TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií

Studijní program: B2612 – Elektrotechnika a informatika

Studijní obor: 1802R022 – Informatika a logistika

Generování vstupních dat modelu na konfigurovatelné síti

Generating of input data of a model on configurable mesh

Bakalářská práce

Autor: **Ladislav Neuman**

Vedoucí práce: **doc. Ing. Jiřina Královcová, Ph.D.**

Konzultant: **Ing. Zuzana Capeková**

V Liberci 29. 5. 2009

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 o právu autorském, zejména § 60 (školní dílo).

Beru na vědomí, že TUL má právo na uzavření licenční smlouvy o užití mé BP a prohlašuji, že **s o u h l a s í m** s případným užitím mé bakalářské práce (prodej, zapůjčení apod.).

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím bakalářské práce a konzultantem.

Datum:

Podpis:

Poděkování

Zde bych chtěl poděkovat svým rodičům za podporu během celého studia, vedoucímu mé práce doc. Jiřině Královcové, která se mnou měla nekonečnou trpělivost, a v neposlední řadě konzultantce Ing. Zuzaně Capekové.

Abstrakt

Obsahem práce bylo vytvoření výpočetní sítě a generování vstupních dat nutných pro výpočet proudění pomocí simulačního nástroje Flow123D. V první části bakalářské práce je popsán postup vytvoření výpočetní sítě s konfigurovatelnou základní geometrií zadané zájmové oblasti. Dále práce obsahuje přehled formátů používaných datových souborů potřebných pro konkrétní výpočet proudění na výpočetní síti. V třetí části je popsána implementace jednoúčelové aplikace pro generování datových souborů. Závěrečná část je věnována možnosti využití DirectX v programovacím prostředí Borland Delphi pro zobrazování sítí používaného typu.

Klíčová slova: GMSH, FLOW123D ,DirectX, Direct3D, Delphi

Abstract

The purpose of this bachelor dissertation is to create a computational mesh and to generate input data needed for flow calculation using simulation tool Flow123D. In the first part of this bachelor dissertation the method of creating computational mesh with configurable basic geometry of denoted field of interest is described. This work contains also the basic information of data files needed for particular simulation of underground flow. In the third part implementation of single purpose application for generating data files is analyzed. The last part concerns possibilities of using DirectX in programming-environment Borland Delphi for visualizing mesh of used type.

Key words: GMSH, FLOW123D, DirectX, Direct3D, Delphi

Obsah

Úvod

S rozvojem výpočetní techniky a jejím stále se zvětšujícím výkonem došlo zároveň i k rozvoji softwarových nástrojů a různých metod matematické simulace. Simulační modely se stále častěji využívají před samotnou realizací daného projektu, čímž se dosahuje zefektivnění například sanačního procesu, úpravě výrobního procesu, nebo přepracování evakuačního plánu.

Jedním z těchto softwarových nástrojů je Flow123D. Program je vyvíjen na Ústavu nových technologií a aplikované informatiky Technické univerzity v Liberci a slouží pro simulace proudění a transportu látek v saturovaném horninovém prostředí. Vzhledem k tomu, že tento nástroj je v současné době stále ve vývoji, není k dispozici mnoho materiálů ani dalších podpůrných programů.

Cílem práce bylo v první řadě vytvořit výpočetní síť s konfigurovatelnou základní geometrií pro zadanou zájmovou oblast a dále potom implementovat jednoúčelovou počítačovou aplikaci, která bude generovat základní datové soubory pro simulaci proudění na příslušné síti.

Práce je formálně členěna do čtyř částí, kde první část se věnuje zájmové oblasti, jejím parametrům a postupu vytváření geometrie této oblasti. Druhá kapitola je zaměřena na formáty souborů, které jsou využívány programem Flow123D. Další část se věnuje vzhledu aplikace, jejím datovým strukturám použitým k uložení načtených informací a metodám jejich zpracovaní. V poslední části jsou popsány možnosti využití knihovny DirectX respektive Direct3D pro zobrazení sítě zájmové oblasti.

1 Příprava geometrie zadané oblasti

Aby bylo možné generovat vstupní data konfigurovatelné sítě, je nutné nejdříve tuto síť vytvořit. Prvním úkolem se tedy stalo vytvoření geometrie zájmové oblasti, podle které lze posléze programem GMSH vygenerovat požadovanou síť.

K vytvoření geometrie oblasti je zapotřebí mít k dispozici určitá základní data o zájmové oblasti. Těmito daty jsou myšleny informace např. o umístění hraničních bodů oblasti, její členění dle výskytu hornin, případně rozložení tektonických zlomů.

Tato data mohou být poskytována ve dvou základních formách, a to grafické, nebo datové. Mezi data, která byla o oblasti poskytnuta pouze v grafické podobě patří například mapa oblasti generovaná programem GIS (geografický informační systém) a profily zájmové oblasti. Profily oblasti byly vytvořeny strukturním geologem právě za účelem vytvoření 3D obrazu zájmové oblasti. V datové podobě byly poskytnuty informace o hranicích zvodní. Zvodeň je hydraulicky jednotná a souvislá akumulace gravitačních podzemních vod v hornině.

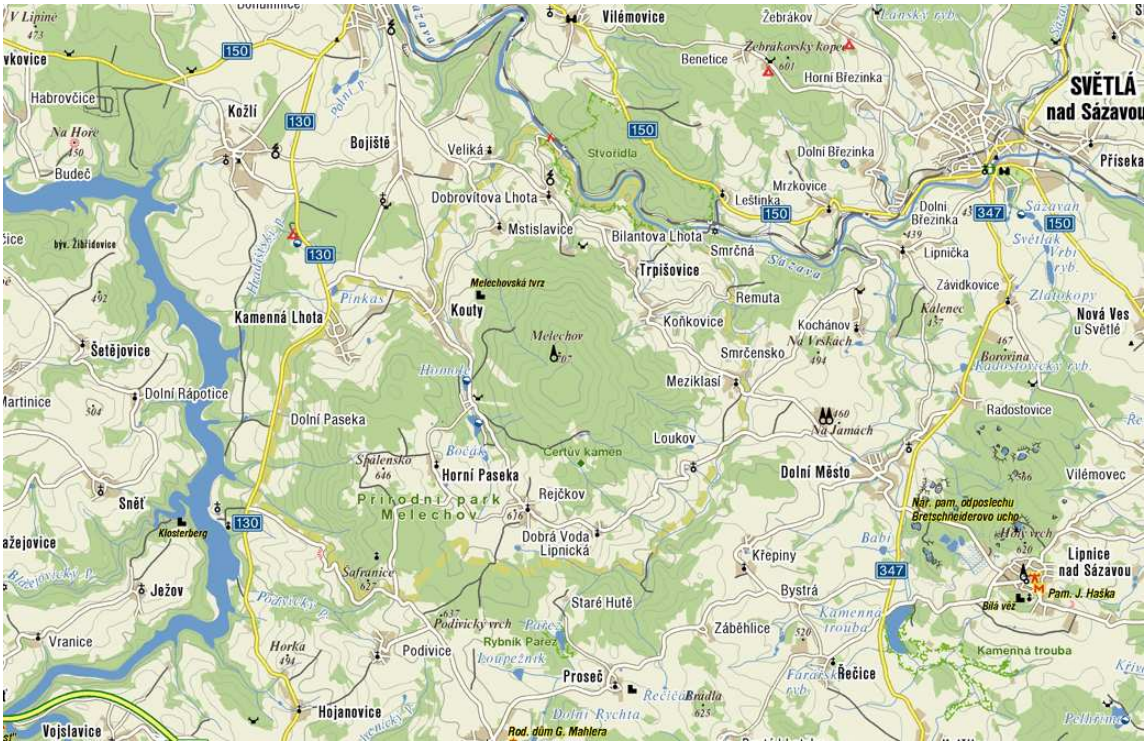
Po prostudování poskytnutých dat lze přistoupit k vytvoření geometrie sítě. Geometrie byla vytvořena pomocí programu GMSH. GMSH je nekomerční generátor FEM sítí s integrovaným CAD rozhraním. Princip FEM sítí spočívá v diskretizaci spojitého kontinua do určitého (konečného) počtu prvků, přičemž zjišťované parametry jsou určovány v jednotlivých uzlových bodech.

Program GMSH je navržen jako pomůcka pro vytváření sítí akademického typu s parametrickým vstupem a pokročilými vizualizačními schopnostmi. Skládá se ze čtyř základních modulů: *geometry*, *mesh*, *solver* a *post-processing*. Výhodou tohoto programu je, že pod podmínkou zachování GNU GPL licence ho lze využívat zcela zdarma. Tento program je možné volně stáhnout na Internetu a to například zde [1].

1.1 Modelovaná oblast

Nejprve bylo nutné zvolit oblast, v jejímž rámci se bude vytvářet geometrie, ze které bude posléze generována 3D síť. Jako zájmová oblast byla zvolena část okolí vrcholu Melechov. Oblast se rozkládá přibližně na rozloze 67 km^2 a je součástí Českomoravské vrchoviny. Kóta vrcholu se nalézá ve výšce 707,7 m a tvoří jasnou dominantu celého okolí. Tak jak je vidět na obr. 1, poblíž této oblasti leží město Světlá

nad Sázavou, nebo také Lipnice nad Sázavou. Jedná se o relativně členitý terén, který tímto komplikuje modelování povrchových částí sítě.

