

**SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE**

Fakulta informatiky a informačných technológií

**VIZUALIZÁCIA INFORMÁCIÍ V OBOHATENEJ REALITE**

Dokumentácia k riadeniu projektu

Členovia tímu č.4:

BC. MATÚŠ CIMMERMAN

BC. IRINA DYOMINA

BC. MICHAL FAŠÁNEK

BC. JAROSLAV GAZDÍK

BC. DENIS ILLÉS

BC. FILIP JURČACKO

BC. DALIBOR MÉSZÁROS

Predmet:

Tímový projekt I

Vedúci:

Ing. Peter Kapec, PhD.

Akademický rok:

2015/2016

# Obsah

Úvod.....	3
<b>1 Členovia tímu.....</b>	<b>3</b>
1.1 Rozdelenie manažérskych úloh .....	5
1.2 Podiel práce .....	5
<b>2 Aplikácie manažmentov .....</b>	<b>6</b>
2.1 Manažment dokumentácie .....	6
2.2 Manažment podpory vývoja .....	6
2.3 Manažment testovania.....	7
2.4 Manažment prehliadky kódu .....	7
2.5 Manažment rizík .....	7
2.6 Manažment plánovania úlohy.....	8
2.7 Manažment vývoja.....	8
<b>3 Sumarizácia šprintov .....</b>	<b>9</b>
3.1 Šprint 1 .....	9
3.2 Šprint 2 .....	9
3.3 Šprint 3 .....	9
3.4 Šprint 4 .....	10
3.5 Šprint 5 .....	10
<b>4 Používané metodiky.....</b>	<b>11</b>
4.1 Metodika písania a komentovania zdrojového kódu.....	11
4.2 Metodika vytvárania úloh v nástroji na evidenciu úloh .....	11
4.3 Metodika komunikácie .....	11
4.4 Metodika manažmentu verzií .....	12
4.5 Metodika testovania .....	12
4.6 Metodika prehliadky kódu.....	12
4.7 Globálna retrospektíva .....	12
<b>5 Zoznam kompetencií tímu .....</b>	<b>14</b>
<b>Príloha A Metodiky .....</b>	<b>15</b>
A.a Metodika prehliadky kódu.....	15
A.b Metodika komunikácie .....	17
A.c Metodika manažmentu verzií .....	20

A.d	Metodika písania a komentovania zdrojového kódu.....	22
A.e	Metodika vytvárania úloh v nástroji na evidenciu úloh .....	27
A.f	Metodika testovania .....	28
<b>Príloha B</b>	<b>Zápisnice.....</b>	<b>32</b>
B.a	Prvý týždeň .....	32
B.b	Druhý týždeň.....	33
B.c	Tretí týždeň.....	34
B.d	Štvrtý týždeň.....	35
B.e	Piaty týždeň .....	36
B.f	Šiesty týždeň.....	38
B.g	Siedmy týždeň.....	42
B.h	Ôsmy týždeň .....	45
B.i	Deviaty týždeň .....	48
B.j	Desiaty týždeň.....	51
B.k	Jedenásty týždeň.....	53
B.l	Dvanásty týždeň.....	56
<b>Príloha C</b>	<b>Exporty úloh z Jiry.....</b>	<b>58</b>
C.a	Úlohy z prvého šprintu .....	58
C.b	Úlohy z druhého šprintu .....	61
C.c	Úlohy z tretieho šprintu.....	64
C.d	Úlohy zo štvrtého šprintu .....	69
C.e	Úlohy z piateho šprintu .....	73
<b>Príloha D</b>	<b>Retrospektívy šprintov .....</b>	<b>77</b>
D.a	Po prvom šprinte.....	77
D.b	Po druhom šprinte .....	78
D.c	Po treťom šprinte .....	79
D.d	Po štvrtom šprinte.....	80
D.e	Po piatom šprinte.....	81

# Úvod

Tento dokument slúži ako dokumentácia k predmetu Tímový projekt na Fakulte informatiky a informačných technológií. Opisuje prácu na projekte Vizualizácia informácií v obohatenej realite a to hlavne z pohľadu riadenia a manažmentu v tíme.

V kapitole 1 sú opísaní členovia tímu, ich role a podiel práce na dokumentácii. Kapitola 2 detailne opisuje aplikované manažmenty pre riadenie v našom tíme. Sumár doterajších šprintov je opísaný v kapitole 3. Metodiky pre podporu riadenia a manažmentu sú stručne popísané v kapitole 4, pričom ich celé znenie je priložené v prílohe A.

V prílohe B zápisnice a v prílohe C exporty z nástroja pre manažment tímu. V prílohe D sa nachádzajú retrospektívy z doterajších šprintov.

## 1 Členovia tímu

BC. MATÚŠ CIMMERMAN

- absolvent študijného odboru informatika na bakalárskom stupni na FIIT. Aktuálne študent programu informačné systémy na inžinierskom stupni. Najviac programuje v jazyku Java a Scala. Pracuje v Orange Slovensko ako IT Trainee v OSS & BI engineering tíme. Vo voľnom čase sa venuje vývoju pre platformu Android a projektom, ktoré používajú RaspberryPi

BC. IRINA DYOMINA

- absolventka študijného odboru aplikovaná informatika na Paneurópskej vysokej škole. V súčasnosti študuje softvérové inžinierstvo na FIIT STU a pracuje v Asseco Central Europe ako IT Tester. Preferuje programovanie v Jave, vo voľnom čase sa venuje web programovaniu a počítačovej grafike.

BC. MICHAL FAŠÁNEK

- študent informačných systémov, preferujúci vývoj v Java. V súčasnosti pracuje ako technická podpora v Hewlett Packard a webový administrátor - junior pre Fabrici consulting. Vo voľnom čase záujmovo študuje oblasť umelej inteligencie.

#### BC. JAROSLAV GAZDÍK

- študent informačných systémov na FIIT STU. Preferuje programovací jazyk Java. Momentálne pracuje ako Java Junior Developer v Atose, odkiaľ má skúsenosti s tvorbou webových stránok a testovaním aplikácií pomocou JUnit testov. Vo voľnom čase sa venuje oblasti počítačového videnia a počítačovej grafiky.

#### BC. DENIS ILLÉS

- študent informačných systémov, ktorá na bakalárskom štúdiu programoval prevažne v jazyku C rozšíreného do knižnicu CUDA. Preferuje programovanie v Jave. Vo voľnom čase sa venuje úprave a vytváraniu grafiky, tvorbe propagačných materiálov v nástrojoch od Adobe (PhotoShop, InDesign).

#### BC. FILIP JURČACKO

- študent softvérového inžinierstva, z programovacích jazykov preferuje C a Javu. Má ročnú pracovnú skúsenosť s vývojom webových aplikácií na pozícií Junior Java Developer a momentálne pracuje ako Junior Malware Analyst v ESETe. Vo voľnom čase sa venuje programovaniu Android aplikácií a vzdelávaniu v oblasti počítačovej bezpečnosti.

#### BC. DALIBOR MÉSZÁROS

- absolvoval bakalársky stupeň študijného oboru informatika na FIIT STU. Vo všeobecnosti preferuje programovací jazyk C++, alternatívne programuje v Jave. Má základné skúsenosti s počítačovou grafikou a tvorbou webových stránok. Vo voľnom čase sa venuje štúdiu japončiny a písaniu skriptov (napr. AutoHotkey).

## ***1.1 Rozdelenie manažérskych úloh***

Člen tímu	Rola v tíme
BC. MATÚŠ CIMMERMAN	Manažér komunikácie, rozvrhu a plánovania
BC. IRINA DYOMINA	Manažér rozsahu projektu
BC. MICHAL FAŠÁNEK	Manažér webu a podpory vývoja
BC. JAROSLAV GAZDÍK	Manažér dokumentácie
BC. DENIS ILLÉS	Manažér rizík
BC. FILIP JURČACKO	Manažér vývoja
BC. DALIBOR MÉSZÁROS	Manažér kvality a hlavný architekt

## ***1.2 Podiel práce***

BC. MATÚŠ CIMMERMAN

- dok. k riadenie: úvod, podiel práce, rozdelenie manažérskych úloh, manažment podpory vývoja, manažment rizík, manažment plánovania úlohy, manažment vývoja, sumarizácia šprintov 1 a 2
- dok k inž. dielu: globálne ciele, úvod

BC. IRINA DYOMINA

- dok. k riadenie: metodika prehliadky kódu, manažment prehliadky kódu
- dok k inž. dielu: moduly systému, globálne ciele

BC. MICHAL FAŠÁNEK

- dok. k riadenie: úvod, zoznam kompetencií tímu, globálna retrospektíva
- dok k inž. dielu: moduly systému, globálne ciele

BC. JAROSLAV GAZDÍK

- dok. k riadenie: metodika manažmentu verzií, manažment dokumentácie
- dok k inž. dielu: moduly systému, globálne ciele

BC. DENIS ILLÉS

- dok. k riadenie: metodika vytvárania úloh v nástroji na evidenciu úloh, manažment rizík, sumarizácia šprintu 3
- dok k inž. dielu: úvod, globálne ciele, inštalčná príručka
- formátovanie dokumentov

BC. FILIP JURČACKO

- dok. k riadenie: metodika testovania, manažment testovania
- dok k inž. dielu: inštalčná príručka, globálne ciele

BC. DALIBOR MÉSZÁROS

- dok. k riadenie: metodika komunikácie, manažment vývoja
- dok k inž. dielu: inštalčná príručka, globálne ciele

## 2 Aplikácie manažmentov

Každý člen je zodpovedný za príslušnú oblasť manažmentu. Na dodržiavanie metodík a manažment danej oblasti zodpovedá príslušný manažér.

### 2.1 Manažment dokumentácie

Dokumentácia k projektu bola vytváraná od začiatku práce na projekte. Na pravidelnom týždennom stretnutí sa zakaždým spisuje zápisnica. Túto zápisnicu má na starosti každý týždeň iný zapisovateľ. Zápisnicu tvorí vyhodnotenie úloh z minulého týždňa, písaný obsah priebehu stretnutia a zadanie úloh na budúci týždeň.

Na konci každého šprintu bola spísaná retrospektíva šprintu. Táto retrospektíva rekapituje, čo sa nám počas šprintu podarilo, čo nešlo podľa našich predstáv a čo by bolo treba zlepšiť do budúcnosti.

Dokumentácia inžinierskeho diela bola vytvorená v spoločnom Google Docu. Prácu na dokumentácii rozdeľuje medzi členov tímu primárne manažér dokumentácie podľa toho, čo kto robil a čo je jeho úlohou v tíme.

### 2.2 Manažment podpory vývoja

Manažment podpory vývoja riadi problematiku správneho písania a komentovania zdrojových kódov v tomto projekte. Je dôležité, aby sa metodika, ktoré boli pre túto oblasť vytvorené, riadil každý člen tímu. Metodika bola čiastočne formalizovaná na konci tretieho šprintu. Do konca zimného semestra bude komunikovaná vo finálnej verzii a upravovaná len ad-hoc.

Na tento manažment priamo nadväzuje manažment prehliadky kódu kde sme schopní včas, pred zaradením zdrojového kódu do projektu, odhaliť nedostatky, ktoré vznikli počas vývoja. Najčastejšie to môže byť nedostatočné alebo nesprávne komentovanie kódu podľa predpisu. Či zlé formátovanie zdrojového kódu.

Pre uľahčenie formátovania si každý nastavil editor vývojového prostredia rovnako. Zodpovednosť za kód, ktorý je odovzdaný na prehliadku a žiadosť o zaradenie do projektu, nesie programátor. Každý by si mal teda dodržiavať príslušnú metodiku a riadiť sa ňou v čo najväčšej možnej miere, výnimky môžu nastať.

### ***2.3 Manažment testovania***

Každý si testuje svoj vlastný kód, a ak je to možné, vytvorí aj unit testy, ktoré vykonajú zvolený kód s daným očakávaním výsledkom.

Pred tým, ako sa ukončí šprint, prebehne regresné a akceptačné testovanie. Akceptačné testovanie má za úlohu overiť správnosť pridanej funkcionality. Regresné testovanie má za úlohu overiť, či sme nepokazili iný starší kód ktorého sa nové zmeny priamo netýkajú.

V prípade že odhalené chyby nevieme opraviť ihneď, sú k nim vytvorené úlohy v JIRE, ktoré sa následné riešia v ďalšom šprinte.

### ***2.4 Manažment prehliadky kódu***

V tímoch, kde pracujú viacerí programátori, je dobre pre dosiahnutie väčšej kvality robiť prehliadky kódu. Po dokončení napísania kódu programátor posunie svoju úlohu v JIRE do „Ready to Review“ a dohodne sa s ostatnými členmi tímu, kto bude robiť prehliadku kódu. Nový kus kódu sa prečíta, zjavné chyby budú ihneď opravené. Správca prehliadky kódu overí či je kód porozumiteľný, či je dostatočne okomentovaný, skontroluje formátovanie a dodržiavanie pravidiel napísania kódu v jazyku C++. Následne kód prejde overenie code-review nástrojmi a kontrolu warningov pri kompilácii, preverí sa ako po týchto zmenách funguje program. V prípade nefunkčnosti alebo pri chybách v kóde, autor kódu bude oznámený že treba svoju prácu prerobiť. Keď je všetko v poriadku, úlohu považujeme za spravenú a predlžujeme pracovať nad projektom ďalej.

### ***2.5 Manažment rizík***

Od prvých stretnutí sme priebežne identifikovali niekoľko rizík, ktoré môžu negatívne ovplyvniť vývoj projektu. Skutočnosť, že pracujeme na rozbehnutom a pomerne veľkom projekte je najväčšie riziko, s ktorým sme sa museli od začiatku aktívne vysporiadať. Museli sme



pochopiť dôležité časti zdrojového kódu, o čo sme sa aktívne snažili štúdiom kódu, opravovaním chýb, či konzultovaním s členmi tímu a tiež s vedúcim projektu. Ďalšie podstatné riziká, ktoré by bolo potrebné opisovať sme neidentifikovali.

## ***2.6 Manažment plánovania úloh***

Pre manažment a plánovanie úloh sme po otestovaní viacerých nástrojov, zvolili systém JIRA licencovaný pre fakultu. Na začiatku prvého šprintu sme naplnili backlog a identifikovali kategórie úloh. Kategórie úloh budú v priebehu projektu pribúdať, pretože mapujú epic míľniky.

Na začiatku každého šprintu sme diskutovali o úlohách a o ich výbere z backlogu do začínajúceho šprintu. Každú úlohu sme ohodnotili story pointami. Následne Scrum master pridelil úlohy každému z členov tímu. V polovici šprintu sme zhodnotili stav úloh a prípadne podnikli potrebné kroky, aby sme šprint ukončili čo najlepšie.

Na konci každého šprintu sme spravili retrospektívu za posledným šprintom a zhodnotili čo sme robili zle, ale hlavne čo a ako môžeme zlepšiť do budúceho šprintu. Priebežný status úloh exportujeme na granularite jedného týždňa, aby sme mali stopu o priebehu úloh počas šprintu.

## ***2.7 Manažment vývoja***

Pri projekte o takejto veľkosti bolo nutné použiť nástroje pre manažment verzií, zvolili sme si nástroj Git. Na projekte pokračujeme z existujúceho projektu, ktorý sa nachádza v vzdialenom repozitári na Githube. Tu sme si vytvorili vlastnú kópiu projektu, ktorú budeme počas dvoch semestrov udržiavať, pričom na konci by sa tieto projekty mali spojiť.

Práca s verziovacím nástrojom Git je riadená príslušnou metodikou. V skratke, každá nová funkcionálna, ktorú implementuje jeden z členov by mala byť vytváraná v samostatnej vetve. Po dokončení práce vo vetve je odoslaná vo forme pull-requestu (zlúčenie) do develop vetvy vo vzdialenom repozitári. Pred zlučovaním (angl. merge) musí byť vykonaná prehliadka kódu aspoň dvomi členmi, toto opäť podrobne opisuje príslušná metodika.

Pri vývoji je dôležité aby boli zmeny komitované (angl. commit) do vetvy v primerane veľkých, resp. malých inkrementoch z dôvodu prehľadnosti zmien, ktoré boli vykonané. Na konci každého šprintu musia byť všetky hotové funkcionality spojené v develop vetve vo vzdialenom repozitári a následne by mali byť spojené do hlavnej vetvy, ktorá je upravovaná vždy len na konci šprintu.

## 3 Sumarizácia šprintov

### 3.1 Šprint 1

Na začiatku sme sa oboznamovali s projektom a jeho smerovaním. Identifikovali sme naše hlavné ciele projektu.

Postupne sme sa všetci členovia tímu zoznamovali s nástrojmi a metodikami pre riadenie a manažment tímu a to napríklad Jira, či Git.

Počas prvého šprintu sme vytvorili najdôležitejšie metodiky potrebné pre vývoj a fungovanie tímu. Tieto metodiky zatiaľ ešte neboli formalizované. Vytvorili sme webové sídlo projektu a tiež sme sa snažili “automatizovať” proces vytvárania zápisníc a exportov úloh z Jiry. Začali sme pracovať na implementačných úlohách, ktoré sme vybrali do šprintu.

Na konci šprintu sme spätne počas retrospektívy identifikovali niekoľko vecí, ktoré je potrebné zlepšiť do ďalšieho šprintu. A to najmä: prácu, pridelovanie a ohodnotenie úloh, lepšie automatizovať základnú rutinnú prácu (zápisnice), nedostatočná viditeľnosť projektu pre product ownera, zlepšiť dekompozíciu úloh, veľa úloh ostalo v stave in-progress.

### 3.2 Šprint 2

Úspešne sme dokončili všetky neimplementačné úlohy a podarilo sa nám automatizovať rutinnú prácu. Stále sa málo implementačných úloh posunulo do stavu Done a ostali v stave ready-to-review. Hlavnou príčinou bola zlá aplikácia metodiky pre prehliadky kódu. Znamená to, že vetvy boli dokončené a bol vytvorený pull-request, ale nikto nespravil prehliadku kódu a spojenie do develop vetvy vo vzdialenom repozitári.

