VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

3D zpracování fotografie v rámci systému FOTOM^{NG}

3D Image Processing in System FOTOM^{NG}

2014 Jakub Hendrych

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Jakub Hendrych**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: 3D zpracování fotografie v rámci systému FOTOM NG
3D Image Processing in System FOTOM NG

Zásady pro vypracování:

Cílem diplomové práce je návrh a realizace modulu pro 3D modelování objektů pomocí rekonstrukce polygonální reprezentace sítě ze série snímků zájmových objektů. Součástí práce bude také vyhodnocování základních vlastností rekonstruované plochy (obsah pláště, těžiště, objem, plocha průřezu).

1. Seznamte se s problematikou počítačového zpracování fotografií a metod 3D modelování objektů na počítači.
2. Seznamte se s prostředním NetBeans a programovacím jazykem JAVA.
3. Seznamte s fotogrammetrickým systémem FOTOM verze 2008 - modul Fotom3 a systémem FOTOM-NG.
4. Navrhněte nový modul na 3D modelování objektů s možností základní analýzy objektů v rámci systému FOTOM-NG.
5. Implementujte navržený modul v systému FOTOM-NG a proveďte zhodnocení dosažených výsledků.
6. Vypracujte uživatelskou a programátorskou dokumentaci.

Seznam doporučené odborné literatury:

[1] Heiko Böck: NetBeans Podrobný Průvodce Programátora, 2010, Computer Press, Czech Republic, ISBN: 978-80-251-3116-9, 320 Pages
[2] Heiko Böck: The Definitive Guide to the NetBeans Platform 6.5, May 2009, Apress, USA ISBN: 978-1-4302-2417-4, 450 Pages
[3] Lačezar Ličev: Analýza, modelování, rozpoznávání a vizualizace procesu měření objektů na snímcích, 128 str., Knihy vydané prostřednictvím www.vydejteknihu.cz, Computer Press, a.s., ISBN 978-80-2513-296-8, EAN 9788025132968

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **doc. Ing. Lačezar Ličev, CSc.**

Datum zadání: 01.09.2013

Datum odevzdání: 07.05.2014

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 30. dubna 2014 ..... Jakub Rudzyk .....

Rád bych poděkoval vedoucímu mé diplomové práce, panu doc. Ing. Lačezaru Ličevovi, Csc., který mi poskytl cenné rady a připomínky v průběhu zpracování diplomové práce. Dále bych chtěl poděkovat mé rodině a přátelům, kteří mě podporovali v průběhu celého studia.

Abstrakt

V mé diplomové práci s názvem 3D zpracování fotografie v rámci systému FOTOM^{NG}, se zabývám implementací nového modulu na platformě NetBeans, který bude připojen do systému FOTOM^{NG}. Tento modul slouží k vytvoření 3D modelu pozorovaného objektu a disponuje velkou škálou funkcí pro manipulaci a analýzu. Vstupní data modulu jsou tvořena sérií snímků, například ultrazvuku krční tepny. Dále se v mé práci zmiňuji o měřící technice fotogrammetrii, teoretickým podkladem provázející 3D modelování a problematikou spojenou s procesem vývoje nového modulu. Samotná implementace je realizována v jazyce Java s využitím knihovny Java3D. Cílem je také navrhnout uživatelskou příručku a programátorskou dokumentaci.

Klíčová slova: 3D modelování, 3D animace, FOTOM^{NG}, fotogrammetrie, platforma NetBeans, Java, Java3D, modul, měřický snímek, transformace

Abstract

In my master thesis, titled 3D Image Processing in System FOTOM^{NG}, I deal with the implementation of a new module on the NetBeans platform, which will be attached into the FOTOM^{NG} system. This module serves to create a 3D model of the observed object and has a large range of functions for manipulation and analyzing. Input data of the module are consists of series of images, such as ultrasound of carotid artery. Next, in this thesis I writting about photogrammetric measurment technique, theoretical basis for 3D modeling and issues associated with the developing process of a new module. The implementation itself is realized in Java with using of the Java3D library. I will design user guide and programming documentation.

Keywords: 3D modeling, 3D animation, FOTOM^{NG}, photogrammetry, NetBeans platform, Java, Java3D, module, measurment images, transformation

Seznam použitých zkratek a symbolů

2D	–	Two Dimensions - dvojrozměrný
3D	–	Three Dimensions - trojrozmerný
API	–	Application Programming Interface
BMP	–	Bitmap image file
DOC	–	Formát textového dokumentu
DVD	–	Digital Versatile Disc
EMG	–	Electromyography
FEM	–	Finite Element Method
GIS	–	Geographic Information System
GUI	–	Graphic User Interface
IDE	–	Integrated Development Environment
JAI	–	Java Advanced Imaging
JDK	–	Java Development Kit
JPEG	–	Joint Photographic Experts Group
JRE	–	Java Runetime Environment
MRI	–	Magnetic Resonance Imaging
MVC	–	Model, View, Controller
NG	–	Next Generation
PDF	–	Portable Document Format
RCA	–	Rich Client Application
URL	–	Uniform Resource Locator
VRML	–	Virtual Reality Modeling Language
VVUÚ	–	Vědeckovýzkumný uhelný ústav
XML	–	Extensible Markup Language

Obsah

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Trendem současných vědních oborů je co nejrychleji a nejpřesněji analyzovat získaná data. Mezi typy takové analýzy můžeme považovat samotnou 3D vizualizaci. Díky vizualizaci měřeného objektu, získáme další perspektivu na zkoumaná data, než jsou například tabulky a grafy. Od roku 1997 byla pod vedením katedry informatiky na VŠB-TU Ostrava realizována myšlenka na vznik systému FOTOM. Tento systém zpracovával a následně analyzoval série snímků na základě vědního oboru důlní fotogrammetrie. Vývoj pokračoval přidáváním nových funkcí a schopností zpracovávat nejen důlní, ale i například lékařské snímky. Výsledkem se stal systém FOTOM 2008, který se skládal z několika částí, kde každá byla zaměřena na odlišnou analýzu a funkcionalitu. Postupem času bylo rozhodnuto, že je systém zastaralý, což dalo podnět k vytvoření nové verze. Byly shromážděny všechny zkušenosti a výhody předchozích verzí a začal vývoj systému FOTOM^{NG}. Velkou výhodou byla samotná implementace na platformě NetBeans kvůli jeho modularitě. Proto mohl být systém jednoduše rozvíjen o další moduly a funkce do podoby komplexního nástroje pro analýzu snímků.

V rámci mého diplomového projektu se zabývám vývojem nového modulu pro 3D modelování, který bude součástí systému FOTOM^{NG}. Tento nástroj bude schopen na základě série snímků s nadefinovanými objekty vytvořit 3D model pozorovaných objektů. Uživatel bude mít k dispozici celou řadu nástrojů pro manipulaci a především měření daného modelu. V rámci tohoto tvrzení, bych chtěl vyzdvihnout výhody, které tento modul bude zahrnovat. Jedná se například o zobrazení modelu ve čtyřech pohledech, které snímají scénu z různých úhlů. Tím se kompletně eliminuje problém tzv. „relativity úhlu pohledu“, který objasním později nebo znázorňování důležitých údajů přímo u 3D obrazu, jako jsou například obsah, měrné jednotky, úhel natočení, aj. Dále chci zmínit schopnost ihned znázornit grafy modelu pro různé typy dat. Obrovskou výhodou je pozorování grafu současně s 3D modelem a navíc nepotřebujeme pracovat s jiným komplexnějším modulem pro zobrazování grafů. Je nutné uvést i další užitečné funkce, jakou jsou třeba animace, projektový objekt, analýza odchylek, atd. Využiji také analýzy modulu předchozí verze, který je znám pod názvem Fotom3. Při návrhu a samotné implementaci budu klást velký důraz na vytvoření kvalitního nástroje, který odstraní možné nedostatky původního modulu a o případné doplnění dalších užitečných funkcí.

Celá diplomová práce bude rozdělena do různých částí, které popisují proces od teoretických základů nutných pro 3D modelování, až po samotnou implementaci nového modulu. Chci zmínit důležitý prvek první části, kterým bude krátký průzkum, jenž provedu v oblasti 3D modelování v počítačových systémech a jeho využití. Dále

se zaměřím na samotnou měřící techniku - fotogrammetrii. Také si objasním matematický základ a metodiku, spojenou s 3D vizualizací objektů na počítači. Další část bude věnována analýze modulu předchozí verze spolu s funkcemi systému FOTOM^{NG}. Tyto funkce je nutné zdůraznit, protože definují vstup pro vytvářený modul, který je reprezentovaný sérií snímků se zájmovými objekty. Dále vytvořím specifikaci požadavků, budu analyzovat knihovnu Java3D a platformu NetBeans. Poslední část pojedná o problematice spojené s implementací nového modulu. Závěr vývoje bude doprovázen otestováním celého modulu, včetně zhodnocení a vytvoření uživatelské příručky a programátorské dokumentace.

2 Výzkum 3D modelování ve světě a jeho využití

Díky svým výhodám, které jsem popsal v úvodní kapitole, je tato metoda analýzy stále více používaná v různých odvětvích vědních oborů. V této kapitole se věnuji výzkumům v oblasti 3D modelování ve světě v období od roku 2012 - 2014:

- Výzkum prezentuje technologii, která vytváří přesný 3D model distribuce tepla. To vede k odhalení míst, kde dochází k úniku nebo plýtvání energií. Hlavním účelem je úspora energie a případné úpravy budovy. Metoda je založena na laserovém skenování spolu s použitím termálních a optických kamer. Thermal 3D modeling of indoor environments for saving energy [1].

- Účelem této studie je vytvořit 3D biomechanický model objemového svalu s modifikovaným Hillovým modelem pro aktivní napětí z EMG nahrávek. Tím se výrazně zkracuje výpočetní čas a simulace vytváří kvalitní výsledek. 3D volumetric muscle modeling for real-time deformation analysis with FEM [2].

- Článek představuje interaktivní systém pro vytváření 3D modelů zájmových objektů. Odstraňuje nevýhody tvorby modelů, které většinou zahrnují pořizování snímků z různých pohledů. To vede k velké časové náročnosti a navíc nemohou být použity pro různé objekty. iModel: Interactive co-segmentation for object of interest 3D modeling [3].

- Tento dokument představuje vytvoření 3D geologického modelu uhelného dolu v systému GIS. Na základě výsledků byla vytvořena řada profesionálních aplikací, především pro zvýšení bezpečnosti. Three-dimension geological modeling and its application of Wangjialing coal mine [4].

- Popisuje metody využití 3D modelování ke tvorbě 3D map při vojenských operacích v městských oblastech s použitím kombinovaného výsadkového senzoru. Tento senzor produkuje rozsáhlý soubor různorodých dat. Combined airborne sensors in urban environment [5].

- Studie se zabývá, jak vygenerovat 3D model co nejjednodušším a pohodlným způsobem, aby na běžného uživatele nebyly kladeny pokročilé dovednosti. Jednou z možností je vygenerování modelu z 2D snímků nebo různých skic. Multi-scale feature matching between 2D image and 3D model [6].

- Přístup navrhuje dvě metody k modelování 3D scény s více objekty. Odstraňují faktory (komplexní osvětlení, odrazy a stíny), které ovlivňovaly stávající přístupy,

které byly určeny především pro modelování jednoho objektu. 3D modeling of multiple-object scenes from sets of images [7].

- Tento výzkum mobilní robotiky ukazuje možné způsoby, jak vylepšit analýzu scény. Tím se dosáhne přesnějšího rozpoznávání objektů. Architektura využívá různých 2D snímačů, na základě kterých se výsledek převádí do 3D modelu. Active scene analysis based on multi-sensor fusion and mixed reality on mobile systems [8].
- Představuje komplexní analýzu dopadu používáním běžné metody při rekonstrukci pacientovy srdeční geometrie. Vytvořené 3D modely z hrubých dat jsou následně aproximovány podle modelu difúzního tenzoru MRI, který je ve vysokém rozlišení. Accurate reconstruction of 3D cardiac geometry from coarsely-sliced MRI [9].

3 Fotogrammetrie

V této kapitole pojednávám o fotogrammetrii, jakož to měřicí technice. V celkovém měřítku je velice důležité seznámit se s tímto pojmem, jelikož vstupem nového modulu bude právě série fotogrammetrických snímků. V následujícím textu budu čerpat z literatury [10, 11].

Samotný pojem fotogrammetrie vznikl spojením tří řeckých slov: „photos“ světlo, „gramma“ nakresleno a „metron“ měřit. Fotogrammetrie, jakožto vědní obor, se zabývá zpracováním informací na fotografických snímcích. Jinými slovy: rekonstrukcí různých zájmových objektů pomocí geometrických tvarů, měřením jejich rozměrů nebo určováním jejich polohy. Jak už jsem napsal, samotná technika měření nezískává data přímým měřením objektů, ale měřením jejich fotografických snímků. Takový snímek je exaktním středovým průmětem fotografovaného předmětu. Následně potřebujeme odvodit geometrické vztahy mezi předmětem a jeho snímkem. K tomu slouží různé grafické a numerické metody nebo je můžeme odvodit mechanicky, pomocí speciálních přístrojů. Je důležité zdůraznit i velkou výhodu fotogrammetrie, která spočívá v možnosti, pořízené snímky uchovat (digitálně nebo fyzicky) a později celé měření opakovat.

V dnešní době nalezla fotogrammetrie velmi vysoké uplatnění v geodézii a kartografii při vytváření map a plánů, dále pro armádní účely při pozorování a vytváření taktických map nepřátelských objektů, v lékařství, v důlním průmyslu a v mnoha dalších odvětvích. Dokonce i u lidského zraku, kdy rozlišujeme velikosti a tvary momentálně pozorovaných objektů. V poslední době se do popředí dostává i snímkování pomoci tzv. bolidových kamer, které zaznamenávají průlety meteorů a bolidů Zemskou atmosférou. I tehdy je při vyhodnocování snímků použita fotogrammetrie. Moderní fotogrammetrie používá digitální snímky, které jsou následně zpracovány na počítači.

Velkou výhodou je využití této metody jako prostředek ke sledování průběhu a dokumentace. Na základě tohoto tvrzení uvedu dva konkrétní příklady, kdy byla využita fotogrammetrie. Prvním příkladem je diagnostika důlního díla při měření VVUÚ Ostrava – Radvanice, kde na základě série snímků budeme schopni pozorovat různé odchylky, které indikují deformace. Mohou se tak včas odhalit různá bezpečnostní rizika. Kompletní postup, včetně ilustrace a dalších příkladů, je uveden v [10]. Druhým příkladem je využití fotogrammetrie při ultrazvukovém vyšetření krční tepny. Výstupem většinou bývají video záznamy nebo samotná série snímků. V případě videa je nutné sestříhání a vytvoření požadované série. Vyšetření se provádí v důsledku onemocnění koronární aterosklerózou, což je usazování tukových tělísek. To má za následek zužování krční tepny, čímž se snižuje průtok krve do mozku. Pokud provedeme více takových vyšetření, budeme schopni určit, k jakým změnám došlo a podle toho pokračovat v další léčbě.