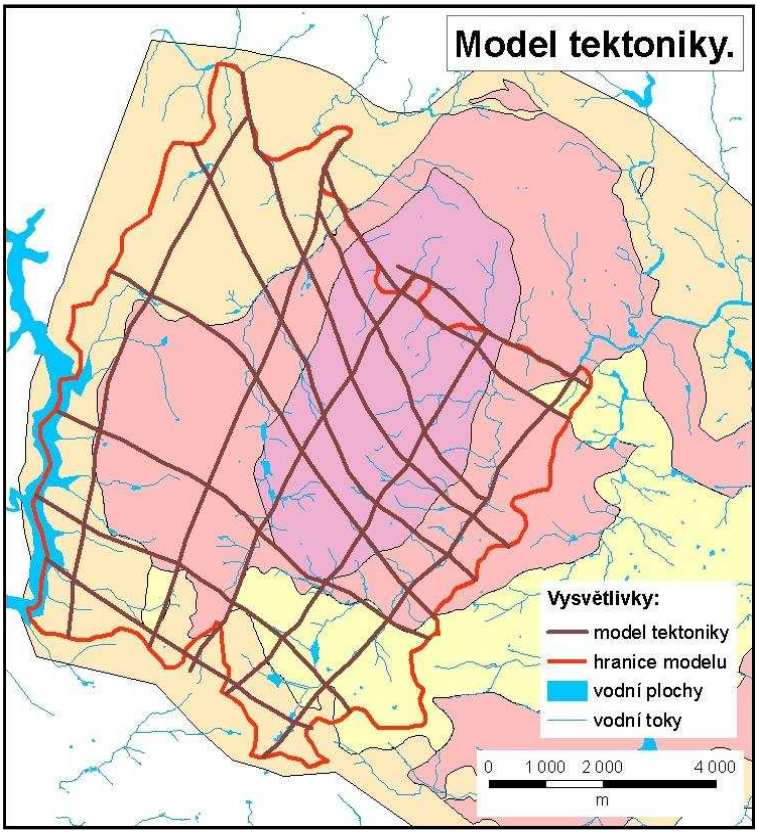


Obr. 1: Mapa okolí vrcholu Melechov

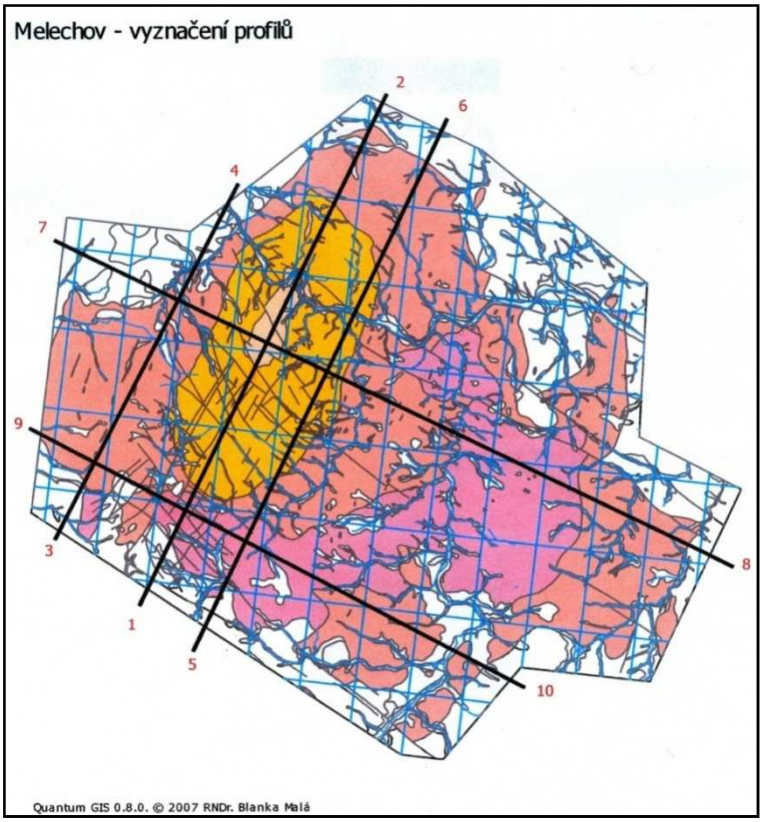
Hranice oblasti byly zvoleny na základě návrhu hydrogeologů tak, aby byly tvořeny zejména rozvodnicemi (rozvodnice ohraničují povodí podpovrchových vod), částečně propojené hydroliniemi řek Sázavy a Želivky.

Vymezení oblasti lze vidět na obr. 2, kde je tato hraniční linie označena červenou barvou. Na obr. 2 a obr. 3 lze vyčíst informace o rozložení čtyř dominantních hornin, které se v zájmové oblasti nacházejí. Jednotlivé horniny jsou od sebe barevně odlišeny. Na obr. 3 je navíc vidět rozmístění příčných řezů vedoucích zájmovou oblastí. Ukázky těchto dvou příčných řezů (profilů) vytvořených strukturním geologem lze vidět na obr. 4. Ostatní profily viz Příloha A.

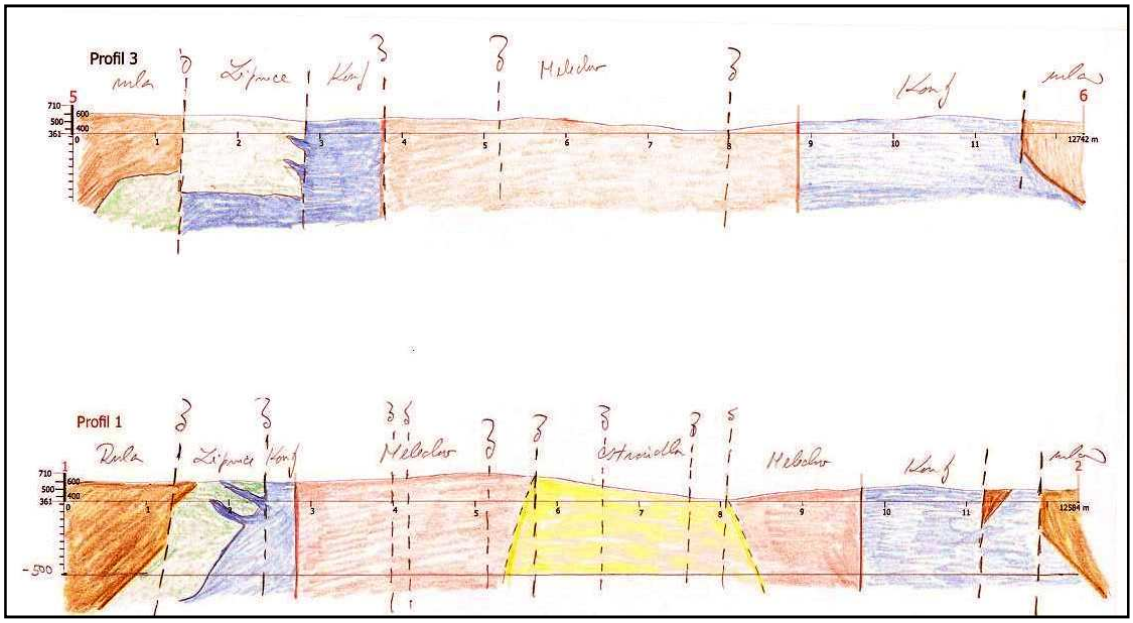
V modelovaném území se také nacházejí hydrogeologicky významné zlomy, které jsou na obr. 2 vyznačeny hnědou barvou.



Obr. 2: Model tektoniky a hranice zájmové oblasti



Obr. 3: Vyznačení profilů zájmové oblasti



Obr. 4: Ukázka profilů zájmové oblasti

1.2 Postup vytváření geometrie sítě

K vytvoření geometrie sítě lze využít dvě metody. Jako první se nabízí vytváření jednotlivých elementů ruční editací souboru v textovém tvaru, tzn. vytvoření každé entity zapsáním do textového souboru. Tato metoda se běžně využívá spíše k odstraňování problémů, nebo drobným korekcím.

Druhou, častěji využívanou metodou je využití interaktivního grafického rozhraní programu GMSH. Díky integraci *geometry* modulu do programu GMSH lze vytvářet geometrii sítě graficky pomocí uživatelsky přívětivého menu a tahů myši. Tento modul poskytuje jednoduché CAD rozhraní vycházející z okrajové definice entity. Využívá se při tom hierarchické stavby geometrie, kde výchozím prvkem je bod. Z bodů lze definovat přímky, z přímek plochy. Pomocí hraničních ploch lze následně vytvářet objemy. Tímto způsobem definujeme základní (elementární) entity. Základní entity ještě mohou být následně seskupovány do tzv. fyzických entit.

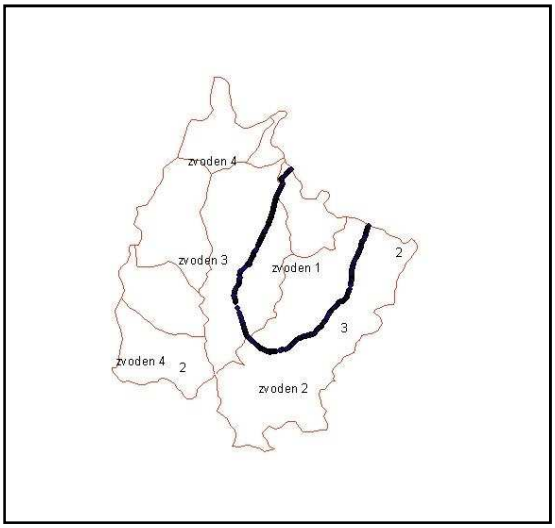
Vytváření obou typů entit se vyznačuje jedním společným znakem, tím je unikátní číslo, které musí být entitám přiřazeno při vytváření. To znamená, že každý bod musí mít své unikátní číslo, každá přímka musí mít své unikátní číslo atd.

Dříve zmíněné menu umožňuje nejenom entity vytvářet, ale případně i zvolené entity odstraňovat. Subjekty lze přidávat či odstraňovat selekcí vybraných prvků v návrhovém prostředí.

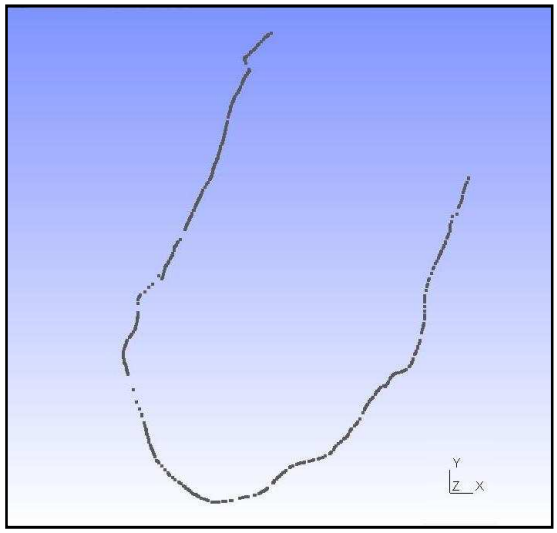
Samotné vytvoření geometrie lze shrnout do několika následujících kroků:

- Návrh první horizontální 2D vrstvy pomocí dat vyexportovaných z GIS
- Doplnění 2D geometrie o významné tektonické zlomy
- Podle příčných řezů a potřeb simulace navrhnou počet horizontálních vrstev sítě případně doplnit vrstvy, které nelze vytvořit pomocí první horizontální vrstvy
- Doplnění geometrie o plochy a objemy, pokud tak nebylo prováděno průběžně
- Dle příčných řezů a povrchové geologické mapy oblasti doplnit geometrii o fyzické objemy a plochy

Pro vytváření první horizontální vrstvy byla využita data pro tento účel vyexportovaná z GIS a upravená pro použití v GMSH. Tato data byla ve formě hraničních bodů (jejich 2D souřadnic) jednotlivých zvodní. Nešlo je využít přímo, ale nejdříve je bylo nutné upravit. Vzorová situace je znázorněna na obr. 5 a 6.



Obr. 5: Hranice zvodně 1 zobrazené v GIS



Obr. 6: Hranice zvodně 1 zobrazené v GMSH

Takto získané body jsou následně redukovány tak, aby hranice nebyla již od okamžiku vytváření geometrie členěna na velký počet krátkých přímek. Dalším krokem bylo propojení zbylých bodů liniovými elementy, čímž byly získány hranice dané zvodně. Vzhledem k tomu, že tato konkrétní oblast se nachází na hranici zájmové oblasti pro nadefinování plochy, bylo nutné doplnit zvodeň o okrajové body zájmové oblasti.

