje, testuje se počet_bodů v triangulaci. Když je menší než tři, automaticky pomocí metody savePoint() uloží bod do zásobníku. Jakmile počet bodů je větší než 3, pomocí podmínky *exist_trojúhelník* zkoušíme, zda je možno vytvořit trojúhelník. Když tomu tak není, ukládáme nový bod do zásobníku. Tento postup se opakuje do té doby, než můžeme vytvořit první trojúhelník. Pak jen stačí otestovat, jestli je zásobník prázdný. Když tomu tak není, pomocí metody reInsertPoint() se body znovu vloží do triangulace, která by v tento okamžik měla vypadat stejně jako na obrázku 5.4..



obr. 5.4: triangulace po zadání bodu č. 5

5.2 TIN již existuje

V tomto kroku již existuje TIN. Mohou tedy nastat dvě varianty. Bod bude ležet uvnitř konvexního obalu nebo mimo něj, jak ukazuje obrázek 4.5.



obr. 4.5: volání metod divideTriangle() a modifyConvexHull()

Pro reprezentaci konvexního obalu byla využitá třída LinearRing z knihovny com.vividsolutions.jts.geom.*; Tato třída obsahuje řadu metod. V tomto případě je použita:

- *.contains(Geometry)* vrátí true, jestli obsahuje geometrický obrazec, který je zadán jako parametr.

Nový bod se tedy pomocí metody contains() otestuje, zda se nachází uvnitř konvexního obalu, či mimo něj. Pak se následně spouští příslušná metoda:

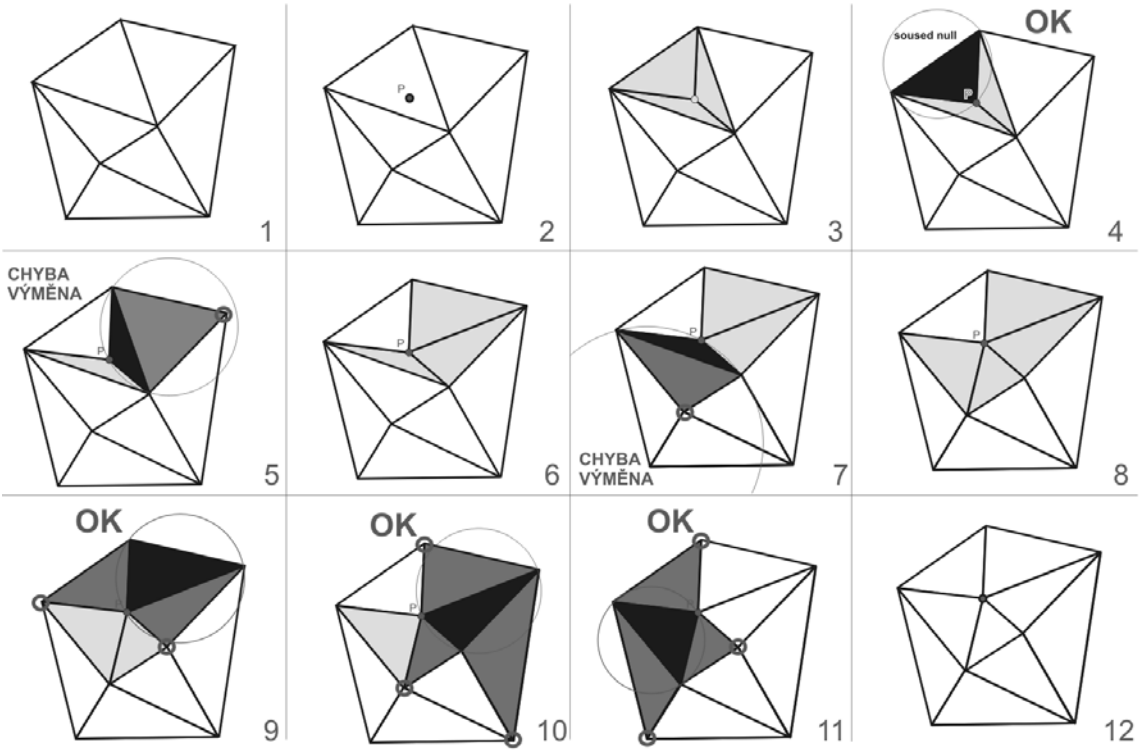
- nový bod leží uvnitř konvexního obalu - ***divideTriangle()***
- nový bod leží mimo konvexní obal - ***modifyConvexHull()***

Máme tedy dvě možnosti:

5.2.1 Metoda *divideTrinagle()*

Tato metoda se spouští, když se nový bod nachází uvnitř konvexního obalu. Obrázek 5.6

ukazuje, jak se postupuje v úpravě triangulace.



obr. 5.6: Delaunay test, postup triangulace

- *černý trojúhelník:* právě testovaný trojúhelník na Delaunay test
- *světle šedý trojúhelník:* nově vzniklý trojúhelník, který čeká na test
- *tmavo šedý trojúhelník:* je to soused testovaného trojúhelníku, kde jeho volný bod je testován, zda se nachází v opsané kružnici

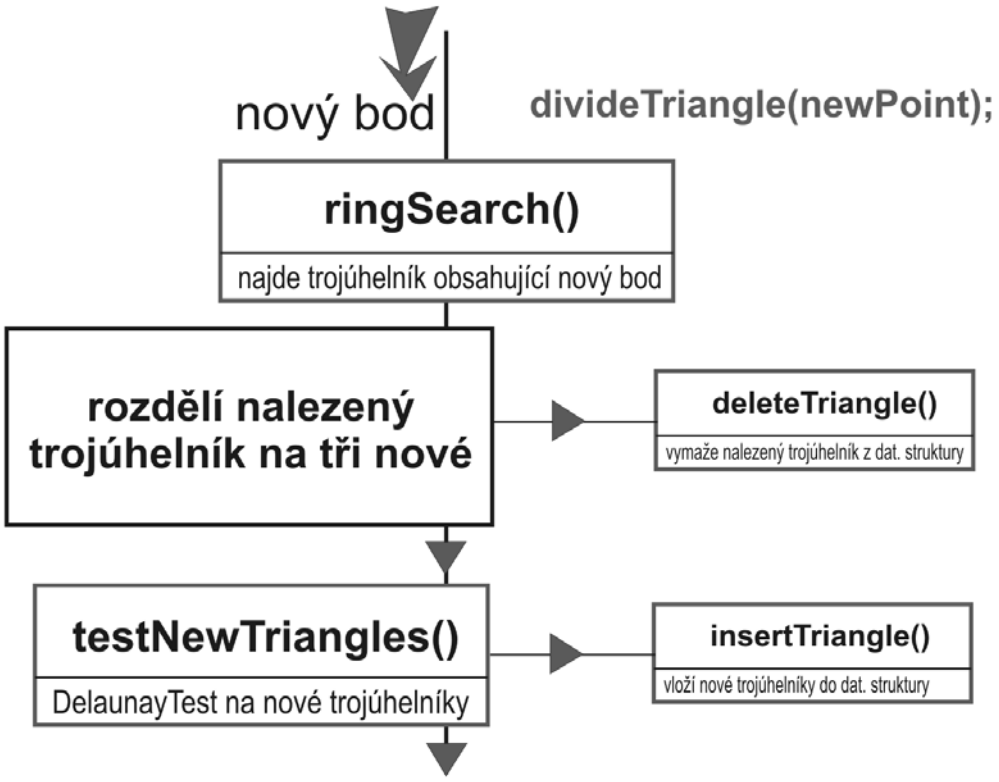
1) Na obrázku č.1 je vidět existující triangulace.
2) Do triangulace vstupuje nový bod P. Musíme najít trojúhelník, který ho obsahuje.
3) Našli jsme trojúhelník, ten jsme rozdělili na tři nové trojúhelníky. Původní trojúhelník byl vymazán z datové struktury.
4) U každého nově vzniklého trojúhelníku je potřeba provést Delaunay test. Čili vytvořit opsanou kružnici a zjistit, jestli se uvnitř kružnice nenachází některý z bodů triangulace. V našem případě je to v pořádku. Trojúhelník se uloží do datové struktury.
5) Pokračujeme v testu. Na tomto obrázku je vidět případ, kdy v opsané kružnici se nachází bod. Došlo k selhání testu, je nutno provést opravu.
6) Oprava se provádí tak, že se prohodí společná hrana trojúhelníků a vzniknou tak dva nové. Ty se samozřejmě musí znovu otestovat. Proto jsou nakresleny šedivou barvou.