4 3D modelování

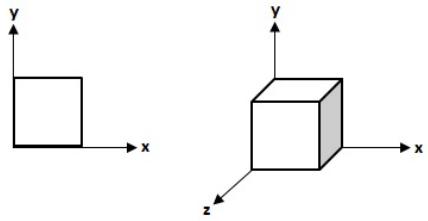
Tuto kapitolu věnuji teorii spojené s 3D modelováním. Cílem bude objasnit pojmy, které jsou důležité v pozdějším vývoji nového modulu. Následující text se opírá o informace v publikacích [10, 12].

4.1 Modelování v prostoru

Pokud chci modelovat objekt, potřebuji ho umístit do virtuálního prostoru. V tomto případě rozlišujeme dva typy prostorů, ve kterých lze provést onu vizualizaci objektu.

Prvním zástupcem je 2D prostor, který je popsán dvěma rozměry. Tedy obsahuje pouze souřadnicové osy x a y. Ve 2D můžeme vytvářet geometrické obrazce, jako jsou například úsečky, křivky, kružnice, elipsy, různé typy n-úhelníků, atd. Pro modelování ve 2D platí, že objekty nemají objem a pokud vytvoříme množinu všech bodů, vždy se budou nacházet v jedné rovině. Využití nalezneme při vytváření grafů, tvorbě technických výkresů, atd.

Druhým zástupce je 3D prostor. Ten, na rozdíl od 2D prostoru, přidává navíc třetí souřadnicovou osu z. Tím modelovaný objekt získá hloubku, tudíž můžeme začít určovat i objem. 3D prostor poskytne pozorovateli vytvářet objekty reálného světa, tak jak jsou vnímány lidským zrakem. Existuje několik způsobů jak modelovat objekty v 3D prostoru. Jedním ze způsobů je skládání základních 2D útvarů ve složitější 3D objekty. Například pokud chceme vytvořit krychli, použijeme k její konstrukci šest čtverců. V praxi se využívá polygonů, díky kterým jsme schopni modelovat i velmi složité objekty, jako jsou lidské tváře nebo různé členité tvary [12].



Obrázek 1: Rozdíl mezi 2D a 3D prostorem

4.2 Souřadnicová soustava

V předchozí kapitole 4.1 jsem zmínil pojem souřadnicové osy. Pokud mluvíme o 3D modelování, jedná se o tři navzájem kolmé orientované přímky (x, y, z), které se protínají

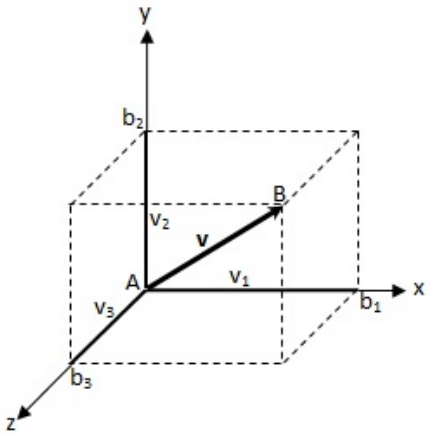
v jednom bodě. Tento bod je počátek souřadnicové soustavy. Taková soustava je také známá jako Kartézského souřadnicová soustava.

V praxi se setkáme se dvěma typy Kartézského souřadnicových soustav (dále jen KSS) ve 3D prostoru. Jedná se o pravotočivou a levotočivou soustavu. Rozdíl mezi těmito typy spočívá v orientaci jednotlivých os. K názornému vysvětlení se nejčastěji používá poučka levé a pravé ruky – palec je osa x, ukazováček je osa y a prostředníček je osa z (nezapomenout na kolmost). V případě, že použijeme levou ruku, jedná se o levotočivou KSS. Pokud použijeme pravou ruku, jedná se o pravotočivou KSS. V praxi může celá souřadnicová soustava různě rotovat, avšak musí být vždy zachována kolmost jednotlivých os. Většinou se setkáme spíše s pravotočivou KSS, ta je také použita na obrázku 1.

4.3 Objekty ve 3D prostoru

Z předchozích kapitol 4.1 a 4.2 už víme, co je to 3D prostor a jak se v něm orientovat. Dalším krokem bude vytvoření nějakého objektu. Důležitým předpokladem pro vytváření objektů ve 3D prostoru je ohodnocení souřadnicových os.

Ve 3D modelování máme možnost vytvářet různě složité obrazce od polotovarů až po výsledný 3D model, například automobil. Avšak každý model je složen z elementárních částí a to jsou bod a vektor. Proč zmiňuji pouze tyto dva objekty? Pokud se zaměříme na nějaké konkrétní API pro tvorbu 3D modelů zjistíme, že vždy používají tyto dva objekty. A z nich následně vytváří například polygony, atd.



Obrázek 2: Bod a vektor ve 3D prostoru

Na obrázku 2 můžeme pozorovat body A, B a vektor $\boldsymbol{v}$ ve 3D prostoru. Jak jsem již napsal dříve, většina API vyžaduje pro práci s body a vektory jejich souřadnice. Nejdřív

obecně určíme souřadnice jednotlivých bodů, tedy počáteční bod $A[a_1, a_2, a_3]$ (všechny souřadnice bodu jsou v počátku soustavy) a koncový bod $B[b_1, b_2, b_3]$. Nyní potřebujeme určit souřadnice vektoru $\boldsymbol{v}$, tedy obecně $\boldsymbol{v} = (v_1, v_2, v_3)$. Souřadnice v_1, v_2, v_3 dopočítáme jako rozdíl souřadnic koncového a počátečního bodu, tedy:

$v_1 = b_1 - a_1$,
$v_2 = b_2 - a_2$,
$v_3 = b_3 - a_3$.

4.4 Transformace

Pokud máme vytvořen nějaký objekt ve 3D prostoru, potřebujeme s ním i nějak manipulovat. Ať už z pohledu interakcí nebo pro různé výpočty, které jsou nutné pro správnou orientaci snímků v sérii vůči referenčnímu profilu. Výchozím neboli referenčním profilem nazýváme většinou první snímek série. V celém diplomovém projektu použiji pouze tři nejjednodušší transformace – otočení (rotaci), posun (translaci) a změnu měřítka (scale). Parametry pro tyto transformace si určíme z poloh dvojice sobě odpovídajících vlícovacích bodů. Vlícovací body určují vztah mezi polohou v globálním souřadném systému (poloha ve skutečném světě) a polohou lokálních vlícovacích bodů (poloha bodů na snímku). Ve všech případech transformací budu pracovat s homogenními souřadnicemi [10].

Pod pojmem rotace rozumíme otočení kolem některé z os soustavy o určitý úhel α. Transformační matice pro rotace kolem jednotlivých os vypadají následovně:

$$\boldsymbol{R_x}(\alpha) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) & 0 \\ 0 & \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \text{ pro rotaci kolem osy } x;$$

$$\boldsymbol{R_y}(\alpha) = \begin{pmatrix} \cos(\alpha) & 0 & \sin(\alpha) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \text{ pro rotaci kolem osy } y;$$

$$\boldsymbol{R_z}(\alpha) = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) & 0 & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \text{ pro rotaci kolem osy } z;$$

Pod pojmem translace rozumíme posunutí o vektor $\boldsymbol{v} = (v_1, v_2, v_3, 1)$, který udává směr a o jakou vzdálenost proběhne posunutí. Transformační matice pro translaci vypadá následovně:

$$\boldsymbol{T}(\boldsymbol{v}) = \begin{pmatrix} 1 & 0 & 0 & v_x \\ 0 & 1 & 0 & v_y \\ 0 & 0 & 1 & v_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Pod pojmem scaling rozumíme změnu měřítka, která se provede násobením velikosti jednotlivých souřadnicových os, podle faktorů s_x, s_y, s_z. Transformační matice pro změnu měřítka vypadá následovně:

$$\boldsymbol{S}(s_x, s_y, s_z) = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Je nutné dodat, že všechny uvedené transformace lze skládat do jedné matice, což vede ke zrychlení výsledného vykreslování. Skládáním je myšleno postupné násobení jednotlivých transformačních matic. V tomto případě je důležité, v jakém pořadí je násobení provedeno, jelikož násobení matic není komutativní. Jinými slovy, pokud vytvoříme transformační matici $\boldsymbol{A}$ násobením tří elementárních transformací v pořadí $\boldsymbol{R} \times \boldsymbol{S} \times \boldsymbol{T}$ a potom vytvoříme transformační matici $\boldsymbol{B}$ násobením tří elementárních transformací v pořadí $\boldsymbol{T} \times \boldsymbol{S} \times \boldsymbol{R}$, budou výsledné matice $\boldsymbol{A}$ a $\boldsymbol{B}$ odlišné, tedy $\boldsymbol{A} \neq \boldsymbol{B}$.

4.5 Projekce

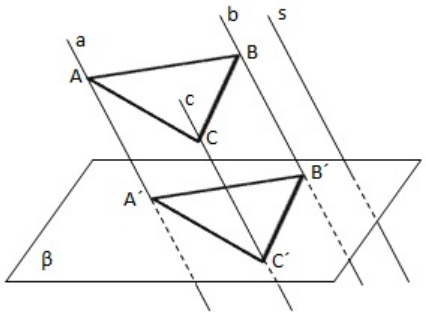
Poslední otázkou zůstává, jak daný 3D model zobrazit na 2D zařízení. Pod 2D zařízením si můžeme představit například monitor počítače. Právě k tomu účelu slouží projekce. V praxi existuje celá řada projekcí, avšak v novém modulu využiji pouze následující typy:

- Paralelní – rovnoběžné promítání
- Perspektivní – středové promítání

4.5.1 Rovnoběžné promítání

U rovnoběžného promítání (obrázek 3) si představme rovinu β (neboli průmětna, reprezentuje 2D zobrazovací zařízení) a přímku s, která udává směr promítání. Přímka s nesmí

být rovnoběžná s rovinou β, jelikož by se průměty bodů obrazce A, B, C nikdy nezobrazily na rovině β. Princip spočívá v promítání bodů obrazce A, B, C na rovinu β pomocí vzájemně rovnoběžných promítacích přímek a, b, c ve směru přímky s. Promítání vytvoří obrazec s body A', B', C' na rovině β.



Obrázek 3: Rovnoběžné promítání obrazce na rovinu

Je důležité si uvědomit, že pokud je přímka s kolmá s rovinou β (tedy v úhlu $90°$), tak výsledným obrazem nové přímky p bude bod, jelikož všechny body přímky p mají totožné promítací přímky. Podle toho, jaký úhel svírá rovina β s přímkou s, rozlišujeme toto promítání na pravoúhlé a kosoúhlé. Využití rovnoběžného pravoúhlého promítání nalezneme například u technických výkresů. Pro úplnost také uvedu transformační matici této projekce, kde dochází k zanedbání souřadnice z:

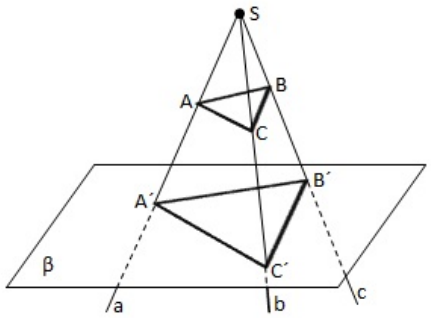
$$\boldsymbol{P} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

4.5.2 Středové promítání

U středového promítání (obrázek 4), vycházejí promítací přímky a, b, c z jediného bodu S. Tento druh promítání respektuje optický model, který vyjadřuje lidské vidění reálného světa. Modeluje proporcionální změnu předmětů při vzrůstající vzdálenosti od pozorovatele (například vzdalující se koleje se sbíhají do jediného bodu), tudíž poskytne dobrý pozorovací vněm na průmětně β [10](rovina, která může opět reprezentovat nějaký druh 2D zobrazovacího zařízení).

V praxi se většinou bod $S[x, y, z]$ označuje jako oko pozorovatele nebo jako kamera. Hodnota souřadnice z pak udává vzdálenost pozorovatele od průmětny. V případě, že

bude modelovaný objekt příliš velký, můžeme zvětšit tuto vzdálenost. Tím se kamera oddálí od průmětny a sledovaný objekt se zmenší. Druhým řešením je manipulace velikosti samotného objektu.



Obrázek 4: Středové promítání obrazce na rovinu

Opět uvedu i transformační matici této projekce:

$$\boldsymbol{P} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{d} & 1 \end{pmatrix}$$

4.6 Výpočty pro analýzu 3D modelu

Fakt, že je model vytvářen ve 3D prostoru, mi umožní provádět různé výpočty, které jsou velmi důležité pro celkovou analýzu. Podle zadání tohoto diplomového projektu, musím být schopen provádět výpočty objemu, obsahu, těžiště a povrchu pláště pro definované zájmové objekty. Tvorbou těchto objektů se budu zabývat v kapitole 5.1.1. Objekty jsou modelovány pomocí bodů, průsečíků, kružnic nebo polygonů. Pochopitelně nebudu provádět všechny tyto výpočty pro každý typ objektu. Například pro bod nebo průsečík nelze počítat obsah.

V následném textu shrnuji všechny důležité vzorce planimetrie a stereometrie. Tím získám potřebný teoretický základ, který budu potřebovat při implementaci funkcí pro jednotlivé grafy.

Polygon

V následujících výpočtech považuji polygon jako nepravidelný n-úhelník, který vymezuje část roviny pomocí lomené čáry. Polygon musí mít minimálně tři vrcholy.

- Objem - na první pohled je zřejmé, že u polygonu nelze počítat objem, jelikož se jedná o rovinný útvar. Avšak u samotného 3D modelování musím vypočítat objem nějakého objektu od jednoho průřezu ke druhému, kdy tyto průřezy jsou definovány jako polygony o stejném počtu hran. Na základě tohoto faktu, mohu využít vzorce pro výpočet objemu mnohostranného komolého jehlanu:

$$V = \frac{v}{3}(S_1 + \sqrt{S_1 S_2} + S_2), \tag{1}$$

  kde v označuje výšku mnohostranného komolého jehlanu, tedy vzdálenost mezi horní a dolní podstavou, S_1 je obsah dolní podstavy a S_2 je obsah horní podstavy.