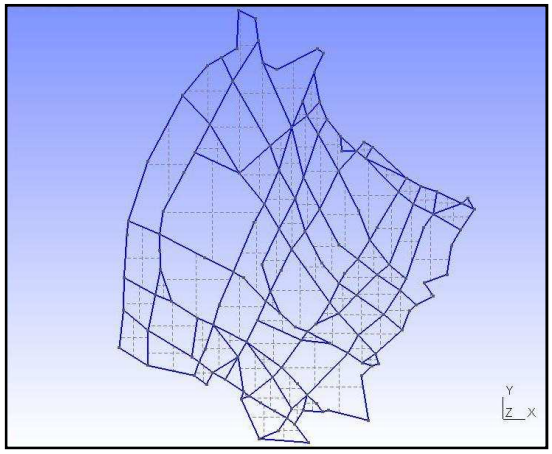
1.3 Popis geometrie sítě

Jak už bylo v předchozím textu řečeno, hranice a jednotlivé členění první horizontální vrstvy bylo vytvořeno pomocí dat exportovaných z programu GIS. Zároveň s tím byly do této vrstvy zakomponovány významné tektonické linie tak, jak je lze vidět na obr. 2 a obr. 7. Tímto způsobem vzniklá geometrie byla ještě doplněna o základní plochy, čímž vznikla kompletní 2D geometrie tak, jak ji lze vidět na obr. 7.

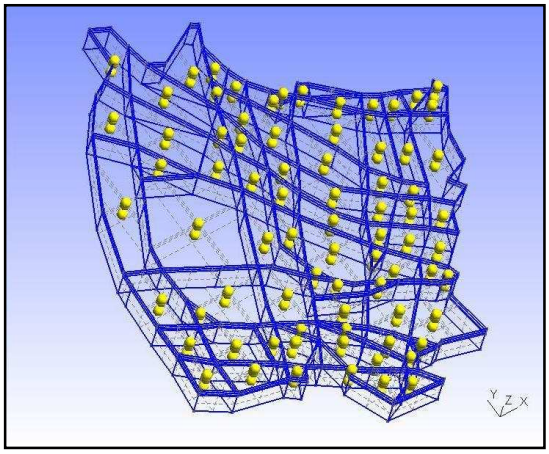
Z příčných řezů oblasti byla vybrána zájmová oblast ve výškové hladině od 361 metrů nad mořem do hloubky 639 metrů pod hladinu moře.

V oblasti bylo provedeno několik výzkumných vrtů. Při těchto vrtech bylo zjištěno, že se zde nacházejí dvě významné subhorizontální poruchy. Tyto dvě subhorizontální poruchy jsou do geometrie zakomponovány jako další horizontální vrstvy ve výškových hladinách 261 m. n. m. a 161 m. n. m. Protože GMSH používá vlastní skriptovací jazyk, který umožňuje využívat tzv. proměnných, lze provádět některé úpravy souřadnic změnou hodnoty právě těchto proměnných. Geometrie byla vytvořena tak, aby bylo možné měnit umístění (Z souřadnice) jednotlivých vrstev samostatně (proměnné *Z1* až *Z4*), nebo lze měnit výškové umístění celé geometrie (proměnná *ZB*).

Vertikální tektonické zlomy a subhorizontální poruchy jsou v geometrii reprezentovány 2D elementy (fyzickými plochami) se samostatně konfigurovatelnými materiálovými vlastnostmi.



Obr. 7: První horizontální vrstva s tek. zlomy



Obr. 8: Kompletní geometrie sítě

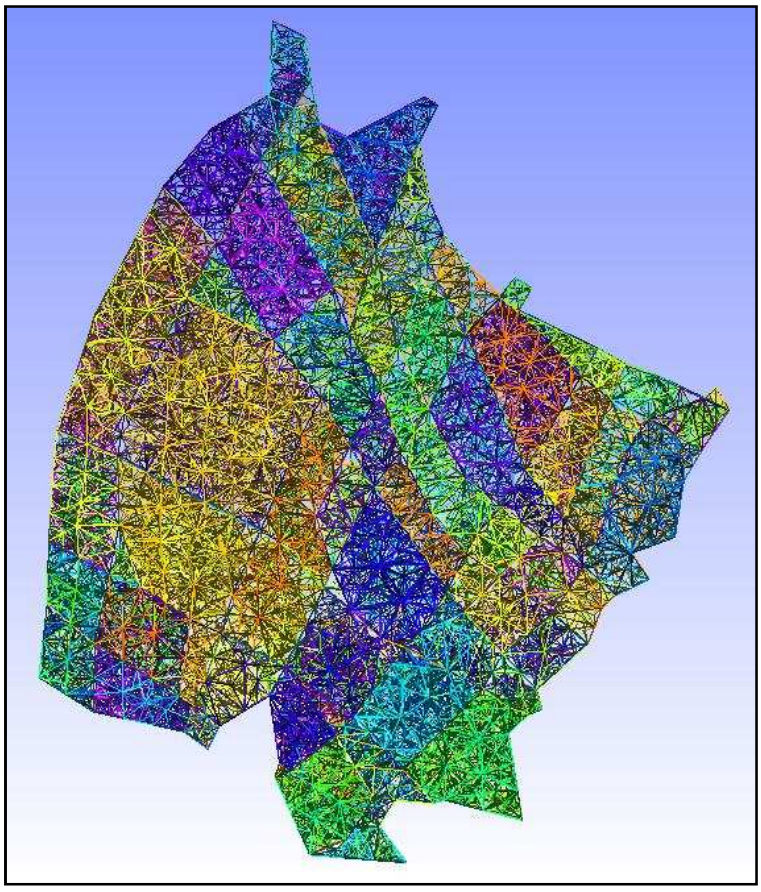
Při vytváření jednotlivých základních i fyzických entit dochází k jejich automatickému číslování. Toto číslování lze buďto ponechat, nebo číslování pozměnit

ruční úpravou textového souboru. U takto rozsáhlých sítí je vhodné pozměnit číslování alespoň fyzických entit tak, aby bylo možné rychle určit, kde se tato fyzická entita nachází. Za předpokladu, že daná síť bude ještě dále upravována, případně rozšiřována, je zvláštní číslování dobrý předpoklad pro rychlou orientaci a tudíž i práci se sítí.

Vzhledem k tomu, že byla požadována 3D síť, obsahuje geometrie zájmové oblasti nejen horizontální ale i vertikální plochy, které jsou následně seskupeny do základních objemů, tak jak lze vidět na obr. 8. Pomocí hraničních ploch jsou následně definovány také objemy. Vzdálenost jednotlivých vrstev včetně „mezivrstvy“ lze libovolně měnit. Ačkoliv je základními strukturami pokryta celá geometrie sítě, fyzické objemy jsou využity pouze v místech, která byla nutná pokrýt sítí.

1.4 Generování sítě

Pod pojmem generování sítě si lze představit rozklad ploch, objemů atd. na požadované elementy. Po vygenerování sítě programem GMSH, je každému elementu přiřazeno číslo elementární a fyzické entity, ke které patří.



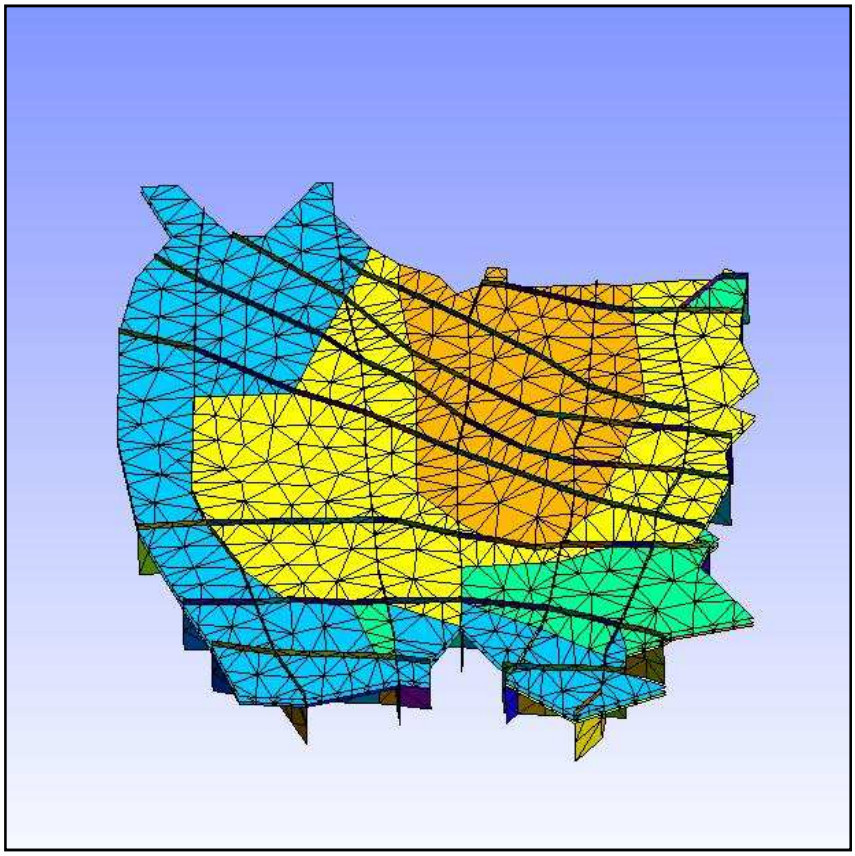
Obr. 9: Základní vygenerovaná síť

Jak už bylo řečeno, samotné generování sítě dané oblasti se provádí algoritmem, který je obsažen v *mesh* modulu programu GMSH. Podrobnost vytvářené sítě můžeme ovlivnit změnou tzv. charakteristické délky. Charakteristická délka nemusí být uvedena (v tom případě je použita přednastavená hodnota), nebo ji lze uvést například jako čtvrtý parametr při vytváření bodů geometrie sítě. Velikost prvků se pak vypočítá lineární interpolací uvedených charakteristických délek. Bližší informace o algoritmu výpočtu velikosti jednotlivých elementu jsou uvedeny v dokumentaci [1]. V této práci je využita možnost definice charakteristické délky v rámci definic bodů. Její hodnota je pro všechny body rovna proměnné *Len_bas* umístěné v textovém souboru geometrie sítě.

GMSH umožňuje také provést takzvanou optimalizaci generované sítě. Tu provedeme v případě, že chceme, aby měly všechny elementy pokud možno stejnou délku hraničních úseček. Optimalizací také většinou odstraníme elementy s nulovým objemem, které se zde po vygenerování mohou vyskytnout. Pro generování sítě jsou využívány 1D, 2D a 3D elementy. Konkrétně v případě Flow123D se jedná o přímku, trojúhelník, a čtyřstěn.

GMSH provádí generování elementů v celé oblasti geometrie. Do souboru ale uloží pouze elementy, které jsou zahrnuty v některé z fyzických entit. Při generování sítě docházelo k některým problémům. Jedním z nich byl i problém neuložení některých elementů do výsledné sítě, ačkoliv tento element spadal do jedné z definovaných fyzických entit. Tento problém byl v krajním případě řešen úpravou původní geometrie se zachováním požadovaných struktur.

Program GMSH dokáže zobrazit síť celou, nebo jednotlivé její části. Této vlastnosti se využívá převážně při kontrole vygenerované sítě. Při kontrole sítě se také využívá schopnosti GMSH barevně odlišit například fyzické entity, elementární entity atd.