Zisťujeme nedostatočnú komunikáciu v tíme a tiež s vlastníkom produktu. Navrhujeme riešenie tohto problému, ktorý by mohol mať zásadný dopad na celý projekt. Dôsledky nedostatočnej komunikácie boli: nikto neurobil prehliadku dokončenej funkcionality, otázky súvisiace s úlohy sme komunikovali príliš neskoro s vlastníkom produktu.

Do budúceho šprintu by sme mali zlepšiť komunikáciu a prácu na pridelených úlohách, aktívne komunikovať stav projektu vlastníkovi produktu.

### 3.3 Šprint 3

Počas priebehu tretieho šprintu sa nášmu tímu podarilo úspešne dokončiť veľkú časť implementačných úloh, za čo vďačíme zlepšeniu organizácii tímu z pohľadu komunikácie.

V predchádzajúcom šprinte bolo poukázané na to, že mnohé implementačné úlohy boli schopné sa dostať do revidovanej sekcie, ale neboli nikdy patrične revidované. Tento nedostatok

bol odstránený politikou, ktorú sme si určili pre tretí šprint. Vyžadovali sme aby boli jednotlivé kódy patrične zrevidované, zostavené a otestované na funkcionality ostatnými členmi tímu.

Tento jednoduchý cieľ sme dokázali dosiahnuť pomocou hromadného informovania jednotlivých členov o dostupnosti úlohy na testovanie. Následne sa očakávalo verejné prihlásenie dobrovoľníkov, ktorý by danú implementáciu vyskúšali podľa štandardných metodík. Týmto jednoduchým prístupom sme boli schopní otestovať veľké množstvo navrhovaných zmien v štruktúre existujúceho softvéru, pričom sme odhalili nemálo nedostatkov a následne ich úspešne odstránili.

Bolo poukázané na to, že jednotliví členovia boli schopní úspešne sledovať svoje vlastné komunikačné kanály a verejné kanály. Ale najmä bolo poukázané na to, že sa naučili a osvojili si prístup sledovania cudzích komunikačných kanálov ostatných členov a ich doterajšiu produkciu v ich pridelených vetvách.

### **3.4 Šprint 4**

Počas štvrtého šprintu sa nám nepodarilo úspešne dokončiť všetky naplánované úlohy. Bolo to hlavne z dôvodu zlého odhadnutia ich zložitosti a nedostatočnej dekompozície úloh. Úlohy boli definované na príliš vysokej granularite. Zistili sme, že aj úlohy s ohodnotením 5SP je potrebné dekomponovať na viac úloh alebo podúloh.

Identifikovali sme, že potrebujeme manažéra, ktorý bude aktívne dohliadať na spájanie zdrojových kódov do vzdialeného repozitára. Spájanie by malo byť vždy vykonané na konci šprintu, pričom sa spájajú len prehliadnuté a dokončené kódy. Toto ošetrujú príslušné metódy.

### **3.5 Šprint 5**

Posledný a zároveň najkratší šprint sme sa venovali najmä dokončovaniu dokumentácie a príprave na prezentáciu riadenia v tímovom projekte. Ďalej sme pripravovali projekte na prezentáciu pri príležitosti dňa otvorených dverí na FIIT. Dokončili sme implementačné úlohy, ktoré bolo možné stihnúť za tento skrátený šprint a zvyšné sme zachovali so snahou ich dokončiť do Vianoc, resp. cez Vianočné sviatky.

## 4 Používané metodiky

Súčasťou projektu sú metodiky, ktoré aplikujeme pri riadení a manažmente v príslušných oblastiach. Potreby a benefity metodík, ktoré aplikujeme v tomto projekte sú stručne opísané v tejto kapitole. Celé znenie každej metodiky je uvedené v prílohe A.

### 4.1 Metodika písania a komentovania zdrojového kódu

Metodika je určená pre všetkých členov tímu. Čiastočne pomáha aj pri procese prehliadky kódu.

Hovorí o základných konvenciách pri písaní a dokumentovaní zdrojového kódu. Cieľom je aby každý človek produkoval konzistentný a čitateľný kód pre ostatných členov, prípadne pre iných programátorov, ktorí sa s projektom predtým nestretli. Hovorí tiež o správnom spôsobe komentovania, ktoré umožní automatické generovanie dokumentácie príslušným nástrojom, v našom prípade Doxygen. Častým problémom je rozdielne nastavenie vývojového prostredia, je nutné aby mal každý člen nastavené rovnaké odsadzovanie bielymi znakmi. Nekonzistencia spôsobuje problémy pri spájaní vetiev do vzdialeného repozitára a neprehľadnosť zmien.

### 4.2 Metodika vytvárania úloh v nástroji na evidenciu úloh

Metodika je určená hlavne pre administrátora nástroja Jira. Admin práva majú pridelení všetci členovia vrátane produktového vlastníka. Pridávanie je sprístupnené všetkým, avšak hlavné zásahy – mazanie úloh, spustenie a ukončenie šprintu, môže vykonávať iba hlavný administrátor.

Dokument jasne definuje postupy pri vytváraní nových úloh. Pred jednotlivými šprintami je produktový backlog rozširovaný o nové úlohy. Úlohy sú následne kategorizované na *implementačné* a *ostatné*. Úlohy s podobnými vlastnosťami sú zorganizované do kategórií (epics) – documents, bugs, warnings, *zariadenia*. Každá úloha je ohodnotená story pointami. Nástroj Jira je nám nápomocná pri vytváraní retrospektívy a exporty pridelených úloh sa nemusia manuálne písať.

### 4.3 Metodika komunikácie

Metodika opisuje jednotlivé spôsoby komunikácie medzi jednotlivými členmi tímu a existujúce komunikačné siete a kanály, ktoré sa za daných situácií používajú, tiež opisuje prioritu sledovaných kanálov s odôvodneniami.

Tento dokument bol vypracovaný na základe štandardného členenia komunikácie na hovorovú a písanú, aktívnu a pasívnu. Detailne opisuje najmä písanú komunikáciu, pričom pou-

kazuje na dôležitosť sledovania pasívnych komunikačných kanálov, ktoré si musia ustrážiť jednotliví členovia tímu osamote (napr. pridelený branch na GitHub-e, pridelená úloha v Jire). Členovia tímu sú tiež povinní mať všeobecný prehľad, ktorý je poskytnutý skupinovú emailovou komunikáciou, pričom sa odporúča nepoužívať súkromné spôsoby komunikácie, aby mal informačný prínos celý tím. Aktívne spôsoby komunikácie sú používané v prípade nutnej spolupráce nad zjednotenou úlohou, kde je vyžiadaná prítomnosť každého člena tímu.

Metodika tiež hovorí o povinnostiach jednotlivých členov, ktoré musia dodržiavať pre jednoduchšie a priamočiarejšie vykonávanie všetkých procesov riadenia tímového projektu.

#### ***4.4 Metodika manažmentu verzií***

Tento projekt využíva Gitflow metodiku manažmentu verzií. Projekt má vytvorený Git repozitár, do ktorého sa ukladajú jednotlivé verzie aplikácie. Gitflow metodika sa zaoberá manažmentom Git vetiev a má celkom striktné pravidlá ich tvorby a zlučovania. Táto metodika popisuje aké vetvy využívame, ako vytvárať nové vetvy a akým spôsobom ich zlučovať naspäť do Master vetvy.

#### ***4.5 Metodika testovania***

Cieľom metodiky testovania je popísať členom tímu ucelené postupy, ako počas vývoja testy vytvárať a vykonávať. Metodika popisuje unit testy, ich vytváranie a vykonávanie v QtCreator-e, regresné testovanie a akceptačné testovanie.

#### ***4.6 Metodika prehliadky kódu***

Pre dosiahnutie väčšej kvality, po napísaní kódu prejde prehliadkou. Code review metodika popisuje jednotlivé kroky, ktorých treba dodržiavať sa pri prehliadke kódu, na čo treba pamätať, aké nástroje použiť.

#### ***4.7 Globálna retrospektíva***

V tejto podkapitole sumarizujeme všetky procesy, ktoré vznikli v našom tíme a projekte. Identifikujeme tri kategórie procesov, tie ktoré potrebujeme vytvoriť a aplikovať, procesy ktoré sú nefunkčné alebo zbytočné a procesy, ktoré potrebujeme zaviesť.

Vytvoriť nové procesy:

- Riadenie a tvorba dokumentácia na priebežnej báze.
- Commitovať zmeny na menšej granularite.
- Dodržiavať striktné metodiky, ktoré boli vytvorené pre podporu vývoja a riadenia v tíme.

- Neoficiálne stretnutia tímu v laboratóriu.

Eliminovať nefunkčné procesy:

- Všetky úlohy musia byť dokončené v šprinte, v ktorom boli pridelené, okrem blokujúcich a výnimočných stavov.
- Priebežne pracovať na úlohách.

Pokračovať v procese:

- Dekomponovať úlohy.
- Prehliadky kódu aspoň dvomi členmi pred spojením do vzdialeného repozitára.
- Komunikovať problémy a nejasnosti s vlastníkom produktu.
- „Automatizovaná“ rutinná práca (zápisnice, retrospektíva)

## **5 Zoznam kompetencií tímu**

### **Matúš Cimerman**

- Vedúci tímu
- Kontrola správneho písania a komentovania zdrojového kódu

### **Denis Illés**

- Plánovanie úloh v Jire
- Identifikácia a riešenie rizík

### **Dalibor Mészáros**

- Komunikácia v tíme
- Hlavný architekt projektu

### **Irina Dyomina**

- Prehliadky kódu

### **Michal Fašánek**

- Verziovanie projektu

### **Jaroslav Gazdík**

- Dokumentácia

### **Filip Jurčacko**

- Testovanie

# Príloha A Metodiky

## *A.a Metodika prehliadky kódu*

Code review metodika popisuje roly aj ich úlohy, ktoré treba spraviť pri prehliadke kódu. Vysvetľuje jednotlivé kroky, ktorých treba dodržiavať sa pri prehliadke kódu, na čo treba pamätať, aké nástroje použiť. Metodikou sa riadia všetci programátori, ktorí dopĺňujú alebo menia kód, a ostatní členovia tímu, ktorí budú vykonávať samotnú prehliadku kódu. Všetci členovia tímu sa majú riadiť touto metodikou pred odovzdaním upraveného kódu pre dosiahnutie väčšej kvality produktu.

Prehliadka kódu je dôležitá činnosť, pretože od zmien v kóde závisí funkčnosť programu a jeho správne fungovanie. Prehliadka slúži na overenie a kontrolu správnosti napísania kódu, jeho kvality a dopadu jeho zmien na funkčnosť celého systému.

### *Roly a zodpovednosti*

- Programátor – napísanie nového kódu a úpravy už napísaného kódu. Po vykonaní zmien v kóde jeho zverejnenie, a oznámenie členov tímu, že kód je pripravený na Code Review. Po skontrolovaní iným členom tímu zapracovanie pripomienok a doplnenie kódu a následné zverejnenie pre opätovnú prehliadku.
- Reviewer (kontrolór) – skontrolovanie správnosti kódu a jeho funkčnosti podľa dohodnutých noriem a pravidiel, zhodnotenie kvality kódu, oznamovanie programátora o dopadu kontroly kódu, prípadne jeho vrátenie.

### *Proces prehliadky kódu*

- Programátor napíše nový kód alebo urobí zmeny v starom kóde a po overení, že všetko funguje tak, ako programátor chcel, zverejní ho pomocou nástroju Github.
- Reviewer kódu ho prezrie, v prípade akýchkoľvek otázok spýta sa autora kódu, ako to myslel. Zjavne malé chyby hneď opraví.
- Kontrolór má skontrolovať či je kód v dostatočnej miere okomentovaný, či sú tie komentáre porozumiteľné a či nie sú zbytočné.
- Kontrolór skontroluje formátovanie, či sú dodržané pravidla napísania kódu v jazyku C++.
- Následne kontrolór preverí kód špeciálnymi nástrojmi – cppcheck a cpplint.
- Po opravení komentárov a formátovania, ak to bolo nutné, kontrolór skompiluje kód a prevedie kontrolu warningov pri kompilácii.



- Reviewer spusti program a overí jeho funkčnosť po vykonaní zmien.
- Po overení kódu kontrolór oznámi programátora, či všetko bolo v poriadku alebo prácu treba prerobiť, povie ktoré časti kódu treba prepísať. Ak programátor prerábať kód nemusí, kontrolór vystaví pull-request v nástroji Github so všetkými svojimi drobnými zmenami.
- V prípade, že kód je napísaný správne, úloha programátora je skončená. V opačnom prípade, programátor opravuje svoj kód a po zapracovaní všetkých pripomienok zase zverejní kód pre jeho prehliadku.

## **A.b Metodika komunikácie**

Metodika je určená pre všetkých členov tímu. Slúži na opis jednotlivých komunikačných kanálov, opis ich použitia a jednoznačné vymedzenie situácií a priorít, v ktorých sa majú použiť. Táto metodika zakazuje používanie niektorých komunikačných kanálov, najmä súkromných kanálov, ktoré neposkytujú možnosť archivovať informácie. Ďalej táto metodika opisuje štruktúru formálnych tímových stretnutí a komentovanie k vykonaným zmenám.

### **Pojmy**

JIRA – nástroj na evidovanie úloh

GitHub – repozitár pre kontrolu revízií

### **Používané komunikačné kanály**

Z hľadiska tímovej komunikácie rozoznávame nasledujúce komunikačné kanály a ich hlavné využitia:

- **Aktívne**
  - Skupinová instantná textová komunikácia (skupinový chat)
    - Zahrňuje komunikáciu len s členmi tímu
    - Používa sa v prípade aktívnej skupinovej práce – skupinové písanie dokumentov, dokumentácií a riešenie minoritných organizačných problémov
    - Zakazuje sa používanie súkromných kanálov, aby mala komunikácia prínos aj pre ostatných členov tímu
  - Hovory so zdieľanou obrazovkou
    - Požívajú sa pri praktizovaní metód extrémneho programovania
  - Komunikácia tvárou v tvár
    - Základný všeobecný komunikačný kanál pre jednotlivých členov tímu
    - Tiež je možné túto formu komunikácie obohatiť o vedenie zápisov, prácu pri tabuli, aby mala komunikácia fyzický výstup
    - Rozšírenie tejto formy komunikácie je skupinová diskusia
  - Stretnutia
    - Hlavný formálny a neformálny tímový komunikačný kanál
- **Pasívne**
  - Emailový zoznam a skupinové správy (mailing list)
    - Zahrňuje komunikáciu aj s členmi tímu, ale aj s vedúcim

- Používané v prípade kladenia dôležitých otázok, riešenia problémov a organizácie tímu
- JIRA komentáre
  - Slúžia na dopĺňovanie potrebných informácií k jednotlivým úlohám, neslúžia na diskusiu k implementáciám, na tú slúži emailový zoznam
- GitHub komentáre
  - Slúžia na hlásenie nájdených chýb a hlásenie úspešných testovaní danej vetvy

## Priebeh stretnutí

Z hľadiska formálnosti a neformálnosti majú tímové stretnutia daný nasledujúci priebeh:

- **Oficiálne formálne stretnutie na konci šprintu**
  - Časovo vyhradené na pravidelné tímové stretnutia, ktoré sa uskutočňujú každý týždeň
  - Stretnutie otvára vedúci a člen tímu, ktorý na minulom stretnutí viedol zápisnicu
  - Skontroluje sa stav jednotlivých úloh, členovia s priradenou úlohou sa vyjadria k stavu
  - Diskutuje sa k priebehu šprintu, navrhujú sa organizačné zmeny pre zlepšenie funkcionality tímu
  - Uzatvorenie šprintu
  - Identifikujú sa a pridávajú nové úlohy
  - Ohodnocujú sa úlohy a ich pridelujú jednotlivým dobrovoľným členom pre nasledujúci šprint
  - Otvorenie šprintu
  - Otvorená diskusia a riešenie individuálnych problémov
- **Oficiálne formálne stretnutie počas aktívneho šprintu**
  - Časovo vyhradené na pravidelné tímové stretnutia, ktoré sa uskutočňujú každý týždeň
  - Stretnutie otvára vedúci a člen tímu, ktorý na minulom stretnutí viedol zápisnicu
  - Hlási sa postup na jednotlivých úlohách členov tímu
  - Pýtajú sa otázky k úlohám a diskutujú sa možné riešenia
  - Otvorená diskusia a riešenie individuálnych problémov

- **Neformálne stretnutia**
  - Uskutočňujú sa po dohodnutí tímu
  - Určené na písanie retrospektívy a doplňovanie zápisníc
  - Otvorená diskusia

### **Riešenie problémov, kladenie a zodpovedanie otázok**

Počas práce na tímovom projekte vznikajú často otázky, ktoré sa nemôžu odkladať na oficiálne stretnutia, ale musia sa zodpovedať okamžite. Tieto otázky sa identifikujú nasledujúco:

- **Všeobecné otázky s nízkou informačnou hodnotou**
  - Nevyžadujú si pozornosť vedúceho
  - Otázky sú triviálne a nemajú potenciál vzbudiť dlhú diskusiu
  - Používa sa **skupinová instantná textová komunikácia**
  - **Nepoužívajú** sa súkromné správy
- **Otázky týkajúce sa organizácie a implementácie**
  - Vyžadujú si pozornosť celého tímu, alebo vedúceho
  - Majú vysokú informačnú hodnotu a diskusie k nim si vyžadujú archiváciu
  - Používa sa **emailový zoznam a skupinové správy**

### **Komentovanie vykonaných zmien**

Pri vykonávaní implementačných zmien si vyžadujú jednotliví členovia a úlohy odozvu od ostatných členov. Pri hlásení považujeme formu odozvy ako potvrdenie funkčnosti, alebo hlásenie chýb.