7) Na obrázku 7 je opět vidět chyba v testu. Nutná výměna.
8) Triangulace po výměně hrany. Trojúhelníky jsou opět světle šedé, je tedy potřeba je otestovat.

9,10,11) pokračujeme v testu. Zde už chyba nenastala.

12) Poslední obrázek zobrazuje hotovou triangulaci po vstupu bodu P.

Na tomto postupu je už vidět, jak algoritmus bude pracovat. Na obrázku 5.7. je to ukázáno pouze schématicky.

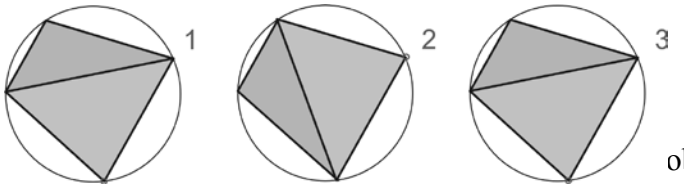


obr. 5.7: schématické znázornění metody divideTriangle()

Do metody *divideTriangle()* vstoupí nový bod. Zavolá se metoda *ringSearch()*, ta nalezne trojúhelník, který obsahuje nový bod. Dále metoda rozdělí nalezený trojúhelník na tři nové a vymaže nalezený trojúhelník z datové struktury. Následuje metoda *testNewTriangles(),* která otestuje nové trojúhelníky a vloží je do datové struktury.

5.2.2 Nebezpečná kružnice

Při implementaci Delaunayho testu hrozí jedno nebezpečí. Je to nebezpečná kružnice, kdy na opsané kružnici trojúhelníku leží i náš testovaný bod. Na obrázku 5.8 je tento stav.



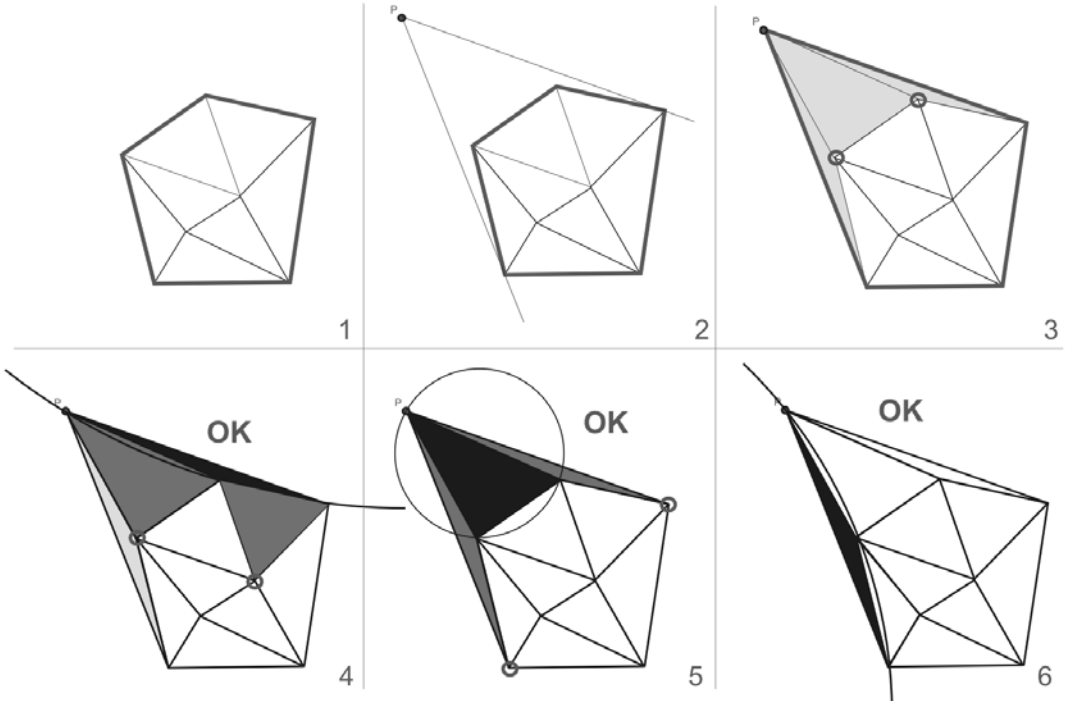
obr: 5.8 nebezpečná kružnice

1) začínáme testovat trojúhelník. Testovaný bod je uvnitř a je tedy potřeba udělat výměnu diagonál.
2) provedla se výměna diagonál a je potřeba otestovat nové trojúhelníky. Opět nastala chyba a prohodíme diagonály.
3) dostáváme se do původního stavu a je nutné trojúhelníky opět otestovat.

Zde je vidět reálné nebezpečí, kdy se algoritmus může zacyklit. Proto je nutné poloměr opsané kružnice zmenšit o malou diferenci. Tím dojde k tomu, že bod, který leží na hraně kružnice, je mimo a nedojde k výměně diagonál.
Tento způsob ale vede k tomu, že triangulace na množině stejných bodů není vždy identická, a to v případě, kdy body do inkrementální metody pošleme v opačném pořadí. Na místech, kde body jsou na hraně nebezpečné kružnice, budou pak prohozené diagonály. Je to z toho důvodu, že se testoval nejprve druhý trojúhelník.

5.2.3 Metoda *modifyConvexHull()*

Metoda na změnu konvexního obalu se spouští v druhém případě, když nový bod leží mimo konvexní obal. Postup výpočtu ukazuje obrázek 5.9.



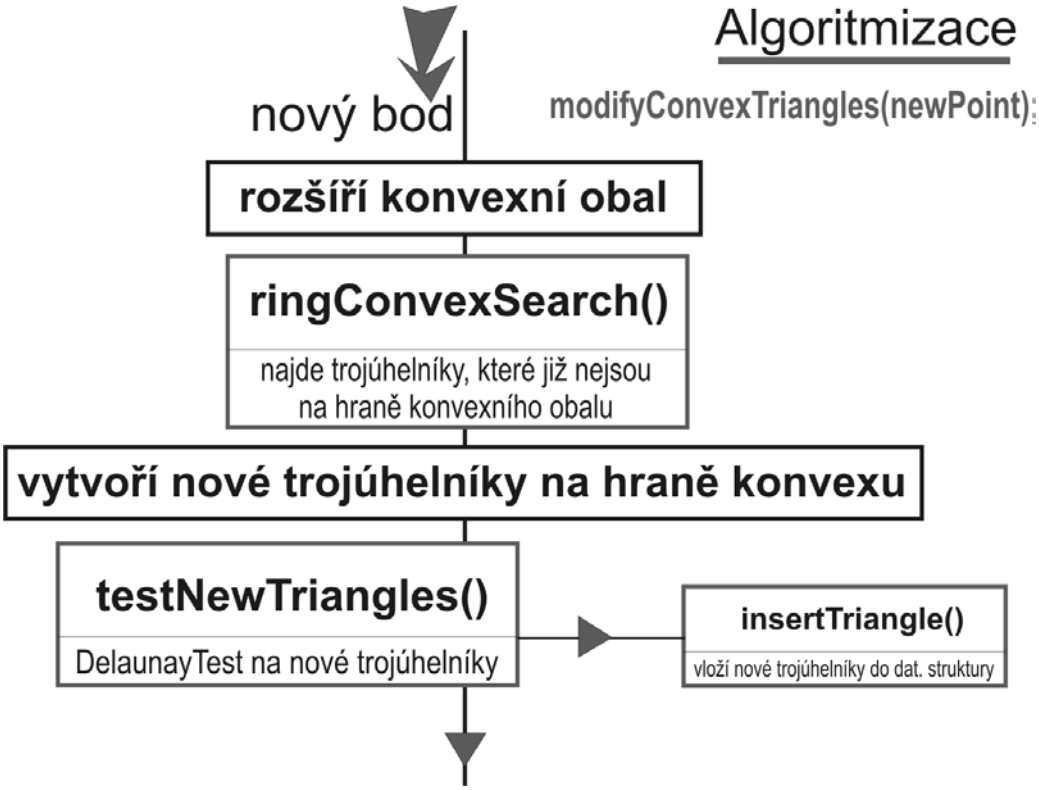
obr. 5.9: postup triangulace pro bod mimo konvexní obal

- *černý trojúhelník:* právě testovaný trojúhelník na Delaunay test
- *světle šedý trojúhelník:* nově vzniklý trojúhelník, který čeká na test
- *tmavo šedý trojúhelník:* je to soused testovaného trojúhelníku, kde jeho volný bod je testován, zda se nachází v opsané kružnici

1) Na obrázku č.1 je vidět existující triangulace.
2) Do triangulace vstupuje nový bod, který leží mimo konvexní obal.
3) Dojde ke změně konvexního obalu. Důležité jsou body, které z konvexního obalu vypadly. Tyto body budou totiž generovat nově vzniklé trojúhelníky.
4) Nově vzniklé trojúhelníky se musí opět otestovat Delaunayho testem.