- Povrch pláště - obdobně, jako tomu bylo u objemu, i zde u výpočtu povrchu pláště, musím uvažovat s mnohostranným komolým jehlanem. Do výsledného povrchu pláště nesmím započítávat povrchy obou podstav, proto použiji tento univerzální vzorec pro výpočet bočního pláště mnohostranného komolého jehlanu:

$$S_{pl} = \frac{1}{4}n(a+b)\sqrt{\cot^2\left(\frac{\pi}{n}\right)|a-b|^2 + 4v^2}, \tag{2}$$

  kde n označuje počet vektorů, které propojují vrcholy obou podstav, a a b označují délky hran dolní a horní podstavy a v označuje výšku mnohostranného komolého jehlanu.

- Obsah - při výpočtu použiji následující vzorec:

$$S = \frac{1}{2}\left(\begin{vmatrix} x_1 & x_2 \\ y_1 & y_2 \end{vmatrix} + \begin{vmatrix} x_2 & x_3 \\ y_2 & y_3 \end{vmatrix} + \cdots + \begin{vmatrix} x_n & x_1 \\ y_n & y_1 \end{vmatrix}\right), \tag{3}$$

  kde x a y jsou souřadnice jednotlivých vrcholů polygonu a matice $\boldsymbol{M}$ označuje determinant. Vše můžu rozepsat následovně:

$$S = \frac{1}{2}(x_1y_2 - x_2y_1 + x_2y_3 - x_3y_2 + \cdots + x_ny_1 - x_1y_n) \tag{4}$$

- Těžiště - použiji následující vzorce pro výpočet souřadnic těžiště T_x a T_y:

$$T_x = \frac{1}{6S}\sum_{i=0}^{n-1}(x_i + x_{i+1})(x_iy_{i+1} - x_{i+1}y_i), \tag{5}$$

$$T_y = \frac{1}{6S}\sum_{i=0}^{n-1}(y_i + y_{i+1})(x_iy_{i+1} - x_{i+1}y_i), \tag{6}$$

  kde x a y jsou souřadnice jednotlivých vrcholů a S označuje obsah polygonu (viz vzorec (4)).

Kružnice

U následujících výpočtů považuji kružnici jako pravidelný mnohoúhelník o vysokém počtu hran. Vyplývá to ze samotného použití API pro 3D modelování, kde nelze definovat kružnici tak, jak ji známe. Praktický rozdíl bude zanedbatelný a to, jak ve výsledku výpočtu, tak i opticky.

- Objem - v případě, že bych při 3D modelování používal kružnici, stačilo by využít vzorce pro výpočet objemu rotačního kužele. Ale jak jsem již napsal v odstavci výše, kružnice jsou reprezentovány pravidelným mnohoúhelníkem. Proto využiji vzorec (1) pro výpočet mnohostranného komolého jehlanu.

- Povrch pláště - i zde by byla možnost využít vzorce (2) pro povrch pláště mnohostranného komolého jehlanu. Avšak pro jednoduchost využiji vzorce pro výpočet plochy pláště komolého kužele. Kvůli velkému počtu hran v polygonu, který reprezentuje kružnici, je rozdíl výsledků obou možností zanedbatelný:

  $$S_{pl} = \pi(r_1 + r_2)s, \tag{7}$$

  kde r_1 a r_2 označují poloměry dolní a horní podstavy komolého kužele a s označuje výšku stěny. Tu lze jednoduše spočítat pomocí Pythagorovy věty:

  $$s = \sqrt{(r_1 - r_2)^2 + v^2}, \tag{8}$$

  kde v označuje výšku komolého kužele, tedy vzdálenost mezi dolní a horní podstavou.

- Obsah - při výpočtu použiji následující vzorec:

  $$S = \pi r^2, \tag{9}$$

  kde r označuje poloměr dané kružnice.

- Těžiště - nepotřebuji počítat, jedná se o střed kružnice. Tyto informace jsou známé při použití nástroje pro definici objektů – jedná se o místo, kde uživatel klikne při vytváření nové kružnice.

Průsečík

Průsečík je reprezentován průnikem dvou úseček, které mají uživatelsky definovanou délku a obsahují počáteční a koncové body.

- Objem - neurčuje se.

- Povrch pláště - pokud předpokládám průsečík definovaný na jednotlivých snímcích, tak ve 3D modelu získám dvě různoběžné roviny se společnou průsečnicí. Obě roviny mohu považovat za lichoběžníky, na které aplikuji vzorec pro obsah. Nesmím zapomenout vynásobit každou rovinu dvěma, jelikož jsme v prostoru a rovina má dvě strany (plochy). Vzorec jde samozřejmě zjednodušit, ale pro názornost ho ponechám v původním tvaru. Podmínkou pro správnost tohoto vzorce je dodržení stejných souřadnic průsečíku na jednotlivých snímcích:

$$S_{pl} = \left[\frac{2(a+b)v}{2}\right] + \left[\frac{2(c+d)v}{2}\right], \tag{10}$$

  kde a, b jsou délky horní a dolní strany prvního lichoběžníku a c, d toho druhého. Proměnná v označuje výšku obou lichoběžníků, tedy vzdálenost obou průsečíků v 3D prostoru.

- Obsah - neurčuje se.

- Těžiště - nepotřebuji počítat. Souřadnice těžiště získám z jiného modulu, který se zabývá tvorbou zájmových objektů na jednotlivých snímcích.

Bod

U bodu neurčuji objem, plochu pláště ani obsah. Souřadnice těžiště bodu je samotný bod. V případě, že body z jednotlivých snímků modeluji ve 3D prostoru, získávám lomenou čáru, která je složená z úseček.

5 FOTOM^{NG}

Seznámení s celým systémem je velmi důležité pro pochopení, jak získám data, tedy, co bude vstupem vyvíjeného modulu 3D modelování. FOTOM^{NG} je komplexní nástroj schopný zpracovávat a analyzovat fotogrammetrické snímky. Vývoj této aplikace začal už v roce 2008. Hlavním důvodem byly zvyšující se požadavky, kterým už předešlé verze systému FOTOM nemohly nadále konkurovat. FOTOM^{NG} musel klást velký důraz na modularitu. Ta měla vést k jednoduchému integrování nových funkcí, tedy k rozšiřování systému. Proto bylo rozhodnuto, že implementace proběhne na platformě NetBeans (v jazyce Java), díky které se stal FOTOM^{NG} plnohodnotnou aplikací, odpovídající moderním trendům. Základem bylo vytvoření samotného jádra, včetně prototypu GUI pod vedením katedry informatiky VŠB-TU Ostrava.

Stav systému je ve fázi testování a dalšího vývoje. Avšak už nyní obsahuje velké množství modulů, které jsou ve stádiu, kdy mohou být uživatelem plně využívané. Jedná se například o paletu nástrojů, díky které definujeme zájmové objekty na snímcích, dále velký počet filtrů, 2D animace, 2D modelování a mnoho dalších.

V plánu je vývoj dalších funkcí, jako jsou 3D rekonstrukce s využitím radonové transformace, voxelové grafiky a další. Jinou myšlenkou je tvorba obsahově menších systémů, které integrují jen některé moduly a to podle toho, pro jaký obor bude systém určen. To povede ke zjednodušení manipulace pro uživatele, který například nepotřebuje využívat některé funkce systému.

Jak jsem se zmínil v prvním odstavci, systém pracuje s fotogrammetrickými snímky. V praxi si pod tímto pojmem můžeme představit například nějakou sérii ultrazvukových snímků (takovou sérii budu mít k dispozici při vývoji a testování).

5.1 Moduly a funkce

Po provedení všech náležitostí pro správnou funkčnost systému (instalace JDK nebo JRE a balíčku JAI) se po inicializaci všech modulů zobrazí defaultní uživatelské rozhraní. Už zde jsme schopni pozorovat podobnost s NetBeans IDE. Rozhraní je separováno do pěti hlavních oken, kdy každé okno je implementováno jiným modulem (tato okna je možné spravovat přes příslušnou položku v hlavním menu). V levém horním rohu se nachází okno prohlížeče, které poskytuje přehled všech měření (adresářů) a jejich snímků. Pod tímto oknem je umístěn manažér objektů. Ten zobrazuje pole názvů aktuálních pozorovaných objektů. Uprostřed celého systému je situováno okno pro editor, který pracuje s jednotlivými snímky. Pod pojmem editor si můžeme představit třeba okno 2D animací nebo modelování. Systém řazení dalších editorů funguje na principu záložek. V pravém

horním rohu se nachází paleta, která obsahuje celou řadu nástrojů pro definici zájmových objektů nebo pro grafickou úpravu fotogrammetrických snímků. Poslední částí je okno vlastností. Poskytuje uživateli doplňující informace, například o daném objektu. Okna a funkce důležité pro 3D modelování podrobněji popisuji v následujících kapitolách, kde čerpám z literatury [11].

5.1.1 Paleta nástrojů

Tento modul poskytuje uživateli panel nástrojů, ve kterém jsou zobrazeny dostupné nástroje pro měření, definici zájmových objektů, skicování a algoritmy pro automatickou detekci. Výhodou této palety je jednoduchost přidávání nových nástrojů. Stačí implementovat novou třídu, která reprezentuje požadovaný nástroj a paleta jej přidá do nabídky [11]. Vše je přehledně rozděleno do několika sekcí, podle typu nástroje:

Měření

- Vlícovací body – jeden z nejdůležitějších nástrojů, protože definuje referenční body na zpracovaném snímku. Podle těchto bodů je pak možné určit pozici snímku v prostoru. Dále se využívá u série snímků, které jsou na sobě závislé, tzn. na snímcích, kde je pozorován stejný objekt. U série pak podle vlícovacích bodů určujeme posunutí nebo otočení snímků vůči referenčnímu a podle toho provádíme transformace (viz kapitola 4.4). Vlícovací body také vyjadřují poměr mezi souřadnicemi udávaných v měrných jednotkách (mm, cm, m) a obrazovými jednotkami (pixely). Při jejich vytvořeni je potřeba zadat měrné jednotky, reálné souřadnice a polohu snímku v sérii.

Definice objketů systému FOTOM

- Polygon – nejpoužívanější nástroj. Pomocí polygonu definujeme oblasti, které nejsou pravidelné, a proto nelze použít nástroj kružnice. Avšak v kombinaci obou nástrojů můžeme pozorovat rozdíly obsahu sledovaného objektu. Například, jaký obsah má tepna na ultrazvuku (definice polygonem) a jaký by měla mít (definice kružnicí). Rozdílem dostaneme obsah usazeného cholesterolu. Sledovanými parametry je těžiště, obsah polygonu v měrných i obrazových jednotkách (musí být definovány vlícovací body) a počet hran.
- Kružnice – slouží pro měření odchylek a vzdáleností na snímcích kruhového tvaru (důlní šachty, tepny, atd.). Je definována dvěma parametry a to souřadnicemi středu

a poloměrem. U kružnice sledujeme obsah, jak v obrazových, tak i v měrných jednotkách (musí být definovány vlícovací body).

- Průsečík – pomocí průsečíku definujeme jediný bod. K tomu využíváme dvě úsečky, vytvořené pomocí čtyř bodů. Průsečíkem těchto dvou úseček je námi sledovaný bod.
- Bod – nejjednodušší nástroj k definování jediného bodu zájmu. Sledujeme souřadnicovou polohu bodu na snímku.
- Další nástroje v položce definice objektů – tyto nástroje jsou z pohledu 3D modelování irelevantní, uvádím je jen pro úplnost – mřížka, matice.

Pokročilé metody pro definice objektů

- Aktivní kontura – jeden z algoritmů pro tvorbu segmentace a následného polygonu. Uživatel si tahem myši vytvoří konturu kolem sledovaného objektu. Pak se spustí algoritmus. Nabízí se taky možnost zadání vlastního počtu bodů v polygonu nebo další parametry ovlivňující průběh algoritmu.
- Fast Marching Level-Set – další algoritmus pro snadné vytvoření polygonu. Uživatel si vybere místo, kde chce provést segmentaci. Celý proces je závislý na počtu iterací, které můžeme následně přidávat. Opět se nabízí možnost definice vlastního počtu bodů v polygonu a mnoha dalších parametrů.
- Detekce tepny – je posledním algoritmem, který nám po výběru místa automaticky detekuje tepnu. Velkou výhodou je i vytvoření kružnice, která indikuje obvod tepny. Další způsob jak můžeme sledovat obsah usazeného cholesterolu. I tento algoritmus poskytuje vlastní definici počtu bodů.

5.1.2 Filtry

Tato položka menu nám poskytne celou řadu metod pro úpravu fotogrammetrických snímků. Mezi ně patří například prahování, několik funkcí pro práci s barvami snímku, pro zaostření a rozmazání nebo máme možnost využít několik metod pro detekci hran. Velkou výhodou je i tvorba vlastních uživatelských filtrů. Většina těchto funkcí slouží k efektivnímu nalezení hran a následnému definování zájmových objektů:

- Prahování - jedná se o velmi jednoduchou metodu, která je založená na rozlišování jasových bodů (pixelů) od pozadí. Je nutné najít jasový práh, který by oddělil zájmové objekty od pozadí. Výsledkem je pak obraz v binárním tvaru, kde hodnotu

1 mají ty body, které náleží objektu. Hodnota 0 pak patří pozadí [11]. K dispozici je také rozšíření, které umožňuje výběr barvy pro jasové body náležící objektu.

- Detekce hran – pod pojmem hrana si představme místo na snímku, kde se prudce mění jas. Tento fakt jsme schopni využít u identifikace sledovaných oblastí na obrazovém snímku. Každý typ detektorů hran využívá různých technik, proto se jejich aplikace hodí pro odlišné snímky. V systému FOTOM^{NG} jsou k dispozici následující detektory hran. Tento text čerpá z literatury [13]:
  - Prewitt – využívá k výpočtu gradientu v bodě $[X, Y]$ hodnoty obrazové funkce všech okolních pixelů. Následně se spočítá průměr derivace ve směrech v osách x a y. Tato technika přispívá k redukci šumu.
  - Canny – cílem je vytvořit hranový detektor s optimálními výsledky, které netrpí citlivostí na šumu. Musí splňovat kritéria, jako jsou minimalizace chybné detekce hran, přesnost nalezených hran a jednoznačná identifikace hrany. Samotný detektor běží v následujících algoritmech – vyhlazení, nalezení gradientu, non-maximální suprese a hysterezní prahování. Kompletní rozbor techniky je uveden v literatuře [13].
  - Kirsch – počítá v osmi různých směrech. Technika využívá k výpočtu gradientu v bodě $[X, Y]$ všech okolních bodů, podle směru hrany.
  - Laplacian – používá druhou derivaci průběhu jasu, ve směru napříč hranou, kde nabývá dvou extrémů opačného znaménka. Hranu detekuje v místě, kde se tato znaménka mění. Směr hrany se dopočítá právě pomocí operátoru Laplace. Technika je citlivá na šum, proto je dobré využít různých metod pro potlačení šumu, například Gaussova eliminace šumu.
  - Sobel – podobný jako Prewitt, s tím rozdílem, že se pro výpočet derivací v jednotlivých směrech používá vážený průměr. Klade větší důraz centrální buňce, větší citlivost na šum v obraze. Ideální pro detekci vertikálních a horizontálních hran.
  - Další – detekce technikou FreiChen, High-Pass a Low-Pass filtr, směrový filtr a Median filtr.