- **Hlásenie fungujúceho riešenia**
  - Jednoduchá forma a štruktúra
  - Oznamiť funkčnosť vetvy
  - Opísať spôsob testovania
  - Oznamiť použitý systém
- **Hlásenie chýb**
  - Štruktúrovaný text a stručná forma
  - Oznamiť objavenie chyby
  - Opísať vlastnými slovami problém
  - Pri hlásení chýb doložiť evidenciu vo forme textových súborov obsahujúcich výpis z konzoly, kompilovania a chybové hlášky
  - Oznamiť použitý systém

## **A.c Metodika manažmentu verzíí**

Metodika sa zaoberá manažmentom rôznych verzíí softvéru, konkrétne v interakcii s Git repositárom projektu. Metodika pokrýva, akým spôsobom sa tvoria a zlučujú vetvy, aké vetvy v projekte tvoríme a aké príkazy používame.

Používame vetvy: Master, Develop, Feature, Hotfix

**Master** - hlavný projekt

**Develop** - branchnutá z Master vetvy, projekt má vlastnú Develop vetvu, na konci šprintu sa zlúči späť do Master vetvy.

**Feature** - branchnutá z Develop vetvy, každý nový kus funkcionality (task v Jire), ktorý sa programuje, musí mať vlastnú Feature vetvu, po dokončení a validácii kódu sa zlúči späť do Developu, neinteraguje s Master vetvou.

**Hotfix** - vetva na rýchly fix (u nás z developu), zlučuje sa do developu, navyšuje aktuálnu verziu.

Vždy zlučujeme (merge) cez Shell a s prepínačom „--no-ff“.

### **Commit messages:**

Pri Commitovani zmien commitujeme s prepínačom -m, za ktorým uvedieme commit message. Na začiatok commit message uvedieme tagy:

[FIX] - opravili sme nejakú chybu z minula, bugfix, hotfix a podobne

[ADD] - pridali sme novú funkcionality, súbor, ...

[DOC] - pridali sme dokumentáciu, komentáre...

[REF] - pre refaktoring

[FMT] - formátovanie textu, úprava

[TEST] - pre testy

Za tým veľmi stručne (a výstižne) opíšeme, aké zmeny sme spravili. Správy by mali byť krátke, no pokrývať všetko, čo sme v commite spravili.

### **Pull requesty:**

Po dokončení práce, keď sme *ready to review* sa dáva *pull request* na vetvu, do ktorej sa bude mergovať. Pull request sa robí z GUI GitHubu (pravý horný roh), alebo „\$ git request-pull {meno\_commitu} {URL}“. Pozn. doporučujem robiť cez GUI

Po odsúhlasení Pull requestu sa potom pristúpi k zlúčeniu.

### **Niektoré užitočné commandy:**

```
$ git submodule update --init --recursive
```

```
//update submodulov (dependencies)
```

```
$ git checkout -f meno_branch
```

```
//checkout branch aj napriek lokálnym zmenám, budú zahodene
```

```
$ git status
```

```
//vypíše všetky vykonané zmeny
```

```
$ git stash / $ git stash pop
```

```
//uloží stav projektu do stash, z ktorého sa da potom tento stav pop-núť, dobré na prenos zmien medzi vetvami
```

### **Tvorba feature branchu:**

```
$ git checkout -b "feature/meno-feature" develop
```

```
//Switched to a new branch "feature/meno-feature"
```

### **Mergovanie hotového feature:**

```
$ git checkout develop
```

```
//Switched to branch 'develop'
```

```
$ git merge --no-ff meno-feature
```

```
$ git push origin develop
```

### **Tvorba hotfix branch-u:**

```
$ git checkout -b "hotfix/nazov-co-fixujem" master
```

```
//Switched to a new branch "hotfix-{cislo_verzie}"
```

```
$ git commit -m "Sprava, co som spravil"
```

### **Uzatvorenie Hotfix branchu:**

```
$ git checkout develop
```

```
//Switched to branch 'develop'
```

```
$ git merge --no-ff "hotfix/nazov-co-fixujem"
```

## ***A.d Metodika písania a komentovania zdrojového kódu***

Metodika je určená pre všetkých členov tímu. Čiastočne pomáha aj pri procese prehliadky kódu. Tento dokument hovorí o základných konvenciách pri písaní a dokumentovaní zdrojového kódu. Cieľom je aby každý člen produkoval konzistentný a čitateľný kód pre ostatných členov, prípadne pre iných programátorov, ktorý sa s projektom predtým nestretli. Hovorí tiež o správnom spôsobe komentovania, ktoré umožní automatické generovanie dokumentácie príslušným nástrojom, v našom prípade Doxygen1. Častým problémom je rozdielne nastavenie vývojového prostredia, je nutné aby mal každý člen nastavené rovnaké odsadzovanie bielymi znakmi. Nekonzistencia spôsobuje problémy pri spájaní vetiev do vzdialeného repozitára a neprehľadnosť zmien.

### **Slovník pojmov**

- Code review – prehliadka zdrojového kódu,
- Doxygen – nástroj pre automatické generovanie dokumentácie zo zdrojového kódu podľa komentárov v kóde,
- QT Creator – vývojové prostredie používané v projekte.

### **Roly a zodpovednosti**

<b>Rola</b>	<b>Zodpovednosť</b>
Programátor	<ul style="list-style-type: none"><li>• Písanie komentárov do zdrojového kódu,</li><li>• Písanie zdrojového kódu s použitím nastavení, ktoré vymedzuje táto metodika</li></ul>
Člen tímu, ktorý vykonáva code review	<ul style="list-style-type: none"><li>• Kontrola, či je zdrojový kód písaný a komentovaný podľa príslušných metodík</li></ul>
Manažér dokumentáci	<ul style="list-style-type: none"><li>• Generovanie dokumentácie zo zdrojového kódu projektu s nástrojom Doxygen</li></ul>

### **Komentovanie zdrojového kódu**

Komentovanie zdrojového kódu používame dva typy komentárov:

- jednoriadkový, na jeden riadok do 80 znakov, stručne opisuje časť kódu,
- blokový komentár, opisuje väčšiu časť kódu, zväčša metódu alebo triedu.

Komentáre k triedam a metódam píšeme v hlavičkových súboroch (.h), teda pri definícii triedy a jej metód. Komentáre sú písané v anglickom jazyku.

### Komentár metódy

Komentáre k metódam uvádzame pred definíciou metódy. Začiatok komentára označíme znakmi `/**` a pokračujeme na ďalšom riadku. Každý riadok komentára musí byť označený kľúčovým slovom. Ak text pre kľúčové slovo presiahne jeden riadok, potom nový riadok začína iba znakom `*`. Komentár je uzavretý znakmi `**/`.

Kľúčové slovo	Popis	Počet v jednom blokovom komentári
@author	Autor kódu	1 – n
@param <meno parametra>	Vstupný parameter metódy	0 – n
@return	Opis návratovej hodnoty	0 – 1
@brief	Stručný popis metódy	1
@note	Detailný popis metódy	0 – n
@bug	Stručný popis známej chyby alebo problému v metóde	0 – n
@todo	Opis funkcionality, ktorá má byť v budúcnosti implementovaná	0 - n
@date	Dátum kedy vznikla metóda	0 - 1

Autor a opis metódy sú povinnými parametrami vždy, ostatné parametre len v prípade keď to má zmysel. Programátor je povinný komentovať každú časť kódu, ktorú považuje za potrebnú doplniť komentárom. Ukážka blokového komentára metódy:

```
/**
 * @author Jozef Novak
 * @brief brief description
 * @param a parameter a description
 * @param b parameter b very very long
 * description
 * @return return value description
 * @note detailed method description
 * @bug this method should not use global variable
```



```
* @todo description of functionality to implement
*/
```

## Komentár triedy

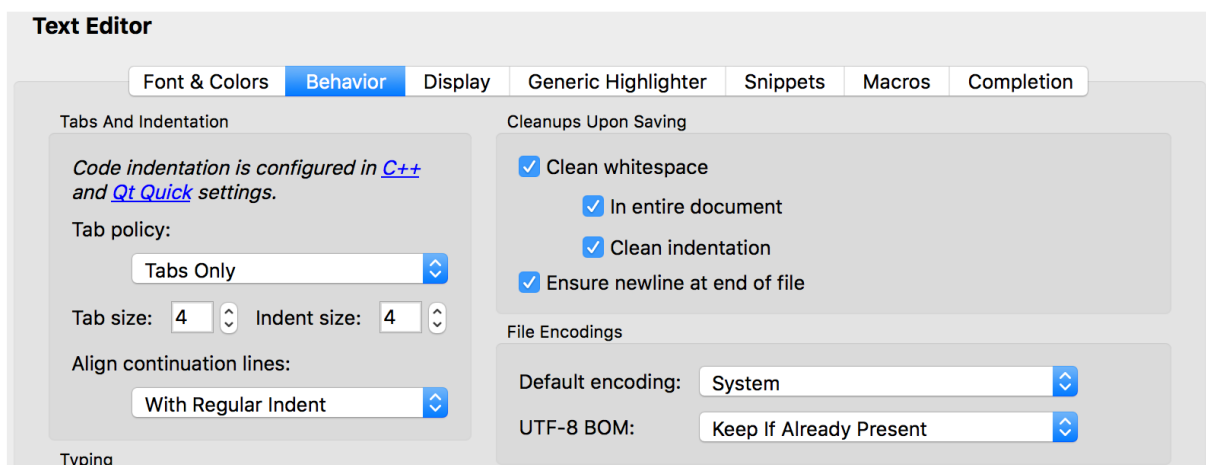
Komentáre k triedam píšeme pred definíciu triedy v hlavičkovom súbore. Postupujeme rovnako ako pri písaní komentárov pre metódy. Časť *brief* píšeme v tvare „Class <ClassName> <stručný popis triedy>“. Ukážka takto vytvoreného komentáru triedy:

```
/**
 * @brief ClassArucoThread main class for aruco thread
 * @author Autor: Jozef Novak
 * @date 12.12.2015
 * @note This class provides a thread for tracking Aruco.
 */
```

## Nastavenie editora

Biele znaky, ktoré vznikajú pri odsadzovaní kódu spôsobujú problémy pri verziovaní projektu nástrojom Git (generuje to umelé zmeny). Preto je potebné konzistentné nastavenie vývojového prostredia všetkými členmi. V našom projekte používame vývojové prostredie QT Creator.

Výber v hornej lište Tools -> Options, následne v ľavom menu položku Text Editor a záložka Behavior. Nastaviť podľa obrázku nižšie.



Obrázok 1 - nastavenie editora

## Konvencie v názvoch a prístupe k elementom

- Názov každej (okrem makra) začína malým písmenom a každé ďalšie slovo v názve začína veľkým (napr. *coreWindow*).
- Názov metódy musí byť sloveso, napr. *computeLine()*.

- Metódy, ktoré vracajú booleovu hodnotu pravda/nepravda musia začínať slovami „is“ alebo „has“, napr. *isVisible()*.
- Názov, ktorý predstavuje typ triedy musí začínať veľkým písmenom a každé ďalšie slovo začína tiež veľkým písmenom (napr. *ColorsListType*).

Elementy zoraduje v hlavičkových súboroch podľa obmedzenia prístupu nasledovne:

1. public
2. private
3. protected

### Zaradovanie hlavičkových súborov

Definujeme hierarchiu v akej sa musia pripájať projekty do zdrojových kódov:

1. Hlavičkové súbory projektu
2. Hlavičkové súbory z QT a OSG
3. Systémové hlavičkové súbory

Ďalšie pravidlá pri zaradovaní:

- Hlavičkový súbor by mal obsahovať všetky premenné, s ktorými pracuje cpp súbor.
- Zaradujeme len to čo je potrebné, snažime sa minimalizovať nepotrebné zaradovanie hlavičkových súborov.
- Ak je v triede použitý pointer na typ, stačí použiť doprednú deklaráciu a netreba zaradovať celý hlavičkový súbor typu. Toto neplatí pri dedení.

### Písanie slotov a signálov

Sloty a signály slúžia na komunikáciu medzi objektami. Signály sú vysielané objektom keď sa zmenil jeho vnútorný stav, ktorý by mohol byť zaujímavý pre majiteľa objektu. Signály sú verejné (angl. public) funkcie. Odporúčané je signály vysielat' len z triedy, ktorá signál definuje alebo z jej podtried. Slot je funkcia, ktorá je volaná keď je k nemu príslušný signál vysielaný.

Signál so slotom prepojíme pomocou funkcie *connect*, príklad:

```
connect(chb_vertigo, SIGNAL(clicked()), this, SLOT(toggleVertigo())); □
```

Poradie je nasledovné:

1. Odosielateľ signálu
2. Signál
3. Prijímateľ signálu
4. Metóda, ktorá sa má vykonať

## Odporúčania a praktiky

### *Zlé praktiky:*

- nikdy nepoužívať using namespace a tiež keyword using.
- nepoužívať 0 (nulu) ako NULL pointer, vždy použiť nullptr (ak kompilátor podporuje C++11), inak NULL.
- otvorenie pull-requestu bez predchádzajúcej aplikácie metodiky pre prehliadku kódu, warningy počas kompilácie.
- pri testovaní pointrov, použite (!myPtr) alebo (myPtr); nepoužívajte myPtr != nullptr.
- neporovnávajúte premenné spôsobom `x == true`, alebo `x == false`; použite `(x)` alebo `(!x)`.

### *Dobré praktiky:*

- inicializovať všetky atribúty v konštruktore cez initialization list,
  - ak je v konštruktore keyword new, potom v deštruktore musí byť delete,
  - inicializovať v takom poradí v akom sú zapísané v triede, resp. usporiadať od najväčších (pointer, trieda, double, int, char, ...)
- hlavičkové súbory sú rozdelené do troch blokov a mali by byť zoradené podľa abecedy:
  - hlavný hlavičkový súbor, Foo.h v Foo.cpp,
  - súbory štandardnej knižnice: #include <vector>
  - ostatné súbory externých projektov: #include <osg/PolygonMode>
- prefixy premenných by mali mať výpovednú hodnotu pre čitateľa kódu:
  - k = konštanta (napr. kLimit)
  - g = globálna premenná (napr. gConfig)
  - a = argument (napr. aCnt)
  - C++ špecifické prefixy:
    - ! s = static member (napr. sPrefChecked)
    - ! m = member (napr. mLength)

## ***A.e Metodika vytvárania úloh v nástroji na evidenciu úloh***

Metodika je určená predovšetkým hlavnému administrátorovi nástroja Atlassian JIRA, ktorý je zodpovedný za správne nastavenie, zaradenie a ohodnotenie úloh. Cieľom dokumentu je poskytnúť jednotný spôsob zakladania úloh, ktorým zabezpečíme konzistenciu a prehľad priebehu riešení problémov týkajúcich sa projektu.

### **Pojmy:**

JIRA – nástroj na sledovanie úloh

ISSUE – úloha

SPRINT – definovaný časový úsek (u nás 2 týždne)

PRODUCT BACKLOG – zoznam úloh, ktoré sa neriešia v aktuálnom šprinte

SPRINT BACKLOG – zoznam úloh, ktoré sme zaradili do šprintu a majú byť splnené do konca šprintu

### **Vytvorenie úlohy:**

- Create issue
- Zadať jednoznačný názov
- Zaradiť do správnej kategórie – LABEL: *implementaion* alebo *other*
- Pripojiť príslušný epicLink – warning, bug, document, AR glasses,...
- Opísať úlohu v časti Description
- Kliknúť create

Následne vytvorenú úlohu môžeme ohodnotiť story pointami. Na čísle sa zhodneme použitím scrum pokeru, kde sa zoberie priemerná hodnota s tým, že člen s najvyšším a najnižším hodnotením okomentuje svoju voľbu.

Pri vytvorení šprintu sú úlohy presunuté z product backlog do sprint backlog. Množstvo súvisí od predošlých šprintov. Ak sme nestihli vyriešiť všetky úlohy z predošlého šprintu, tak nasledujúci šprint bude obsahovať menší počet úloh (= suma story pointov úloh bude menšia). V opačnom prípade je počet úloh pre šprint zvýšený.

Stavy úloh sa môžu nachádzať pre *other*: TODO, INPROGRESS, DONE a pre *implementation*: TODO, INPROGRESS, READYTOREVIEW, DONE.

## **A.f Metodika testovania**

Cieľom metodiky testovania je popísať členom tímu ucelené postupy, ako počas vývoja testy vytvárať a vykonávať. Metodika popisuje Unit testy, ich vytváranie a vykonávanie v QtCreator-e, regresné testovanie a akceptačné testovanie.

### **Pojmy**

QtCreator - vývojové prostredie

JIRA - nástroj na sledovanie úloh

### **Unit testovanie**

Unit testovanie prebieha počas celého šprintu, unit testy vytvárajú a vykonávajú autori na vlastnom kóde. Na vytváranie unit testov používame framework QTestLib.

### **Vytvorenie unit testov v QtCreator-e**

Vstup: funkcie projektu

Výstup: testovacie funkcie

Zodpovedný: programátor

Najprv vytvoríme testovaciu triedu, ktorá dedí od QObject. Jednotlivé testovacie funkcie musia byť deklarované pod private slots.

```
#include <QTest/QTest>
class TestQString: public QObject
{
    Q_OBJECT
private slots:
    void toUpper();
};
```

Následne implementujeme jednotlivé testovacie funkcie.

```
void TestQString::toUpper()
{
    QString str = "Hello";
    QVERIFY(str.toUpper() == "HELLO");
}
```

### **Vykonanie unit testu:**

Vstup: testovacie funkcie

Výstup: výsledky testov

Zodpovedný: programátor

### Príkaz na spustenie testu má formát:

```
testname [options] [testfunctions[:testdata]]...
```

príklad:

```
/myTestDirectory$ testQString toUpper
```

### Vyhodnotenie unit testu:

Vstup: výsledky testov

Výstup: oprava prípadných chýb

Zodpovedný: programátor

### Výsledky testu sú v nasledovnom formáte:

```
***** Start testing of TestQString *****  
Config: Using QTest library 4.8.5, Qt 4.8.5  
PASS : TestQString::toUpper()  
Totals: 1 passed, 0 failed, 0 skipped  
***** Finished testing of TestQString *****
```

Výstup obsahuje zoznam všetkých vykonaných testovacích funkcií, kde každý test skončí buď úspechom alebo zlyhaním. Programátor identifikuje chyby na základe testov ktoré zlyhali, následne ich opraví.