5,6) Všechny trojúhelníky jsou v pořádku.

Zde je tedy vidět postup pro vstupující bod, který leží mimo konvexní obal. Schématické znázornění algoritmu ukazuje obrázek 5.10.



obr. 5.10: metoda modifyConvexHull()

Do metody *modifyConvexHull()* vstoupí nový bod, který nám rozšíří konvexní obal. Je potřeba uložit body, které nám vypadly z konvexního obalu. Ty se následně využijí pro konstrukci nových trojúhelníků. Metoda *ringConvexSearch()* nalezne trojúhelníky na hraně bývalého konvexu. Ty se pak uloží jako sousedé nových trojúhelníků. Nové trojúhelníky se dále otestují metodou *testNewTriangles()* na Delaunayho podmínku. Otestované trojúhelníky se vloží do datové struktury.

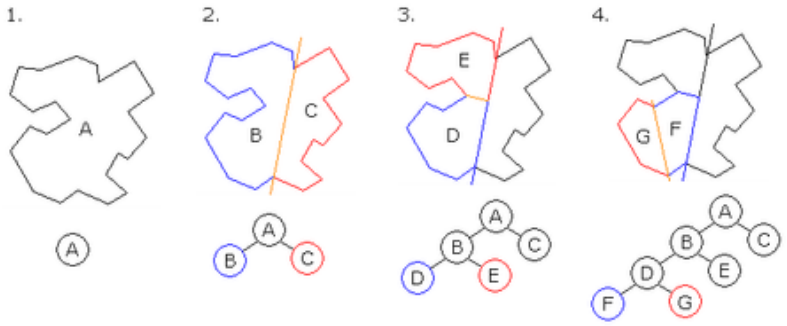
6 Datová struktura

Rychlost výpočtu triangulace je především závislá na datové struktuře, která drží trojúhelníky a na vyhledávacích algoritmech nad touto strukturou. Ve třídě IncrementalDT bylo postupně otestováno několik datových struktur. Vycházelo se z [Hort06], [Hero04], [web3], [web4].

- ***nesetříděný spojový seznam*** - první byl použit klasický nesetříděný spojový seznam *LinkedList* z knihovny java.util z java CoreAPI. Hledání spočívalo v průchodu celým seznamem a každý trojúhelník byl testován, zda obsahuje nový bod. Toto řešení se nakonec ukázalo, jako scela nevhodné.
- ***setříděný spojový seznam*** - s touto strukturou se dosahovalo lepšího času. Klíčem pro uložení do datové struktury byla největší x-ová souřadnice. Hledání v této struktuře spočívalo v nalezení menšího klíče a od tohoto místa se testovaly trojúhelníky, jestli obsahují hledaný bod.
- ***hybridní setříděný spojový seznam*** - jednalo se o jakýsi hybrid mezi hash tabulkou a spojovým seznamem. Celá oblast, kde se nacházely body, se rozdělila na několik podoblastí, což byly jednotlivé buňky hash tabulky. Vlastní oblast byla reprezentována setříděným spojovým seznamem. Toto není rozhodně špatné řešení, ale jeho použití by bylo ideální pro výpočet triangulace, kde vstupem by byla setříděná data.
- ***K-D Strom*** - poslední strukturou, která byla testována a ukázala se jako ideální pro práci s prostorovými daty a v GIS aplikacích, je vícedimenzionální K-D strom. Těžiště trojúhelníku je klíčem pro ukládání do stromu.

6.1 BSP stromy

BSP *(Binary space partitioning)* je binární strom, který je vhodný pro prostorová data. Vlastní prostor rozděluje na části, které jsou organizovány v datové struktuře binárního stromu.



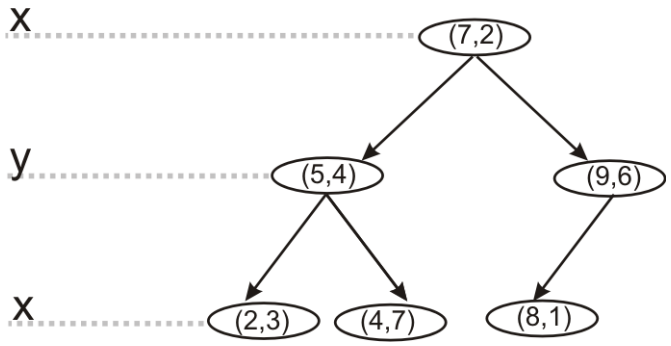
obr. 6.1: BSP Tree, [web18]

Tento datový model používá ve svých aplikacích firma Bentley. U BSP Tree je problematické odebírání prvků z datové struktury. Proto jsem se rozhodl pro použití K-D stromů, [web4].

6.2 K-D stromy

K - Dimensional tree je vícedimenzionální strom pro reprezentaci prostorových dat vyšších dimenzí. Jedná se opět o binární strom. Data jsou v něm organizována podle vícedimenzionálních klíčů. V případě, že se jedná o prostorová data, bude se jednat o souřadnice bodu. Operace nad binárním stromem, jako je vkládání, vymazání a vyhledávaní prvků, ukazují následující obrázky, [web4].

6.2.1 Vložení prvku do K-D stromu

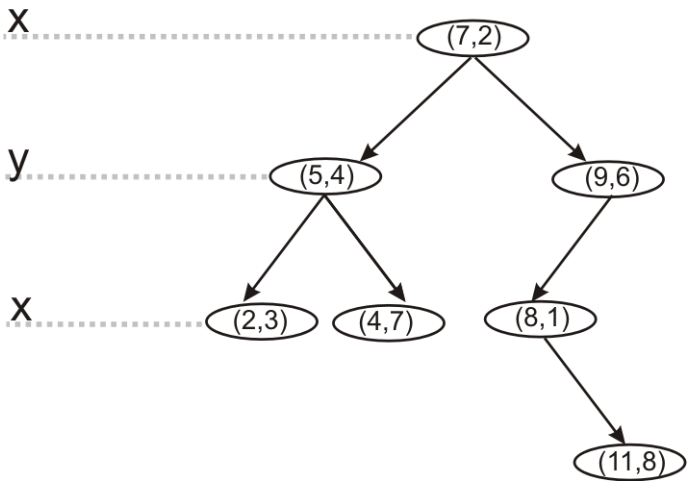


obr. 6.2: vkládání do K-D stromu 2D, [web4]

Na obrázku 6.2 je znázorněno vkládaní prvků do stromu. Jedná se o 2D strom. Vlastní organizace dat je následující. V každé úrovni stromu nebo hloubce se porovnává jiná souřadnice. V našem případě v hloubce 0 podle x, pak v hloubce 1 podle y. Tento proces se stále opakuje.

Příklad: Do datové struktury vložíme prvek s klíčem (11,5).

- bod s tímto klíčem se nejprve podle x-ové souřadnice porovná s uzlem (7,2). Jelikož 11>7 jde doprava.
- porovná vrchol (9,6) podle y-ové souřadnice. Jelikož 5<6, jde doleva
- porovná vrchol (8,11) podle x-ové souřadnice. Jelikož 11>8, jde doprava. tento uzel je již null, proto na toto místo uloží prvek s klíčem (11,5), obr. 4.13



obr. 6.3: změna stromu po vložení prvku (11.8), [web4]

6.2.2 Vymazání prvku v K-D stromu

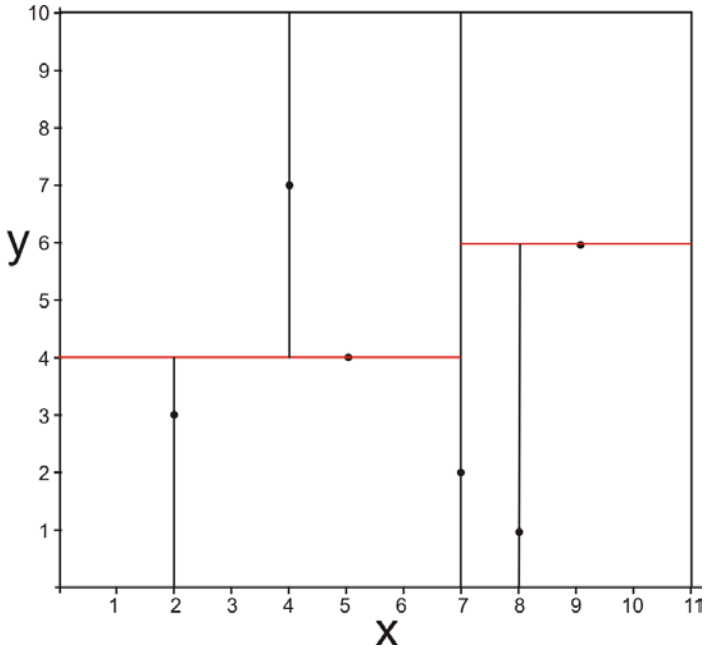
Nejjednodušším způsobem vymazání uzlu, je dát mu parametr deleted. Tímto způsobem je to řešeno i v datové struktuře IncrementalDT, [web4].