5.2 Předchůdce - FOTOM 2008

Tuto kapitolu chci věnovat starší verzi systému FOTOM^{NG}. Je to velmi důležitá kapitola z pohledu specifikace požadavků. Seznámím se s modulem Fotom3 starého systému, který se zabýval právě 3D modelováním.

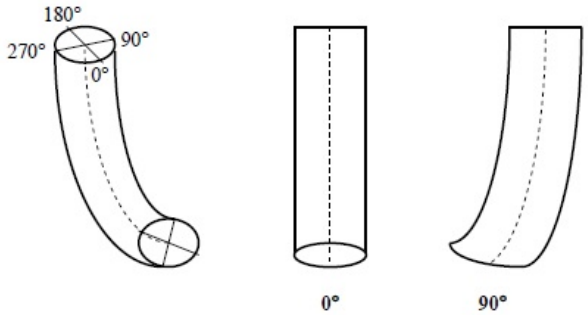
Celý systém FOTOM 2008 (implementovaný v jazyce C++) byl založen na základním modulu Fotom1, který fungoval jako rozcestník k ostatním modulům. Nicméně, byla zde možnost, kdy každý modul mohl pracovat a spouštět se samostatně. V případě, že uživatel začal pracovat na nějakém rozsáhlém měření, kde používal moduly pro 3D modelování, 2D modelování, 2D animace a ještě měl spuštěný základní modul, ztížilo to jeho práci z pohledu přehlednosti a jednoduchosti manipulace s celým systémem. Tento nedostatek kompletně odstraňuje, díky platformě NetBeans, systém FOTOM^{NG}.

5.2.1 Analýza modulu Fotom3

Nyní zanalyzuji předchůdce, modul Fotom3 a popřípadě odhalím nedostatky, kterým se budu chtít vyhnout při implementaci nebo navrhnu řešení problému. Podrobnější analýze, která popíše jednotlivé funkce, se budu věnovat v kapitole 6.2. Poslouží mi jako základní specifikace požadavků, kterou popřípadě rozšířím o další užitečné funkce.

Při spuštění modulu, byl uživatel vyzván k výběru snímkové série a po potvrzení a kontrole podmínek byl spuštěn samotný modul. Jelikož neexistovalo společné jádro, nemohly být jednotlivé snímky předávány mezi moduly.

Modul poskytoval uživateli čtyři synchronizované pohledy se schopností přepínat se do jediného zvětšeného pohledu. Pokud došlo k nějaké změně na jednom pohledu, projevila se tato změna i na ostatních. Každý pohled byl definován jinou polohou kamery. Tím se odstranil nechtěný efekt „relativity úhlu pohledu“. Představme si nějaký objekt, který se jeví jako rovný. Avšak pokud na něj pohlédneme z jiného úhlu, zjistíme deformaci tak, jak je to zobrazeno na obrázku 5, získaného z literatury [14].

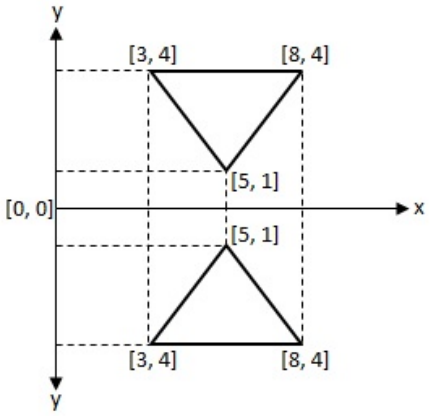


Obrázek 5: Relativita úhlu pohledu

Nevýhodou modulu Fotom3 byla složitá interakce s 3D modelem. Uživatel měl k dispozici interakce jako posun, zvětšování, zmenšování, rotace, změnu měřítka, atd. Pokud chtěl provést jednu z interakcí, musel na liště nástrojů stisknout požadované tlačítko.

Tím se rapidně prodlužovala doba pro manipulaci a navíc tlačítka zabíraly místo už v tak přeplněné liště nástrojů. V novém modulu chci tento nedostatek odstranit využitím dalších tlačítek myši, popřípadě na klávesnici.

Dále jsem odhalil problém, který je způsobený převodem všech souřadnic objektů ze 2D fotogrammetrických snímků do 3D prostoru. Jelikož je na snímcích počátek souřadnicové soustavy umístěn vpravo nahoře, tak následné převedení do 3D prostoru, bez potřebné úpravy způsobí, že budou všechny objekty zrcadlově převrácené (osová souměrnost), tak jak to lze vidět na obrázku 6. Základem problému je, že definice objektů na 2D snímky používá levotočivou souřadnicovou soustavu, kdež to 3D prostor ve většině případů používá pravotočivou souřadnou soustavu – viz kapitola 4.2. Tento nedostatek způsobí minimálně špatnou orientaci uživatele ve 3D modelu. Abych tomu předešel, budu muset provést výpočet, který každou y souřadnici odečte od maximální y souřadnice. Maximální y souřadnici získám z výšky daného snímku. Tím se jednotlivé objekty budou modelovat ve 3D prostoru stejně, jak byly definovány na fotogrammetrických snímcích.



Obrázek 6: Použití stejných souřadnic mezi typy souřadnicových systémů

Uživatel mohl používat funkci pro obarvování jednotlivých objektů. Nejprve musel tuto funkci aktivovat pomoci tlačítka na liště nástrojů, poté si vybrat jednu ze šesti barev, které byly taktéž umístěny na liště nástrojů. V dalším kroku se označil požadovaný objekt v seznamu, který byl situován v levé části okna modulu. Opět není potřeba mít tyto barvy přímo na panelu nástrojů, kde zbytečně zabírají místo. Navíc jsme byli omezeni pouze na šest různých barev. To jednoduše vyřeším pomocí palety barev, která se zobrazí v dialogovém okně a navíc nám poskytne celé barevné spektrum. Problém se také může vyskytnout, pokud uživatel zvolí obarvování pomocí bílé barvy. Jelikož je možné zvýraznit jednotlivé průřezy modelu (průřezy určují data z jednotlivých snímků série) bílou

barvou, uživatel by pak neviděl zvýrazněné oblasti. Toto vyřeším uživatelsky nastavitelnou barvou průřezu. Jelikož chci vytvořit moderní nástroj, budu implementovat funkci, která bude obarvovat objekty pomocí myši přímo ve 3D modelu.

Každý ze čtyř pohledů má v levém horním rohu umístěny informace o zobrazovaném objektu. Jedná se například o název objektu, úhel otočení, jednotky, obsah, atd. Avšak i přesto, že jsou zobrazeny 3D modely více objektů, jsou popisky názvu a obsahu stále omezeny jen na první objekt. Nový modul bude poskytovat kompletní data a to podle toho, co bude zobrazeno. Další problém se vyskytl u přibližování, kdy jednotlivé popisky mizí za kameru, tudíž, když si uživatel přiblíží celý 3D model, neuvidí tyto důležité údaje. Možností, jak vyřešit tento problém, je ukotvit popisky na dané plátno.

Bohužel, modul postrádá funkci, která by zajistila možnost fyzické dokumentace zobrazených výsledků. A to, jak 3D modelu, tak i zobrazovaných grafů. V závěrečné fázi implementace chci vytvořit funkci, která zajistí uživateli možnost tisku jednotlivých pohledů.

Samotné GUI modulu obsahovalo nabídku menu, kde uživatel nalezl položky pro nastavení animací, scény, atd. Tyto položky přesunu přímo na lištu nástrojů, aby byly uživateli plně k dispozici.

5.2.2 Závěr analýzy

I přesto, že modul Fotom3 starého systému obsahoval nějaké nedostatky, je výborným nástrojem pro 3D modelování. Díky této analýze, mám základní představu, jak bude nový modul vypadat. Navíc jsem v kapitole 5.2.1 navrhl možná řešení pro dané nedostatky.

6 Návrh modulu

Nastává moment, kdy je potřeba navrhnout nový modul. Postupně jsem prošel všechny potřebné fáze. Nejdřív proběhla studie teoretických základů, které zahrnují znalosti fotogrammetrie (3) a 3D modelování (4). Pak jsem se seznámil se stávajícím systémem a podrobně zanalyzoval předchůdce nového modulu (5).

V následujících kapitolách zmiňuji základní části pro návrh modulu. Úvod bude věnován vizi navrhovanému modulu, která vyplývá ze zadání diplomového projektu a předchozích zkušeností s vývojem systémů FOTOM. Dále chci specifikovat obecné požadavky, které budou kladeny na implementaci. Poté navrhnu základ GUI. V závěru kapitoly budu specifikovat softwarové požadavky, které zahrnují znalosti platformy NetBeans, ale především knihovny Java3D určené pro vytváření 3D grafiky.

6.1 Vize

Navrhovaný modul s názvem 3D modelování bude zahrnovat všechny funkce, které byly implementovány v systému FOTOM 2008, modul Fotom3. Pochopitelně tyto funkce mohou být rozšířeny nebo doplněny o další nové. V případě zjištění nějakých nedostatků v modulu Fotom3, je potřeba předejít stejným chybám. Veškeré požadavky jsou systematicky popsány v následující kapitole 6.2.

Modul 3D modelování umožní uživateli rekonstrukci objektů pomocí polygonální reprezentace sítě ze série fotogrammetrických snímků zájmových objektů. Součástí bude také vyhodnocování základních vlastností rekonstruované plochy, jako je povrch pláště, těžiště, objem a obsah průřezu. Modul bude integrován do systému FOTOM^{NG} spolu s položkou pro jeho spuštění umístěnou v hlavním menu systému.

Modul poskytne uživatelsky příjemné prostředí s potřebnou nápovědou formou českého překladu (jak se modul ovládá, popisky jednotlivých funkcí). Jednotlivé parametry zadávané uživatelem budou ošetřeny, včetně spuštění celého modulu. Tím je myšlena kontrola podmínek, které musí splňovat všechny snímky série. Podporu modulu budou zaručovat dokumenty ve formě uživatelské příručky a programátorské dokumentace.

6.2 Specifikace obecných požadavků

6.2.1 Pohledy a interakce

Typy pohledů

Po spuštění modulu přes menu položku 3D modelování se uživateli zobrazí čtyři pohledy. Fungují jako čtyři kamery, snímající modelované objekty z různých směrů. V

následujících případech počítám s pravotočivou souřadnicovou soustavou. První kamera snímá model vodorovně, podle osy z. Druhá kamera snímá svisle s otočením modelu o $90°$, taktéž podle osy z. Třetí a čtvrtá kamera poskytuje pohled podle osy x, kdy čtvrtý pohled poskytuje uživateli rozdílné interakce (viz požadavky na interakce).

Zobrazení modelu

Model bude zobrazen v pohledech podle dat, která získá z jednotlivých snímků série, tedy z jejich definic zájmových objektů včetně vlícovacích bodů. Model musí být schopen modelovat z polygonů, kružnic, průsečíků, bodů a zobrazovat jednotlivé vlícovací body.

Interakce s modelem v jednotlivých pohledech

Rotace modelu (kolem osy x) v prvních třech pohledech bude synchronizována, tzn. v případě rotace v prvním pohledu, rotují modely i v dalších dvou. Rotace ve čtvrtém pohledu funguje nezávisle na ostatních a to kolem osy x i y. Obdobně bude přizpůsoben i posun (translace). Změna měřítka bude synchronizována pro všechny pohledy. Jedná se o změnu vzdálenosti mezi všemi snímky s možností měnit měřítko i ve všech osách. Dále uživatel musí mít k dispozici interakci, která mu umožní procházet jednotlivé průřezy modelu, které jsou reprezentovány snímky série. Průřez bude obarven odlišnou barvou než je barva modelu a zobrazí se na všech pohledech. Přiblížení a oddálení bude umožněno pouze na čtvrtém pohledu. Uživatel může přepínat každý pohled do maximalizovaného pohledu.

Popisky v pohledech

Každý pohled bude obsahovat šest popisků s nejdůležitějšími údaji. První popisek zobrazuje názvy všech objektů ve scéně. Druhý popisek se aktualizuje při rotování modelem a zobrazuje úhel rotace. Třetí popisek zobrazuje jednotky, ve kterých je série měřena. Čtvrtý popisek zobrazuje obsah průřezu všech objektů v modelu pro daný průřez. Pátý popisek udává pozici snímku v celé sérii. Poslední popisek zobrazuje název souboru, podle označeného průřezu, tedy ze kterého souboru jsou data získána.

6.2.2 Základní funkce objektů a scény

Nastavení objektů a scény

Poskytne hlavní nastavení pro objekty a scénu. Uživatel bude mít možnost si vybírat, který objekt se zobrazí ve středu souřadnicové soustavy. Dále selekce barvy průřezů pomocí palety, včetně zbarvení popisku souboru na jednotlivých pohledech. Výběr mezi typy propojování profilů (jehlanovité, válcovité), tím získáme jiný pohled na data ze snímků série. Dále funkce, která nastaví hustotu bodů v generovaných kružnicích. Od

toho se odvíjí počet spojnic mezi kružnicemi jednotlivých snímků. Dále bude mít uživatel možnosti, zdali chce propojovat vlícovací body přímkou, měnit měřítko i pro osy x a y nebo používat transformaci. U této funkce máme možnost pozorovat například vliv kvality snímků (jak byly pořízeny), ale hlavně zajistí správnou návaznost objektů z dalších snímků. Tím se vytvoří reálný 3D model.

Volba objektů

Poskytne uživateli možnost vybrat si, které objekty chce zobrazovat. Zvýší se tím přehlednost celého 3D modelu, kdy série obsahuje více objektů.

Resetování objektů a scény

V případě resetu objektů, se zobrazí všechny objekty série. Uživateli bude poskytnuta funkce, která resetuje i celou scénu. Jinými slovy, vrátí vše do defaultního zobrazení, tzn. polohu i nastavení objektů. Užitečné v případě, že uživatel dokončil analýzu a chce začít novou, tak nemusí restartovat celý modul.

6.2.3 Rozšířené funkce objektů a scény

Funkce přední a zadní stěny

Tyto funkce umožní zobrazovat jen požadovanou část 3D modelu a bude synchronizována pouze na třetím a čtvrtém pohledu. Přední stěna bude postupovat od pozorovatele, kdežto zadní stěna k pozorovateli.