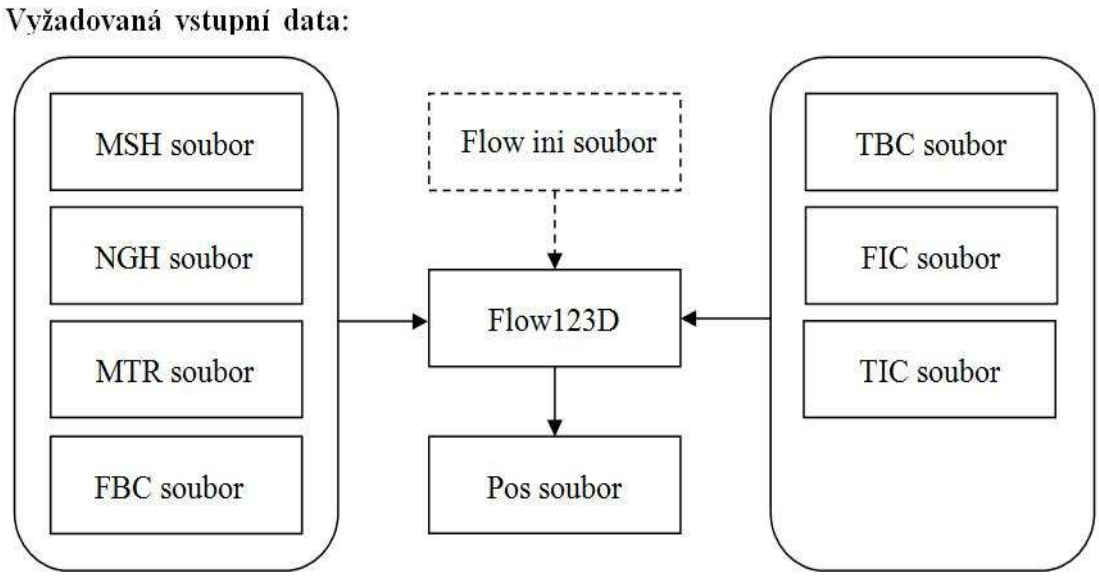
Obr. 10: Ukázka zobrazení pouze fyzických ploch sítě

Na obrázku č. 10 jsou zobrazeny fyzické plochy (v místě tektonických zlomů a horizontálních puklin) pokryté sítí 2D elementů. Jak je vidět, v některých částech se tyto zlomy shodují s hranicemi jednotlivých zvodní.

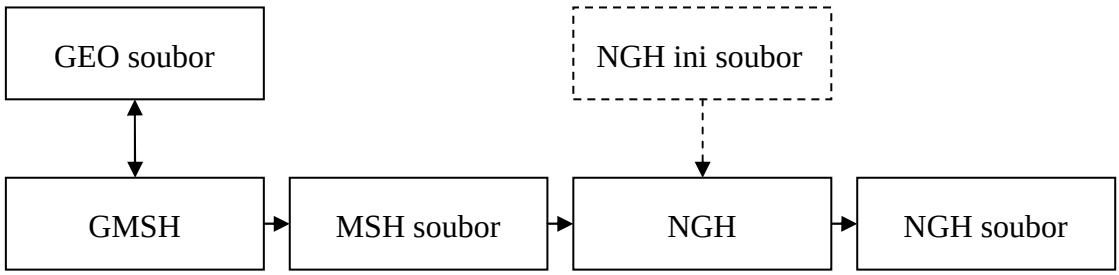
2 Formát datových souborů

Datové soubory jsou dány specifikací v dokumentaci Flow123D [2]. Pro výpočet proudění jsou nutné čtyři typy souborů. Těmito soubory jsou: soubor sítě s běžně používanou příponou msh, soubor topologie sítě (zaznamenané sousednosti elementů) s běžně používanou příponou ngh, dále pak soubor fyzikálních vlastností (běžně používaná přípona mtr) a soubor okrajových podmínek proudění obvykle s příponou bcd. Na výstupu potom program ukládá výsledky do souboru pos (GMSH), nebo do obecného textového souboru. Používání uvedených přípon není povinné, ale bývá to dobrým zvykem.

Na obrázcích 11 a 12 lze shlédnout schéma používaných vstupních a výstupních souborů programu Flow123D. Toto schéma je převzato z dokumentace programu Flow123D [2].



Obr. 11: Schéma práce programu Flow123D s datovými soubory



Obr. 12: Schéma generování souboru sousedností

Protože jedním z cílů této práce bylo vytvoření aplikace za účelem generování vstupních dat modelu, lze jeho soubory z pohledu aplikace rozdělit na vstupní a výstupní. Jako vstupní jsou používány soubor sítě a soubor topologie sítě, za výstupní lze z pohledu aplikace chápat mtr a bcd soubor.

2.1 Vstupní soubory

Vstupními soubory aplikace jsou dva soubory v textovém tvaru. Tyto soubory tvoří technické minimum pro generování výstupních souborů a práci se sítí obecně.

2.1.1 MSH soubor

Formát tohoto souboru vychází z GMSH systému. Msh formát ve verzi 2.0 je rozdělen do tří oddílů, které vymezují formát souboru. Jednotlivé oddíly jsou uzavřeny mezi příznaky počátku a konce bloku:

- **$MeshFormat - $EndMeshFormat** – Hlavička souboru
- **$Nodes - $EndNodes** – Informace o bodech (uzlech)
- **$Elements - $EndElements** – Informace o elementech

$MeshFormat - $EndMeshFormat:

Mezi tyto dva příznaky jsou vloženy pouze tři parametry. Prvním parametrem je verze souboru, v současné době je tento soubor ve verzi 2.0. Druhým parametrem je způsob uložení dat v souboru. V této práci se využívají pouze textová data ve formátu ASCII. Tato hodnota je celočíselného typu a pro ASCII formát je rovna nule. Třetím a posledním parametrem je hodnota rovnající se velikosti (v bytech) čísel s plovoucí

desetinnou čárkou. V současné době je podporována pouze hodnota odpovídající *data = sizeof(double)* tedy 8.

Příklad struktury hlavičky souboru:

```
$MeshFormat
2 0 8
$EndMeshForma
```

$Nodes - $EndNodes:

První informací uvedenou mezi těmito příznaky je počet bodů, které jsou zde zapsány. Na následujících řádcích jsou uvedeny informace o každém jednotlivém bodu, kde první číslo určuje pořadí (index) uzlu. Vzhledem k tomu, že se jedná o trojrozměrný systém, následují souřadnice na ose X, Y a Z. Jednotlivé uzly nemusejí být uvedeny tak, jak vyjadřuje jejich číslování, ale jejich pořadí muže být libovolně pozměněno.

Příklad struktury bloku vrcholů:

```
$Nodes
2
1 1 0 0
index souřadnice_X souřadnice_Y souřadnice_Z
$EndNodes
```

$Elements - $EndElements:

Obdobně jako u bloku vrcholů je zde nejdříve uveden počet elementů. Následují informace o elementech, z kterých se síť skládá. Informace jsou uloženy obdobně jako vrcholy. Rozdíl je v tom, že se jedná o proměnný počet parametrů, tudíž nelze tyto informace načítat jako blok dat, ale po částech. Struktura uložených informací je zhruba takováto: první hodnota značí index elementu, následuje typ elementu a počet využitých tagů. Následující položky jsou závislé od typu elementu a počtu tagů.

Dle uvedeného počtu tagů následuje počet hodnot odpovídající tomuto číslu. V současné době se využívají 1 až 3 tagy. Poslední hodnoty, které lze u definice elementu nalézt, jsou vrcholy, ze kterých se tento element skládá. Msh soubor v současné době podporuje 11 typů elementů identifikovatelných dle čísla a to:

1 Přímka (2 vrcholy)

2 Trojúhelník (3 vrcholy)

3	Čtyřúhelník (4 vrcholy)
4	Čtyřstěn (4 vrcholy)
5	Šestistěn (8 vrcholů)
6	Hranol (6 vrcholů)
7	Jehlan (5 vrcholů)
8	Přímka druhého řádu (3 vrcholy)
9	Trojúhelník druhého řádu (6 vrcholů)
11	Čtyřúhelník druhého řádu (10 vrcholů)
15	Bod (1 vrchol)

V současné době je Flow123D schopen provádět výpočty pouze s elementy typu přímka, trojúhelník a čtyřstěn. Jakékoliv jiné elementy vedou k ukončení výpočtu. Podrobnější Informace o jednotlivých elementech jsou uvedeny v [1] a [2].

2.1.2 NGH soubor

Tento soubor je obdobně jako msh soubor rozdělen do dvou částí. Tyto části, jsou umístěny mezi příznaky:

- **$NeighbourFormat - $EndNeighbourFormat** – Hlavička souboru
- **$Neighbours - $EndNeighbours** – Informace o sousednostech

$NeighbourFormat - $EndNeighbourFormat:

Jak už bylo řečeno, struktura ngh souboru je obdobná jako u souboru msh, proto jsou mezi tyto příznaky vloženy rovněž pouze tři parametry. Prvním parametrem je verze souboru. Dle verze souboru lze posléze dohledat, jak jsou uložená data strukturována. V současné době je používána verze 1.0. Druhý a třetí parametr je naprosto stejný jako u msh souboru.

$Neighbours - $EndNeighbours:

Prvním parametrem tohoto bloku je počet sousedností, které jsou v souboru uloženy. Tím je umožněno měnit pořadí vzhledem k číslování sousedností.

Na následujících řádcích jsou uloženy postupně index sousednosti a typ sousednosti. Podle typu sousednosti je dána i struktura uložených dat. Aktuálně lze sousednosti rozdělit na čtyři druhy:

10 – vrcholová sousednost
11 – hranová, nebo stěnová sousednost
20 – kompatibilní sousednost, element s elementem o jednu dimenzi nižší
30 – nekompatibilní sousednost, společná část objemů

Dle typu sousednosti, případně přidruženého parametru počtu sousedností, lze odvodit množství dat, které bude následovat. Více informací o způsobu uložení jednotlivých sousedností lze nalézt v uživatelském manuálu programu Flow123D [2].

Příklad struktury bloku sousedností:

```
$Neighbours
2
0   11 2 1 0 3 1
1   20 56 47 3 1
$EndNeighbours
```

2.2 Výstupní soubory

Implementovaná aplikace má umožnit generovat soubory, a to soubor fyzikálních vlastností a soubor okrajových podmínek proudění.