6.2.3 Vyhledávání prvku v K-D stromu

Vyhledání existujícího prvku je velmi jednoduché. Použijeme stejného algoritmu pro vložení. Jediný rozdíl je, že v každém kroku testujeme, jestli klíč je identický, [web4].

6.2.4 Vyhledávání nejbližšího prvku v K-D stromu

K-D strom umožňuje vyhledávat nejbližší prvek. Na obrázku 6.4 je znázorněno jak K-D strom rozdělí oblast na menší obdélníky.



obr. 6.4: rozdělení oblasti K-D stromem [web4]

Vyhledání nejbližšího prvku tedy znamená nalezení nejmenšího možného obdélníku a prohledání okolních oblastí, kde by se ještě mohl nacházet bližší bod, [web4].

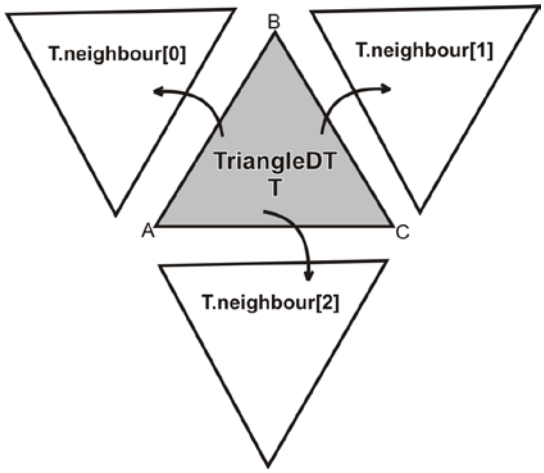
6.3 Datová struktura v IncrementalDT

Trojúhelníky jsou ukládány do K-D stromu. Klíče do K-D stromu, jsou těžiště trojúhelníků. Tato volba přinesla řadu výhod:

- těžiště trojúhelníků je v TIN jedinečným a jednoznačným číslem. Není možné, aby existovala v triangulaci dvě stejná těžiště. To by znamenalo, že existují dva stejné trojúhelníky nebo jsou přes sebe. V obou případech se jedná o chybu, která nemůže nastat.
- další výhodou je snadná optimalizace vyhledávacích algoritmů. Delaunayho triangulace vytváří trojúhelníky, které se co nejvíce blíží trojúhelníkům rovnostranným. Proto těžiště reprezentuje jakýsi střed plochy, která je vymezená trojúhelníkem. Můžeme tedy předpokládat, že pokud do triangulace vstoupí nový bod a nalezneme nejbližší těžiště k tomuto novému bodu, bude nový bod s velkou pravděpodobností ležet v tomto trojúhelníku.

6.3.1 Síťový model

Síťový model datové struktury spočívá v přidání topologické informace každému trojúhelníku o jeho sousedech.



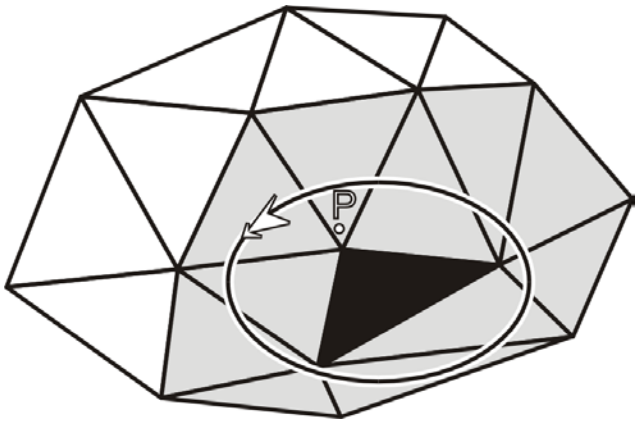
obr. 6.5: síťový model datové struktury

To znamená, že když bod padne do trojúhelníku T, nemusíme již vyhledávat okolní trojúhelníky, protože T v sobě nese odkazy na sousedy. V klasickém provedení musíme minimálně jednou projít datovou strukturu pro nalezení sousedů. V nejhorším případě, kdy jeden soused bude null, musíme projít celou strukturu. Pokud dojde k výměně diagonál musíme tento postup několikrát opakovat. Toto uspořádání přineslo velké časové zrychlení celého algoritmu, ale bohužel i vyšší paměťové nároky, [Stro02].

6.3.2 Vyhledávací algoritmy

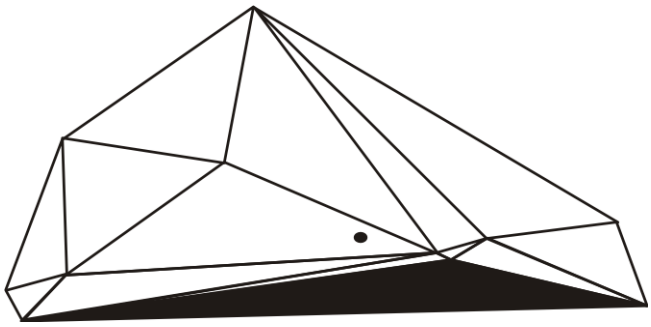
Primární vyhledávací algoritmus je K-D strom. Program najde nejbližší těžiště k novému bodu a otestuje, zda nalezený trojúhelník obsahuje bod. Při testování této metody byla účinnost nalezení správného trojúhelníku přibližně 56%. To znamená, že ve 44% případů, neobsahuje trojúhelník s nejbližším těžištěm nový bod. V původním kódu jsem to řešil průchodem celého stromu a testováním všech trojúhelníků.

Ve finální verzi IncrementalDT je implementováno prstencové vyhledávání. Nalezený trojúhelník K-D stromem se otestuje. Jestliže neobsahuje nový bod, testují se v prstenci jeho sousedé. Opět se s výhodou využil síťový model.



obr. 6.6 prstencové vyhledávání

Obrázek 6.6 ukazuje, jak to funguje. Nalezený černý trojúhelník neobsahuje nový bod, proto se spouští metoda *ringSearch()*. Tato metoda vezme prvního existujícího souseda a otestuje. Pak pokračuje dalším svým sousedem. Vždy se hlídá, aby testovaný trojúhelník měl minimálně jeden vrchol totožný s nalezeným trojúhelníkem T. Dále je potřeba ukládat odkazy na již testované trojúhelníky do zásobníku, aby nedošlo k zacyklení.
Při testování tohoto algoritmu byla již účinnost nalezení trojúhelníka 99.93%. Bohužel v 0.07% algoritmus selže a trojúhelník nenajde. Je to hlavně v místech u hrany konvexního obalu, kdy trojúhelníky jsou velmi dlouhé a nalezené nejbližší těžiště může být až o několik prstenců dále. Obrázek 6.7 ukazuje tento případ.



obr. 6.7: selhání metody *ringSearch()*

Černý trojúhelník je trojúhelník s nejbližším těžištěm. Je vidět, že když provedeme prstencové vyhledávání, trojúhelník obsahující bod nenajdeme. Musíme tedy projít celý strom. V *IncrementalDT* je to realizováno pomocí rekurzivního průchodu preorder.