Drátěný a texturovaný model

Funkce poskytnou různé kombinace, jak zobrazit 3D model. V případě drátěného modelu bude model zobrazen pouze pomocí lomených čar a spojnic. Uživatel uvidí, kde jsou definovaný jednotlivé body a jak se mění na dalším snímku. U texturovaného modelu bude k dispozici nabídka několika typů textur. Po výběru budou stěny vyplněny touto texturou a uživatel získá realističtější pohled na 3D model. Navíc může pozorovat, jak se stěny jednotlivých objektů překrývají. Kombinací získáme výhody obou druhů zobrazení. V případě, že nezvolíme žádný, model se nezobrazí. Zvýraznění průřezu bude pro oba druhy jinak koncipováno. U drátěného modelu budou zvýrazněny lomené čáry, kdežto u textury bude obarvena celá plocha průřezu.

Volba projekce

K dispozici bude výběr mezi dvěma typy projekcí. Prvním defaultním typem je perspektivní projekce a druhým je paralelní projekce. O těchto projekcích se věnuji v kapitole 4.5.

Minimalizovaný a maximalizovaný pohled

Jedná se o rozšíření interakce, která poskytne stejnou možnost pro přepínání pohledů do maximalizovaného pohledu pomocí dialogového okna pro výběr pohledu a tlačítek.

Grafy

Než se zobrazí samotný graf, bude uživatel definovat místo, kterého pohledu se graf zobrazí nebo o jaký typ dat se bude jednat – těžiště, plocha pláště, obsah, objem. Uživatel si může zvolit i samotné nastavení grafů, bude se jednat o velikost textu, velikost bodů a velikost aktuálně zvoleného bodu.

Grafy pro těžiště musí zahrnovat průběhy pro souřadnici X a Y, kde na ose x se vynesou reálné hodnoty v měrných jednotkách a na ose y se u všech grafů uvádí poloha snímku v sérii. Dále popisky grafů, které určují průměrnou, minimální a maximální hodnotu pro souřadnice X a Y včetně aktuální hodnoty v obou grafech. Ta bude stejně jako u ostatních grafů synchronizovaná s procházením mezi jednotlivými průřezy, včetně barvy aktuálního bodu. V grafu pro plochu pláště a obsahu se na osu x vynesou reálné hodnoty ve čtverečních jednotkách. Pro graf objemu jsou na osu x vyneseny reálné hodnoty v krychlových jednotkách. Výpočty jsem uvedl v kapitole 4.6.

Nastavení popisků

Volbou této funkce získá uživatel možnost kontrolovat zobrazení všech popisků na jednotlivých pohledech 3D modelu. Může tak vypínat nebo zapínat zobrazení informací, aby například zvýšil přehlednost.

Barvení objektů

Tento nástroj poskytne uživateli možnost obarvovat jednotlivé objekty série pomocí barvy zvolené z palety. Po zvolení barvy klikne myší na objekt 3D modelu, který se podle ní obarví. Tato funkce zvýší přehlednost a navíc můžeme pozorovat, jak se jednotlivé objekty překrývají. Barvení objektů bude koncipováno pro drátěný i texturovaný model. V případě drátěného modelu se obarví jednotlivé lomené čáry a spojnice a u texturovaného modelu se smíchá barva s texturou. Resetování objektu ani funkce volby objektu nesmí mít vliv na obarvení, tedy nesmí jej zrušit.

6.2.4 Funkcionalita animací

Nastavení animací

K dispozici bude jednoduché nastavení, kde uživatel definuje prodlevu ve vykreslování jednotlivých profilů série v milisekundách s defaultní hodnotou 500ms. Důležitý bude typ animace, který rozlišuje pět druhů. Jedná se o animaci vpřed, kdy proběhne

jedna iterace od prvního profilu po poslední. Dále opakovaná animace vpřed, kdy se cyklicky vykreslují profily od prvního po poslední. Pak animace vzad, kde ubývají profily od posledního po první. Obdobně pak funguje i opakovaná animace vzad avšak cyklicky. Posledním typem bude animace vpřed a vzad, kdy se cyklicky opakuje nejdříve vykreslení vpřed následované vykreslením vzad.

Přehrávání automatických animací

Pod tímto názvem bude mít uživatel k dispozici funkce pro manipulaci s přehráváním animací. Jedná se o klasická tlačítka spustit, pozastavit a zastavit, doprovázená tlačítky pro zrychlení animace o dvojnásobnou rychlost. Tlačítko pro pozastavení, zastaví nebo zapne animaci u daného snímku. Tlačítko pro zastavení, ukončí celou animaci a vrátí model do původního stavu.

Ruční animace

Jedná se o typ animace, kdy nebude využita prodleva. Uživatel používá tlačítka pro posun dopředu a zpátky. Tím animuje jednotlivé profily série manuálně. Procházení pomocí ruční animace bude omezeno intervalem prvního a posledního snímku série.

6.2.5 Funkcionalita měření

Kalibrace

Pod touto funkcí bude možnost zkoumat odchylky na snímcích. Na třetím pohledu budou zobrazeny kalibrační osy, pomocí kterých uživatel nastaví kalibrační body X_1, X_2, Y_1 a Y_2. Při této kalibraci se sleduje ořezání modelu na prvním a druhém pohledu podle pohybu kalibračních os, ze kterého můžeme vidět dané odchylky.

Funkce projektového objektu

Tato funkce na základě aktuálního průřezu a zvoleného objektu vytvoří projekci průřezu přes celou délku 3D modelu, tedy od prvního až po poslední snímek. Budeme moci pozorovat, jak se jednotlivé objekty a jejich průřezy liší od takto projektovaného objektu.

6.2.6 Doplňující funkcionalita

Ověření správné série

Před spuštěním celého modulu musí proběhnout kontrola série na vstupu, aby byla zaručena správná funkčnost 3D modelování. Budou zkontrolovány následující parametry – existence vlícovacích bodů a jejich reálných hodnot, nadefinování hodnoty pozice

snímku v sérii, zdali snímky obsahují vhodné objekty pro 3D modelování, jsou-li definované jednotky a zda mají polygony jednoho objektu stejný počet bodů. V případě porušení podmínky, musí být vyvolána výjimka, která informuje uživatele o dané chybě.

Doplňující vlastnosti série

Aby bylo možné provést 3D modelování, musí série obsahovat minimálně dva fotogrammetrické snímky. Dále objekt nemusí být definován na každém snímku. V takovém případě bude návaznost jednotlivých průřezů pro dané místo přerušena. Pokud série nebude kompletní, tedy budou-li scházet některé snímky, musí se tomu přizpůsobit i vzdálenost jednotlivých průřezů ve 3D modelu.

Ostatní požadavky

Uživatel bude moci využívat funkce tisku pohledů a rychlé nápovědy pro manipulaci s 3D modelem. Dále možnost přidávat další sérii, avšak s rozdílnou základní barvou. Můžeme tak porovnávat více měření mezi sebou. Všechny uživatelské vstupy musí být ošetřeny vůči špatnému zadání parametrů a jednotlivé funkce budou vybaveny českou nápovědou, včetně české lokalizace celého modulu. Pro co nejlepší podporu je nutné vytvořit uživatelskou příručku a programátorskou dokumentaci.

6.3 Návrh GUI

U tohoto prvotního návrhu GUI budu vycházet z uživatelského rozhraní, které bylo použito u modulu Fotom3, avšak upraveného podle specifikace požadavků v předchozí kapitole 6.2. Jelikož se jedná o úplně jiné programovací jazyky, bude výsledná implementace komponent zcela odlišná.

Okno modulu 3D modelování bude situováno jako editační okno platformy NetBeans. Na základě zachování společného konceptu s dalšími moduly v systému FOTOM ^{NG} bude GUI složeno ze dvou částí. První část bude lišta nástrojů, kterou umístím horizontálně v horní části okna. K vytvoření lišty použiji $JToolBar$. Zde budou umístěny jednotlivé ikony funkcí v oddělených blocích pomocí separátorů, které zvýší přehlednost. Druhá část bude tvořená kontejnerem, ve formě $JScrollPane$, na kterém zobrazím čtyři pohledy na 3D model. Pro tento panel jsem zvolil nákresový manažer $GridLayout$ s 2x2 buňkami. Do jednotlivých buněk pak vložím samotná plátna.

Pokud hovořím o návrhu GUI, je nutné zmínit dialogová okna, kterých bude v tomto modulu velké množství. Využiji je například u nastavení scény a objektů, pro výběr objektů, nastavení grafů a textů nebo pro nastavení animací, atd. Návrhy se liší podle využití dialogového okna, ale vždy je použit nějaký nákresový manažer, aby se dosáhlo vzhledu moderní aplikace.

V praxi by návrh GUI doprovázel i prvotní náčrt a podrobnější popis. V mém případě se spokojím pouze s tímto textem, jelikož mám představu, jak bude celkový vzhled vypadat. A to díky specifikaci požadavků (kapitola 6.2) i analýze modulu Fotom3 (kapitola 5.2.1).

6.4 Platforma NetBeans

Následující kapitola je věnována platformě NetBeans a čeprá z literatury [11, 15], na které je implementován celý systém FOTOM^{NG}. Platforma NetBeans patří mezi RCA aplikace. To jsou takové aplikace, které pracují pouze na straně klienta, kde nejen data zobrazují, ale také je zpracovávají. Jedná se o modulární a rozšiřitelný aplikační framework pro Swing aplikace. Tato modularita přináší mnoho výhod, jako je například zjednodušení vytváření, přidávání a mazání nových prvků aplikace pro uživatele nebo zjednodušení aktualizací již nainstalovaných prvků aplikace. Avšak největší výhodou této platformy je, že programátor nemusí vyvíjet žádné základní funkce, což vede k ušetření mnoha hodin vývoje [11].

6.4.1 Jádro platformy

Jádro Netbeans Runetime Container (dále jen NRC) na sebe přebírá veškerou zodpovědnost za běh celé aplikace. Nahrává všechny dostupné NetBeans moduly a sestavuje je do jedné Swing aplikace. Dále dovoluje dynamické instalování a odinstalování modulů, které jsou v tu chvíli dostupné. Jinak řečeno NRC je prostředí, které pochopí, jakou funkci mají jednotlivé moduly. Také pracuje s životními cykly modulu (spuštění, vypnutí, nahrávání a uvolňování modulu) a umožňuje komunikaci s ostatními moduly. Minimální sestava NRC pro běh aplikace se skládá z těchto modulů [15]:

- Bootstrap – spouští se dříve než všechny ostatní moduly, aby umožnil NRC pochopit, jaká je funkce daného modulu. Následuje načtení a poskládání do jedné aplikace.
- Utilities – poskytuje základní komponenty, například pro komunikaci mezi moduly.
- Startup – nastartuje aplikaci, inicializuje Module System a File System.
- File System – zpřístupní virtuální datový systém s přístupem nezávislým na platformě. Na začátku se použije k načtení zdrojů jednotlivých modulů do aplikace.
- Module System – tento modul odpovídá za správu všech modulů aplikace, jejich načítání, aktivace, deaktivace. U tohoto modulu je nutno zmínit práci se soubory `manifest.mf`. Tyto soubory popisují základní vlastnosti jednotlivých modulů a jejich závislosti na jiných modulech. Nalezneme zde také další atributy, jako jsou

například krátký a dlouhý popis modulu, lokalizace souboru s vlastnostmi, lokalizace souboru `layer.xml`, název modulu, definice závislostí mezi moduly nebo verzi modulu [11].

6.4.2 Definování souborového systému

Každý modul přidaný do aplikace má svůj vlastní souborový systém, který je reprezentován souborem `layer.xml`. Formátem takového souboru je hierarchický souborový systém se složkami, soubory a atributy. Při slučování jednotlivých modulů do jedné aplikace se tyto soubory spojí a vytvoří tzv. $MultiFileSystem$. Můžeme na něj pohlížet, jako na rozhraní mezi modulem a platformou NetBeans. V praxi můžeme pomocí souborového systému vytvořit například strukturu hlavního menu aplikace [11].

V souboru `layer.xml` také definujeme akce, které reagují na uživatelské vstupy do aplikace, například po stisku na nějakou položku v menu. Akce v tomto souboru se registrují uvnitř bloku složky $Actions$.

6.4.3 Okno TopComponent

Knihovna Windows System poskytuje třídu $TopComponent$, která je potomkem třídy $JComponent$. Na této třídě jsou založena všechna okna, která se integrují do platformy NetBeans. Všichni potomci, kteří budou implementováni, se můžou během své existence nacházet v různých stavech, například otevřený, uzavřený, viditelný, neviditelný, aktivní a neaktivní [11].

6.5 Knihovna Java3D

Při výběru této knihovny, vhodné pro 3D modelování, jsem se musel řídit několika požadavky. Potřeboval jsem knihovnu, schopnou vytvořit 3D model včetně všech funkčních požadavků uvedených v kapitole 6.2. Dále, aby byla používaná a měla velkou podporu komunity. To je výhoda v případě, že by se vyskytly nějaké nejasnosti při implementaci. A pochopitelně bezplatnost celé knihovny. Bohužel, knihovna Java3D není součástí JDK a je proto nutná dodatečná instalace. Abych tomu předešel, bude knihovna (verze 1.5.1) součástí systému FOTOM^{NG}.

Java3D je přídavkem ke knihovnám jazyka Java a slouží k zobrazování 3D grafiky. Třídy této knihovny poskytují rozsáhlé rozhraní, dostačující ke tvorbě kvalitních 3D modelů, animací nebo her. Java3D staví na již existujících technologiích, jako je DirectX, OpenGL a také umožňuje začlenit objekty vytvořené 3D modelovacími nástroji, jako je TrueSpace nebo VRML modely [16]. V následujícím textu čerpám z literatury [17].

6.5.1 Vytvoření scény

Pod pojmem scéna, si můžeme představit 3D virtuální vesmír, do kterého umístíme nějaký pozorovaný model a kameru, která jej pozoruje. Java3D nám poskytuje třídy, ze kterých vytvoříme celkový graf scény. Ten je tvořen z různých objektů, které reprezentují například geometrie tvarů, vzhledy tvarů, světla, atd. Graf scény můžeme definovat jako datovou strukturu tvořenou uzly (což jsou instance tříd knihovny Java3D) a vztahy mezi nimi (rodič – potomek nebo reference).

6.5.1.1 Uzly grafu scény V knihovně Java3D rozlišujeme tři základní druhy uzlů. V následujícím seznamu uvedu tyto druhy a popíšu základní zástupce, které budu využívat při samotné implementaci:

- Skupina (*Group*) - *BrancheGroup* (dále jen BG), tato stěžejní skupina umožňuje seskupovat různé objekty scény, usnadní se tím následná manipulace. Dovoluje také aplikovat oddělení nebo připojení skupiny z celkového grafu scény. *TransformGroup* (dále jen TG), hlavním účelem je aplikování transformací na všechny potomky této skupiny. Například si můžeme představit jídelní stůl s deseti židlemi. V případě, že chci každý z těchto nábytků otočit o určitý úhel, musím aplikovat transformaci na každý z nich. Ale pokud jsou potomky TG, stačí nastavit transformaci pouze na tuto skupinu. V praxi se využívá kombinace obou skupin, kdy je TG potomkem BG. Existují i další skupiny, například *OrderedGroup*, *SharedGroup* nebo *Primitive*, která poskytne jednoduché vkládání základních objektů jako je krychle, kužel, válec a koule.