2.2.1 MTR soubor

Účel mtr souboru je uložení informací o fyzikálních vlastnostech materiálů. Materiálem se myslí určitá část prostředí, která má stejné fyzikální vlastnosti. Každý materiál je jasně určen svým jedinečným číslem. Pro každý materiál jsou v souboru zaznamenané jednotlivé hodnoty, kterým odpovídá daná fyzikální veličina. Struktura tohoto souboru je postavena na stejném principu, jako ostatní soubory. Data jsou rozčleněna do osmi bloků:

- **$MaterialFormat - $EndMaterialFormat** – Hlavička souboru
- **$Materials - $EndMaterials** - Určuje typ materiálu

- **$Storativity - $EndStorativity** – Udává hodnotu storativity
- **$Sorption - $EndSorption** – Udává sorpci materiálu
- **$DualPorosity - $EndDualPorosity** – Udává pórovitost materiálu
- **$SorptionFraction - $EndSorptionFraction** – Obsahuje fyzikální parametry sorbce
- **$Geometry - $EndGeometry –** Upřesňuje informace o 1D a 2D materiálech
- **$Density - $EndDensity** – Určuje hustotu materiálu

V souboru je pro každé číslo materiálu nutné zadat potřebné fyzikální vlastnosti. Flow123D jako číslo materiálu u daného elementu používá první tag tj. číslo fyzické entity přidělené při vytváření geometrie. Pro každou fyzickou entitu tedy musí být zaznamenány fyzikální parametry v souboru mtr.

Vzhledem k tomu, že struktura všech bloků je obdobná, není nutné zde popisovat každý blok zvlášť. Hodnoty dalších parametrů nebo postup jejich výpočtu lze pro každý blok dohledat v manuálu programu Flow123D [2].

Příklad struktury bloku materiálu:

```
$Materials
2
1000   21     0.0001
2000   21     0.0002
$EndMaterials
```

2.2.2 BCD soubor

V tomto souboru jsou uloženy hodnoty okrajových podmínek na jednotlivých okrajových stěnách sítě. Soubor je rozdělen stejně jako ngh soubor do dvou bloků:

- **$BoundaryFormat - $EndBoundaryFormat** – Hlavnička souboru
- **$BoundaryConditions - $EndBoundaryConditions** – Okrajové podmínky

$BoundaryFormat - $EndBoundaryFormat:

Těmito příznaky je ohraničena standardní hlavička datových souborů programu Flow123D. Hlavička obsahuje 3 základní parametry: verzi souboru, typ souboru

a velikost čísel (v bytech) s plovoucí desetinnou čárkou. Aktuální používaná verze souboru je 1.0.

$BoundaryConditions - $EndBoundaryConditions:

V tomto bloku jsou uloženy vlastní informace o okrajových stěnách, proto je tento blok poněkud složitější. Prvním parametrem je počet okrajových podmínek sítě. Na dalších řádcích jsou uvedeny konkrétní okrajové podmínky. První uvedenou hodnotou je číslo okrajové podmínky, následuje typ podmínky. Okrajové podmínky dělíme na 3 typy:

1 Dirichletova typu
2 Neumannova typu
3 Newtonova typu

Pro řešení úloh transportu látek diferenciálními rovnicemi se na hranicích zvodnělého prostředí zadávají tzv. okrajové podmínky hodnotami funkce, jejíž řešení hledáme. Protože hledaným řešením je obvykle tlak vyjádřený hladinou transportované látky, okrajová podmínka, která je udávaná piezometrickou výškou (hladinou) se nazývá okrajová podmínka 1. typu (Dirichletova). Okrajová podmínka zadaná průtokem transportované látky se nazývá okrajovou podmínkou 2. typu (Neumanova). Okrajová podmínka zadaná lineární kombinací podmínek 1. a 2. typu je podmínka 3. druhu, neboli Newtonova. Aby výpočty v programu Flow123D proběhly v pořádku, vyžaduje se, aby alespoň jedna z okrajových podmínek byla Dirichletova nebo Newtonova typu.

Typ	Data	Popis
1	Skalární_hodnota	Předepsané hodnoty tlaku nebo piez. výšky.
2	Proudění	Předepsané hodnoty toku přes hranice.
3	Skalární_hodnota σ	Skalární hodnoty a σ součinitel

Tab. 1: Typy okrajových podmínek

Jak lze vidět v Tab. 1, třetí uvedená informace je závislá na typu okrajové podmínky. Následuje způsob určení místa okrajové podmínky. Data o okrajové podmínce lze do souboru uložit třemi různými způsoby:

1 – Podmínka vrcholu

2 – Okrajová podmínka stěny

3 – Okrajová podmínka elementu, který je jednou stěnou na okraji oblasti

Způsob	Data	Popis
1	ID_Vrcholu	ID vrcholu, zapsaného v msh souboru
2	ID_elementu ID_stěny	ID Elementu a číslo jeho stěny
3	ID_elementu	ID Elementu

Tab. 2: Způsob zadávaní dalších dat o okrajové podmínce

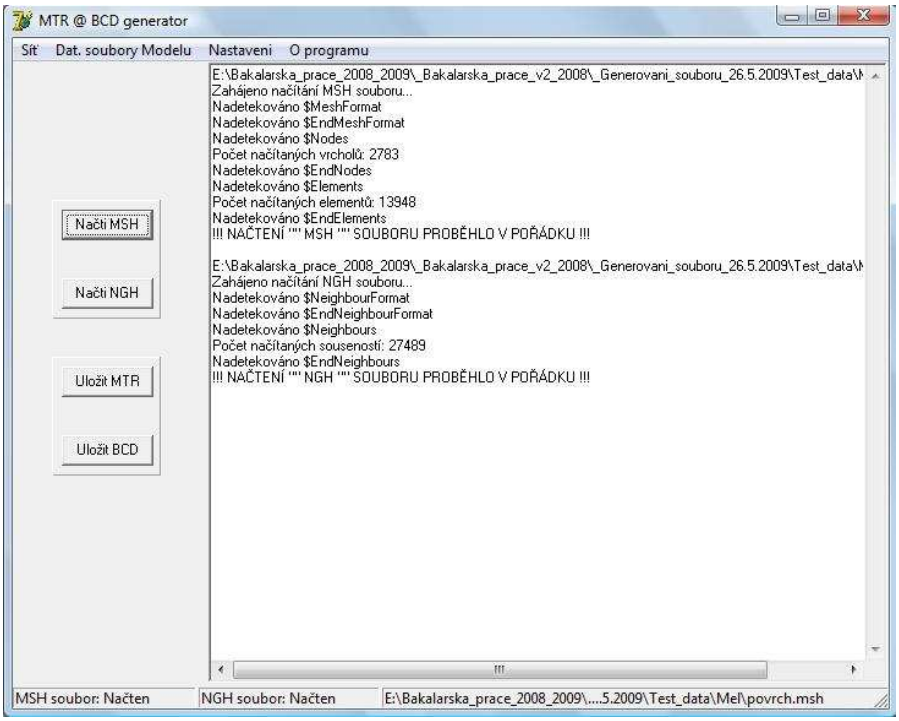
Podle způsobu určení okrajové podmínky následuji další data, tak jak jsou zobrazeni v tab. 2. Předposlední uvedenou hodnotou je počet tagů, které jsou u okrajové podmínky uvedeny. V současné době je využíván pouze jeden tag a ten označuje skupinu okrajových stěn.

3 Vytvořený program

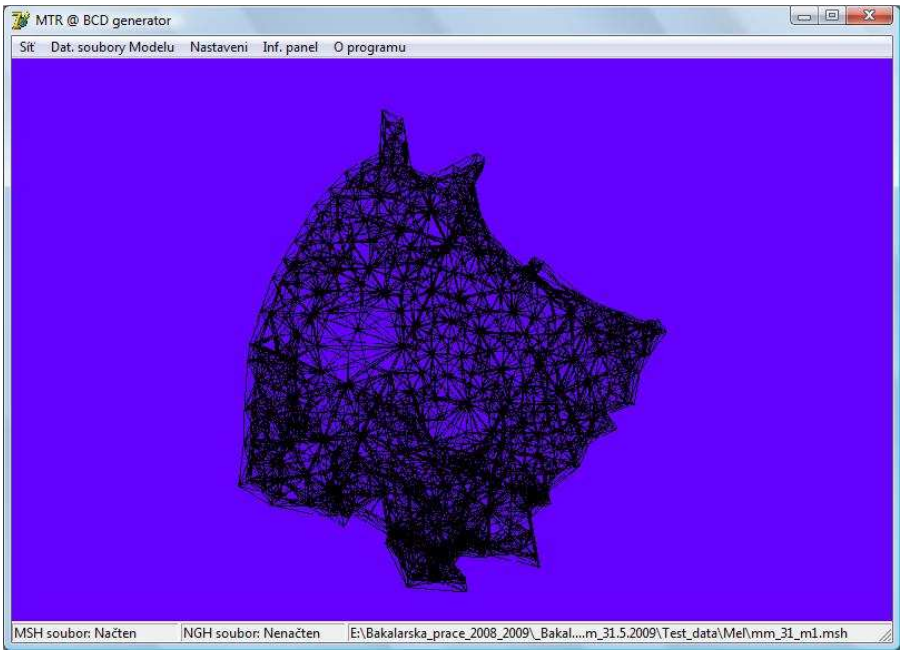
Jako vývojové prostředí využité k tvorbě programu bylo zvoleno Borland Delphi Studio 7. Vytvořeny byly dvě aplikace, první verze pouze zobrazuje informace o prováděných činnostech, druha dokáže zobrazit i právě zpracovávanou síť. Vnitřní strukturou ukládání dat a generováním výstupních souborů se jedná o identické aplikace.

3.1 Uživatelské rozhraní

Na obr. 13 a 14 lze vidět společné prvky uživatelského rozhraní, kterým je uživatelské menu, které slouží pro interakci uživatele s programem. Například k zpřístupnění možnosti vygenerovat mtr a bcd datové soubory dojde pouze tehdy, pokud je například u bcd souboru načten ngh i msh soubor. U mtr souboru stačí, aby byl načten pouze msh soubor.



Obr. 13: Uživatelské rozhraní bez DirectX



Obr. 14: Uživatelské rozhraní s DirectX

Společným prvkem je rovněž takzvaný Status Bar. Tento prvek slouží k informování uživatele o tom, u kterého souboru bylo provedeno načtení do vnitřních struktur programu, nebo z jakého adresáře byla načtena data o síti.

U programu, kde není formulář aplikace využit k zobrazení zpracovávané sítě, byla umístěna další čtyři tlačítka. Uvedená tlačítka umožňují zrychlit uživateli výběr akce, kterou chce provést. V pravé části je uživatel pomocí informačního panelu informován, které akce jsou aplikací právě prováděny. Zároveň je tato část využita k upozornění uživatele v případě, že došlo k nějaké chybě.

K hlášení chyb při načítání souboru je využita funkce s jedním přípustným argumentem. Tím je číslo chyby, které je posléze vyhodnoceno a této hodnotě odpovídající textový řetězec použit jako návratová hodnota. Takovýmto způsobem je zajištěna úprava varovných hlášení na jednom místě, a není tedy nutné prohledávat celý kód aplikace.

3.2 Použité vlastní datové struktury

Po prostudování struktury datových souborů využívaných programem Flow123D bylo zjištěno, že by bylo vhodné nadefinovat vlastní datové typy. Tyto typy je nutné vytvořit tak, aby umožňovaly přístup ke každé položce zvlášť, ale také, aby bylo možné dynamicky měnit počet ukládaných hodnot.