7 Testovací třída

Pro třídu IncrementalDT byla napsána testovací třída *IncrementalDTTest.java*, kde pomocí knihovny *JUnit* se vytvořily tyto testy:

public void testDelaunayCircles(){..}
- otestuje všechny trojúhelníky na Delaunay test

public void testRightCountCircle(){..}
- *otestuje u všech trojúhelníků, zda je správně vypočtena opsaná kružnice*

public void testAllIsTriangle(){..}
- *otestuje, zda body, které generují trojúhelník, nejsou v přímce*

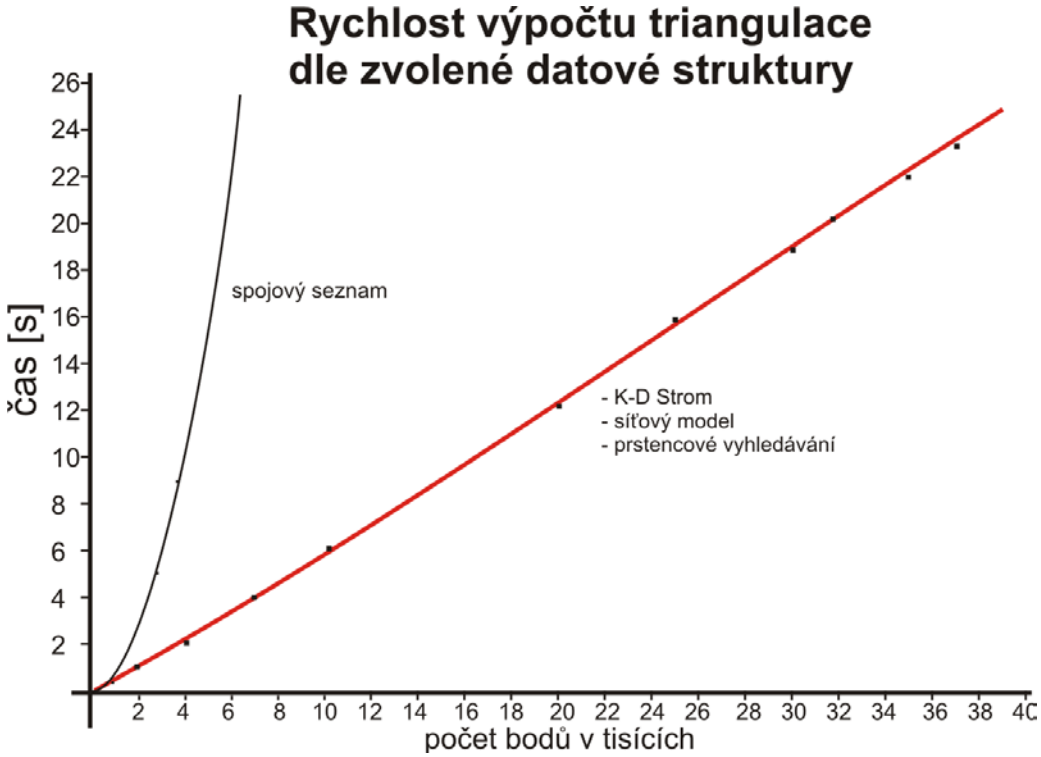
public void testAllPointsExistInTIN(){..}
- *otestuje, zda všechny vstupující body jsou obsaženy v TIN*

public void testDupliciteTriangles(){..}
- *hledá duplicitní trojúhelníky*

8 Zhodnocení třídy IncrementalDT

Třída IncrementalDT má dvě veřejné (public) metody:

insertPoint(Coordinate) - metoda pro vstup bodů typu *Coordinate*. *Coordinate* je datový typ z knihovny JTS.
LinkedList getLinkedList() - vrátí spojový seznam (knihovna java.util) z vytvořených trojúhelníků.
Třída byla mnohokrát odzkoušena a jeví se jako zcela funkční. Zhodnotit, jak je celý algoritmus rychlý, může nejlépe následující obrázek.



obr. 8.1: závislost počtu bodů na času výpočtu

Graf ukazuje časovou závislost výpočtu triangulace za použití spojového seznamu. Tato křivka má přibližně kubický charakter. Druhá křivka je časová závislost za použití všech zrychlovacích prostředků: K-D strom, síťový model a prstencové vyhledávání. Když se použijí tyto tři prostředky, je výsledná časová závislost téměř lineární. To je velmi dobrý výsledek. Maximálně optimalizovaná datová struktura je velice náročná na paměťové nároky počítače. Na počítači s operační pamětí 512MB program po zadání 40000 bodů selže a vypíše chybové hlášení „out of memory“. Řešením by bylo držet datovou strukturu v souboru na HDD. To by sice znamenalo výrazné zpomalení, protože datová propustnost pevného disku je mnohonásobně nižší než RAM paměti. Velkou výhodou této varianty by byl téměř neomezený počet vstupujících bodů do triangulace.

9 Pevné hrany

Poslední část této práce pojednává o pevných hranách v triangulaci. Algoritmus ve třídě *IncrementalDT* bylo poměrně složité modifikovat natolik, aby bylo možné počítat TIN s pevnými hranami. Proto se implementovala statická třída *TINWithFixedLines* a statická metoda *countTIN(LinkedList trians, LinkedList hLines).* Parametrem této metody jsou dva spojové seznamy.

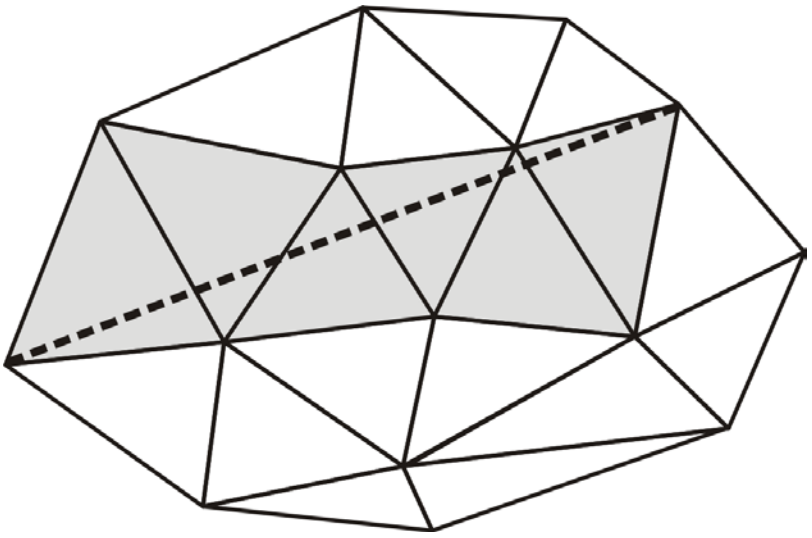
trians - vypočtená trojúhelníková síť

hLines - pevné hrany, počáteční a koncový bod

metoda vrátí spojový seznam LinkeList se změněnými trojúhelníky v okolí pevných hran, [Koho06], [Stro03].

9.1 Algoritmus pro výpočet TIN s pevnými hranami

Do metody countTIN vstupuje jako parametr vypočtená trojúhelníková síť a pevné hrany, které jsou definované počátečním a koncovým bodem.



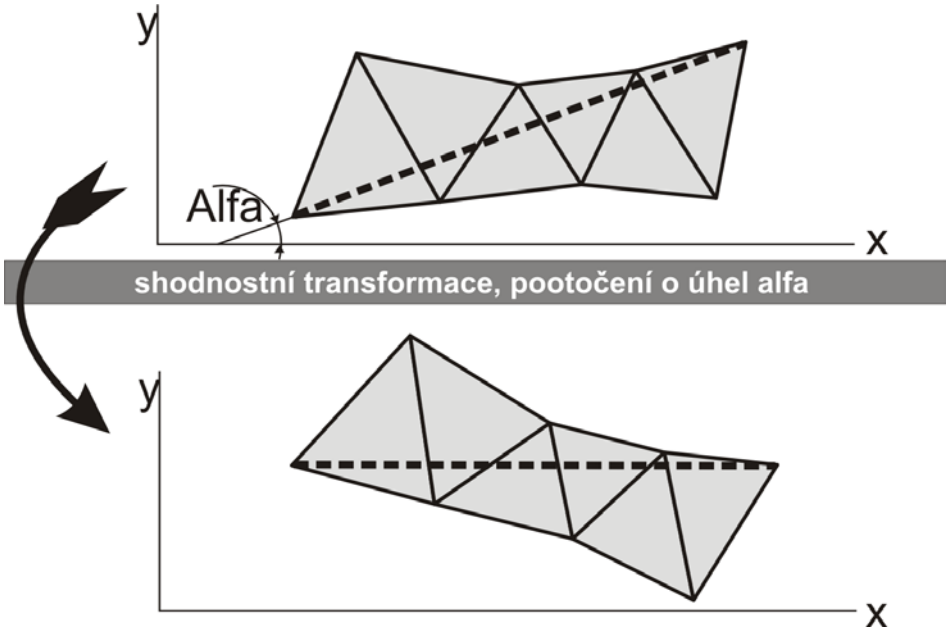
obr. 9.1: trojúhelníky ke změně po vstupu pevné hrany

Na obrázku 9.1 je vidět existující TIN, do kterého vstupuje pevná hrana, značená čárkovaně. Světle šedou barvou jsou znázorněny trojúhelníky, které se budou muset změnit.