- List (*Leaf*) - Na rozdíl od skupin, listy nemohou mít žádné potomky. *Behavior* nám definuje chování 3D modelu například v reakci na pohyb myši nebo stisk klávesy. Avšak nejdůležitějším listem je *Shape3D*, který specifikuje všechny geometrické objekty a jejich vzhled. Jednotlivé geometrie popíšu v kapitole 6.5.2. Na základě tohoto listu budu schopen vytvářet všechny potřebné 3D modely. Jako další listy mohu uvést například *Fog*, *Light* nebo *Sound*.

- Komponenty uzlů (*NodeComponent*) - Zahrnují všechny komponenty, pomocí kterých můžu definovat vlastnosti geometrií, vzhledů, textur, atd. *Appearance* určuje vzhled objektu, na který bude aplikován. Obsahuje atributy, které určují například barvu, vlastnosti čar, bodů, vykreslení 2D a 3D textu, atd. *Geometry* - zde definujeme geometrie nutné pro *Shape3D*. Mezi další komponenty můžu uvést například *Texture*, *PolygonAttributes*, *ColoringAttributes* nebo *PointAttributes*.

6.5.1.2 Vztahy uzlů grafu scény

Při definování vztahu rodič – potomek mezi jednotlivými uzly grafu scény, musíme dbát určitých pravidel. Například skupiny mohou mít velký počet potomků, avšak vždy jen jednoho rodiče. Jak už bylo řečeno, list nemůže mít žádného potomka a vždy jen jednoho rodiče. Dalším příkladem vztahů jsou reference. Ty použijeme například u komponent uzlů, kdy se u $Shape3D$ odkazujeme na geometrii a vzhled daného tvaru.

6.5.2 Vytvoření objektu

Při vyváření objektů potřebujeme znát několik informací a to, kde chceme objekt vytvořit, jak bude vypadat a z čeho jej vytvoříme. Předem si ujasněme, že Java3D využívá pravotočivé souřadnicové soustavy (kapitola 4.2), kde osa x nabývá kladné hodnoty od středu souřadné soustavy doprava, osa y nahoru a osa z vychází z monitoru směrem k pozorovateli.

Vždy, když modelujeme nějaký 3D model, který je složen z různého počtu tvarů, musíme se rozhodnou, jak tyto tvary vytvoříme. K tomu nám slouží třídy $GeometryArray$. Ty poskytují tvorbu bodů pomocí $PointArray$, vektorů pomocí $LineArray$, trojúhelníků pomocí $TriangleArray$ nebo čtyřúhelníků pomocí $QuadArray$. Nevýhodou těchto tříd je, že neposkytují znovu použití stejných vektorů. Proto se využívají i třídy $LineStripArray$ pro propojení čar, $TriangleStripArray$ pro trojúhelníky, které sdílejí hrany a $TriangleFanArray$ pro trojúhelníky sdílející stejný vrchol. Pro každou z těchto tříd pak definujeme 3D souřadnice jednotlivých bodů. Takto vytvořenou geometrii aplikujeme ($addGeometry()$) tvaru třídy $Shape3D$ spolu s definovaným vzhledem ($setAppearance()$).

Pokud budeme chtít výsledný tvar vystavit transformacím, umístíme jej jako potomka skupiny $TransformGroup$ ($addChild()$). Transformace nám poskytuje třída $Transform3D$, kterou následně aplikujeme ($setTransform()$) na celou skupinu. Skupinu $TransformGroup$ můžeme zastřešit pomocí další skupiny typu $BrancheGroup$. Tím jsme vytvořili vizuální část scény grafu. Následuje příklad zkráceného zdrojového kódu, kdy chceme vytvořit červený obdélník otočený o $90°$:

```
QuadArray qa = new QuadArray(4, QuadArray.COORDINATES);
qa.setCoordinate(0, new Point3d(0.0, 0.0, 0.0));
qa.setCoordinate(1, new Point3d(1.0, 0.0, 0.0)); //atd. pro dalsi dva body
Shape3D shape = new Shape3D();
shape.addGeometry(qa); //pridani geometrie
Appearance app = new Appearance(); //vzhled
app.setColoringAttributes(new ColoringAttributes(new Color3f(Color.RED), ColoringAttributes.
    SHADE_GOURAUD));
shape.setAppearance(app);
```

```
BrancheGroup bg = new BrancheGroup();
TransformGroup tg = new TransformGroup();
Transform3D tr = new Transform3D
tr.rotZ(Math.PI/2);  //rotace
tg.setTransform(tr);  //aplikovani  transformace
tg.addChild(shape);
bg.addChild(tg)
```

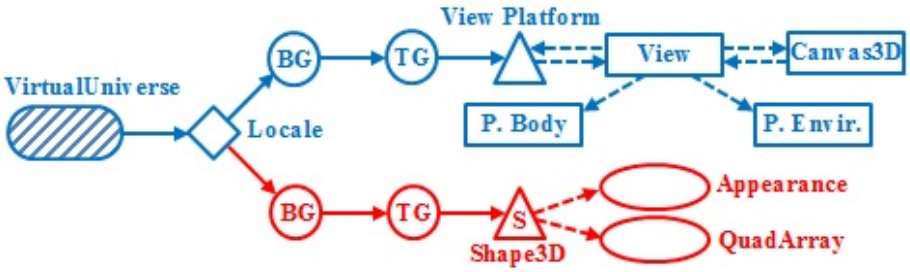
Výpis 1: Příklad vytvoření objektu v Java3D

6.5.3 Vytvoření kamery

Pokud chceme vytvořit kameru, popřípadě kamery, sledující 3D model, potřebujeme definovat její vlastnosti. Základem je $ViewPlatform$, která kontroluje pozici, orientaci a měřítko kamery. Pokud ji umístíme do skupiny $TransformGroup$, budeme schopni s ní manipulovat a nastavit ji podle našich představ. Dalším důležitým prvkem je pohled ($View$), který obsahuje všechny důležité parametry pro vykreslení 3D scény z jednoho bodu pozorování. Parametry jsou myšleny specifika pro samotnou kameru ($PhysicalBody$), fyzické prostředí scény ($PhysicalEnvironment$) a hlavně plátno pro vykreslení 3D grafiky ($Canvas3D$).

6.5.4 Příklad scény 3D virtuálního vesmíru

Na obrázku 9 můžeme vidět, jak vypadá graf scény pro virtuální vesmír, podle pravidel z kapitoly 6.5.1. Modrá barva označuje minimální graf scény nutný pro zobrazení, včetně definování kamery (kapitola 6.5.3). Červená barva označuje část grafu scény pro vizualizaci objektu z kapitoly 6.5.2. Nutno dodat, že $Locale$ slouží jako hlavní kontejner pro skupiny $BrancheGroup$ zobrazovaných objektů a všech kamer.



Obrázek 7: Příklad grafu scény v Java3D

7 Realizace modulu

V předchozích kapitolách jsem se seznámil s teorií a problematikou 3D modelování, analýzou předchůdce a specifikací požadavků na nový modul. Také byly objasněny základy pro práci v knihovně Java3D a na platformě NetBeans. Díky tomu jsem mohl implementovat modul 3D modelování, který je začleněn do systému FOTOM^{NG}.

V této kapitole se věnuji struktuře vytvořeného modulu a zároveň popíšu, jak je řešena funkcionalita v jednotlivých třídách. Konec kapitoly je zaměřen na ověření funkčnosti modulu, včetně ukázky konečného vzhledu. Třídní diagram a závislost balíčků jsou součástí přílohy této diplomové práce.

7.1 Struktura modulu

Při návrhu struktury modulu jsem se snažil zachovat stávající trendy systému FOTOM. Jinými slovy, chtěl jsem dodržet stejnou konvenci, která byla použita i u dalších modulů. Výhodou je pak lehčí údržba, kdy neztrácíme čas pochopením struktury každého modulu. Navíc má každý modul, včetně tohoto, vytvořenou programátorskou dokumentaci. Struktura modulu je rozvržena do následujících pěti balíčků:

1. Balíček prezenční vrstvy - fotom3dmodeling
2. Balíček API - fotom3dmodelingapi
3. Balíček akcí - fotom3dmodelingcontroller
4. Balíček výjimek - fotom3dmodelingexceptions
5. Balíček ikon a textur - fotom3dmodeling.img

Jak už jsem napsal dříve, chtěl jsem zachovat stejnou konvenci, proto je struktura modulu postavena na architektuře návrhového vzoru MVC. Balíček prezenční vrstvy zobrazuje data tříd balíčku pro API. Jedná se například o třídu reprezentující celý 3D model. Co se týká ovládání, tak to je z 90% umístěno přímo v prezenční vrstvě formou různých akcí, které reagují na stisk tlačítka na panelu nástrojů nebo v reakci na událost vyvolanou pohybem myši. Tedy upravují výše zmíněnou třídu 3D modelu a změny zobrazují uživateli skrz prezenční vrstvu. Balíček se zdroji, jako jsou ikony a textury, využívá prezenční vrstva a třídy s výjimkami jsou použity při kontrole snímkové série pro API modulu.

V následujících podkapitolách popíšu jednotlivé balíčky a jejich třídy. Součástí budou ukázky zdrojových kódů a vysvětlení, jak byla řešena samotná funkcionalita.

7.1.1 Balíček prezenční vrstvy

V tomto balíčku jsem implementoval všechny třídy prezenční vrstvy. Jedná se o třídu $TopComponent$, která slouží jako hlavní okno celého modulu. Dále třídy devíti dialogových oken, které slouží například pro nastavení, nápovědu, atd. Balíček také obsahuje třídy reprezentující virtuální vesmír pro 3D model a grafy, česky překlad a soubor `layer.xml`.

`Modeling3dTopComponent.java`

Tato třída, dědící ze třídy $TopComponent$ (kapitola 6.4.3) implementuje samotné GUI modulu 3D modelování, o kterém jsem psal v kapitole 6.3. Konstruktor je volán z akce, která je přiřazená k položce 3D modelování v hlavním menu systému. Následuje kontrola celé snímkové série. V případě úspěchu se spustí metoda $display3D()$, která vytvoří nový virtuální vesmír pro 3D model a grafy. Tím je myšleno přidání modelu a kamer do scény. Tato metoda také vytváří čtyři pohledy, pomocí tříd $Canvas3D$, které implementují jednotlivé kamery. Ke každému 3D plátnu jsou také přiřazeny popisky a adaptéry na různé uživatelské akce, jako jsou například stisk klávesy, táhnutí myší a to podle požadavků uvedených v kapitole 6.2.1. Tím je defaultní zobrazení hotovo a čeká se na nějaký uživatelský vstup.

Nalezneme zde i velké množství událostí, především pak na stisk nějakého tlačítka na panelu nástrojů. V případě, že se jedná o tlačítko pro nastavení funkce, spustí se příslušné dialogové okno. Jinak jsou přímo provedeny metody manipulující s virtuálním vesmírem 3D modelu nebo grafu.

Dialogová okna

Popisovat všech devět dialogových oken by bylo příliš rozsáhlé, většina z nich pracuje na podobném principu, kdy pro zobrazení využívám třídy $DialogDescriptor$. Tato třída v konstruktoru přijímá jako parametr třídu $JPanel$, na které jsem implementoval těla jednotlivých dialogových oken. Konstruktory těchto panelů obsahují parametry, podle kterých se při zobrazení nastaví komponenty. Parametry získává například ze třídy virtuálního vesmíru, atd. Uživatel tak vždy uvidí aktuální nastavení. Mezi tyto panely pro dialogová okna patří například `SettingsPanel.java` pro nastavení objektů a scény, `SetGraphPanel.java` pro nastavení grafů nebo `SetAnimationPanel.java` pro nastavení animací.

`Universe.java`

Tato třída reprezentuje virtuální vesmír pro 3D model. Základem jsou metody pro přidání ($addAllCameras()$) a nastavení ($setAllCameras()$) všech kamer, včetně přidání

samotného modelu do scény ($addModelToScene()$). Jak už vyplynulo ze specifikace požadavků, jsou pohledy jedna, dva a tři odlišné od pohledu čtyři a to ve smyslu interakcí s 3D modelem. Například při rotaci v prvních třech pohledech nesmí rotovat model ve čtvrtém pohledu, atd. Jelikož mi knihovna Java3D neumožnila zkombinovat všechny čtyři pohledy v jednom virtuálním vesmíru, jediným možným řešením bylo umístit první tři kamery s instancí modelu do jednoho kontejneru $Locale$ a čtvrtou kameru s další instancí modelu do druhého kontejneru. Při implementaci jsem tudíž musel počítat se dvěma modely, avšak ve výsledku se to nijak neprojeví, synchronizace je zachována, například změna barvy 3D modelu se projeví na všech pohledech.

Velkou výhodou knihovny Java3D je jednoduché vytváření vlastních nebo už předem implementovaných interakcí uživatele s 3D modelem. Ve výpisu zdrojového kódu 2 můžeme vidět přiřazení rotace čtvrtému pohledu pomocí myši ve všech osách. K rotaci je využitá modifikovaná třída $MyMouseRotate$, o které se zmíním v kapitole 7.1.4.

```
BoundingSphere boundingSphere = new BoundingSphere(new Point3d(0.0,0.0,0.0), Double.
    MAX_VALUE);
mouseRotate4 = new MyMouseRotate(modelTransformGroup);
mouseRotate4.setFactor(0.02, 0.02);
mouseRotate4.setSchedulingBounds(boundingSphere);
modelBranchGroup.addChild(mouseRotate4);
```

Výpis 2: Přiřazení rotace 3D modelu

Třída $BoundingSphere$ definuje kulovou oblast kolem vloženého bodu s požadovaným poloměrem. Tím se zajití oblast, pro kterou bude tato interakce fungovat. Proměnné $modelTransformGroup$ a $modelBranchGroup$ jsou druhy skupin, které jsem popisoval v kapitole 6.5.1.1. Parametry metody $setFactor()$ zajišťují rychlost otáčení v osách x a y. Obdobně jsou interakce aplikovány i v ostatních pohledech, pokud chci zamezit rotaci nebo translaci v určité ose, změním parametr na hodnotu 0.0.

Třída `Universe.java` dále zajišťuje zobrazování 3D modelu v animacích podle časovače ($Swing.Timer$), který je definován ve třídě `Modeling3dTopComponent.java`. Zobrazování 3D modelu jsem ovlivňoval pomocí vzhledu jednotlivých částí modelu, tedy změnou atributů $Appearance$. Atributy jsou myšleny například $ColoringAttributes$ pro barvy, $RenderingAttributes$ pro vykreslování, $TextureAttributes$ pro textury a mnoho dalších. Stejný princip je také využíván u změn mezi drátěným nebo texturovaným modelem, obarvováním objektů ve 3D modelu, atd. Nalezneme zde také metody zajišťující funkcionalitu kalibračních os nebo metodu pro vytvoření středového kříže.