Výpis č. 1: Datový typ TNodes

```
type
  TNodes = record
    coordinateX: Single;
    coordinateY: Single;
    coordinateZ: Single;
    barva: LongWord;
  end;
```

Definice typu *TNodes*, uvedená ve výpisu č. 1, vychází z potřeb a principů DirectX. Vykreslování objektů pomocí bodů je samo o sobě složité, proto se využívá takzvaných vrcholů neboli vertexů. Vrcholy jsou vlastně zvláštní případ bodu. Mají taktéž souřadnice na ose X, Y, Z, ale na jejich souřadnicích dochází ke spojování hran útvaru. U tohoto typu je dále uvedena barva. Ta je uvedena z toho důvodu, že při definici takzvaného vertex bufferu (datová struktura pro uchovávání vertexů) musí DirectX znát další parametry vrcholu. Těmito vlastnostmi může být jako zde barva, nebo například informace o míchání. U programu, který DirectX knihovnu nevyužívá, je položka barva nepotřebná. V tom případě by bylo možné využít typ *TPoint*, který pouze neobsahuje barvu.

Dalším vytvořeným typem je typ *TElement*, uvedeným ve výpisu č. 2. Byl navrhnut tak, aby reflektoval variabilitu načítaných dat.

Výpis č. 2: Datový typ TElements

```
type
  TElements = record
    elementType: Byte;
    numberOfTags: Byte;
    tagList: array of Integer;
    nodeList: array of Integer;
    indexList: array of integer;
    indexBuffer: IDirect3DIndexBuffer9;
  end;
```

Tomu odpovídají tři dynamická pole, kde v poli *tagList* jsou uloženy jednotlivé hodnoty tagů, tak jak byly načteny z msh souboru. Stejným způsobem jsou data uložena

v poli *nodeList*. U tohoto typu je zvláštním případem dynamické pole *indexList*, které slouží zobrazovací části DirectX. Z tohoto pole jsou kopírována data do index bufferu, respektive za jeho pomoci je vytvořen objekt *indexBuffer* typu *IDirect3DIndexBuffer9.* Vice v oddílu DirectX.

Posledním datovým typem, který je zde využit pro „dlouhodobější“ uložení dat je typ *TNeighbours*. Ten je uveden ve výpise č. 3.

Výpis č. 3: Datový typ TNeighbours

```
type
  TNeighbours = record
    typ: Byte;
    sousednosti: array of integer;
  end;
```

Jedná se o nepříliš složitou datovou strukturu, kde typ označuje druh okrajové podmínky, a v dynamickém poli *sousednosti* jsou následně uložena data o sousednostech, tzn. po dvojicích uloženy elementy a jejich stěny.

Ačkoliv jsou vždy využívána dynamická pole, ze struktur načítaných souborů je předem známo množství načítaných dat. Tím je umožněno pole nastavit nejdříve na požadovanou velikost a až poté provést zápis informací.

3.3 Metodika načítání dat ze souboru

Za předpokladu, že bude síť modelované oblasti vygenerována programem GMSH, by bylo vhodné, aby se tyto dvě aplikace chovaly vzhledem k načítání souboru obdobně. Bylo provedeno několik testů chování programu GMSH při načítání dat ze souboru msh. Těmito testy bylo zjištěno hned několik zajímavých poznatků.

První získanou informací se stala základní podmínka zahájení sekvence načítání. Tato posloupnost příkazů je zahájena pouze tehdy, pokud je na prvním řádku uveden příznak *$MeshFormat*. Pokud tam tento příznak počátku hlavičky není, je uživateli nahlášen *syntax error ($).* Tato podmínka zahájení načítání dat ze souboru je v této aplikaci ponechána.

Druhým příkladem získaného poznatku je, že pokud je v souboru uveden počáteční příkaz typu *$MeshFormat* je jedno, jaký textový řetězec ve stejném řádku

následuje. Program pravděpodobně očekává řetězec o určité délce, který vyhodnocuje z načteného řádku.

Další informace vytěžená z testování se týkala nevyžadování ukončovacího příznaku daného bloku. Tím je myšleno, že pokud je v souboru umístěn příznak začátku bloku a informace, které má blok obsahovat, jsou v pořádku načteny, není při chybějícím příznaku konce bloku nahlášena chyba.

V případě msh souboru je vždy po příznaku začátku bloku hlavičky na dalším řádku očekávána informace o verzi souboru atd. Ostatní nekonzistentní informace samozřejmě vedou k ukončení načítání. Za nekonzistentní informace jsou považovány například textové řetězce nebo znaky mezi číselnými hodnotami.

Při zkoumání těchto vlastností byla nalezena vlastnost, která na rozdíl od vynechávání prázdných řádku nebyla ponechána. Touto vlastností je přerušení načítání elementů, jestliže je mezi nimi nalezen řádek s textovým řetězcem. Ačkoliv je načítání přerušeno, není o tom uživatel informován a zobrazeny jsou pouze ty elementy, které se nacházejí před inkriminovaným textovým řetězcem.

Ukázka dat, které lze načíst:

```
$MeshFormat
2 0 8
Prázdný řádek
$EndMeshFormat
Prázdný řádek
$Nodes
2783
Prázdný řádek
1 2067.9  2000.36 475
Prázdný řádek
3 2074.46 0.37    475
```

Načítací algoritmus, uvedený ve výpisu č. 4, striktně vyžaduje, aby v souboru byly umístěny jak příznaky začátku bloků, tak ukončení bloku. Pokud je v neukončeném bloku umístěn příznak začátku jiného bloku, je ignorován, respektive považován za běžný textový řetězec.

Díky podobné stavbě obou načítaných souborů je stavba algoritmu obdobná pro všechny bloky. Mění se pouze informace, které jsou načítány mezi příznaky.

Výpis č. 4: Ukázka algoritmu načítání hlavičky msh souboru

```
AssignFile(F, cesta);
Reset(F);
Readln(F, radek);
slovo := Copy(radek,0,11);
if slovo = '$MeshFormat' then
  begin
   {$IOChecks OFF}
   Readln(F, v1, v2, v3);
   {$IOChecks ON}
   IOError := IOResult;
  if (IOError<>0) or (v1<>2) or (v2<>0) or (sizeof(double)<>v3) then
    error :=  TRUE;
   while (not EoF(F)) and (not error) do
     begin
      Readln(F, radek);
      slovo := Copy(radek, 0, 14);
      if slovo = '$EndMeshFormat' then
         break;
     end;
  end;
```

3.4 Generování mtr souboru

Při generování mtr souboru se vychází z jeho struktury definované v manuálu programu Flow123D. V něm je uvedena jeho základní struktura, dle které se řídí i zapisování hodnot do souboru. Před vytvořením vlastního souboru je nutné zjistit, jaké fyzické entity se v síti nacházejí. To je provedeno pomocí procedury *ZjistiFyzickéObjemy*. Kód této procedury je uveden ve výpisu č. 5.

Výpis č. 5: Kód procedury ZjistiFyzickéObejmy

```
procedure TForm1.ZjistiFyzickeObjemy(var fyzObjemy: TPole;
                                     var fyzObjNizsRadu: TPole);
      var
        I, tagObjemu: integer;
      begin
      for I := 1 to high(poleElementu) do
       begin
        tagObjemu:= poleElementu[I].tagList[0];
        ZapisPokudNeobsahuje(fyzObjemy, tagObjemu);
        if (poleElementu[I].elementType = 1) or
             (poleElementu[I].elementType = 2) then
          begin
           tagObjemu:= poleElementu[I].tagList[0];
           ZapisPokudNeobsahuje(fyzObjNizsRadu, tagObjemu);
          end;
       end;
      end;
```

V této proceduře dochází k prohledávání pole, ve kterém jsou uloženy informace o elementech. Z tohoto pole je vždy využit pouze tag, který je v poli tagů (*tagList*) na nulté pozici a udává příslušnost elementu k fyzické entitě.

Pokud číslo fyzické entity ještě není obsaženo ve *fyzObjNizsRadu* , dojde k rozšíření tohoto pole a zapsání hodnoty na poslední pozici. Pro úplnost lze dodat, že typ *TPole* je definován jako pole proměnných typu *integer*.

Výsledné pole fyzických objemů je poté setříděno a jednotlivé položky zapsány do souboru s požadovanou strukturou jednotlivých bloků (tzn. dle předem daných parametrů vytvořena základní struktura souboru). Tato část se provede obdobně u všech bloků kromě bloku *$Geometry*.

Tento blok je používán pro doplnění parametrů u 1D a 2D materiálů, od toho se odvíjí struktura procedury používané k vyhledání materiálů tohoto typu. Protože se 1D elementy ve vytvořené síti nepředpokládají, není nutné v proceduře zahrnout vyhledávání elementů tohoto typy.

V předchozí proceduře je ve stejném cyklu, jako u vyhledávání všech fyzických entit, prováděna kontrola typu elementu, pokud je nalezen 2D prvek je provedeno testování, zda je obsažen v poli *fyzObjNizsRadu*. Pokud ne, je pole opět rozšířeno a hodnota zapsána na vytvořenou pozici.

Před samotným výpisem jsou tímto způsobem uložené hodnoty v poli setříděny a zapsány s požadovanou strukturou přidružených dat. Ukázka zápisu jednoho z bloků je uvedena ve výpisu č. 6.

Výpis č. 6: Ukázka zápisu jednoho bloku mtr souboru

```
AssignFile(F, cesta);
Rewrite(F);
.  .  .  .  .  .
WriteLn(F,'$Geometry');
for I:= Low(pole2DObjemu) to High(pole2DObjemu) do
  begin
   writeln(F, IntToStr(pole2DObjemu[I]) +#9+ ‘2‘ +#9+‘0.01‘);
  end;
WriteLn(F,'$EndGeometry');
.  .  .  .  .  .
CloseFile(F);
```

3.5 Generování bcd souboru

Při vytváření bcd souboru se vychází ze struktury souboru, která je popsána v kapitole 2 Formáty datových souborů. Aby bylo možné tento soubor vygenerovat, je nutné nejdříve zjistit, které elementy (jejich stěny) se nacházejí na okraji sítě. K tomu slouží procedury *ZjistiOkrajiveElementy* a *ZapisPokudOkraj*.

Aby bylo možné zjistit, zda element na některé ze stěn nemá souseda, je nejdříve nutné provést transformaci dat, která jsou uložena ve struktuře *poleSousednosti*. Tomuto se věnuje procedura *ZjistiOkrajiveElementy*, jejímž úkolem je postupně prohledat tuto strukturu a zapsat do pomocné proměnné (*sousedeElementu*, jedná se o dvourozměrné pole celočíselných hodnot) informace o sousedech elementu. Rozdílem možných sousedů elementu a počtu skutečných sousedů lze následně postupně zjistit celkový počet okrajových podmínek. Ukázku části kódu takovéto procedury lze vidět ve výpisu č. 7.