Při tvorbě algoritmu se hledal takový případ, ***kdy se jednoznačně může říci, že mezi dvěma body bude existovat pevná hrana***. To je jen v případě, ***když tyto dva body budou ležet v konvexním obalu***. Pak vzhledem k definici Delaunay triangulace je toto pravidlo splněno. Na této myšlence je vystavěna třída *TINWithFixedLines.*
To znamená, rozdělit šedou oblast, znázorněnou na obr.9.1, na dvě části. Na horní a dolní triangulaci, kde dělícím prvkem bude pevná hrana.

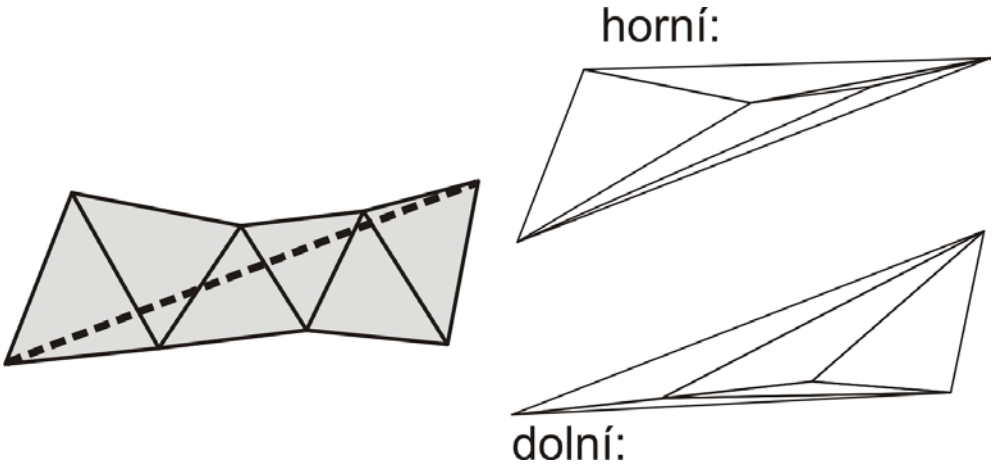
9.2 Horní a dolní triangulace

Horní triangulace jsou body, které leží nad přímkou a koncový a počáteční bod pevné hrany. Dolní triangulace jsou body pod přímkou a koncový a počáteční bod. Na obrázku 9.2 je znázorněn algoritmus pro rozdělení bodů.



obr. 9.2: rozdělení bodu na horní a dolní triangulaci

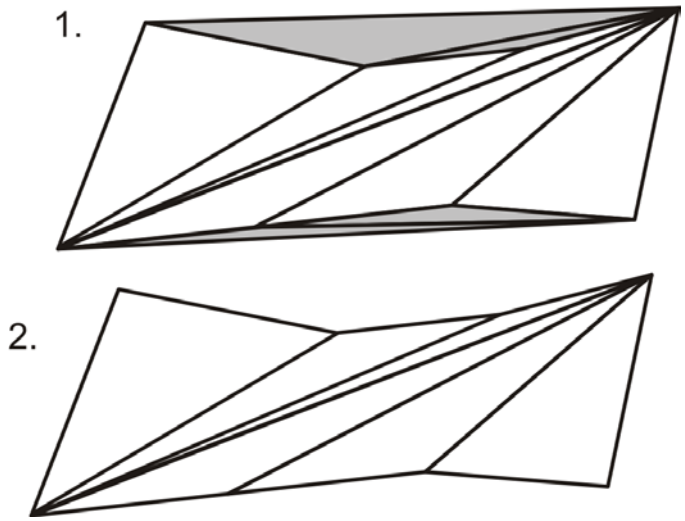
Existuje řada metod, jak to udělat. Ve třídě TINWithFixedLines je to pomocí shodnostní transformace. Ze souřadnic počátečního a koncového bodu pevné hrany se vypočte úhel alfa, o který se pootočí všechny body. Pak vzhledem k tomu, že pevná hrana je vodorovná s osou x, se mohou podle y souřadnice rozdělit body do horní a dolní triangulace. Z těchto dvou množin bodů se vypočte triangulace pomocí třídy *IncrementalDT*.



obr. 9.3: vypočtená horní a dolní triangulace

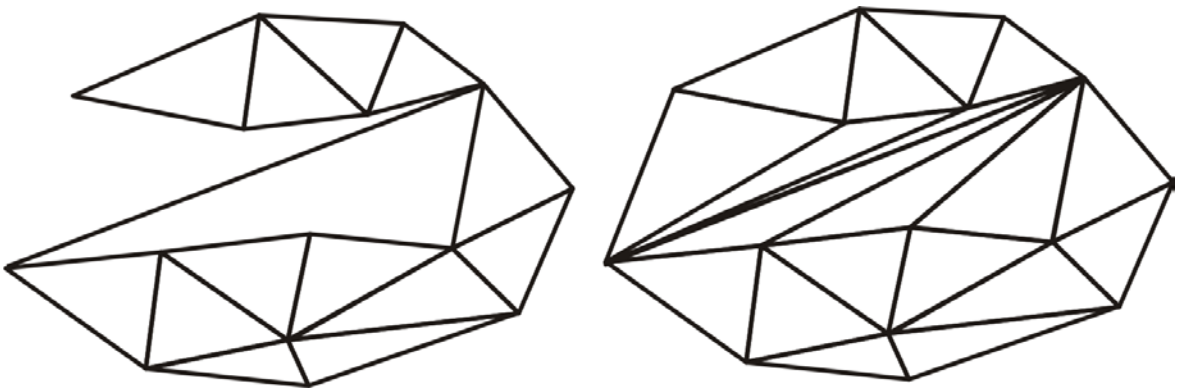
To ukazuje obrázek 9.3. Triangulace se počítá samozřejmě z původních bodů. Nyní

se složí tyto triangulace dohromady. Jsou vidět šedé trojúhelníky, které nám přesahují původní oblast změny.



obr. 9.4: odstranění přesahujících trojúhelníků

Po výpočtu TINu dostaneme vždy konvexní obal, proto musíme přesahující trojúhelníky odstranit. Otestujeme tedy všechny nové trojúhelníky, jestli jejich těžiště obsahuje nějaký původní trojúhelník. Jestliže ano, vložíme ho do datové struktury. Posledním krokem je odstranění původních trojúhelníků. Tím dostáváme nový TIN.



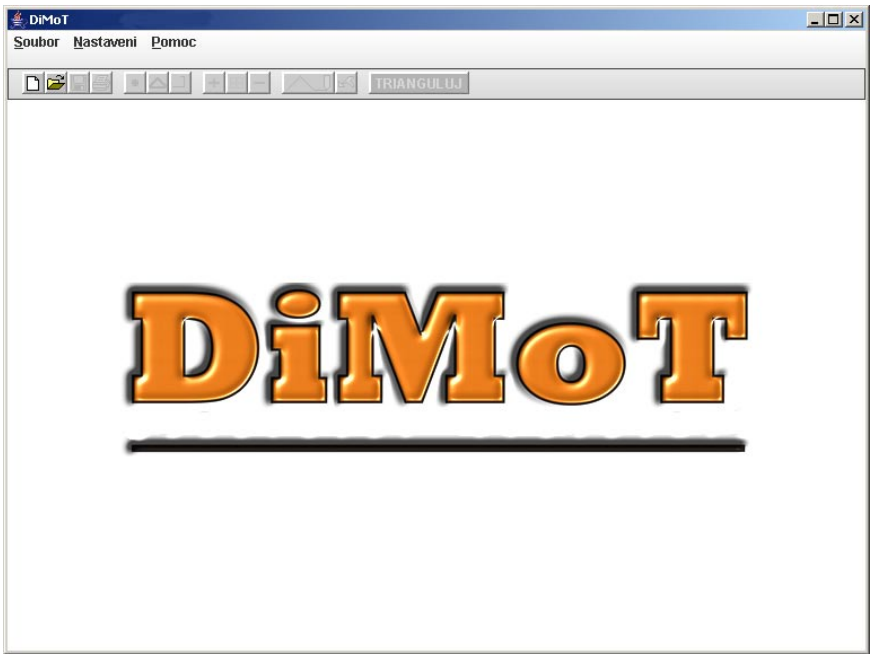
obr. 9.5: odstranění původních trojúhelníků a vložení nových

9.3 Zhodnocení třídy TINWithFixedLines

Metoda nabízí stabilní algoritmus, ale nastal tu jeden problém, který je ještě nedořešen. Pokud body z horní nebo dolní triangulace jsou všechny v konvexním obalu, může vzniknout trojúhelník, který nebude spojen s žádným bodem pevné hrany. To znamená, že jeho opsaná kružnice se bude jen mírně nachylovat k pevné hraně. Tudíž je tu reálné nebezpečí, že bude existovat bod, který bude ležet uvnitř kružnice a tím pádem je porušena Delaunay podmínka. Tato možnost se ještě musí řádně otestovat a případně dořešit.