GraphUniverse.java

Tato třída, podobně jako `Universe.java`, vytváří virtuální vesmír pro generovaný graf, podle uživatelsky specifikovaného nastavení skrz příslušné dialogové okno. Na první pohled se může zdát divné, proč vytvářet 2D grafy pod 3D knihovnou. Nechtěl jsem zbytečně aplikovat další knihovnu pro vytváření grafů (například $JFreeChart$), nemusel jsem proto ani řešit vykreslování grafů a s tím spojenou teorii. Navíc pro účely 2D grafů v tomto modulu je knihovna Java3D zcela dostačující.

Součástí jsou opět metody pro vložení, tentokrát grafu do scény a jedné kamery, která tento graf z pevně dané vzdálenosti sleduje. Dále obsahuje metody určené pro dodatečné výpočty průměrných hodnot jednotlivých veličin. Pro procházení jednotlivých bodů grafu je zde také implementována metoda, která pracuje se změnou vzhledu jednotlivých bodů, jako u `Universe.java`.

layer.xml

V kapitole 6.4.2 jsem tento soubor zmiňoval také v souvislosti s registrací akcí, které jsou volány po stisku na příslušnou položku hlavního menu systému. Následující výpis kódu XML 3 ukazuje registraci akce, reprezentovanou třídou pro přidání série snímků, kterou popisuji v kapitole 7.1.4. V bloku `<file></file>` můžeme také definovat ikony, pozici nebo český překlad.

```
<folder name="Actions">
  <folder name="2Dmodeling">
    <file name="org-fotomapp-fotom3dmodelingcontroller-AddFtmForModeling3dAction.
        instance"></file>
  </folder>
</folder>
```

Výpis 3: Registrace akce na spuštění modulu 3D modelování

7.1.2 Balíček API

Do tohoto balíčku jsem implementoval třídy, které tvoří jádro celého modulu. Jedná se především o třídy, které reprezentují samotný 3D model nebo grafy a celou sérii snímků. Dále jsou zde umístěny třídy pro kamery nebo třída realizující tisk jednotlivých pohledů.

FtmSeries.java

Tato třída reprezentuje celou sérii snímků, které se objeví na vstupu modulu při jeho spuštění. Konstruktor je volán ve třídě `Modeling3dTopComponent.java`, kdy kontroluji, zdali série splňuje všechny náležitosti pro 3D modelování. Parametrem konstruktoru

je pole objektů typu $FtmObject$, kde každý obsahuje informace o daném fotogrammetrickém snímku. Tyto informace zahrnují například definované zájmové objekty, které byly na snímku vytvořeny pomocí nástrojů uvedených v kapitole 5.1.1, včetně jejich názvů a hlavně reálných a obrazových souřadnic v levotočivém souřadném systému. Dále definované vlícovací body, pozici snímku v sérii, atd.

Z těchto dat získám vše potřebné pro vytvoření 3D modelu a grafů. Jednotlivá data jsou setříděna. Například pokud snímky obsahují rozdílné nástroje (polygon, kružnice, průsečík, bod), tak jsou rozděleny do příslušných datových struktur, podle daného typu nástroje. Třída dále obsahuje metody, které na základě získaných dat počítají hodnoty obsahu, objemu, plochy pláště nebo těžiště a to podle vzorců uvedených v kapitole 4.6. Také jsou zde implementované metody, které kontrolují splnění podmínek vhodné série pro 3D modelování a mnoho dalších užitečných metod.

`Model.java`

V předchozí kapitole jsem zmiňoval metodu, která přidává 3D model do scény. A právě 3D model je reprezentovaný touto třídou, kdy všechna potřebná data jsou získána ze třídy `FtmSeries.java`.

Nyní vysvětlím, jak jsou jednotlivé tvary umístěny do souřadného systému a jak z nich vytvořím kompletní 3D model. Hned na začátku bylo potřeba určit, jakým způsobem budu vkládat 2D souřadnice získané z dat snímků do 3D prostoru přes třídu $Point3d$, která má jako parametry konstruktoru souřadnice x, y, z (viz výpis 4). Jednak je zde rozdíl mezi typy souřadnicových systémů, musel jsem brát ohled i na interakce poskytnuté knihovnou Java3D, které šly ovlivnit skrz metodu $setFactor()$, ale také na pozici jednotlivých kamer. Nakonec jsem se rozhodl pro následující řešení. Podle orientace os knihovny Java3D popsaných v kapitole 6.5.2, určuje souřadnici x hodnota pozice snímku v sérii. Na souřadnice y se vynáší rozdíl maximální výšky snímku a y souřadnice bodu 2D objektu tak, jak jsem navrhnul řešení problému v kapitole 5.2.1. A na souřadnici z se vynáší x souřadnice bodu 2D objektu.

```
LineStripArray poly = new LineStripArray (point.length+1, LineStripArray.COORDINATES, stripC);
for (int j = 0; j < point.length; j++){
    Point3d polygonPoint = new Point3d(globalZ[n], s[n] - point[j].y, point[j].x);
    poly.setCoordinate(j, polygonPoint);
}
Point3d polygonPoint = new Point3d(globalZ[n], s[n] - point[0].y, point[0].x);
poly.setCoordinate(point.length, polygonPoint);
```

Výpis 4: Vytvoření geometrie polygonu ve 3D prostoru

Ve zjednodušeném výpisu zdrojového kódu 4, je vytvořena geometrie pro polygon získaný z dat jednoho snímku série na pozici n. Třída $LineStripArray$ vytváří lomenou čáru na základě definovaných bodů $polygonPoint$. Za cyklem je definován ještě jeden bod, aby se lomená čára spojila se svým začátkem a vytvořila tak uzavřenou oblast polygonu. Pole $globalZ$ udává pozici snímku v sérii, pole s obsahuje výšky jednotlivých snímků a v poli $point$ jsou uloženy všechny souřadnice polygonu získaných ze snímku. V originálním kódu jsou ještě jednotlivé body vystaveny transformacím, aby se docílilo přesného zobrazení pozorovaného objektu a pochopitelně je celá geometrie přiřazena tvaru $Shape3D$ (viz výpis 1). K danému polygonu se ještě vytváří spojnice, reprezentované geometrií $QuadArray$ (vhodný pro aplikaci textur), aby se zajistila návaznost mezi polygony jednotlivých snímků. Pokud tento postup aplikujeme na celou sérii, na scéně to vytvoří 3D model. Obdobně to funguje i pro kružnici, průsečík a bod.

Ještě je nutné dodat, že veškeré souřadnice jsou v obrazových jednotkách (pixelech) a při výpočtech pro grafy využívám měrných jednotek. Převod na měrné jednotky dosáhnu vynásobením poměrem reálné vzdálenosti vlícovacích bodů ke vzdálenosti vlícovacích bodů v pixelech.

`Graph.java`

Podobně jako tomu je u `Model.java`, tak tato třída reprezentuje graf, který je přidán na scénu pomocí příslušné metody. Jelikož grafy jsou 2D zobrazení, ve všech konstruktorech třídy $Point3d$ ignoruji souřadnici z a vždy ji nastavím na hodnotu 0.0. A samotný graf vynáším pouze v souřadnici x a y.

Postup vytvoření jednotlivých grafů pro obsah, objem, těžiště a plochu pláště je obdobný. Postupně vytvářím jednotlivé geometrie a skládám je do výsledných grafů, tzn. osy grafu, jednotlivé dílky os s popisky, vedlejší pomocné osy a samotnou křivku s body. Na závěr ještě celý graf umístím do správné pozice pomocí transformace posunu, ale to už je jen kvůli vzhledu.

Metody pro výpočty jednotlivých veličin jsou umístěny ve třídě `FtmSeries.java` a dodatečné výpočty ve třídě `GraphUniverse.java`. Podstatou je vždy určit rozsahy hodnot jednotlivých os a pomocí měřítka vynášet body křivky. Osa y je určena sledovanou veličinou a na osu x vynáším pozice snímků v sérii. Dále jsem musel vyřešit vykreslování textu a to pro popisky grafů nebo pro popisky jednotlivých pohledů. Ale o tom až v následující části, kterou věnuji kamerám.

`Camera.java` a `GraphCamera.java`

Obě třídy nám reprezentují kameru, která pozoruje 3D model nebo graf. V případě 3D modelu jsou vytvořeny čtyři instance této třídy, každá pro jiný pohled. V těchto třídách se

definují všechny vlastnosti, které jsem uvedl v kapitole 6.5.3. Velkou výhodou je přistoupit skrz metodu $postRender()$ k plátnu $Canvas3D$ a zapsat na něj text, přesně jak je to vidět na výpisu 5.

```
@Override
public void postRender() {
    this.getGraphics2D().setColor(Color.white);
    this.getGraphics2D().setFont(new Font("Italic",  Font.BOLD, 11));
    this.getGraphics2D().drawString("string",  xPos, yPos);
    this.getGraphics2D().flush(false);
}
```

Výpis 5: Výpis textu na 3D plátno

Tento způsob zápisu má výhody například ve vysoké kvalitě zobrazeného textu, možnost nastavovat styl, barvu a velikost nebo kdykoliv změnit obsah řetězce. Avšak velikou nevýhodou je obtížné pozicování textu, kdy musíme zadat x a y souřadnici bodu, odkud bude text vykreslován. Navíc jsem zjistil, že pokud uživatel bude chtít zapsat několik řádků textu pod sebe a pro vynechání použije prázdný řetězec, metoda $postRender()$ to pochopí jako poslední řetězec a celé vykreslení se ukončí. Řešením je použití řetězce s mezerou.

Tento typ zápisu textu je výhodný například u vykreslení různých informací, jako jsou například popisky jednotlivých pohledů, názvy pohledů, atd. Avšak u dynamicky měnících se pozic textů jsou už nedostačující. Proto existuje další způsob, využitý například u popisků dílků v grafu nebo u kalibračních os, pomocí třídy $Text2D$. Ta rovněž umožňuje změnu velikosti, barvy nebo stylu písma, avšak s výhodou použití skupin $BrancheGroup$ a $TransformGroup$.

`Print.java`

Tato třída poskytuje tisk komponenty $JScrollPane$, takže můžeme tisknout všechny zobrazené pohledy, včetně grafů a popisků. Navíc poskytne volbu vzhledu stránky tak, jak to bývá u většiny editorů. S příslušným softwarem můžeme tisknout i do souborů `PDF` nebo obrázkových formátů `JPEG`, `BMP`, atd.

7.1.3 Balíček akcí

`AddFtmForModeling3dAction.java` a `AddNextFtmForModeling3dAction.java`

V tomto balíčku nalezneme tyto dvě třídy. Obě jsou registrované v souboru `layer.xml`. První třída slouží pro otevření celého okna 3D modelování (viz výpis 6). Také zajišťuje, že položku hlavního menu nelze označit, dokud nebudou v okně prohlížeče označeny

minimálně dva soubory. Druhá funguje pro přidání další série k našemu již existujícímu 3D modelu. Můžeme tak porovnávat obě série mezi sebou. Akci nelze provést, dokud není aktivní okno 3D modelování a nejsou označeny minimálně dva soubory v prohlížeči.

```
Modeling3dTopComponent tc = new Modeling3dTopComponent();
if (tc.VerifyFtmSeries(activatedNodes)){
    tc.open();
    tc.requestActive();
}
```

Výpis 6: Otevření okna modulu 3D modelování

`MyMouseRotate.java`

Hlavním účelem vytvoření této třídy pro interakci rotování s 3D modelem byla schopnost získat a zaznamenat daný úhel rotace v osách x a y. Tyto údaje jsou využity v popiscích jednotlivých pohledů.

7.1.4 Balíček výjimek a balíček ikon a textur

V balíčku výjimek je umístěno šest tříd, které jsou vyvolány v případě, že nebyla splněna podmínka pro přidání správné série snímků pro 3D modelování. Tyto podmínky jsou popsány v kapitole 6.2.6. Jedná se například o třídy výjimek - `ObjectException.java`, `UnitsException.java`, `PolygonPointsException.java`, atd.

V posledním balíčku jsou uchovány všechny použité ikony a textury. Ve zdrojovém kódu pak přímo využívám URL k jednotlivým obrázkům jako parametr metody $getClass().getResource(url)$.

7.2 Použitý software při vývoji

Nový modul 3D modelování jsem implementoval ve vývojovém prostředí NetBeans IDE verze 7.4, včetně nainstalovaného JDK verze 1.7.0.51. Dále byla nutná i instalace balíčku JAI verze 1.1.3 pro JDK a JAI CLASSPATH verze 1.1.3. Realizace 3D grafiky proběhla díky balíčku Java3D, který je nyní součástí systému a není jej nutno instalovat. Veškerý software, nutný pro správnou funkci systému FOTOM^{NG}, je umístěn v příloze této diplomové práce.