Framework QTestLib umožňuje simulovať udalosti na GUI ako napríklad kliky na menu a tlačidlá, preto je možné pre našu aplikáciu vytvoriť aj komplexné unit testy.

### Regresné testovanie:

Pred ukončením šprintu sa vykonajú všetky doposiaľ vytvorené unit testy. Ich výstup ukáže zmeny správania existujúceho kódu, ktoré sú pravdepodobne zapríčinené novšími zmenami.

## **Akceptačné testovanie:**

Ako posledné pred ukončením šprintu prebieha akceptačné testovanie. Akceptačné testovanie je robené manuálne, overuje funkcionality používateľských scenárov, pridanej funkcionality.

### **Príprava testu:**

Vstup: požiadavka na funkcionality, používateľský scenár

Výstup: testovací scenár

Zodpovedný: Product Owner

Akceptačný test sa najprv vytvorí, v podobe scenára ktorý naplňa požiadavku na funkcionality. Obsahuje nasledovné:

**Názov** – obsahuje názov testovacieho scenára

**Účel** – obsahuje čo je daným testovaním overované

**Vstupné podmienky** – podmienky, ktoré musia byť splnené pred vykonávaním jednotlivých akcií

**Výstupné podmienky** – podmienky, ktoré musia byť splnené po vykonaní jednotlivých akcií

**Akcia** – akcia v systéme, ktorú používateľ vykoná

**Očakávaná reakcia** – reakcia, ktorá je očakávaná po vykonaní danej akcie

**Skutočná reakcia** – reakcia, ktorá po vykonaní danej akcie skutočne nastala

Príklad testu:

<b>ID</b>	1	<b>Názov</b>	Navigácia v 3D priestore prostredníctvom 3d myši	
<b>Účel</b>	Overenie funkčnosti 3D myši			
<b>Vstupné podmienky</b>	3D myš je pripojená k počítaču			
<b>Výstupné podmienky</b>				
<b>Krok</b>	<b>Akcia</b>	<b>Očakávaná reakcia</b>	<b>Skutočná reakcia</b>	
1	Pohyb do strán	Posunutie kamery doľava a doprava		
2	Pohyb hore a dole	Posunutie kamery hore a dole		
3	Pohyb otáčanie a naklápanie	Rotácia		

#### Vykonalie testu:

Vstup: testovaný softvér, testovacie scenáre

Výstup: doplnené testovacie scenáre o skutočné výsledky

Zodpovedný: Product Owner

Testy vykonáva Product Owner podľa predpísaných krokov v testovacích scenároch.

#### Vyhodnotenie testu:

Vstup: doplnené testovacie scenáre o skutočné výsledky

Výstup: oprava prípadných chýb

Zodpovedný: programátori

V prípade že odhalené chyby nevieme opraviť ihneď, sú k nim vytvorené úlohy v JIRE, ktoré sa následné riešia v ďalšom šprinte.



# Príloha B Zázpisnice

## *B.a Prvý týždeň*

**Autor:** Matúš Cimerman

**Dátum stretnutia:** 30.9.2015

**Prítomní:**

Vedúci: Ing. Peter Kapec, PhD.  
Členovia tímu: Bc. Matúš Cimerman  
Bc. Irina Dyomina  
Bc. Michal Fašánek  
Bc. Jaroslav Gazdík  
Bc. Denis Illés  
Bc. Filip Jurčacko  
Bc. Dalibor Mészáros

**Priebeh stretnutia:**

- Diskusia o projekte,
  - Historia,
  - Na čo sluzi a čo momentálne dokazuje,
  - Predstavenie vizie.
  
- Diskusia o:
  - SCRUMe,
  - O riadení a nástrojoch riadenia,
  - O tom ako budeme komunikovať mimo stretnutí.

**Úlohy na budúce stretnutie:**

ID	Riešiteľ	Opis úlohy
-	Volba "sefa" tímu	Tím si zvolí zodpovedného člena za tím, ktorý bude riešiť prípadne problémy.
-	Vyber hlavného architekta	Hlavný architekt projektu, mal by mať najlepší prehľad o všetkých častiach systému.
-	Vyber nástroja na task mgmt	Trello, Jira, Redmine alebo niečo iné?
-	Navrhnut web tímu	Webová stránka projektu

## ***B.b Druhý týždeň***

**Autor:** Matúš Cimerman

**Dátum stretnutia:** 30.9.2015

### **Prítomní:**

Vedúci: Ing. Peter Kapec, PhD.

Členovia tímu: Bc. Matúš Cimerman

Bc. Irina Dyomina

Bc. Michal Fašánek

Bc. Jaroslav Gazdík

Bc. Denis Illés

Bc. Filip Jurčacko

Bc. Dalibor Mészáros

### **Stav úloh z minulého stretnutia:**

<b>ID</b>	<b>Riešiteľ</b>	<b>Opis úlohy</b>	<b>Stav</b>
-	Tím	Voľba "šéfa tímu"	done
-	Tím	Voľba hlavného architekta	done
-	Tím	Výber nástroja na task mgmt	done, Jira
-	Fašánek	Web projektu	in-progress

### **Priebeh stretnutia:**

- oboznamovanie sa s projektom a jeho časťami,
- diskusia o BDD/TDD frameworku pre cpp,
- diskusia o potrebe logovania udalostí v projekte a výbere knižnice pre logovanie,

### **Úlohy na budúce stretnutie:**

<b>ID</b>	<b>Riešiteľ</b>	<b>Opis úlohy</b>
-	Tím	Výber frameworku na BDD/TDD development.
-	Tí	Výber knižnice pre logovanie v cpp programoch.

## ***B.c Tretí týždeň***

**Autor:** Matúš Cimerman

**Dátum stretnutia:** 7.10.2015

### **Prítomní:**

Vedúci: Ing. Peter Kapec, PhD.

Členovia tímu: Bc. Matúš Cimerman

Bc. Irina Dyomina

Bc. Michal Fašánek

Bc. Jaroslav Gazdík

Bc. Denis Illés

Bc. Filip Jurčacko

Bc. Dalibor Mészáros

### **Stav úloh z minulého stretnutia:**

<b>ID</b>	<b>Riešiteľ</b>	<b>Opis úlohy</b>	<b>Stav</b>
-	Tím	Výber frameworku na BDD/TDD development	done, vybraný framework Igloo
-	Tím	Výber frameworku pre logovanie v cpp programoch	done, vybraná knižnica Pantheios
-	Fašánek	Web projektu	in-progress

### **Priebeh stretnutia:**

- oboznamovanie sa so systémom,
- príprava backlogu,
- riešenie inštalačných problémov, kompilácia projektu.

### **Úlohy na budúce stretnutie:**

<b>ID</b>	<b>Riešiteľ</b>	<b>Opis úlohy</b>
-	Cimerman	Gitflow - nastudovať
-	Illés	Jira - vytvorenie projektu, nastavenie
-	Illés, Mészáros	Aktualizácia inštalačného manuálu

## *B.d Štvrtý týždeň*

**Autor:** Matúš Cimerman

**Dátum stretnutia:** 14.10.2015

### **Prítomní:**

Vedúci: Ing. Peter Kapec, PhD.

Členovia tímu: Bc. Matúš Cimerman

Bc. Irina Dyomina

Bc. Michal Fašánek

Bc. Jaroslav Gazdík

Bc. Denis Illés

Bc. Filip Jurčacko

Bc. Dalibor Mészáros

### **Stav úloh z minulého stretnutia:**

ID	Riešiteľ	Opis úlohy	Stav
-	Cimerman	Gitflow - naštudovať	done
-	Illés	Jira - vytvorenie projektu, nastavenie	done
-	Illés, Mészáros	Aktualizácia inštalačného manuálu	done

### **Priebeh stretnutia:**

- oboznamovanie sa so systémom,
- plnenie backlogu,
- riešenie inštalačných problémov, kompilácia projektu.

### **Úlohy na budúce stretnutie:**

ID	Riešiteľ	Opis úlohy
-	Cimerman	gitflow - vytvoriť metodiku
-	Gazdik	dokumentácia - vytvoriť metodiku
-		Zjednotiť metodiky: ako programovať v cpp, gitflow, formátovanie kódu, ...
23	Všetci	[Task] Nastaviť konzistentne editor

## ***B.e Piaty týždeň***

**Autor:** Filip Jurčacko

**Dátum stretnutia:** 21.10.2015

### **Prítomní:**

Vedúci: Ing. Peter Kapec, PhD.

Členovia tímu: Bc. Matúš Cimerman

Bc. Irina Dyomina

Bc. Michal Fašánek

Bc. Jaroslav Gazdík

Bc. Denis Illés

Bc. Filip Jurčacko

Bc. Dalibor Mészáros

### **Stav úloh z minulého stretnutia:**

<b>ID</b>	<b>Riešiteľ</b>	<b>Opis úlohy</b>	<b>Stav</b>
78	Cimerman	Vytvoriť GitFlow metodiku	in--progress
1	Fašánek	Web projektu	done
7	Gazdík	Zápisnice + šablóna	done
34	Mészáros + Illés	Aktualizácia OSG	done
10	Illés	Spustenie bez cotire, doplnenie includov	done
9	Mészáros	Opravovanie warningov	done
12	Jurčacko	"&Multi-select mode" bug	in progress
23	Všetci	Nastaviť konzistentne editor	done

### **Priebeh stretnutia:**

- oboznamovanie sa so systémom,
- plnenie backlogu + poker pre druhý šprint
- riešenie inštalčných problémov, kompilácia projektu.

**Úlohy na budúce stretnutie:**

<b>ID</b>	<b>Riešiteľ</b>	<b>Opis úlohy</b>
50	Jurčacko	Vytvoriť Zápisnicu w5
78	Cimerman	Vytvoriť GitFlow metodiku
56	Mészáros	Opraviť warningy - cppCheck
58		Opraviť warningy - MSVC
57	Cimerman	Opraviť warningy - CppLint
12	Jurčacko	"&Multi-select mode" bug
80	Illés	Aktualizovať inštalačný návod
41	Cimerman	Vytvoriť dok. s retrospektívou
49	Cimerman	Generovanie dokumentácie a diagramov
2	Fašánek	Nahrať dokumenty na webstránku

## ***B.f Šiesty týždeň***

**Autor:** Dalibor Mészáros

**Dátum stretnutia:** 28.10.2015

**Prítomní:**

Vedúci: Ing. Peter Kapec, PhD.

Členovia tímu: Bc. Matúš Cimerman

Bc. Irina Dyomina

Bc. Michal Fašánek

Bc. Jaroslav Gazdík

Bc. Denis Illés

Bc. Filip Jurčacko

Bc. Dalibor Mészáros

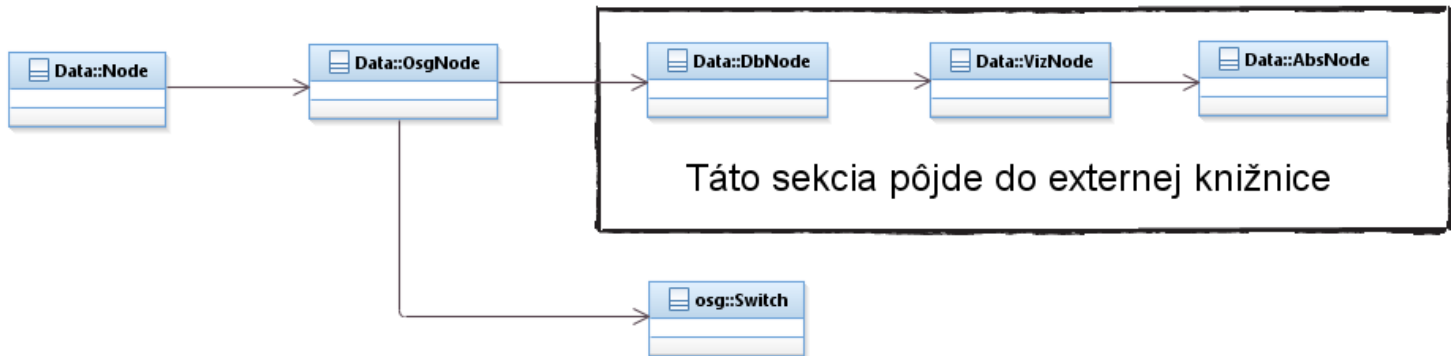
**Stav úloh z minulého stretnutia:**

<b>ID</b>	<b>Riešiteľ</b>	<b>Opis úlohy</b>	<b>Stav</b>
50	Jurčacko	Vytvoriť Zápisnicu w5	done
78	Gazdík, Cimerman	Vytvoriť GitFlow metodiku	done
56	Mészáros, Jurčacko, Gazdík	Opraviť warningy - cppCheck	in progress
58	Mészáros	Opraviť warningy - MSVC	in progress
57	Cimerman	Opraviť warningy - CppLint	in progress
12	Jurčacko	"&Multi-select mode" bug	in progress
80	Illés	Aktualizovať inštalačný návod	in progress
41	Cimerman	Vytvoriť dok. s retrospektívou	done
49	Cimerman	Generovanie dokumentácie a diagramov	in progress
2	Fašánek	Nahráť dokumenty na web-stránku	done

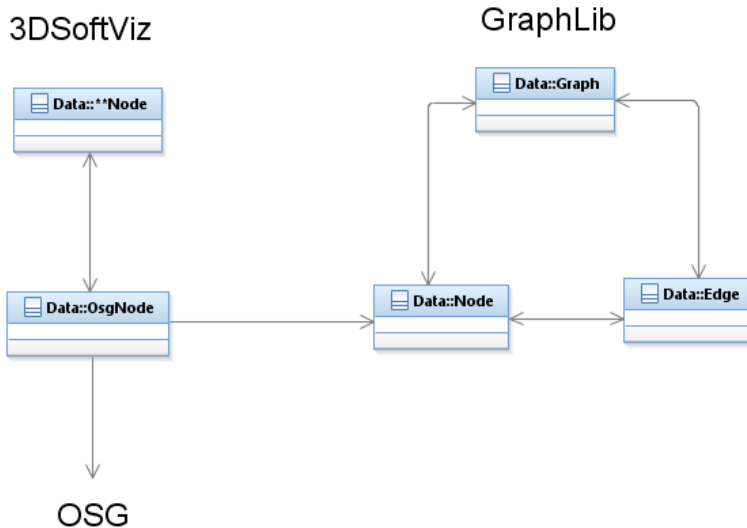
**Priebeh stretnutia:**

- Diskutovali sme zmeny, problémy a riešenia vo vetve "update-aruco"

- Prediskutovali sme našu zverejnenú GitFlow metodiku a dohodli sme sa na dodržovaní opísaných metodík
  - Ohodnotili sme všetky implementačné problémy story pointami, ktorým chýbali zadané hodnoty
  - Popridelovali sme zvyšné nepriradené úlohy jednotlivým členom
  - Zaoberali sme sa problematikou refaktORIZÁCIE pre vzťahy tried "Node", "OsgNode" a "DbNode", "VizNode", "AbsNode"
- Úlohou refaktORIZÁCIE je vyňať spojenie "DbNode", "VizNode", "AbsNode" a zabezpečiť pripojenie k projektu vo forme externej knižnice



- RefaktORIZÁCIU treba aplikovať na sekcie "Node", "Edge" a čiastočne "Graph"



- Odhadovaný čas takejto komplikovanej refaktORIZÁCIE vychádza približne na jeden mesiac až celý semester
- Zaoberali sme sa problematikou 3D myšky a jej integrácie do projektu
  - Treba zistiť aké header súbory treba pridať z ponúkaného SDK
  - Treba modifikovať Cmake a cmake/Find\*, aby bolo možné pridať náš modul



- Je potrebné vytvoriť novú triedu - interface a core
- Interface - minimálne množstvo tried, aby sme mohli vymeniť core v prípade potreby
- Interface - mal by obsahovať v najhoršom prípade zvlášť triedy pre zachytávanie pohybu, rotácie a stlačenia
- Core - trošku viac tried, ako v prípade interface, majú obsahovať priame ovládanie cez SDK
- Namapovať jednotlivé pohyby podľa dizajnu Kinect tried
  - Mali by sme zmazať "ransac" a "vector3" v Kinecte
- Mali by sme si rozmyslieť ako namapujeme jednotlivé pohyby pre "orbitálny fixný" a "voľný letecký" pohyb.
- Na ďalšom stretnutí zozbierame výsledky, dokumenty a pokračujem ďalším šprintom
- Mali by sme si pozrieť informácie k "logForCpp", alebo "logForCxx", ktoré by potenciálne mohli vymeniť QDebug
- V rámci TDD (alebo BDD) by sme si mohli pozrieť čo je "Igloo", ako sa s tým pracuje
- V namespace "OsgModeling" je problém, že "Cube", "Sphere", "SpikySphere" sú mimo svojho namespace
- "HistoryBuffer" by sa mal nachádzať v namespace "Kinect"
- Mali by sme preskúmať adresár 3DSoftVizu a odstrániť nepotrebné súbory:
  - zbytočné .bat súbory
  - staré databázy
  - upratať obrázky v resources/
  - resources/img/ obsahujú staré fotky

### Úlohy na budúce stretnutie:

ID	Riešiteľ	Opis úlohy
49	Cimerman	Generovanie dokumentácie a diagramov
80	Illes	Aktualizovať inštalačný návod
63	Gazdik, Fašánek	3Dconnexion Mouse - preštudovať SDK
83	Cimerman	Vytvoriť metodiku - ako programovať v C++
79	Dyomina	Vytvoriť CodeReview metodiku
41	Meszaros	Vytvoriť dok. s retrospektívou
13	Illes	Tlačidlo "Load function calls" po otvorení priečinka zhodí program

12	Jurcacko	Tlačidlo "&Multi-select mode" vracia cez "getId" NULL, čo zhodí program
57	Cimerman	Opraviť warningy - CppLint
56	Meszaros	Opraviť warningy - cppCheck
58	Meszaros	Opraviť warningy - MSVC
69	Gazdik, Fašánek	3Dconnexion Mouse - vytvoriť diagramy
70	Gazdik, Fašánek	3Dconnexion Mouse - vytvoriť modul v projekte
82	Gazdik, Fašánek	3Dconnexion Mouse - vytvoriť core class / interface

## ***B.g Siedmy týždeň***

**Autor:** Matúš Cimerman

**Dátum stretnutia:** 4.11.2015

**Prítomní:**

Vedúci: Ing. Peter Kapec, PhD.