10 DiMoT

Na základě těchto dvou metod *IncrementalDT* a *TINWithFixedLines* vznikla aplikace jménem Dimot. Což je zkratka pro digitální model terénu. Jedná se o jednoduchou prohlížečku bodů v S-JTSK. Program dále zobrazuje TIN, umožňuje kreslit pevné hrany a generuje TIN s pevnými hranami. Vstupem je soubor typu *.stx a výstupem je soubor typu shapefile.



obr.9.1: DiMoT - java aplikace

popis menu:

 - založí novou pracovní plochu

 - otevře soubor *.stx

 - uloží TIN do shapefile

 - vrstva bodů

 - vrstva TIN

 - vrstva pevných hran

 - plus zoom

 - fit zoom

 - minus zoom

 - kreslí pevné hrany

 - zpět pevné hrany

- trianguluje oblast

V příloze je uvedeno několik obrázků z programu DiMoT. Vlastní program je uložen na přiloženém CD v adresáři DiMoT. Zde spuštění souboru ***start.bat*** se spustí DiMoT.

11 Porovnání programu DiMoT

Porovnání bylo provedeno po vizuální stránce s komerčním softwarem Atlas DMT od společnosti Atlas s.r.o. a s open source programem OpenJump PIROL Edition.

11.1 Atlas DMT

Základ programového systému Atlas, umožňuje zpracovávat výškopisná data z textových

souborů nebo z geodetických zápisníků, fotogrammetrie a z programu Kokeš.
DMT Atlas zpracovává najednou až 10 000 000 bodů [web2]

11.1.1 Hodnocení

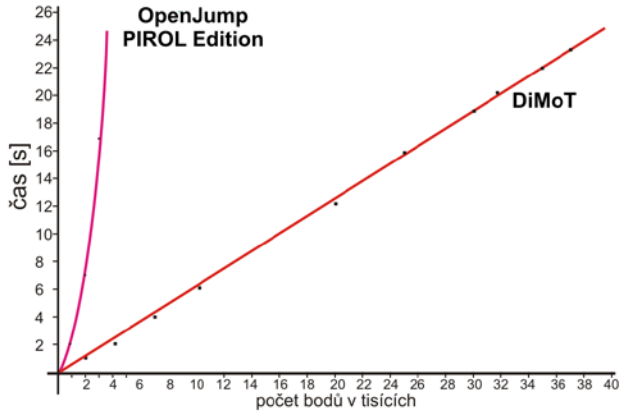
Byly vybrány dva soubory souřadnic typu stx, které byly vytvořeny při měřičské praxi v Nečtinách. Z těchto souborů se v programu DiMoT a Atlas vytvořila trojúhelníková síť TIN a vzájemně se porovnala. Výsledek je naprosto shodný. Tento způsob kontroly metody *IncrementalDT* jen potvrdil výsledky vytvořené testovací třídy, které byly rovněž v pořádku. Jednotlivé výstupy na obrazovku (*screen*) jsou uvedeny v příloze A.

11.2 OpenJump PIROL Edition

OpenJump PIROL Edition je projekt, který je vyvíjen na německé univerzitě v Osnabrücku. Základem tohoto projektu je standardní verze OpenJump, která je rozšířena o několik plugin modulů implementovaných na této univerzitě. Jeden z těchto modulů řeší Delaunayho triangulace. [web11]

11.2.1 Hodnocení

Jelikož OpenJump PIROL Edition je také Java aplikace, je možné porovnávat čas výpočtu. Pro porovnání se vytvořily dva vstupní soubory s identickými souřadnicemi. Textový soubor typu stx a shapefile pro OpenJump. Na těchto souborech bodů byl pak měřen čas výpočtu triangulace viz obr.11.



obr. 11: čas výpočtu triangulace v programu DiMoT a OpenJump

12 Získání zdrojových kódů

Zdrojové kódy lze získat z SVN a to na adrese (svn://mapserver.zcu.cz/bakalarky/bezdek). Zde jsou uloženy zdrojové kódy modulů *IncrementalDT, TINWithFixedLines* a zdrojové kódy knihovny *B2DTree*, což je upravený K-DTree. Ostatní požadované knihovny jsou standardní součástí Geotools.

13 Závěr

V rámci této práce byla přiblížena a popsána knihovna Geotools. Implementované demonstrační programy ukazují jednoduchost a efektivnost použití. Dále byl vyvinut modul na výpočet Delaunayovy triangulace, který přináší velice rychlé a stabilní řešení algoritmu s inkrementálním vkládáním. Tento modul je jakýmsi základem, který se dále rozšířil o výpočet pevných hran v triangulaci.

Řada optimalizací, které byly použity, přinesly podstatné zrychlení celého výpočtu. Toto zrychlení nejlépe ilustruje rychlostní test programu OpenJump s programem DiMoT, jenž představuje aplikaci vytvořenou na vizualizaci implementovaných tříd. V tomto rychlostním testu byl DiMoT mnohonásobně rychlejší, jak ukazuje obrázek 11 v kapitole 5.

Geotools v současné době neposkytují metody pro práci nad DMT, jako je například výpočet pevných hran, či výpočet vrstevnicového modelu. Tyto nedostatky bych rád řešil v navazující diplomové práci. Do budoucna by se implementoval plugin modul pro software uDig, který bude řešit generování vrstevnic. Vytvořený modul by se mohl distribuovat přes internetové stránky oddělení Geomatiky Fakulty aplikovaných věd a mohl by být dobrou reklamou pro naše oddělení. Snahou bude prosadit tento plugin modul jako standardní součást softwaru uDig.

.

Seznam použitých zkratek:

BSP Binary space partitioning (Binární prostorové dělení)
CVS Concurrent Version System (SCM systém)
DMT digitální model terénu
EPSG European Petroleum Survey Group (Evropská petrolejářská průzkumná společnost)
GFDL GNU Free Documentation License (volně přístupná koncese GNU pro dokumenty)
GIS Geograpics information system (Geografické informační systém)
GML Geograpics Markup Language (geografický značkový jazyk)
GNU GNU‘s Not Unix (projekt zaměřený na svobodný software, spravuje licence)
GPL General Public License (všeobecná veřejná licence GNU)
GUI Graphics User Interface (grafické uživatelské prostředí)
HDD Hard Disk Drive (pevný disk)
ISO International Organization for Standardization (Mezinárodní organizace pro normalizaci)
JRE Java Runtime Enviroment (Java provozní prostředí)
JTS Java Topology Suite (Java soubor topologií)
JVM Java Virtual Machine (Java virtuální stroj)
K-D K-dimensional tree (k- dimensionální strom)
LGPL Lesser General Public License (menší všeobecná veřejná licence GNU)
OGC The Open Geospatial Consortium (otevřené mezinárodní konsorcium)
RAM Random Access Memory (pamět s libovolným přístupem)
SCM Source Code Management (správa zdrojových kódů)
SDK Standart Development Kit (standardní vývojářská sada)
SVN Subversion revision control system (nadstavba CVS)
WFS Web Feature Service (webová objektová služba)
WMS Web Map Service (webová mapová služba)
TIN Triangulated Irregular Network (nepravidelná trojúhelníková síť)
XML Extensible Markup language (rozšiřitelný značkovací jazyk)

Použité zdroje - literatura:

[Hero04] Herout, P. *Učebnice jazyka Java.* Dotisk 1.vydání. České Budějovice: Koop, 2004. ISBN 80-7232-115-3.