7.3 Ověření funkčnosti

Celý modul 3D modelování byl kompletně implementován podle specifikace požadavků uvedených v kapitole 6.2. V první fázi testování, jsem se zaměřil na kontrolu série, která

musí splňovat podmínky pro správnou funkčnost modulu. Vytvořil jsem testovací sérii, která úmyslně nesplňovala jednotlivé podmínky. **Test proběhl v pořádku.**

Dále jsem testoval jednotlivé uživatelské vstupy proti špatnému zadávání hodnot. Jelikož byly všechny, například číselné vstupy, ošetřeny komponentou $JSpinner$, které byl nastaven model pouze pro zadávání čísel v určitém rozmezí, nehrozí špatné zadání parametrů. **Test proběhl v pořádku.**

Pro vývoj a testování funkcí modulu jsem vytvořil sérii sedmi snímků, které obsahovaly zájmové objekty. Tyto zájmové objekty byly vytvořeny všemi druhy nástrojů pro tvorbu objektů, tedy polygon, kružnice, průsečík a bod. Podmínkou bylo, že například polygon stejného objektu nebyl na všech snímcích. To ve výsledku znamená přerušení 3D modelu v určitém místě, jelikož nejsou pro tento řez data dostupná. Na této sérii jsem otestoval všechny funkce ze specifikace požadavků. **Test proběhl v pořádku.**

Dále jsem testoval ovlivňování funkcí jinýma. Například, pokud nám probíhá cyklická animace, nesmí její podstatu ovlivnit použití jiné funkce, třeba změna projekce. Jinými slovy, nesmí to vyvolat žádnou chybu. Otestoval jsem všechny možné kombinace. **Test proběhl v pořádku.**

I přesto, že funkce pracují správně a nevyvolají žádnou chybu, může se uživatel svým nedopatřením dostat do situace, kdy je mu znemožněno správně používat celý modul. Uvažujme například funkci přední nebo zadní stěny. Uživatel dlouhým pohybem myši posune stěnu do takové pozice, že mu zmizí celý 3D model. A navíc tato stěna může být v neznámé vzdálenosti. Proto jsem jednotlivé funkce ošetřil tak, aby k takovým situacím nedocházelo. Konkrétně na příkladu, který jsem zmínil, je pohyb stěny omezen jen na interval dvojnásobné délky 3D modelu. Navíc se při vypnutí této funkce, stěny vrátí do defaultní polohy. V záloze je ještě k dispozici tlačítko pro reset celé scény. Takto jsem otestoval všechny funkce. **Test proběhl v pořádku.**

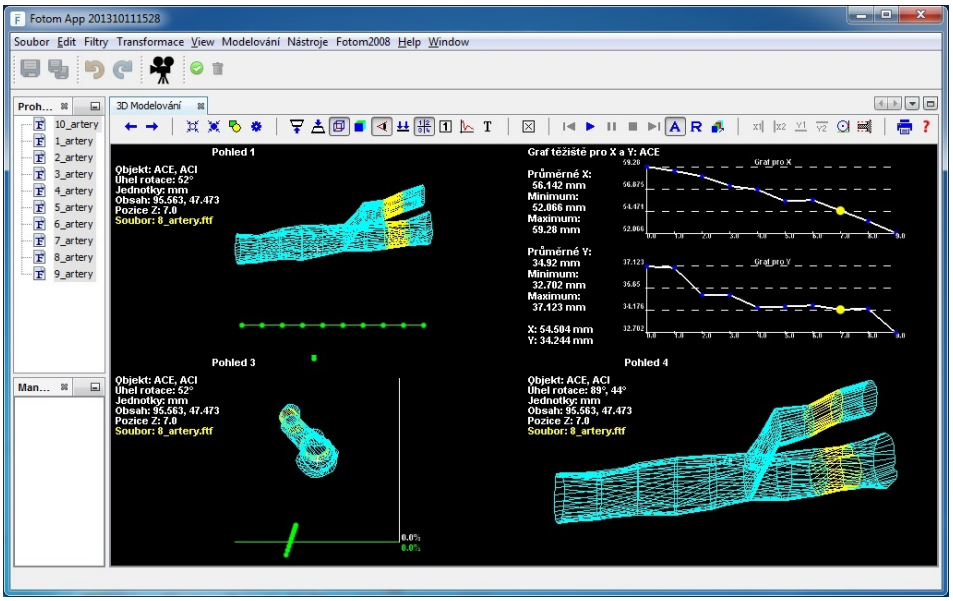
Nutno dodat, že vše bylo ještě jednou testováno na jiném stroji, včetně instalační i archivované distribuce. **Test proběhl v pořádku.**

7.3.1 Shrnutí testování modulu 3D modelování

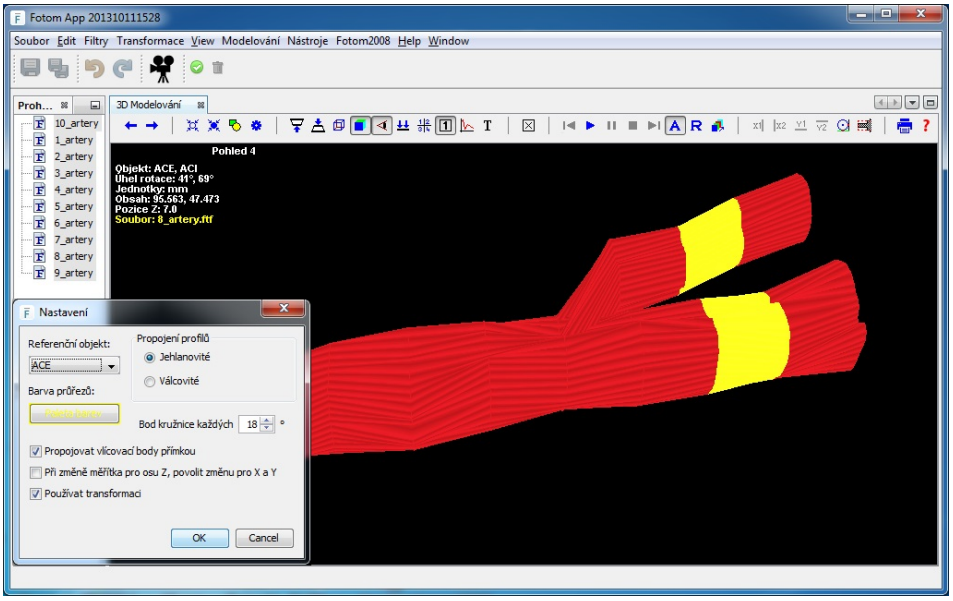
Po kontrole funkčnosti celého modulu, jsem dospěl k názoru, že je připraven k běžnému používání. Nicméně, vždycky může dojít, i přes důkladné testování, k výskytu drobných chyb. To může být v mém případě důsledkem toho, že jsem tento modul testoval i implementoval. V praxi tyto fáze provádí odlišné osoby. Avšak chyby se dají kdykoliv odstranit.

7.4 Konečný vzhled modulu 3D modelování

V následující kapitole uvedu dva obrázky výsledného vzhledu uživatelského rozhraní modulu 3D modelování. Další ukázky jsou uloženy v příloze této diplomové práce.



Obrázek 8: Modul 3D modelování - MultiView, drátěný model



Obrázek 9: Modul 3D modelování - SingleView, texturovaný model

8 Závěr

Tvorba této diplomové práce si prošla dlouhodobým procesem. Na začátku jsem věnoval svoji pozornost teoretickým základům. S pojmem fotogrammetrie jsem se už setkal při zpracování bakalářského projektu a stále více mne překvapuje, jak můžeme tento vědní obor aplikovat v různých odvětvích běžného života. Dále jsem se seznámil s teorií 3D modelování a provedl jsem krátký výzkum ve světě. Účelem bylo zjistit, jak si tato metoda vizualizace stojí na světové úrovni, kde nachází své využití, jakých technologických pokroků dosáhla a zdali se někdo zabývá vývojem nástroje, který je podobný systému FOTOM^{NG} a jeho funkcionalitě. V další fázi projektu jsem se zaměřil právě na tento systém. Provedl jsem analýza stavu systému a jeho nástrojů pro definování zájmových objektů. Tyto objekty určují vstup implementovaného modulu a na jejich základě je vytvářen celý 3D model. Souběžně jsem se také věnoval analýze předchozí verze. Cílem bylo dosáhnout ucelené specifikace požadavků doplněné o vlastní nápady a modifikace, popřípadě odhalit chyby a předejít jim při tvorbě nového modulu.

Samotné implementaci předcházela studie platformy NetBeans a knihovny Java3D, která je určená pro tvorbu 3D grafiky. Nevýhodou této knihovny byla nutnost dodatečné instalace. Z pohledu uživatele je už tak nepraktické instalovat, kromě samotného systému, ještě JDK a JAI. Proto jsem knihovnu integroval jako nový modul systému. Hotový modul 3D modelování je důležitou částí, která doposud v systému FOTOM^{NG} chyběla. Se svou vysokou škálou funkcí poskytne uživateli prostředek pro 3D vizualizaci, ale především nástroj pro měření těchto objektů. Dnes je k dispozici velké množství softwaru, který vytváří 3D modely, avšak s absencí funkcí pro jejich měření. V novém modulu máme možnost práce s grafy, měření odchylek, pozorovat změny různých veličin, změny jednotlivých profilů, atd.

Při vývoji jsem musel řešit několik problémů. Jednalo se například o použití souřadnic 2D snímků ve 3D prostoru. Dále jsem potřeboval vymyslet řešení pro jednotlivé nedostatky, které se vyskytly v předchozí verzi. Problémem bylo také kombinovat jednotlivé interakce pohledů spolu s jejich synchronizací. I samotné nastudování knihovny Java3D vyžadovalo svůj čas. Většině řešení problémů se věnuji v příslušných kapitolách.

Koncept celého systému FOTOM^{NG} má velkou budoucnost a už teď poskytuje uživateli komplexní nástroje pro práci se snímky. Jednou z posledních novinek jsou funkce pro automatické detekce zájmových objektů. To posouvá celý projekt zase o krok dál. V nejbližší době se plánují funkce využívající radonové transformace pro 3D modelování nebo voxelové grafiky. Tato diplomová práce mi pomohla rozšířit své znalosti v oblasti 3D grafiky a moderní fotogrammetrie. Jsem velice vděčný za to, že mi bylo umožněno se na tomto projektu podílet.

9 Reference

[1] BORRMANN, Dorit, Hassan AFZAL, Jan ELSEBERG a Andreas NUCHTER. Thermal 3D modeling of indoor environments for saving energy. *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems* [online]. IEEE, 2012, s. 4538-4539 [cit. 2014-03-15]. DOI: 10.1109/IROS.2012.6386265. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6386265

[2] BERRANEN, Yacine, Mitsuhiro HAYASHIBE, Benjamin GILLES a David GUIRAUD. 3D volumetric muscle modeling for real-time deformation analysis with FEM. *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society* [online]. IEEE, 2012, s. 4863-4866 [cit. 2014-03-15]. DOI: 10.1109/EMBC.2012.6347083. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6347083

[3] KOWDLE, Adarsh, Dhruv BATRA, Wen-Chao CHEN a Tsuhan CHEN. *IModel: Interactive Co-segmentation for Object of Interest 3D Modeling* [online]. s. 211 [cit. 2014-03-15]. DOI: 10.1007/978-3-642-35740-4_17. Dostupné z: http://link.springer.com/10.1007/978-3-642-35740-4_17

[4] WU, Jianjun, Mei LI a Zhenming SUN. Three-dimension geological modeling and its application of Wangjialing coal mine. *2013 21st International Conference on Geoinformatics* [online]. IEEE, 2013, s. 1-4 [cit. 2014-03-15]. DOI: 10.1109/Geoinformatics.2013.6626152. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6626152

[5] DIMMELER, Alwin, Hendrik SCHILLING, Michal SHIMONI, Dimitri BULATOV, Wolfgang MIDDELMANN, Gary W. KAMERMAN, Ove K. STEINVALL, Gary J. BISHOP a John D. GONGLEWSKI. *Combined airborne sensors in urban environment* [online]. 88970U- [cit. 2014-03-15]. DOI: 10.1117/12.2028648. Dostupné z: http://proceedings.spiedigitallibrary.org/proceeding.aspx?doi=10.1117/12.2028648

[6] LEE, Yong Yi, Min Ki PARK, Jae Doug YOO a Kwan H. LEE. Multi-scale feature matching between 2D image and 3D model. *SIGGRAPH Asia 2013 Posters on - SA '13* [online]. New York, New York, USA: ACM Press, 2013, s. 1-1 [cit. 2014-03-15]. DOI: 10.1145/2542302.2542320. Dostupné z: http://dl.acm.org/citation.cfm?doid=2542302.2542320

[7] GRUM, Matthew a Adrian G. BORS. 3D modeling of multiple-object scenes from sets of images. *Pattern Recognition* [online]. 2014, vol. 47, issue

1, s. 326-343 [cit. 2014-03-15]. DOI: 10.1016/j.patcog.2013.04.020. Dostupné z: http://linkinghub.elsevier.com/retrieve/pii/S0031320313002197

[8] KLIMENTJEW, Denis, Sebastian ROCKEL a Jianwei ZHANG. *Active Scene Analysis Based on Multi-Sensor Fusion and Mixed Reality on Mobile Systems* [online]. s. 795 [cit. 2014-03-15]. DOI: 10.1007/978-3-642-37835-5_69. Dostupné z: http://link.springer.com/10.1007/978-3-642-37835-5_69

[9] RINGENBERG, Jordan, Makarand DEO, Vijay DEVABHAKTUNI, Omer BERENFELD, Brett SNYDER, Pamela BOYERS a Jeffrey GOLD. Accurate reconstruction of 3D cardiac geometry from coarsely-sliced MRI. *Computer Methods and Programs in Biomedicine* [online]. 2014, vol. 113, issue 2, s. 483-493 [cit. 2014-03-15]. DOI: 10.1016/j.cmpb.2013.11.013. Dostupné z: http://linkinghub.elsevier.com/retrieve/pii/S0169260713003854

[10] LIČEV, Lačezar. *Analýza, modelování, rozpoznávání a vizualizace procesu měření objektů na snímcích.* Vyd. 1. Brno: Computer Press, 2010, 125 s. ISBN 978-80-251-3296-8.

[11] HENDRYCH, Jakub. *2D animace procesu měření.* Ostrava, 2012. Bakalářská práce. Vysoká škola Báňská - Technická univerzita Ostrava, Fakulta elektrotechniky a informatiky, Katedra informatiky.

[12] HLAVÁČEK, Lukáš. *Analýza objektu z 3D modelu.* Ostrava, 2008. Diplomová práce. Vysoká škola Báňská - Technická univerzita Ostrava, Fakulta elektrotechniky a informatiky, Katedra informatiky.

[13] HUDEČEK, Tomáš. *Rozpoznávání objektů na snímcích v medicíně* Ostrava, 2013. Diplomová práce. Vysoká škola Báňská - Technická univerzita Ostrava, Fakulta elektrotechniky a informatiky, Katedra informatiky.

[14] PYTLÍK, Tomáš. *2D počítačové zpracování fotografie na PC* Ostrava, 2011. Diplomová práce. Vysoká škola Báňská - Technická univerzita Ostrava, Fakulta elektrotechniky a informatiky, Katedra informatiky.

[15] BÖCK, Heiko. *Platforma NetBeans: podrobný průvodce programátora.* Vyd. 1. Brno: Computer Press, 2010, 320 s. ISBN 978-80-251-3116-9.

[16] *Java 3D pro začátečníky* [online]. [cit. 2013-11-25]. Dostupné z: http://www.java3d.org/czech.html

[17] *Java 3D Tutorial* [online]. 2010, 2010-02-12 [cit. 2013-12-02]. Dostupné z: http://www.cs.stir.ac.uk/courses/ITNP3B/Java3D/

Přílohy

Přílohy této diplomové práce jsou uloženy na přiloženém DVD, které obsahuje:

1. Zdrojové kódy modulu 3D modelování
2. Zdrojové kódy systému FOTOM^{NG} s integrovaným modulem 3D modelování
3. Distribuce systému formou instalačního souboru a komprimovaného archivu
4. Dokument samotné diplomové práce ve formě PDF a DOC
5. Uživatelská příručka pro 3D modelování
6. Kompletní uživatelská příručka včetně 3D modelování
7. Programátorská dokumentace
8. Instalační soubory softwaru nutného pro běh systému FOTOM^{NG}
9. Diagramy – třídní a závislost balíčků
10. Testovací měření
11. Obrázky uživatelského rozhraní modulu 3D modelování