Členovia tímu: Bc. Matúš Cimerman

Bc. Irina Dyomina

Bc. Michal Fašánek

Bc. Jaroslav Gazdík

Bc. Denis Illés

Bc. Filip Jurčacko

Bc. Dalibor Mészáros

**Stav úloh z minulého stretnutia:**

<b>ID</b>	<b>Riešiteľ</b>	<b>Stav</b>	<b>Opis úlohy</b>
49	Cimerman	Done	Generovanie dokumentácie a diagramov
80	Illes	Done	Aktualizovať inštalačný návod
63	Gazdik, Fašánek	Done	3Dconnexion Mouse - preštudovať SDK
83	Cimerman	in-progress	Vytvoriť metodiku - ako programovať v C++
79	Dyomina	Done	Vytvoriť CodeReview metodiku
41	Meszaros	Done	Vytvoriť dok. s retrospektívou
13	Illes	Ready to re-view	Tlačidlo "Load function calls" po otvorení priečinka zhodí program
12	Jurcacko	Merge od Garaža	Tlačidlo "&Multi-select mode" vracia cez "getId" NULL, čo zhodí program
57	Cimerman	Ready to re-view	Opraviť warningy - CppLint
56	Meszaros	Ready to re-view	Opraviť warningy - cppCheck
58	Meszaros	Ready to re-view	Opraviť warningy - MSVC
69	Gazdik, Fašánek	To-Do	3Dconnexion Mouse - vytvoriť diagramy

70	Gazdik, Fašánek	Ready to review	3Dconnexion Mouse - vytvoriť modul v projekte
82	Gazdik, Fašánek	To-Do	3Dconnexion Mouse - vytvoriť core class / interface

#### Priebeh stretnutia:

- Diskusia o tom ako správne robiť code review:
  - Mergovať aspoň deň pred stretnutím, resp. pred koncom šprintu.
  - Na stretnutí by už malo byť všetko mergnuté a prejdené stavom ready to review.
  - Každý člen tímu by mal sledovať Jiru, a ktoré tasky sú ready to review.
  - Držať sa code review metodiky.
- Diskusia o taskoch:
  - Prečo nič z implementačných taskov nebolo v stave done? Nikto nesledoval Jiru a nerobil code review.
  - Pracovať na taskoch priebežne, nie jeden/dva dni pred stretnutím.
  - Na konci šprintu by mali byť VŠETKY úlohy v stave *done*, s výnimkou block-erov. Každý člen by teda mal mať svoju vetvu - mergnutú v develop vetve.
  - Na konci šprintu ešte celý kód vždy "upratať", aplikovať metodiky - toto by mal robiť každý ešte pred pull-requestom.
- Každý by už mal dodržiavať gitflow metodiku. Zhodnotili sme, že aktuálna gitflow metodika potrebuje nejaké úpravy, ktoré budú hovoriť o ukončení života vetvy a pull-requestoch.
  - Codereview metodika - upraviť, bola napísaná veľmi všeobecne.
  - Zhodnotili sme ako sa nám darilo, či nedarilo počas druhého šprintu.
  - Ako hlavný problém v našom tíme sme identifikovali komunikáciu.
  - Za šprint by sme mali zvládnuť vyriešiť aspoň 30SP.
  - Otvorenie diskusie o dokumentácii, čo všetko chceme dokumentovať?
  - Zápisnice, retrospektívy a percentuálny podiel na šprinte by mal byť odoslaný najneskôr 1 deň po stretnutí alebo ukončení šprintu.

**Úlohy na budúce stretnutie:**

Task	Opraviť error handling	Filip Jurcacko	Ready to review
Task	Dokumentácia k produktu - riadenie	Filip Jurcacko	To Do
Task	Dokumentácia k produktu - BigPicture	Filip Jurcacko	To Do
Task	Dokumentácia k produktu - 3 šprinty	Filip Jurcacko	To Do
Task	Upload dokumentov na stránku	Michal Fasanek	To Do
Task	Nájsť všetky cudzie kódy	Irina Dyomina	In Progress
Task	Zápisnica W7	Matus Cimerman	In Progress
Bug	Opraviť warningy - OS X	Matus Cimerman	In Progress
Task	Začleniť triedy do príslušných namespaceov	Denis Illes	In Progress
Task	Zmazať zbytočné triedy a vyskúšať program	Dalibor Meszaros	To Do
Task	Vytvoriť metodiku - ako programovať v C++	Matus Cimerman	In Progress
Task	3Dconnexion Mouse - vytvoriť core class / interface	Jaroslav Gazdik	In Progress
Task	3Dconnexion Mouse - namapovať na ovládanie virtuálnej kamery	Michal Fasanek	To Do
Task	3Dconnexion Mouse - vytvoriť modul v projekte	Jaroslav Gazdik	Ready to review
Task	3Dconnexion Mouse - vytvoriť diagramy	Irina Dyomina	To Do
Bug	Opraviť warningy - MSVC	Dalibor Meszaros	Ready to review
Bug	Opraviť warningy - CppLint	Matus Cimerman	Ready to review
Bug	Opraviť warningy - cppCheck	Dalibor Meszaros	Ready to review
Task	Vytvoriť dok. s retrospektívou	Dalibor Meszaros	To Do
Task	linkovať sa na osgModelling	Peter Kapec	In Progress
Bug	Tlačidlo "Load function calls" po otvorení priečinka zhodí program	Denis Illes	Ready to review
Bug	Tlačidlo "&Multi-select mode" vracia cez "getld" NULL, čo zhodí program	Filip Jurcacko	In Progress
Task	Aktualizovať používateľskú príručku	Denis Illes	To Do

## ***B.h*** Ôsmy týždeň

**Autor:** Michal Fašánek

**Dátum stretnutia:** 11.11.2015

**Prítomní:**

Vedúci: Ing. Peter Kapec, PhD.

Členovia tímu: Bc. Matúš Cimerman

Bc. Irina Dyomina

Bc. Michal Fašánek

Bc. Jaroslav Gazdík

Bc. Denis Illés

Bc. Filip Jurčacko

Bc. Dalibor Mészáros

**Stav úloh z minulého stretnutia:**

<b>ID</b>	<b>Riešiteľ</b>	<b>Opis úlohy</b>	<b>Stav</b>
57	Cimerman	Opraviť warningy CppLint	done
84	Meszároš	Zmazať zbytočné triedy	done
70	Gazdík	3DConnexion mouse Vytvoriť modul v projekte	done
88	Dyomina	Nájsť cudzie kódy	done
13	Illés	Tlačidlo "Load function calls" zhodí program	ready to review
96	Jurčacko	Opraviť error handling	ready to review
56	Meszároš	Opraviť CppCheck warningy	ready to review
58	Meszároš	Opraviť MSCV warningy	ready to review
17	Kapec	Linkovať sa na OSG mode- ling	ready to review
12	Jurčacko	Tlačidlo "Multi-select Node" vracia NULL, čo zhodí pro- gram	in progress
85	Illés	Začleniť triedy do príslušných namespace-ov	in progress
86	Cimerman	Opraviť warningy - OSX	in progress

69	Dyomina	3DConnexion mouse vytvorit' diagramy	in progress
82	Gazdik	3DConnexion mouse vytvorit' core class/interface	in progress
72	Fašánek	3DConnexion mouse nama- povat' na ovládanie virtuálnej kamery	to do

#### Priebeh stretnutia:

- Diskutovali sme o úspešnosti prvého týždňa šprintu
- Analyzovali sme stav úloh a pridávali nové úlohy
- Diskutovali sme o dokumentácií a spôsoboch podieľania sa jednotlivých členov
  - Ako vyhodnocovať úlohy
  - Rozpracovanie implementačných úloh
  - Možnosti písania univerzálnej dokumentácie v ktorej by mohli pokračovať aj ďalšie tímy
  - Zmeny v predošlých implementáciách
  - Diagramy
  - Big Picture obsah
- Prístup do miestnosti 1.27
- Referenčný model GitLib
  - namapovanie na 3dConnexion myš
- Súbor v Resource – čo sa ešte používa?

#### Úlohy na budúce stretnutie:

ID	Riešiteľ	Opis úlohy
13	Illés	Tlačidlo "Load function calls" zhodí program
96	Jurčacko	Opraviť error handling
72	Fašánek	3dConnexion mouse namapovať na ovládanie virtuálnej kamery
56	Meszároš	Opraviť CppCheck warningy
58	Meszároš	Opraviť MSCV warningy

17	Kapec	Linkovať sa na OSG modeling
12	Jurčacko	Tlačidlo "Multi-select Node" vracia NULL, čo zhodí program
83	Dyomina	Metodika programovania v C++
11	Illés	Aktualizovať používateľskú príručku
92	Jurčacko	Dokumentácia k produktu – 3 šprinty



## ***B.i Deviaty týždeň***

**Autor:** Matúš Cimerman

**Dátum stretnutia:** 18.11.2015

**Prítomní:** Vedúci: Ing. Peter Kapec, PhD.  
Členovia tímu: Bc. Matúš Cimerman  
Bc. Irina Dyomina  
Bc. Michal Fašánek  
Bc. Jaroslav Gazdík  
Bc. Denis Illés  
Bc. Filip Jurčacko  
Bc. Dalibor Mészáros

### **Priebeh stretnutia:**

- Diskusia o OSG Modelingu,
- Pripraviť demo 3D myši – najlepšie tak, aby bola funkčná do DOD na FIIT.
- Refaktorovať diluculum mimo namespace
  - Lua, LuaGraph a LuaInterface by sa mali vyňať z projektu .
  - Do LuaGraphu a LuaInterface pridať metódy aby Lua nepoužíval Diluculum priamo.
  - Nachádza sa na 5-6 miestach v kóde.
- Diskusia o stereoskopickom zobrazení
  - OSG podpora
- Diskusia o multiview zobrazení - viac monitorov
  - OSG podpora,
  - Existujú example na to – jedno OSG okno rozťahnuté cez 3 monitory.
- Treba otestovať kalibrovanie kamery:
  - Každá kamera ma iné skreslenie,
  - Diplomovka Juríka.
- Zhodli sme sa na tom, že pripojíme výstup z PlantUML do doxygenu
- Otvorili sme diskusiu o udržiavaní robustnej a hodnotnej dokumentácie pre tento projekt, ktorú by tvorili aj nasledujúce tímové projekty po nás.
  - Nájsť možnosti generovania dokumentácie z textových súborov (niečo ako markdown štýlovanie)
  - Predpokladom sú napríklad:
    - dobre okomentovaný zdrojový kód,
    - unit testy,
    - identifikované požiadavky a analýza,
  - Nové implementované moduly do projektu by mali byť ukážkou.

## Stav úloh z minulého stretnutia:

Zapisnica w8	Michal Fasanek	Done	V prílohe export taskov z Jiry. w8-1_status - status uloh z minuleho tyzdna w8_todo-inprogress - zvsyne ulohy na buduci
Opraviť error handling	Filip Jurcacko	Done	Na viacerých miestach v kode je do funkcií posielany argument bool* error ktorý sa vnútri funkcie nastavuje v prípade chyby  Avsak často po vykonaní funkcií buď nieje skontrolovaný vôbec alebo je viacnásobne
Dokumentácia k produktu - riadenie	Filip Jurcacko	Done	BIELIKOVA <a href="http://www2.fiit.stuba.sk/~bielik/courses/tp-slov/tp-main.html">http://www2.fiit.stuba.sk/~bielik/courses/tp-slov/tp-main.html</a>
Dokumentácia k produktu - BigPicture	Filip Jurcacko	Done	BIELIKOVÁ <a href="http://www2.fiit.stuba.sk/~bielik/courses/tp-">http://www2.fiit.stuba.sk/~bielik/courses/tp-</a>
Dokumentácia k produktu - 3 šprinty	Filip Jurcacko	Done	do 18.11. v AIS
Nájsť všetky cudzie kódy	Irina Dyomina	Done	identifikovať všetky cudzie kódy, a nájsť odkiaľ boli zobrať  napr: foldre: src > Noise (hlavička zobrazuje že kód je
Opraviť warningy - OS X	Matus Cimerman	Done	
Začleniť triedy do príslušných namespaceov	Denis Illes	Done	Triedy ako "Cube", "Sphere", "SpikySphere" sa nachádzajú mimo svojho namespaceu (Clustering). Treba ich začleniť. Rovnako ako "HistoryBuffer" (malo by to byť v kinect namespace)
Vytvoriť metodiku - ako programovať v C++	Matus Cimerman	Done	Vytvoriť dokument, v ktorom sa opíše postup pri programovaní v jazyku cpp. Zaznamenajú sa základné princípy, ktorých sa budeme držať pri vývoji softvéru. * astyle,
3Dconnexion Mouse - vytvoriť core class / interface	Jaroslav Gazdik	In Progress	Vytvoriť triedy na spracovanie a konvertovanie údajov získaných z myšky, napísať funkcie, ktoré pohyby v jednotlivých smeroch spracuje a poskytne projektu. Príklad Core triedy - Kinect. Vytvoriť interface, cez ktoré bude komunikovať modul (myš) s existujúcim projektom 3DSoftViz.
3Dconnexion Mouse - namapovať na ovládanie virtuálnej kamery	Michal Fasanek	In Progress	Poskytnuté údaje cez rozhranie použiť na ovládanie virtuálnej kamery. Ovládanie v oboch režimoch - voľný a orbitálny pohyb.
3Dconnexion Mouse - vytvoriť diagramy	Irina Dyomina	Done	Vytvoriť diagram s triedami znázorňujúce začlenenie modulu do projektu. Inšpirovať sa s modulom Kinect.
Opraviť warningy - MSVC	Dalibor Meszaros	Done	Warningy detekované nástrojom MSVC
Opraviť warningy - cppCheck	Dalibor Meszaros	Done	Warningy detekované nástrojom cppCheck
linkovať sa na osgModelling	Peter Kapec	Done	treba odstrániť z projektu súbory v Math/ - Bezier - Curve - Export
Tlačidlo "Load function calls" po otvorení priečinka zhodí program	Denis Illes	Done	Pri otváraní zložky s .lua súbormi program crashne. Zrejme Windows related issue, keďže na iOS funguje bez akýchkoľvek problémov.  Riešenie: zakomentovať ...\\3dsoftviz\\resources\\scripts\\luadb\\extraction\\extractor.lua line 182 (assert) alebo zmazať druhú podmienku > assert(path, "wrong path
Tlačidlo "&Multi-select mode" vracia cez "getId" NULL, čo zhodí program	Filip Jurcacko	Done	Pri označovaní viacerých uzlov program crashne.
Aktualizovať používateľskú príručku	Denis Illes	Done	

## Úlohy na budúce stretnutie:

Upratať nesprávne umiestnené veci z 3dsodtviz	Denis Illes	In Progress	doc - doxygen resources
Aruco example testy	<i>Unassigned</i>	To Do	vysúšať 3dsoftviz\_install\bin\aruco_test_gl.exe a ostatné *_gl.exe  arg. live camera.yml 0.025  freeglut.dll prekopirovať z
Refaktoring časti: Kinect, Leap, OpenCV	Filip Jurcacko	In Progress	inšpirácia - Garaj
OSG Jedno okno - rozťahnuté cez 3 monitory	Matus Cimerman	In Progress	
OSG Multiview	Denis Illes	In Progress	OSQ multiview/window - projektory virtualCave equalizer - skúsiť googlit
Refaktoring nahradiť cudzie kody originalmi	Irina Dyomina	To Do	
Refaktoring - diluculum	Dalibor Meszaros	In Progress	diluculum použitý mimo namespace Lua (triedy LuaGraph, LuaInterface) riešenie: LuaGraph/Interface - pridať metódy
3Dconnexion Mouse - vytvoriť core class / interface	Jaroslav Gazdik	In Progress	Vytvoriť triedy na spracovanie a konvertovanie údajov získaných z myšky, napísať funkcie, ktoré pohyby v jednotlivých smeroch spracuje a poskytne projektu. Príklad Core triedy - Kinect. Vytvoriť interface, cez ktoré bude komunikovať modul (myš) s existujúcim projektom 3DSoftViz.
3Dconnexion Mouse - namapovať na ovládanie virtuálnej kamery	Michal Fasanek	In Progress	Poskytnuté údaje cez rozhranie použiť na ovládanie virtuálnej kamery. Ovládanie v oboch režimoch - voľný a orbitálny pohyb.
AUGREALITY-18 Opraviť CameraManipulator.cpp	<i>Unassigned</i>	To Do	
AUGREALITY-18 Opraviť QtColorPicker.cpp	<i>Unassigned</i>	Ready to review	
AUGREALITY-18 Opraviť NoiseUtils.cpp	<i>Unassigned</i>	Ready to review	
Presunúť "cudzí" kód do dependencies	Peter Kapec	To Do	Zdrojové súbory: NoiseUtils.cpp, QtColorPicker.cpp, CameraManipulator.cpp

## B.j Desiaty týždeň

**Autor:** Dalibor Mészáros

**Dátum stretnutia:** 25.11.2015

**Prítomní:** Vedúci: Ing. Peter Kapec, PhD.  
Členovia tímu: Bc. Matúš Cimerman  
Bc. Irina Dyomina  
Bc. Michal Fašánek  
Bc. Jaroslav Gazdík  
Bc. Denis Illés  
Bc. Filip Jurčacko  
Bc. Dalibor Mészáros

### Priebeh stretnutia:

- Diskusia ku dňu otvorených dverí a otázke prezentovania projektu 3dSoftViz:
  - Podpora zobrazenia Cave
  - Podpora zobrazenia na viacerých obrazovkách (rozšírenie)
  - Zhotoviť fotografickú dokumentáciu na prezentovanie tímu a projektu v akcií
- Boli vykonaný merge pre gitlib a osgmodeling
- Boli diskutované problémy, s ktorými mali problémy jednotliví členovia tímu s ich pridelenými úlohami