Výpis č. 7: Ukázka hlavní části kódu procedury ZjistiOkrajovéElementy

```
doc := length(poleElementu);
SetLength(sousedeElementu, doc);
for I := low(poleSousednosti) to high(poleSousednosti) do
  begin
    if (poleSousednosti[I].typ = 11) then
      begin
       doc:= Length(poleSousednosti[I].sousednosti);
       doc:= doc div 2;
       for Y := 0 to (doc - 1) do
        begin
         indexElementu:= poleSousednosti[I].Sousednosti[Y*2];
         indexSteny:= poleSousednosti[I].Sousednosti[Y*2+1];
         pom:= Length(sousedeElementu[indexElementu]);
         SetLength(sousedeElementu[indexElementu], pom+1);
         sousedeElementu[indexElementu, pom]:= indexSteny;
        end;
      end;
    if (poleSousednosti[I].typ = 20) then
      begin
       indexElementu:= poleSousednosti[I].Sousednosti[1];
       indexSteny:= poleSousednosti[I].Sousednosti[2];
       pom:= Length(sousedeElementu[indexElementu]);
       SetLength(sousedeElementu[indexElementu], pom+1);
       sousedeElementu[indexElementu, pom]:= indexSteny;
      end;
  end;
pom:=0;
for I:= 1 to high(sousedeElementu) do
  begin
    case poleElementu[I].elementType of
       1: doc:= 2;
       2: doc:= 3;
       4: doc:= 4;
    end;
   pom:= pom + (doc - length(sousedeElementu[I]));
  end;
WriteLn(F, IntToStr(pom));
end;
```

Vzniklá datová struktura je předána jako atribut další jmenované proceduře, kde se provádí její vyhodnocení pomocí lokální proměnné *blnPole* typu *array of boolean*. Velikost tohoto pole je vždy nastavena podle maximálního počtu možných sousedů vyhodnocovaného elementu.

Parametrem procedury je datová struktura, kde index řádku odpovídá číslu elementu, a hodnoty uložené ve sloupcích jsou stěny, u kterých byl zjištěný soused. Jak již bylo řečeno, hodnoty ve sloupcích odpovídají stěnám, které mají souseda. Tyto hodnoty jsou použity jako indexy do proměnné *blnPole*, kde je uložena hodnot TRUE, pokud daná stěna má souseda, jinak je zde ponechána hodnota FALSE.

Takto vzniklé pole je využito při samotném výpisu, kdy se provede prohledání pole, zda obsahuje hodnotu FALSE. Pokud je tato hodnota nalezena, je pro zapsání okrajové podmínky využit index elementu (reprezentováno krokem cyklu), dále hodnota proměnné *doc* (reprezentující pořadí okrajové podmínky) a index v poli nalezené hodnoty FALSE (určuje okrajovou stěnu elementu). Tato procedura je znázorněna ve výpisu č. 8.

Výpis č. 8: Procedura ZapišPokudOkraj

```
procedure TForm1.ZapisPokudOkraj(var soubor: textfile;
                                       pole:array of TBoundary);
var
  I, Y, doc: integer;
  blnPole: array of boolean;
begin
doc:= -1;
for I:= 1 to high(pole) do
  begin
   case poleElementu[I].elementType of
     1: SetLength(blnPole, 2);
     2: SetLength(blnPole, 3);
     4: SetLength(blnPole, 4);
   end;
   for Y:= low(blnPole) to high(blnPole) do
       blnPole[Y]:= FALSE;
   for Y:= low(pole[I]) to high(pole[I]) do
     blnPole[pole[I,Y]]:= TRUE;
   for Y:= low(blnPole) to high(blnPole) do
    begin
     if (blnPole[Y] = FALSE) then
      begin
       inc(doc);
       Writeln(soubor,IntTostr(doc)+#9+'1'+#9+FloatToStr(VypTlaku(I,Y))+
              #9+'2'+#9+IntToStr(I)+#9+IntToStr(Y)+#9+'1'+#9+'1');
      end;
    end;
  end;
end;
```

Procedura *VypTlaku()* provede výpočet tlaku každé okrajové podmínky, podle informací zadaných v nastavení programu. Data nutná pro výpočet tlaku jsou: souřadnice referenčního bodu, tlak v tomto bodě a gradient - tlakový spád.

4 DirectX

DirectX je rozhraní mezi hardwarem a softwarem, které vytvořila firma Microsoft pro operační systémy Windows. Je to název skupiny knihoven vytvořených nejen pro tvorbu her a softwaru sloužícího k modelování 3D světa, ale také je lze využít pro přehrávání videa a hudby atd. DirectX se skládá z následujících částí: *DirectX Graphics* (a jeho části *DirectDraw* a *Direct3D***)**, *DirectInput, DirectPlay, DirectSound, DirectMusic*, *DirectShow*, *DirectSetup* a *DirectX Media Objects*. Toto rozdělení je důležité hlavně pro programátora, protože z hlediska uživatele se DirectX šíří jako jeden instalační balík obsahující všechny tyto komponenty. Každá z uvedených komponent má svoji specifickou funkci. V této práci je využita k zobrazování jednotlivých elementů pouze knihovna Direct3D, která tvoří jádro pro programování softwaru ve 3D prostoru.

V době začátku psaní této práce byla sice běžně dostupná zařízení využívající DirectX 10 (10.1). Tato verze bohužel není dostupná pod Windows XP (a nižší), proto byla využita pro programování grafické části aplikace starší verze a to DirectX 9.0c.

Vzhledem k tomu, že programovací prostředí Borland Delphi 7 neposkytuje programátorovi přístup k funkcím grafické knihovny DirectX resp. Direct3D, byly použity unity, které je možné stáhnout na Internetu. Tyto unity jsou dostupné například zde [7]. Následně je nutné unity integrovat do vývojového prostředí.

Pokud v systému nejsou obsaženy dynamické knihovny d3dx9_31.dll a DXErr9ab.dll je nutné pro plnou funkčnost grafické části aplikace tyto knihovny doplnit. Dynamické knihovny lze rovněž stáhnout na Internetových stránkách [7]. Na přiloženém CD jsou k dispozici nejen potřebné unity, ale i dynamické knihovny.

Při používání jednotlivých funkcí byla jako zdroj informací využita diplomová práce [3], která byla vytvořena ve stylu příručky programátora. V následujícím textu se tedy budeme věnovat pouze využití knihovny Direct3D jako prostředku k zobrazené sítě zájmové oblasti.

4.1 Postup vytváření programu pomocí Direct3D

Základním předpokladem programování DirectX je inicializace DirectX. Vytváření objektu zařízení a samotného zařízení Direct3D jsou rutiny, kterým se programátor nevyhne. Tato část se v aplikacích provádí stále stejným způsobem, proto pro naše potřeby postačí zmínit, že se tato operace provádí pomocí procedury *InicializaceD3D.* Popis jednotlivých částí lze nalézt zde [3].

Následuje deklarace procedury za účelem vykreslení nejprve pozadí aplikace a následně i objektů, které definujeme. Ve vytvořené aplikaci tuto část provádí procedura *VykresleníD3D*. Uvnitř této procedury je inicializována ještě další procedura a to *InicializaceMatic*, která slouží k transformaci scény.

Po načtení souboru msh jsou zavolány procedury *VytvoreniVertexBufferu* a *NacteniIB.* Tyto procedury slouží k načtení informací v případě vertex bufferu o vrcholech (souřadnice na ose X, Y, Z a barva) a u index bufferu o indexech vrcholu, které budou využity pro vykreslování primitiv. Problematice vytváření těchto procedur se také věnuje [3], proto jsou zde uvedeny pouze názvy deklarovaných procedur.

4.2 Vykreslování objektů

Vykreslování tvarů (polygonů) může být prostřednictvím Direct3D prováděno v zásadě dvěma metodami: *DrawPrimitive()* a *DrawIndexedPrimitive()*.

Pokud bychom využili *DrawPrimitive*, je nutné definovat typ dat ve vertex bufferu a index vertexu, od kterého se má začít vykreslovat primitiva a počet primitiv, které mají být vykresleny. Z toho vyplývá, že pokud bychom chtěli využít jeden vrchol znovu, je nutné ho do vertex bufferu umístit opakovaně. Tato metoda vzhledem k počtu vrcholů sítě (v řádu tisíců) není vhodným řešením.

Proto v našem případě využijeme metody *DrawIndexedPrimitive()*. U té je nutné, kromě výše uvedených tří bodů, nastavit ještě zdroj indexů (metodou *SetIndices()*). Metoda *SetIndices()* má jako jediný parametr použitý index buffer. V této aplikaci je vytvořen index buffer pro každý objekt samostatně. Princip kreslení pomocí indexování je obdobný jako bez indexování. Nejdříve je nutné definovat nastavení transformací (reprezentované procedurou *InicializaceMatic*). Dalším krokem je

nastavení zdroje vertexů, ze kterého budou data načítána a formát vertexů použitých k vykreslení (reprezentováno metodou *SetFVF()*). Pokud jsou všechny tyto úkony provedeny, lze začít vykreslovat samotné primitivy.

Každý objekt má definován vlastní index buffer. Z toho důvodu je nutné provést nastavení zdroje indexů pro každý objekt samostatně a až poté přistoupit k vlastnímu vykreslení. Procedura použitá k vykreslení jednotlivých elementů je uvedena ve výpisu č. 9.

Výpis č. 9: procedura VykresleniD3D

```
procedure TForm1.VykresleniD3D(var vrcholy: array of TNodes;
var g_pVertexBuffer: IDirect3DVertexBuffer9; var pole: array of
Elements);
var
  I: integer;
begin
  if assigned(g_pZarizeniDirect3D) then
  with g_pZarizeniDirect3D do
   begin
    Clear(0, nil, D3DCLEAR_TARGET or D3DCLEAR_ZBUFFER,
          3DCOLOR_XRGB(cCervena, cZelena, cModra), 1.0,0);
    InicializaceMatic;
    SetStreamSource(0, g_pVertexBuffer, 0, sizeOf(TNodes));
    SetFVF(D3DFVF_CUSTOMVERTEX);
    BeginScene;
    for I:= 1 to high(pole) do
      begin
       if(pole[I].IndexBuffer = nil) then continue;
       SetIndices(pole[I].IndexBuffer);
       DrawIndexedPrimitive(D3DPT_LINELIST, 0, 0, length(vrcholy), 0,
                            (length(pole[I].indexList) div 2));
      end;
    EndScene;
    Present(nil, nil, 0, nil);
   end;
end;
```

Při vytváření aplikace byl nalezen problém, kdy si aplikace využívající Direct3D při vykreslování alokuje příliš mnoho paměti (u Windows Vista, u Windows XP není paměťová náročnost tak velká). Výsledkem je, že zobrazení nepracuje zcela správně pro sítě s větším počtem elementů. Úpravou kódu aplikace a testováním některých částí procedur vytvářející vertex a index buffer byli označeny tři pravděpodobné příčiny tohoto problému.

Prvním z nich je nevhodná kombinace parametrů uvedených při vytváření indexů. Druhým důvodem může být kopírování dat (index a vertex bufferu) do operační paměti, místo do paměti grafické karty. Z toho důvodu by při každém vykreslení následně docházelo k načítání dat z operační paměti do paměti grafické karty a ukládání

dalších blíže nespecifikovaných dat. Tím by se zároveň vysvětloval pomalejší chod aplikace, který by byl způsoben právě přesunem dat.