[Hero04A] Herout, P. *Java - grafické uživatelské prostředí a čeština.* Dostik 1.vydání. České Budějovice: Koop, 2004. ISBN 80-7232-237-0.

[Hort06] Horton, I. *Java 5.* Přeložil Petr Poledňák. 1. vydání. Praha: Neocortex s.r.o., 2006. ISBN 80-86330-12-5.

[Ježe07] Ježek, J. *Implementace nástrojů pro transformace souřadnic: (materiál ke státní doktorské zkoušce)*. Praha: České vysoké učení technické. Katedra mapování a kartografie, 2007.

[Koho05] Kohout, J. *Delaunay triangulation in parallel and distributed enviroment.* State of the Art and Concept of Doctor Thesis. Plzeň: Západočeská univerzita. Fakulta aplikovaných věd, 2005.

[Koli99] Kolingerová, I. *Metody triangularizace v rovině.* Technická zpráva. Plzeň: Západočeská univerzita. Fakulta aplikovaných věd, 1999.

[Stro02] Stropek, I. *Triangulace a rekonstrukce v GIS a tvorba digitálního modelu terénu.* Diplomová práce. Plzeň: Západočeská univerzita. Fakulta aplikovaných věd. Katedra matematiky, 2002. Vedoucí diplomové práce Doc. RNDr. František Ježek, CSc.

Použité zdroje - internet:

[web1] Apache Maven Project [online]. 2002 - 2007 [citováno 25.4.2007]. Dostupné z: http://maven.apache.org/

[web2] Atlas [online]. Poslední úprava: 24.1.2007 [citováno 25.4.2007] Dostupné z: http://www.atlasltd.cz/

[web3] Binary Space Partitioning Trees [online]. Poslední úprava: 26.8.1995 [citováno 25.5.2007]. Dostupné z: http:// www.geocities.com/SiliconValley/2151/bsp.html

[web4] Carnegie Mellon university: School of Computer Science [online]. Poslední úprava: 3.5.2007 [citováno 15.5.2007]. Dostupné z: http://www.cs.cmu.edu/

[web5] ESRI - GIS and Mapping software [online]. Poslední úprava: 2007 [citováno 1.4.2007]. Dostupné z: http://www.esri.com/

[web6] GeoServer [online]. Poslední úprava: 15.2.2006 [citováno 10.4.2007]. Dostupné z: http://geoserver.org/

[web7] GeoTools - The Java GIS Toolkit [online]. [citováno 10.8.2007]. Dostupné z: http://geotools.codehaus.org/

[web8] GeoVista studio [software]. Poslední úprava: 20.10.2006 [citováno 10.4.2007]. Dostupné z: http://www.geovista.psu.edu/index.jsp

[web9] JTS Topology suite [online]. Poslední úprava: 11.4.2007 [citováno 25.4.2007]. Dostupné z: http://www.vividsolutions.com/jts/jtshome.htm

[web10] OGP Surveying&Positioning Committee [online]. Poslední úprava: 2007 [citováno 5.5.2007]. Dostupné z: http://www.epsg.org/

[web11] PIROL-OpenJump [software]. Verze 1.1.2 Dostupné z: http://www.pirol.fh-osnabrueck.de/pirol-openjump.html

[web12] Stránky o Svobodném software [online]. Poslední úprava: 29. 12. 2004 [citováno 6.4.2007]. Dostupné z: http://www.gnu.cz/

[web13] Subversion [online]. Poslední úprava: 2007 [citováno 25.4.2007]. Dostupné z: http://subversion.tigris.org/

[web14] The GeoApi project [online]. Poslední úprava: 15.1.2007 [citováno 5.4.2007]. Dostupné z: http://geoapi.sourceforge.net/stable/site/index.html

[web15] The Open Geospatial Consortium, Inc [online]. Poslední úprava: 25.3.2007 [citováno 10.3.2007]. Dostupné z: http://www.opengeospatial.org/

[web16] Topol Software s.r.o.: Topol software [online]. 1999 - 2007 [citováno 15.4.2007]. Dostupné z: http://www.topol.cz/?doc=3040

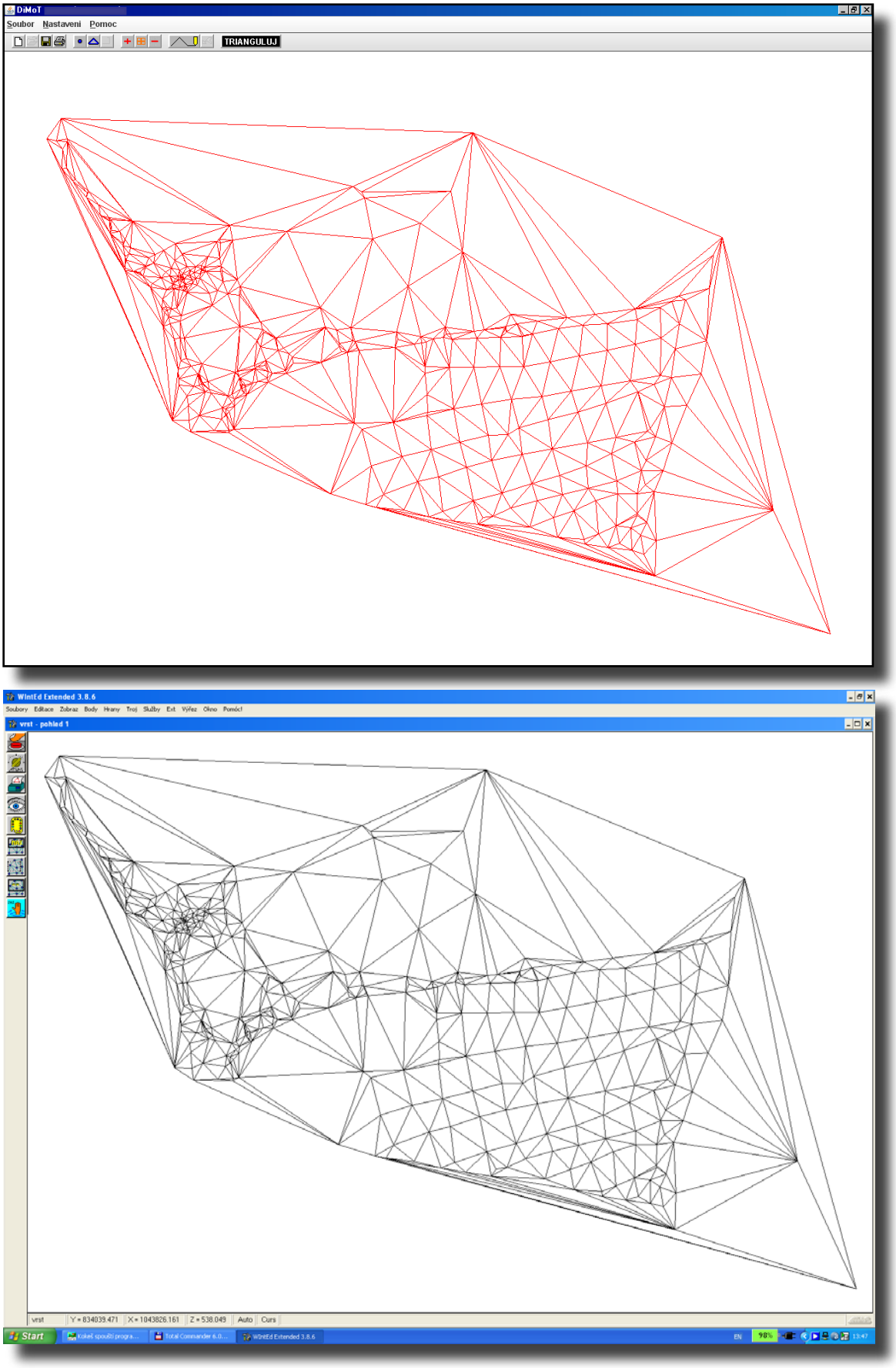
[web17] uDig [software]. Poslední úprava: 5.4.2007 [citováno 10.4.2007]. Dostupné z: http://udig.refractions.net/confluence/display/UDIG/Home

[web18] Wikipedia [online]. [citováno 17.5.2007]. Dostupné z: http://www.wikipedia.org/

Přílohy:

A porovnání výpočtu triangulace v programu DiMoT a Atlas
B struktura přiloženého CD

Příloha A



Porovnání výpočtu triangulace v programu DiMoT a Atlas

Příloha B

Struktura přiloženého CD