### Stav úloh z minulého stretnutia:

Upratať nesprávne umiestnené veci z 3dsodtviz	Denis Illés	In Progress	doc - doxygen resources
Aruco example testy	<i>Unassigned</i>	To Do	vysúšať 3dsoftviz\_install\bin\aruco_test_gl.exe a ostatné *_gl.exe arg. live camera.yml 0.025 freeglut.dll prekopirovať z
Refaktoring časti: Kinect, Leap, OpenCV	Filip Jurčacko	In Progress	inšpirácia - Garaj
OSG Jedno okno - rozťahnuté cez 3 monitory	Matus Cimerman	In Progress	
OSG Multiview	Denis Illés	In Progress	OSQ multiview/window - projektory virtualCave equalizer - skúsiť googlit
Refaktoring nahradiť cudzie kody originalmi	Irina Dyomina	To Do	
Refaktoring - diluculum	Dalibor Meszaros	In Progress	diluculum použitý mimo namespace Lua (triedy LuaGraph, LuaInterface) riešenie: LuaGraph/Interface - pridať metódy
3Dconnexion Mouse - vytvoriť core class / interface	Jaroslav Gazdík	In Progress	Vytvoríť triedy na spracovanie a konvertovanie údajov získaných z myšky, napísať funkcie, ktoré pohyby v jednotlivých smeroch spracuje a poskytne projektu. Príklad Core triedy - Kinect. Vytvoríť interface, cez ktoré bude komunikovať modul (myš) s existujúcim projektom 3DSoftViz.
3Dconnexion Mouse - namapovať na ovládanie virtuálnej kamery	Michal Fasánek	In Progress	Poskytnuté údaje cez rozhranie použiť na ovládanie virtuálnej kamery. Ovládanie v oboch režimoch - voľný a orbitálny pohyb.
AUGREALITY-18 Opraviť CameraManipulator.cpp	<i>Unassigned</i>	To Do	
AUGREALITY-18 Opraviť QtColorPicker.cpp	<i>Unassigned</i>	Ready to review	
AUGREALITY-18 Opraviť NoiseUtils.cpp	<i>Unassigned</i>	Ready to review	
Presunúť "cudzi" kód do dependencies	Peter Kapec	To Do	Zdrojové súbory: NoiseUtils.cpp, QtColorPicker.cpp, CameraManipulator.cpp

## Úlohy na budúce stretnutie:

Upratať nesprávne umiestnené veci z 3dsotviz	Denis Illes	In Progress	doc - doxygen resources .bat delete logging ako modul - <a href="https://github.com/LuaDist/lualogging">https://github.com/LuaDist/lualogging</a>
Aruco example testy	Unassigned	To Do	vysúšať 3dsotviz\install\bin\aruco_test_gl.exe a ostatné *_gl.exe arg. live camera.yml 0.025 freeglut.dll prekopiovať z "OSG\ThirdParty\VC10\x86\bin\" a premenovať na freeglut.dll camera.yml prekopiovať z "_install\share\3dsotviz\config"
Refaktoring časti: Kinect, Leap, OpenCV	Filip Jurcacko	In Progress	inšpirácia - Garaj - gitlib spraviť ako osobitne externé libky da sa samostatne skompilovať, netreba mu 3dsotviz pozreť aj CMakeLists.txt GitGraphImporter a GitGraphUpdater su interface, používajú 3dsotviz a lib
OSG Jedno okno - rozťahnuté cez 3 monitory	Matus Cimerman	In Progress	
OSG Multiview	Denis Illes	In Progress	OSQ multiview/window - projekty virtualCave equalizer - skúsiť googlit tasky - analýza 3 okná - iná kamera, pre každé fullscreen.. pozrieť si example
Refaktoring nahradiť cudzie kody originalmi	Irina Dyomina	To Do	
Refaktoring - diluculum	Dalibor Meszaros	In Progress	diluculum použitý mimo namespace Lua (triedy LuaGraph, LuaInterface) riešenie: LuaGraph/Interface - pridať metódy
3Dconnexion Mouse - vytvoriť core class / interface	Michal Fasanek	In Progress	Vytvoriť triedy na spracovanie a konvertovanie údajov získaných z myšky, napísať funkcie, ktoré pohyby v jednotlivých smeroch spracuje a poskytne projektu. Príklad Core triedy - Kinect. Vytvoriť interface, cez ktoré bude komunikovať modul (myš) s existujúcim projektom 3DSotViz.
3Dconnexion Mouse - namapovať na ovládanie virtuálnej kamery	Michal Fasanek	In Progress	Poskytnuté údaje cez rozhranie použiť na ovládanie virtuálnej kamery. Ovládanie v oboch režimoch - voľný a orbitálny pohyb.
Presunúť "cudzi" kód do dependencies	Peter Kapec	To Do	Zdrojové súbory: NoiseUtils.cpp, QColorPicker.cpp, CameraManipulator.cpp
linkovať sa na osgModelling	Peter Kapec	Done	treba odstrániť z projektu súbory v Math/ Bezier, Curve, Export, Model, Utilities a linkovať sa na osgModeling knižnicu, ktorú treba pridať do dependencies ako submodule
Tlačidlo "&Multi-select mode" vracia cez "getId" NULL, čo zhodí program	Filip Jurcacko	Done	Pri označovaní viacerých uzlov program crashne.

## ***B.k Jedenásty týždeň***

**Autor:** Matúš Cimerman

**Dátum strenutia:** 2.12.2015

**Prítomní:** Vedúci: Ing. Peter Kapec, PhD.  
Členovia tímu: Bc. Matúš Cimerman  
Bc. Irina Dyomina  
Bc. Michal Fašánek  
Bc. Jaroslav Gazdík  
Bc. Denis Illés  
Bc. Filip Jurčacko  
Bc. Dalibor Mészáros

### **Priebeh strenutia:**

- Diskusia k prezentácii RTP,
  - Ako funguje riadenie tímu?
  - Ake problémy nastali s aplikovaním metodík v tíme?
  - Ako si plnia svoje manažerské roly členovia tímu?
- V tomto šprinte sa treba zamerať na dokočenie úloh a ich adekvátnu dokumentáciu v inžinierskom diele.
  - Dokončiť minimálne 3D myš a otvorený refactoring, ktorý bude možné dokončiť.
- Zhodli sme sa na tom, že budeme pracovať na projekte aj cez sviatky a skúškové obdobie aby sme projekt posunuli. Je to z viacerých dôvodov:
  - Projekt je veľký a trvalo nám dlho kým sme sa zoznámili s jeho všetkými súčasťami.
  - Chceme projekt niekam posunúť – implementovať obohatenú realitu s použitím 3D okuliarov, implementovanie CAVE zobrazovania.
- Diskusia k dokumentácii:
  - Dokumentácia by mala pre náš projekt predstavovať skôr technickú správu.
  - Mali by sme ju transformovať do stavu „in-progress“ tak, aby mohli v dokumente pokračovať aj ďalšie tímové projekty
  - Zhodli sme sa, že má zmysel dokumentovať aj rozumnú časť analýzy problému, napríklad pri implementácii nového modulu do projektu.
- Treba otestovať webkameru a sledovanie tváre.

## Stav úloh z minulého stretnutia:

Upratať nesprávne umiestnené veci z 3dsodtviz	Denis Illes	In Progress	doc - doxygen resources .bat delete logging ako modul - <a href="https://github.com/LuaDist/lualogging">https://github.com/LuaDist/lualogging</a>
Aruco example testy	Unassigned	To Do	vysúšať 3dsodtviz\install\bin\aruco_test_gl.exe a ostatné *_gl.exe arg. live camera.yml 0.025 freeglut.dll prekopiovať z "OSG\ThirdParty\VC10\x86\bin\" a premenovať na freeglut.dll camera.yml prekopiovať z "_install\share\3dsodtviz\config"
Refaktoring časti: Kinect, Leap, OpenCV	Filip Jurcacko	In Progress	inšpirácia - Garaj - gitlib spraviť ako osobitne externé libky da sa samostatne skompilovať, netreba mu 3dsodtviz pozret aj CMakeLists.txt GitGraphImporter a GitGraphUpdater su interface, používajú 3dsodtviz a lib
OSG Jedno okno - rozťahnuté cez 3 monitory	Matus Cimerman	In Progress	
OSG Multiview	Denis Illes	In Progress	OSQ multiview/window - projekory virtualCave equalizer - skusit googlit tasky - analýza 3 okná - iná kamera, pre každé fullscreen.. pozrieť si example
Refaktoring nahradiť cudzie kody originalmi	Irina Dyomina	To Do	
Refaktoring - diluculum	Dalibor Meszaros	In Progress	diluculum použitý mimo namespace Lua (triedy LuaGraph, LuaInterface) riešenie: LuaGraph/Interface - pridať metódy
3Dconnexion Mouse - vytvoriť core class / interface	Michal Fasanek	In Progress	Vytvoriť triedy na spracovanie a konvertovanie údajov získaných z myšky, napísať funkcie, ktoré pohyby v jednotlivých smeroch spracuje a poskytne projektu. Príklad Core triedy - Kinect. Vytvoriť interface, cez ktoré bude komunikovať modul (myš) s existujúcim projektom 3DSodtViz.
3Dconnexion Mouse - namapovať na ovládanie virtuálnej kamery	Michal Fasanek	In Progress	Poskytnuté údaje cez rozhranie použiť na ovládanie virtuálnej kamery. Ovládanie v oboch režimoch - voľný a orbitálny pohyb.
Presunúť "cudzí" kod do dependencies	Peter Kapec	To Do	Zdrojové súbory: NoiseUtils.cpp, QColorPicker.cpp, CameraManipulator.cpp
linkovať sa na osgModelling	Peter Kapec	Done	treba odstrániť z projektu súbory v Math/ Bezier, Curve, Export, Model, Utilities a linkovať sa na osgModeling knižnicu, ktorú treba pridať do dependencies ako submodule
Tlačidlo "&Multi-select mode" vracia cez "getId" NULL, čo zhodí program	Filip Jurcacko	Done	Pri označovaní viacerých uzlov program crashne.

## Úlohy na budúce stretnutie:

Upratať nesprávne umiestnené veci z 3dsodtviz	Denis Illes	In Progress	doc - doxygen resources .bat delete logging ako modul - <a href="https://github.com/LuaDist/lualogging">https://github.com/LuaDist/lualogging</a>
Aruco example testy	Unassigned	To Do	vysúšať 3dsodtviz\install\bin\aruco_test_gl.exe a ostatné *_gl.exe arg. live camera.yml 0.025 freeglut.dll prekopiovať z "OSG\ThirdParty\VC10\x86\bin\" a premenovať na freeglut.dll camera.yml prekopiovať z "_install\share\3dsodtviz\config"
Refaktoring časti: Kinect, Leap, OpenCV	Filip Jurcacko	In Progress	inšpirácia - Garaj - gitlib spraviť ako osobitne externé libky da sa samostatne skompilovať, netreba mu 3dsodtviz pozret aj CMakeLists.txt GitGraphImporter a GitGraphUpdater su interface, používajú 3dsodtviz a lib
OSG Jedno okno - rozťahnuté cez 3 monitory	Matus Cimerman	In Progress	
OSG Multiview	Denis Illes	In Progress	OSQ multiview/window - projekory virtualCave equalizer - skusit googlit tasky - analýza 3 okná - iná kamera, pre každé fullscreen.. pozrieť si example
Refaktoring nahradiť cudzie kody originalmi	Irina Dyomina	To Do	
Refaktoring - diluculum	Dalibor Meszaros	In Progress	diluculum použitý mimo namespace Lua (triedy LuaGraph, LuaInterface) riešenie: LuaGraph/Interface - pridať metódy
3Dconnexion Mouse - vytvoriť core class / interface	Michal Fasanek	In Progress	Vytvoriť triedy na spracovanie a konvertovanie údajov získaných z myšky, napísať funkcie, ktoré pohyby v jednotlivých smeroch spracuje a poskytne projektu. Príklad Core triedy - Kinect. Vytvoriť interface, cez ktoré bude komunikovať modul (myš) s existujúcim projektom 3DSodtViz.
3Dconnexion Mouse - namapovať na ovládanie virtuálnej kamery	Michal Fasanek	In Progress	Poskytnuté údaje cez rozhranie použiť na ovládanie virtuálnej kamery. Ovládanie v oboch režimoch - voľný a orbitálny pohyb.
Presunúť "cudzí" kod do dependencies	Peter Kapec	To Do	Zdrojové súbory: NoiseUtils.cpp, QColorPicker.cpp, CameraManipulator.cpp
linkovať sa na osgModelling	Peter Kapec	Done	treba odstrániť z projektu súbory v Math/ Bezier, Curve, Export, Model, Utilities a linkovať sa na osgModeling knižnicu, ktorú treba pridať do dependencies ako submodule
Tlačidlo "&Multi-select mode" vracia cez "getId" NULL, čo zhodí program	Filip Jurcacko	Done	Pri označovaní viacerých uzlov program crashne.



## ***B.1 Dvanásty týždeň***

**Autor:** Matúš Cimerman

**Dátum stretnutia:** 9.12.2015

**Prítomní:** Vedúci: Ing. Peter Kapec, PhD.  
Členovia tímu: Bc. Matúš Cimerman  
Bc. Irina Dyomina  
Bc. Michal Fašánek  
Bc. Jaroslav Gazdík  
Bc. Denis Illés  
Bc. Filip Jurčacko  
Bc. Dalibor Mészáros

### **Priebeh stretnutia:**

- Posledné stretnutie v zimnom semestri.
- Zhodli sme sa, že budeme na projekte pracovať aj cez sviatky a skúškové obdobie aby sme projekt posunuli a dosiahli všetky ciele, ktoré sme si stanovili na začiatku projektu.
- Diskusia najmä o tom, prečo sa nám nepodarilo dokončiť posledné úlohy:
  - Riešili sme hlavne dokumentáciu,
  - Príprava na deň otvorených dverí na FIIT kde sme prezentovali 3Dsoftviz spolu s Kinect-om
  - Posledný šprint trval o týždeň menej ako ostatné šprinty – nedostatok času
  - Nedokončené úlohy ostávajú s cieľom dokončiť ich do Vianoc resp. cez vianočné sviatky

## Stav úloh z minulého stretnutia:

Upratať nesprávne umiestnené veci z 3dsodtviz	Denis Illes	In Progress	doc - doxygen resources .bat delete logging ako modul - <a href="https://github.com/LuaDist/lualogging">https://github.com/LuaDist/lualogging</a>
Aruco example testy	Unassigned	To Do	vysúšať 3dsodtviz\_install\bin\aruco_test_gl.exe a ostatné *_gl.exe arg. live camera.yml 0.025 freeglut.dll prekopiovať z "OSG\ThirdParty\VC10x86\bin\" a premenovať na freeglut.dll camera.yml prekopiovať z "_install\share\3dsodtviz\config"
Refaktoring časti: Kinect, Leap, OpenCV	Filip Jurcacko	In Progress	inšpirácia - Garaj - gitlib spraviť ako osobitne externé líbky da sa samostatne skompilovať, netreba mu 3dsodtviz pozret aj CMakeLists.txt GitGraphImporter a GitGraphUpdater su interface, používajú 3dsodtviz a lib
OSG Jedno okno - rozťahnuté cez 3 monitory	Matus Cimerman	In Progress	
OSG Multiview	Denis Illes	In Progress	OSQ multiview/window - projektory virtualCave equalizer - skusiť googlit tasky - analýza 3 okná - iná kamera, pre každé fullscreen.. pozrieť si example
Refaktoring nahradiť cudzie kody originalmi	Irina Dyomina	To Do	
Refaktoring - diluculum	Dalibor Meszaros	In Progress	diluculum použitý mimo namespace Lua (triedy LuaGraph, LuaInterface) riešenie: LuaGraph/Interface - pridať metódy
3Dconnexion Mouse - vytvoríť core class / interface	Michal Fasanek	In Progress	Vytvoríť triedy na spracovanie a konvertovanie údajov získaných z myšky, napísať funkcie, ktoré pohyby v jednotlivých smeroch spracuje a poskytne projektu. Príklad Core triedy - Kinect. Vytvoríť interface, cez ktoré bude komunikovať modul (myš) s existujúcim projektom 3DSoftViz.
3Dconnexion Mouse - namapovať na ovládanie virtuálnej kamery	Michal Fasanek	In Progress	Poskytnuté údaje cez rozhranie použít na ovládanie virtuálnej kamery. Ovládanie v oboch režimoch - voľný a orbitálny pohyb.
Presunúť "cudzí" kod do dependencies	Peter Kapec	To Do	Zdrojové súbory: NoiseUtils.cpp, QColorPicker.cpp, CameraManipulator.cpp
linkovať sa na osgModeling	Peter Kapec	Done	treba odstrániť z projektu súbory v Math/ Bezier, Curve, Export, Model, Utilities a linkovať sa na osgModeling knižnicu, ktorú treba pridať do dependencies ako submodule
Tlačidlo "&Multi-select mode" vracia cez "getid" NULL, čo zhodí program	Filip Jurcacko	Done	Pri označovaní viacerých uzlov program crashne.