Jako poslední důvod se nabízí chyba v unitě, která poskytuje rozhraní do grafické knihovny Direct3D. Knihovna DirectX je totiž firmou Microsoft psána v jazyce C++ a přeprogramované unity nemusí být, stejně jako žádný program, úplně bez chyb.

Protože jsem testoval aplikaci s různými parametry vytvářeného index bufferu, jeví se jako nejpravděpodobnější druhý a třetí důvod.

Závěr

Cílem bakalářské práce bylo vytvoření geometrie zadané oblasti pomocí programu GMSH s důrazem na vhodnou volbu identifikace fyzických elementů a s ohledem na pozdější snadnou identifikaci umístění elementů v jednotlivých částech sítě. Tato část práce je psána jako stručný návod, jak postupovat při vytváření geometrie sítě, pokud máme k dispozici data o okrajových bodech zvodní a rozložení hornin v dané oblasti. Jedna z podkapitol se věnuje popisu geometrie sítě vytvořené v rámci této bakalářské práce, například umístění samostatně konfigurovatelných vertikálních a horizontálních puklin, nebo možnostem úpravy vytvořené geometrie.

Dále bylo úkolem pomocí programu GMSH a takto vytvořené geometrie sítě provést vygenerování sítě. V této části jsou nastíněny nejen možnosti ovlivnění hustoty elementů, na které je zájmová oblast rozdělena, ale také možnosti kontroly takto vytvořené sítě.

Posledním úkolem této bakalářské práce byl návrh aplikace pro konfiguraci sítě na základě požadavků uživatele, identifikaci okrajových stěn sítě a generování vstupních dat modelu. K tomu bylo nutné prostudovat uživatelský manuál programu Flow123D a GMSH, kde jsou popsány jednotlivé formáty souborů, které měly být využity vytvořenou aplikací.

Aby měl uživatel přehled, se kterým souborem sítě pracuje, bylo po konzultaci s vedoucí bakalářské práce rozhodnuto implementovat do programu zobrazovací část. Protože ne vždy je nutné síť zobrazovat, byly vytvořeny dvě aplikace. Jeden obsahující na formuláři aplikace informační panel a další ovládací prvky. Druhý program demonstruje využití Direct3D k zobrazení zpracovávané sítě.

Vypracování této bakalářské práce pro mne bylo přínosem, protože mi dala možnost nejenom získat informace o postupu vytváření geometrie sítě, ale i prakticky si tento postup vyzkoušet. V této práci jsem se dále zabýval zobrazovací části aplikace, která mi umožnila vytvořit si obrázek o základních principech programování 3D světa a vykreslovaní objektů.

Seznam ilustrací

Seznam použité literatury

[1] Uživatelský manuál generátorů sítí GMSH, http://www.geuz.org/gmsh/

[2] Uživatelský manuál simulačního nástroje Flow123D, Liberec (2008)

[3] Hlavatý Jaroslav, Programování Direct3D v programovacím prostředí Borland Delphi, Diplomová práce, Liberec (2007)

[4] Informační server o programování Builder, http://www.builder.cz/

[5] Knihovna MSDN od Microsoft, http://msdn2.microsoft.com/en-us/default.aspx

[6] Kadlec Václav, Delphi hotová řešení, Brno (2003), Computer Press, ISBM 80-251-0017-0, http://www.cpress.cz/

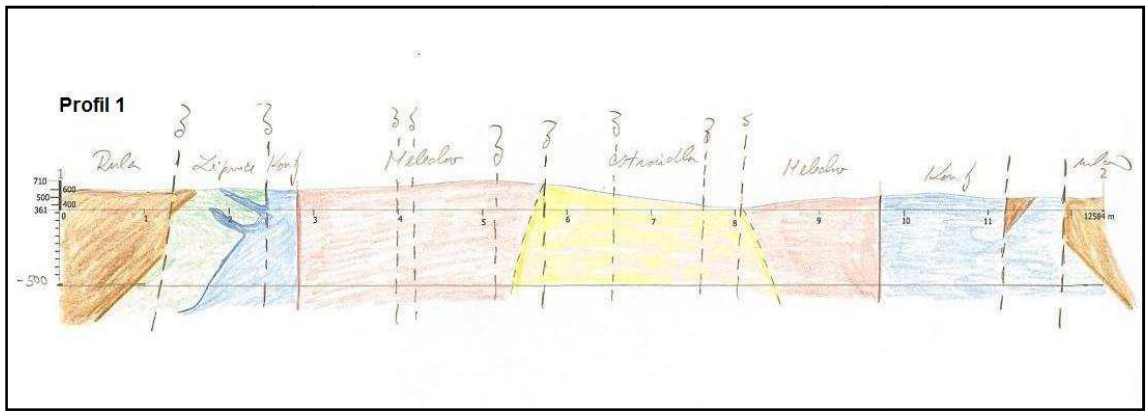
[7] BARKOVOY, A. Clootie graphics pages, http://www.clootie.ru/

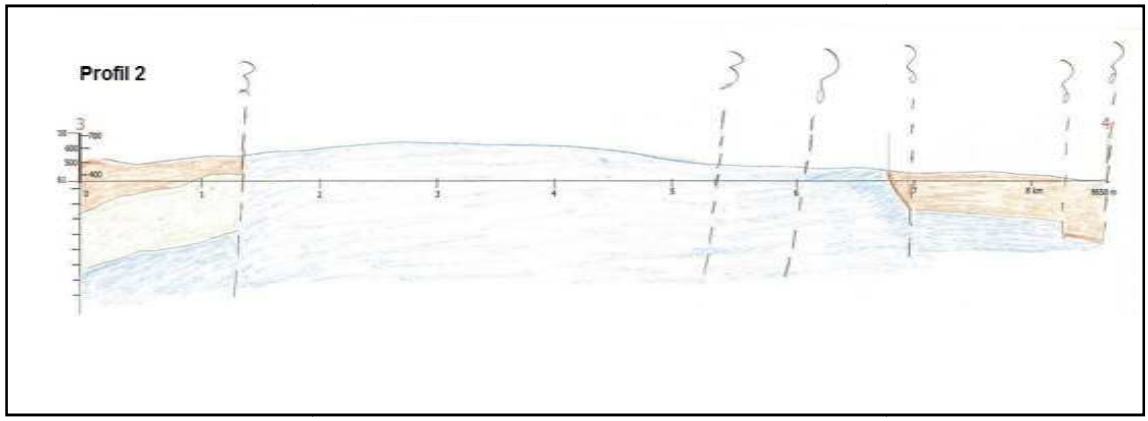
[8] Wikipedie, otevřená encyklopedie, http://cs.wikipedia.org/

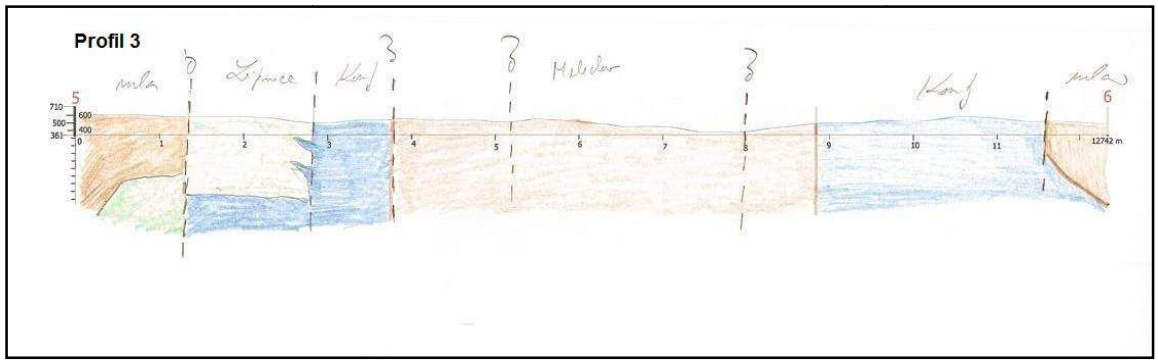
[9] Informační server o programování v Delphi, http://www.delphibasic.uk/

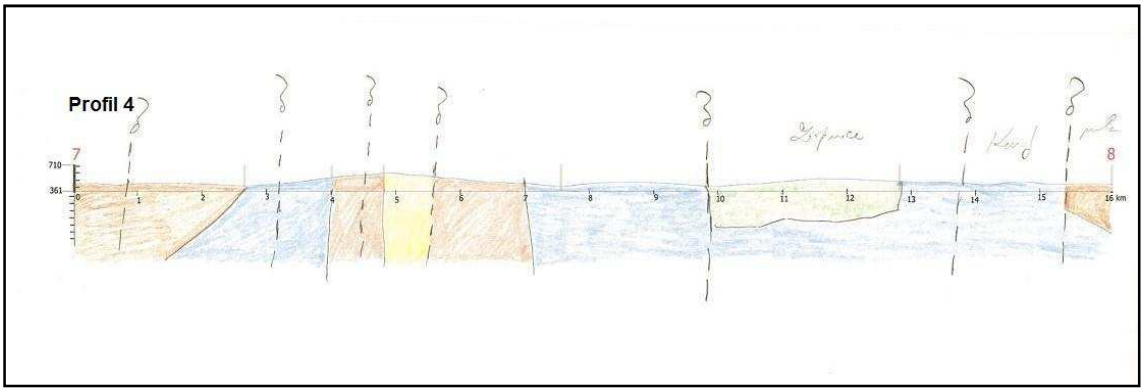
[10] Portál o plánování volného času, http://www.vyletnik.cz/

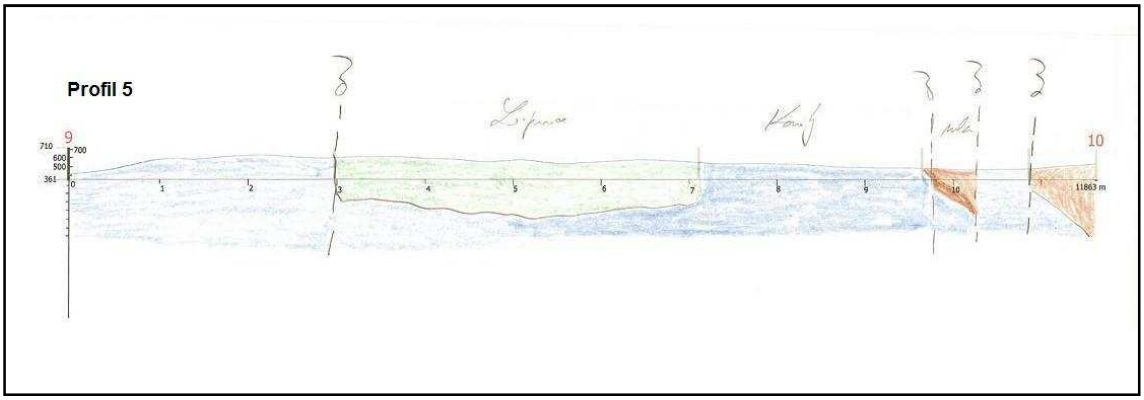
Příloha A Příčné řezy zájmovou oblastí








Profil 4
7
8
710
361
16 km


Profil 5
9
10
710
700
600
500
361
11863 m