## Úlohy na budúce stretnutie:

Upratať nesprávne umiestnené veci z 3dsodtviz	Denis Illes	In Progress	doc - doxygen resources .bat delete logging ako modul - <a href="https://github.com/LuaDist/lualogging">https://github.com/LuaDist/lualogging</a>
Aruco example testy	Unassigned	To Do	vysúšať 3dsodtviz\_install\bin\aruco_test_gl.exe a ostatné *_gl.exe arg. live camera.yml 0.025 freeglut.dll prekopiovať z "OSG\ThirdParty\VC10x86\bin\" a premenovať na freeglut.dll camera.yml prekopiovať z "_install\share\3dsodtviz\config"
Refaktoring časti: Kinect, Leap, OpenCV	Filip Jurcacko	In Progress	inšpirácia - Garaj - gitlib spraviť ako osobitne externé líbky da sa samostatne skompilovať, netreba mu 3dsodtviz pozret aj CMakeLists.txt GitGraphImporter a GitGraphUpdater su interface, používajú 3dsodtviz a lib
OSG Jedno okno - rozťahnuté cez 3 monitory	Matus Cimerman	In Progress	
OSG Multiview	Denis Illes	In Progress	OSQ multiview/window - projektory virtualCave equalizer - skusiť googlit tasky - analýza 3 okná - iná kamera, pre každé fullscreen.. pozrieť si example
Refaktoring nahradiť cudzie kody originalmi	Irina Dyomina	To Do	
Refaktoring - diluculum	Dalibor Meszaros	In Progress	diluculum použitý mimo namespace Lua (triedy LuaGraph, LuaInterface) riešenie: LuaGraph/Interface - pridať metódy
3Dconnexion Mouse - vytvoríť core class / interface	Michal Fasanek	In Progress	Vytvoríť triedy na spracovanie a konvertovanie údajov získaných z myšky, napísať funkcie, ktoré pohyby v jednotlivých smeroch spracuje a poskytne projektu. Príklad Core triedy - Kinect. Vytvoríť interface, cez ktoré bude komunikovať modul (myš) s existujúcim projektom 3DSoftViz.
3Dconnexion Mouse - namapovať na ovládanie virtuálnej kamery	Michal Fasanek	In Progress	Poskytnuté údaje cez rozhranie použít na ovládanie virtuálnej kamery. Ovládanie v oboch režimoch - voľný a orbitálny pohyb.
Presunúť "cudzí" kod do dependencies	Peter Kapec	To Do	Zdrojové súbory: NoiseUtils.cpp, QColorPicker.cpp, CameraManipulator.cpp
linkovať sa na osgModeling	Peter Kapec	Done	treba odstrániť z projektu súbory v Math/ Bezier, Curve, Export, Model, Utilities a linkovať sa na osgModeling knižnicu, ktorú treba pridať do dependencies ako submodule
Tlačidlo "&Multi-select mode" vracia cez "getid" NULL, čo zhodí program	Filip Jurcacko	Done	Pri označovaní viacerých uzlov program crashne.

## Príloha C Exporty úloh z Jiry

### C.a Úlohy z prvého šprintu

Issue Type	Summary	Assignee	Status	Description	Epic Name	Labels
Sub-task	AUGREALITY-5 Aktualizovať OSG 3.4	Denis Illes	Done			implement
Sub-task	AUGREALITY-5 Aktualizovať Aruco	Peter Kapec	Done			implement
Sub-task	AUGREALITY-21 Oboznámiť sa s cppLint	<i>Unassigned</i>	Done			implement
Task	Nastaviť editor v QtCreator	<i>Unassigned</i>	Done	Tab Policy - Tab Only Tab Size - 4, Indent Size - 4 Align Continuation Lines - Not at all		other
Sub-task	AUGREALITY-21 Oboznámiť sa s cppCheck	<i>Unassigned</i>	Done			implement
Task	Vyskúšať nástroje pre C++	<i>Unassigned</i>	Done	Nástroje cppCheck a cppLint		implement

Bug	Tlačidlo "Load function calls" po otvorení priečinka zhodí program	Denis Illes	Done	Pri otváraní zložky s .lua súbormi program crashne. Zrejme Windows related issue, keďže na iOS funguje bez akýchkoľvek problémov. Riešenie: zakomentovať ...\\3dsoftviz\\resources\\scripts\\luadb\\extraction\\extractor.lua line 182 (assert) alebo zmazať druhú podmienku > assert(path, "wrong path passed")		implement
Bug	Tlačidlo "&Multi-select mode" vracia cez "getId" NULL, čo zhodí program	Filip Jurcacko	Ready to review	Pri označovaní viacerých uzlov program crashne.		implement
Task	Includnúť chýbajúce headre pre non-unity build	Denis Illes	Done			implement
Task	Opraviť warningy	Dalibor Meszaros	Done			implement
Sub-task	AUGREALITY-7 Pridať na web zázpisnice z 2-4w	Matus Cimerman	Done			other
Task	Šablóna pre zázpisnice zo strenutí	Jaroslav Gazdik	Done	Vytvorit gdoc sablonu pre zapisnice zo strenuti - pre web.		other

Task	Aktualizovať knižnice v projekte	<i>Unassigned</i>	Done	Knižnice Aruco, OpenSceneGraph 3.4		implement
Task	Nastaviť JIRU podľa potreby	Denis Illes	Done			other
Task	Upraviť inštalačný manuál	<i>Unassigned</i>	Done			other
Task	Nahrať dokumenty na webovú stránku	Michal Fasanek	Done			other
Task	Dokončiť webovú stránku	Michal Fasanek	Done			other

**C.b Úlohy z druhého šprintu**

<b>Issue Type</b>	<b>Summary</b>	<b>Assignee</b>	<b>Status</b>	<b>Description</b>	<b>Epic Name</b>	<b>Labels</b>
Task	Zapísnička w8	Michal Fasánek	Done	V prílohe export úloh z Jiry. w8-1_status - status úloh z minulého týždňa w8_todo-inprogress - zvyšné úlohy na budúci týždeň		other
Task	Gitflow metodiku upraviť	Jaroslav Gazdik	Done	Pill requesty a nič viac...		other
Task	Dokumentácia k produktu - riadenie	Filip Jurcacko	To Do	BIELIKOVÁ <a href="http://www2.fiit.stuba.sk/~bielik/courses/tp-slov/tp-main.html">http://www2.fiit.stuba.sk/~bielik/courses/tp-slov/tp-main.html</a>		other
Task	Dokumentácia k produktu - BigPicture	Filip Jurcacko	To Do	BIELIKOVÁ <a href="http://www2.fiit.stuba.sk/~bielik/courses/tp-slov/tp-main.html">http://www2.fiit.stuba.sk/~bielik/courses/tp-slov/tp-main.html</a>		other
Task	Dokumentácia k produktu - 3 šprinty	Filip Jurcacko	To Do	do 18.11. v AIS		other
Task	Úprava CodeReview metodiky	Irina Dymina	Done	do 19.11 do AIS - BIELIKOVÁ		other
Task	Upload dokumentov na stránku	Michal Fasánek	Done			other
Task	Zápisnička W7	Matus Cimerman	Done			other

Task	Vytvorit metodiku - ako programovat v C++	Matus Cimerman	Done	Vytvorit dokument, v ktorom sa opíše postup pri programovaní v jazyku cpp. Zaznamenajú sa základné princípy, ktorých sa budeme držať pri vývoji softvéru. * astyle, * cpplint, * cppcheck, * spojit s txt na drope.		other
Task	3Dconnexion Mouse - vytvorit core class / interface	Jaroslav Gazdik	In Progress	Vytvorit triedy na spracovanie a konvertovanie údajov získaných z myšky, napísať funkcie, ktoré pohyby v jednotlivých smeroch spracuje a poskytne projektu. Príklad Core triedy - Kinect. Vytvorit interface, cez ktoré bude komunikovať modul (myš) s existujúcim projektom 3DSoftViz.		implement
Task	3Dconnexion Mouse - vytvorit modul v projekte	Jaroslav Gazdik	Done	Pripojiť SDK k projektu - nájsť knižnice, v ktorých sú použité funkcie, začleniť do adresára projektu, nastaviť cesty v CMakeLists.		implement
Task	3Dconnexion Mouse - vytvorit diagramy	Irina Dyomina	In Progress	Vytvorit diagram s triedami znázorňujúce začlenenie modulu do projektu. Inšpirovať sa s modulom Kinect.		implement
Bug	Opraviť warningy - MSVC	Dalibor Meszaros	Done	Warningy detekované nástrojom MSVC		implement
Bug	Opraviť warningy - CppLint	Matus Cimerman	Done	Warningy detekované nástrojom cppLint		implement
Bug	Opraviť warningy - cppCheck	Dalibor Meszaros	Done	Warningy detekované nástrojom cppCheck		implement

Task	Vytvoriť dok. s retrospektívou	Dalibor Meszaros	Done	Zanalyzovať 2. šprint, prekonzultovať s tímom, vyzdvihnúť kladné/záporné stránky šprintu, navrhnúť vylepšenia, zahodiť nefungujúce spôsoby, spísať do dokumentu a odovzdať na upload.		other
Bug	Tlačidlo "Load function calls" po otvorení priečinka zhodí program	Denis Illes	Done	Pri otváraní zložky s .lua súbormi program crashne. Zrejme Windows related issue, keďže na iOS funguje bez akýchkoľvek problémov.  Riešenie: zakomentovať ...\\3dsoftviz\\resources\\scripts\\luadb\\extraction\\extractor.lua line 182 (assert) alebo zmazať druhú podmienku > assert(path, "wrong path passed")		implement
Bug	Tlačidlo "&Multi-select mode" vracia cez "getId" NULL, čo zhodí program	Filip Jurcacko	Ready to review	Pri označovaní viacerých uzlov program crashne.		implement
Task	Aktualizovať používateľskú príručku	Denis Illes	To Do			other



*C.c Úlohy z tretieho šprintu*

Issue Type	Summary	Assignee	Status	Description	Epic Name	Labels
Task	Zapisnica w8	Michal Fasaneck	Done	V prilohe export taskov z Jiry. w8-1_status - status uloh z minuleho tyzdna w8_todo-inprogress - zvsne ulohy na buduci tyzden		other
Task	Opravit error handling	Filip Jurcacko	Done	Na viacerych miestach v kode je do funkcie posielany argument bool* error ktory sa vnuri funkcie nastavu v pripade chyby  Avsak casto po vykonani funkcie bud nieje skontrolovany vobec alebo je viacnasoobne prepisany viacerymi volaniami funkcie nez je skontrolovany.  Fix it.		implement
Task	Gitflow metodiku upravit'	Jaroslav Gazdik	Done	Pill requesty a nic viac...		other
Task	Dokumentacia k produktu - riadenie	Filip Jurcacko	To Do	BIELIKOVÁ <a href="http://www2.fiit.stuba.sk/~bielik/courses/tp-slov/tp-main.html">http://www2.fiit.stuba.sk/~bielik/courses/tp-slov/tp-main.html</a>		other
Task	Dokumentacia k produktu - BigPicture	Filip Jurcacko	To Do	BIELIKOVÁ <a href="http://www2.fiit.stuba.sk/~bielik/courses/tp-slov/tp-main.html">http://www2.fiit.stuba.sk/~bielik/courses/tp-slov/tp-main.html</a>		other

Task	Dokumentácia k produktu - 3 šprinty	Filip Jurcacko	To Do	do 18.11. v AIS		other
Task	Úprava CodeReview metodiky	Irina Dyomina	Done	do 19.11 do AIS - BIELIKOVÁ		other
Task	Upload dokumentov na stránku	Michal Fasaneck	Done			other
Task	Nájsť všetky cudzie kódy	Irina Dyomina	Done	identifikovať všetky cudzie kódy, a nájsť odkiaľ boli zobrať napr: foldre: src > Noise (hlavička zobrazuje že kód je prevzatý) Viewer > Treeltem, TreeModel QOSG > qtcolorpicker OSGQTBrowser > WebViewImage Lua-Graph > LuaGprahTreeltem		implement
Task	Zápisnica W7	Matus Cimerman	Done			other
Bug	Opraviť warningy - OS X	Matus Cimerman	In Progress			implement
Task	Začleniť triedy do príslušných namespaceov	Denis Illes	Done	Triedy ako "Cube", "Sphere", "SpikySphere" sa nachádzajú mimo svojho namespaceu (Clustering). Treba ich začleniť. Rovnako ako "HistoryBuffer" (malo by to byť v kinect namespace)		implement
Task	Zmazať zbytočné triedy a vyskúšať program	Dalibor Meszaros	Done	V projekte sa nachádzajú triedy "ransac" a "vector3", ktoré sú pravdepodobne zbytočné a nepoužívajú sa. Treba ich zmazať a vyskúšať fungovanie projektu bez týchto tried.		implement

Task	Vytvorit metodiku - ako programovat v C++	Matus Cimerman	Done	Vytvorit dokument, v ktorom sa opíše postup pri programovaní v jazyku cpp. Zaznamenajú sa základné princípy, ktorých sa budeme držať pri vývoji softvéru. * astyle, * cpplint, * cppcheck, * spojit s txt na drope.		other
Task	3Dconnexion Mouse - vytvorit core class / interface	Jaroslav Gazdik	In Progress	Vytvorit triedy na spracovanie a konvertovanie údajov získaných z myšky, napísať funkcie, ktoré pohyby v jednotlivých smeroch spracuje a poskytne projektu. Príklad Core triedy - Kinect. Vytvorit interface, cez ktoré bude komunikovať modul (myš) s existujúcim projektom 3DSoftViz.		implement
Task	3Dconnexion Mouse - nama-povať na ovládanie virtuálnej kamery	Michal Fasaneck	In Progress	Poskytnuté údaje cez rozhranie použiť na ovládanie virtuálnej kamery. Ovládanie v oboch režimoch - voľný a orbitálny pohyb.		implement
Task	3Dconnexion Mouse - vytvorit modul v projekte	Jaroslav Gazdik	Done	Pripojiť SDK k projektu - nájsť knižnice, v ktorých sú použité funkcie, začleniť do adresára projektu, nastaviť cesty v CMakeLists.		implement
Task	3Dconnexion Mouse - vytvorit diagramy	Irina Dyomina	In Progress	Vytvorit diagram s triedami znázorňujúce začlenenie modulu do projektu. Inšpirovať sa s modulom Kinect.		implement

Bug	Opraviť warningy - MSVC	Dalibor Meszaros	Done	Warningy detekované nástrojom MSVC		implement
Bug	Opraviť warningy - CppLint	Matus Cimerman	Done	Warningy detekované nástrojom cppLint		implement
Bug	Opraviť warningy - cppCheck	Dalibor Meszaros	Done	Warningy detekované nástrojom cppCheck		implement
Task	Vytvoriť dok. s retrospektívou	Dalibor Meszaros	Done	Zanalyzovať 2. šprint, prekonzultovať s tímom, vyzdvihnúť kladné/záporné stránky šprintu, navrhnúť vylepšenia, zahodiť nefungujúce spôsoby, spísať do dokumentu a odovzdať na upload.		other
Task	linkovať sa na osgModelling	Peter Kapec	Ready to review	treba odstrániť z projektu subory v Math/- Bezier- Curve- Export - Model- Utilities a linkovať sa na osgModeling knižnicu, ktorú treba pridať do dependencies ako submodule		implement
Bug	Tlačidlo "Load function calls" po otvorení priečinka zhodí program	Denis Illes	Done	Pri otváraní zložky s .lua súbormi program crashne. Zrejme Windows related issue, keďže na iOS funguje bez akýchkoľvek problémov.  Riešenie: zakomentovať ...\\3dsoftviz\\resources\\scripts\\luadb\\extraction\\extractor.lua line 182 (assert) alebo zmazať druhú podmienku > assert(path, "wrong path passed")		implement
Bug	Tlačidlo "&Multi-select mode" vracia	Filip Jurcacko	Ready to review	Pri označovaní viacerých uzlov program crashne.		implement

	cez "getId" NULL, čo zhodí program					
Task	Aktualizovať použí- vateľskú príručku	Denis Illes	To Do			other

### C.d Úlohy zo štvrtého šprintu

Issue Type	Summary	Assignee	Status	Description	Epic Name	Labels
Task	Aktualizovat metodiku "Písanie zdrojoveho kodu v C++"	Matus Ci-merman	In Pro-gress	- do 11.12. pred odovzdanim finalnej doku-mentacie Rozsirt metodiku o poznatky z minulych rokov.		other
Task	Nahrat subory na stranku	Michal Fasanek	Done			other
Task	Prezentacia k riadeniu v TP na predmet MSI/MIS	<i>Unassigned</i>	To Do	Viac info na - <a href="http://www2.fiit.stuba.sk/~simko/manazment/rtp-intro-2015.pdf">http://www2.fiit.stuba.sk/~simko/manazment/rtp-intro-2015.pdf</a>		other
Task	Dokumentacia k riade-niu	<i>Unassigned</i>	In Pro-gress			other
Task	Retrospektiva 4. sprint	Matus Ci-merman	Done			other
Task	Zapisnica w11	Matus Ci-merman	Done			other
Task	Inžinierske dielo	<i>Unassigned</i>	In Pro-gress			other
Task	Upratať nesprávne umiestnené veci z 3dsodtviz	Denis Illes	Done	doc - doxygen resources .bat delete  logging ako modul <a href="https://github.com/LuaDist/lualogging">https://github.com/LuaDist/lualogging</a>		imple-ment

Task	Aruco example testy	<i>Unassigned</i>	To Do	<p>vysúšať 3dsoftviz\_install\bin\aruco_test_gl.exe a ostatné *_gl.exe</p> <p>arg. live camera.yml 0.025</p> <p>freeglut.dll prekopiovať z "OSG\Third-Party\VC10\x86\bin\" a premenovať na freeglutdll.dll</p> <p>camera.yml prekopiovať z "_install\share\3dsoftviz\config\"</p>		implement
Task	Refaktoring časti: Kinect, Leap, OpenCV	Filip Jurcacko	To Do	<p>inšpirácia - Garaj - gitlib</p> <p>spraviť ako osobitne externé libky da sa samostatne skompilovať, netreba mu 3dsoftviz pozret aj CMakeLists.txt</p> <p>GitGraphImporter a GitGraphUpdater su interface, používajú 3dsoftviz a lib</p>		implement
Task	OSG Jedno okno - rozťahnuté cez 3 monitory	Matus Cimerman	In Progress			implement
Task	Refaktoring - diluculum	Dalibor Meszaros	Ready to review	<p>diluculum použitý mimo namespace Lua (triedy LuaGraph, LuaInterface)</p> <p>riešenie: LuaGraph/Interface - pridať metódy</p>		implement

Task	3Dconnexion Mouse - vytvorit' core class / interface	Michal Fasanek	In Progress	Vytvorit' triedy na spracovanie a konvertovanie údajov získaných z myšky, napísať funkcie, ktoré pohyby v jednotlivých smeroch spracuje a poskytne projektu. Príklad Core triedy - Kinect. Vytvorit' interface, cez ktoré bude komunikovať modul (myš) s existujúcim projektom 3DSoftViz.		implement
Task	3Dconnexion Mouse - namapovať na ovládanie virtuálnej kamery	Michal Fasanek	In Progress	Poskytnuté údaje cez rozhranie použiť na ovládanie virtuálnej kamery. Ovládanie v oboch režimoch - voľný a orbitálny pohyb.		implement
Sub-task	AUGREALITY-18 Opraviť CameraManipulator.cpp	<i>Unassigned</i>	To Do			implement
Sub-task	AUGREALITY-18 Opraviť QtColorPicker.cpp	<i>Unassigned</i>	Done			implement
Sub-task	AUGREALITY-18 Opraviť NoiseUtils.cpp	<i>Unassigned</i>	Done			implement
Task	Presunúť "cudzí" kód do dependencies	Peter Kapec	To Do	Zdrojové súbory: NoiseUtils.cpp, QtColorPicker.cpp, CameraManipulator.cpp		implement



Task	linkovať sa na osgModelling	Peter Kapec	Done	<p>treba odstranit z projektu subory v Math/</p> <ul style="list-style-type: none"> <li>- Bezier</li> <li>- Curve</li> <li>- Export</li> <li>- Model</li> <li>- Utilities</li> </ul> <p>a linkovat sa na osgModeling kniznicu, ktoru treba pridat do dependencies ako submodule</p>		imple- ment
Bug	Tlačidlo "&Multi-select mode" vracia cez "getId" NULL, čo zhodí program	Filip Jurcacko	Done	Pri označovaní viacerých uzlov program crashne.		imple- ment

*C.e Úlohy z piateho šprintu*

Issue Type	Summary	Assignee	Status	Description	Epic Name	Labels
Task	Aktualizovat metodiku "Písanie zdrojového kodu v C++"	Matus Cimerman	Done	- do 11.12. pred odovzdaním finalnej dokumentácie Rozšíriť metodiku o poznatky z minulých rokov.		other
Task	Nahrat súbory na stránku	Michal Fasánek	Done			other
Task	Prezentácia k riadeniu v TP na predmet MSI/MIS	<i>Unassigned</i>	Done	Viac info na - <a href="http://www2.fiiit.stuba.sk/~simko/manazment/rtp-intro-2015.pdf">http://www2.fiiit.stuba.sk/~simko/manazment/rtp-intro-2015.pdf</a>		other
Task	Dokumentácia k riadeniu	<i>Unassigned</i>	Done			other
Task	Retrospektíva 4. sprint	Matus Cimerman	Done			other
Task	Zápisnica w11	Matus Cimerman	Done			other
Task	Nahradiť cudzie kódy - Example 1. časť	Irina Dyomina	In Progress	src/LuaGraph/LuaGraphTreeItem.cpp include/LuaGraph/LuaGraphTreeItem.h src/QOSG/TreeItem.cpp include/QOSG/TreeItem.h  Ponechať cudzie kódy nezmenené, správnym spôsobom prepojiť s naším projektom. Majú sa nachádzať v dependencies.		implement

Task	CAVE - analýza nástroja Equalizer	Denis Illes	In Progress	Preštudovať využitie, prostriedky, požiadavky. Zohnať programming guide.  <a href="http://www.equalizergraphics.com/index.html">http://www.equalizergraphics.com/index.html</a>		implement
Task	Refaktoring - OpenCV	Filip Jurcacko	To Do	inšpirácia - Garaj - gitlib  spraviť ako osobitne externé libky da sa samostatne skompilovať, netreba mu 3dsoftviz pozret aj CMakeLists.txt  GitGraphImporter a GitGraphUpdater su interface, používajú 3dsoftviz a lib		implement
Task	Refaktoring - Leap	Filip Jurcacko	Ready to review	inšpirácia - Garaj - gitlib  spraviť ako osobitne externé libky da sa samostatne skompilovať, netreba mu 3dsoftviz pozret aj CMakeLists.txt  GitGraphImporter a GitGraphUpdater su interface, používajú 3dsoftviz a lib		implement
Task	Refaktoring - Kinect	Filip Jurcacko	In Progress	inšpirácia - Garaj - gitlib  spraviť ako osobitne externé libky da sa samostatne skompilovať, netreba mu 3dsoftviz pozret aj CMakeLists.txt  GitGraphImporter a GitGraphUpdater su interface, používajú 3dsoftviz a lib		implement

Task	Inžinierske dielo	<i>Unassigned</i>	Done			other
Task	Upratať nesprávne umiestnené veci z 3dsodtviz	Denis Illes	Done	doc - doxygen resources .bat delete  logging ako modul <a href="https://github.com/LuaDist/lualogging">https://github.com/LuaDist/lualogging</a>		imple- ment
Task	Aruco example testy	<i>Unassigned</i>	To Do	vysúšať 3dsoftviz\_install\bin\aruco_test_gl.exe a ostatné *_gl.exe  arg. live camera.yml 0.025  freeglut.dll prekopiovať z "OSG\Third-Party\VC10\x86\bin\" a premenovať na freeglutdll.dll  camera.yml prekopiovať z "_install\share\3dsoftviz\config"		imple- ment
Task	OSG Jedno okno - rozťahnuté cez 3 monitory	Matus Cimerman	In Progress			imple- ment
Task	Refaktoring - diluculum	Dalibor Meszaros	Ready to review	diluculum použitý mimo namespace Lua (triedy LuaGraph, LuaInterface) riešenie: LuaGraph/Interface - pridať metódy		imple- ment

Task	3Dconnexion Mouse - vytvorit' core class / interface	Michal Fasanek	In Progress	Vytvorit' triedy na spracovanie a konvertovanie údajov získaných z myšky, napísať funkcie, ktoré pohyby v jednotlivých smeroch spracuje a poskytne projektu. Príklad Core triedy - Kinect. Vytvorit' interface, cez ktoré bude komunikovať modul (myš) s existujúcim projektom 3DSoftViz.		implement
Task	3Dconnexion Mouse - namapovať na ovládanie virtuálnej kamery	Michal Fasanek	In Progress	Poskytnuté údaje cez rozhranie použiť na ovládanie virtuálnej kamery. Ovládanie v oboch režimoch - voľný a orbitálny pohyb.		implement

# Príloha D Retrospektívy šprintov

## *D.a Po prvom šprinte*

### **Retrospektíva 1. Šprintu**

*Tím #4 - alphaReach*

---

#### **Čo sa nám podarilo počas šprintu?**

- Zlepšiť komunikáciu v tíme.

#### **Čo nešlo podľa našich predstáv?**

- Práca na úlohách.
- Pridelovanie úloh riešiteľom.
- Ohodnocovanie úloh story pointami.
- Automatizácia najzákladnejších úloh, napríklad zápisnice na webe ihneď po stretnutí.
- Viditeľnosť projektu – nedostatočná komunikácia s product ownerom o stave projektu.
- Na konci šprintu nebol dodaný fungujúci inkrement.
- Práca s Jirou – vykazovanie stavu v akom sú úlohy.
- Identifikované úlohy boli nedostatočne popísané a dekomponované.

#### **Čo môžeme zmeniť a zlepšiť do budúceho šprintu?**

- Automatizovať najzákladnejšie úlohy.
- Zlepšiť prácu s Jirou a tým aj vykazovanie stavu v akom je projekt a šprint.
- Špecifickejšie zapisovanie úloh do Jiry.
- Ešte zlepšiť komunikáciu v tíme a s product ownerom.

## *D.b Po druhom šprinte*

### **Retrospektíva 2. Šprintu**

*Tím #4 - alphaReach*

---

#### **Čo sa nám podarilo počas šprintu**

- Dokončili sme všetky dokumentačné úlohy a dokázali sme posunúť väčšinu implementačných úloh do revidáčnej fázy

#### **Čo nešlo podľa našich predstáv**

- Ani jedna implementačná úloha neprešla revidovaním kódu a nebola dokončená
- V tíme bola nedostatočná komunikácia medzi jednotlivými členmi, skupinou a vlastníkom produktu
- Nedostatočná komunikácia zapríčinila, že:
  - Nedohodli sme sa na tom, kto bude vykonávať revidovanie kódu, preto úlohy neprešli revidovaním do sekcie dokončených úloh
  - Neskoro sme sa začali pýtať vlastníka produktu a vedúceho na rady, ako postupovať pri riešení úloh

#### **Čo môžeme zmeniť a zlepšiť do budúceho šprintu**

- Uvedomili sme si, že by sme sa mali aktívne pýtať vlastníka produktu a vedúceho, keďkoľvek nevieme jednoznačne určiť, ako postupovať v riešení úloh
- Nebudeme si odkladať riešenie úloh na poslednú chvíľu, aby sa stihli zrevidovať

## *D.c Po tret'om šprinte*

### **Retrospektíva 3. Šprintu**

*Tím #4 - alphaReach*

---

#### **Čo sa nám podarilo počas šprintu?**

- Úspešne dokončiť takmer všetky implementačné úlohy a teda robiť prehliadky kódu ostatnými členmi tímu.
- Automatizovať rutinné procesy (zápisnice, retrospektíva).
- Zlepšili sme komunikáciu v tíme a tiež s vlastníkom produktu.
- Včas sme komunikovali nejasnosti k úlohám a stavu produktu s vlastníkom produktu.

#### **Čo nešlo podľa našich predstáv?**

- Procesy, ktoré sme až teraz začali používať aktívne:
  - Tvorba dokumentácie nebola zvládnutá z pohľadu riadenia a manažmentu.
- Burndown chart:
  - Skokové zmeny, ktoré implikujú, že sa nepracuje priebežne.
  - Nízka, resp. pomalá „rýchlosť“ (velocity) tímu.

#### **Čo môžeme zmeniť a zlepšiť do budúceho šprintu?**

- Zlepšiť proces tvorby dokumentácie:
  - Priebežná tvorba dokumentácie.
- Zefektívniť prácu na úlohách.



## *D.d Po štvrtom šprinte*

### **Retrospektíva 4. Šprintu**

*Tím #4 - alphaReach*

---

#### **Čo nešlo podľa našich predstáv?**

- Práca na úlohách:
  - Väčšina z úloh nebolo možné dokončiť v tomto šprinte kvôli ich veľkosti

#### **Čo môžeme zmeniť a zlepšiť do budúceho šprintu?**

- Dekomponovať úlohy na maximálne 1-3 SP.
- Spájanie zdrojových kódov na konci šprintu do príslušných vetiev, ktoré ošetruje metódika verziovania kódu.

*D.e Po piatom šprinte*

## **Retrospektíva 5. Šprintu**

*Tím #4 - alphaReach*

---

Posledný a zároveň najkratší šprint (týždeň) išiel podľa našich predstáv, aj napriek tomu, že sme nestihli dokončiť všetky úlohy. Bolo to z dôvodu nedostatku času (kratší šprint ako obvykle) a dokončovania dokumentácie a príprav na deň otvorených dverí na FIIT. Zhodli sme sa, že budeme pracovať na projekte priebežne aj počas vianočných sviatkov a skúškového obdobia.

**SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE**

Fakulta informatiky a informačných technológií

**VIZUALIZÁCIA INFORMÁCIÍ V OBOHATENEJ REALITE**

Dokumentácia k inžinierskemu dielu

Členovia tímu č.4:

BC. MATÚŠ CIMMERMAN

BC. IRINA DYOMINA

BC. MICHAL FAŠÁNEK

BC. JAROSLAV GAZDÍK

BC. DENIS ILLÉS

BC. FILIP JURČACKO

BC. DALIBOR MÉSZÁROS

Predmet:

Tímový projekt I

Vedúci:

Ing. Peter Kapec, PhD.

Akademický rok:

2015/2016

# Obsah

Úvod .....	3
<b>1 Globálne ciele .....</b>	<b>4</b>
1.1 Globálne ciele pre zimný semester .....	4
<b>2 Celkový pohľad na systém .....</b>	<b>6</b>
2.1 Architektúra .....	6
2.2 Dátový model .....	6
2.3 Diagram tried .....	8
<b>3 Prehľad modulov .....</b>	<b>9</b>
3.1 Implementované moduly .....	9
3.2 Plánované moduly .....	10
<b>4 Moduly systému .....</b>	<b>11</b>
4.1 Mouse3d .....	11
4.1.1 Analýza modulu Mouse3d .....	11
4.1.2 Návrh modulu Mouse3d .....	11
4.1.3 Implementácia .....	12
4.2 Equalizer .....	13
4.2.1 Analýza .....	13
<b>5 Inštalčná príručka .....</b>	<b>14</b>
5.1 Návod na Windows .....	14
5.2 Rozšírenie 3DSoftviz o Kinect .....	21
5.3 Rozšírenie 3DSoftviz o 3D myš .....	23
5.4 Návod na nastavenie debuggera v QtCreator .....	23
5.5 Časté problémy .....	24

# Úvod

Inžinierske dielo slúži ako technická dokumentácia k projektu Tímový projekt na fakulte informatika a informačných technológií Slovenskej technickej univerzity v Bratislave. Opisuje štruktúru a postup rozširovania projektu na tému Vizualizácia informácií v obohatenej realite.

V prvej kapitole sa opisujú globálne ciele pre zimný semester, ktoré sa budeme za nasledovné mesiace snažiť dosiahnuť. Okrem zakomponovania nových zariadení sme medzi ciele zaradili aj údržbu a opravy chýb v projekte. Na projekte robilo súčasne viacero študentov a pri spájaní došlo k menším nedostatkom, ktoré spôsobili nefunkčnosť istých funkcií. Na úvod sme si zaumienili riešiť tieto nedostatky.

V druhej kapitole je umiestnený aj celkový pohľad na súčasný systém, z ktorého vychádzame.

V tretej kapitole sú zverejnené zoznamy modulov implementovaných a plánovaných. K modulom prislúcha aj krátky stručný opis.

V štvrtej kapitole sa nachádzajú opisy modulov, ktoré implementujeme do systému. Podrobný priebeh analýzy, návrhu, implementácie a testovania.

V piatej kapitole je podrobný inštalačný manuál. Základ manuálu bol prebratý z predošlých rokov, avšak postupným aktualizovaním a rozširovaním projektu musel byť manuál tiež upravený. Inštalačný návod bol preformátovaný, kvôli prehľadnosti a rozšírené a viacero krokov, ktoré sme spozorovali pri inštalovaní softvéru jednotlivými členmi. Návod je doplnený o obrázkové postupy pre ľahšiu orientáciu.

# 1 Globálne ciele

## 1.1 Globálne ciele pre zimný semester

V prvom semestri sa budeme zaoberať analýzou problémovej oblasti. Jednotliví členovia sa musia zoznámiť s problematikou vizualizácie informácií. Na predmete tímový projekt sa bude vylepšovať a rozširovať existujúci rozsiahli projekt, ktorý má implementovanú funkcionálnosť niekoľko hardvérových zariadení. V projekte je pre každé zariadenie vytvorený modul. Naším cieľom bude zozname modulov rozšíriť.

Zimný semester máme na pláne prevažne úpravu zdrojového kódu. Z dôvodu spájania bakalárskych a diplomových prác došlo k výskytu závažných chýb a nedostatkov. Našou prioritou je dostať projekt do stabilného stavu, aby sme mohli neskôr pracovať na implementovaní nových modulov do funkčného systému. Pre náš projekt je dôležité pridanie hardvérových zariadení na efektívnejšie ovládanie grafov vo virtuálnom svete – 3D myš. Hlavnou výzvou je projekcia grafov na okuliare určené pre zobrazenie rozšírenej reality (AR okuliare).

Po odladení projektu sa začne pracovať na implementácii 3D myšky. Špeciálna myš poskytuje používateľovi 6 úrovní voľnosti (pohybom od seba/k sebe simuluje zoom, pohyb do strán znamená posuv doľava/doprava, stlačenie alebo vytiahnutie spôsobuje posuv hore/dole a otáčanie a naklápanie spôsobuje rotáciu). Ovládanie nahradí potrebu náročnej manipulácie s grafmi prostredníctvom klávesnice a klasickej myšky.

Medzi ďalšie dostupné zariadenia, ktoré máme na pláne implementovať, sú stereoskopické okuliare. Použitím okuliarov a vhodného 3D monitora budeme schopní vnímať grafy, ktoré boli načítané zo súborov alebo vytvorené analýzou volaných funkcií skriptovacieho jazyka Lua ako trojdimenzionálne objekty.

Okrem pripájania nových komponentov máme na pláne rozšíriť aj prezentačnú časť projektu. Prvotnou výzvou je zobrazenie softvéru klasicky na viacerých monitoroch v režime „extended“. Následne vyriešiť paralelné spracovanie dát pre individuálne scény a tak vytvoriť „virtuálnu jaskyňu“ (angl. CAVE).

Najnáročnejšou úlohou bude správne nasadiť funkcionality AR okuliarov. Naša vízia je zobrazenie grafu na objekt, ktorý drží používateľ v ruke a je označený špeciálnou značkou. Otáčaním objektu sa má otáčať aj samotný graf. Špeciálna značka sa nasníma pomocou modulu Aruco, ktorý je už súčasťou projektu a plne funkčný.

Po úspešnom aplikovaní predošlých modulov sa budeme zaoberať ďalšej problematike vizualizácie štruktúry softvéru. Podobne ako pri zobrazovaní grafu volaní, ktorý čerpá dáta zo zdrojových kódov jazyka Lua, sa môže vizualizovať aj štruktúra softvéru na základe modulov. Cieľom je zobraziť jednotlivé moduly ako grafické objekty, ktoré sú vhodným spôsobom prepojené. Uvedomujúc si, že jednotlivé moduly môžu v sebe zahŕňať ďalšie „podmoduly“, treba navrhnuť vnorené objekty, ktoré majú jednoznačne odzrkadľovať štruktúru softvéru.

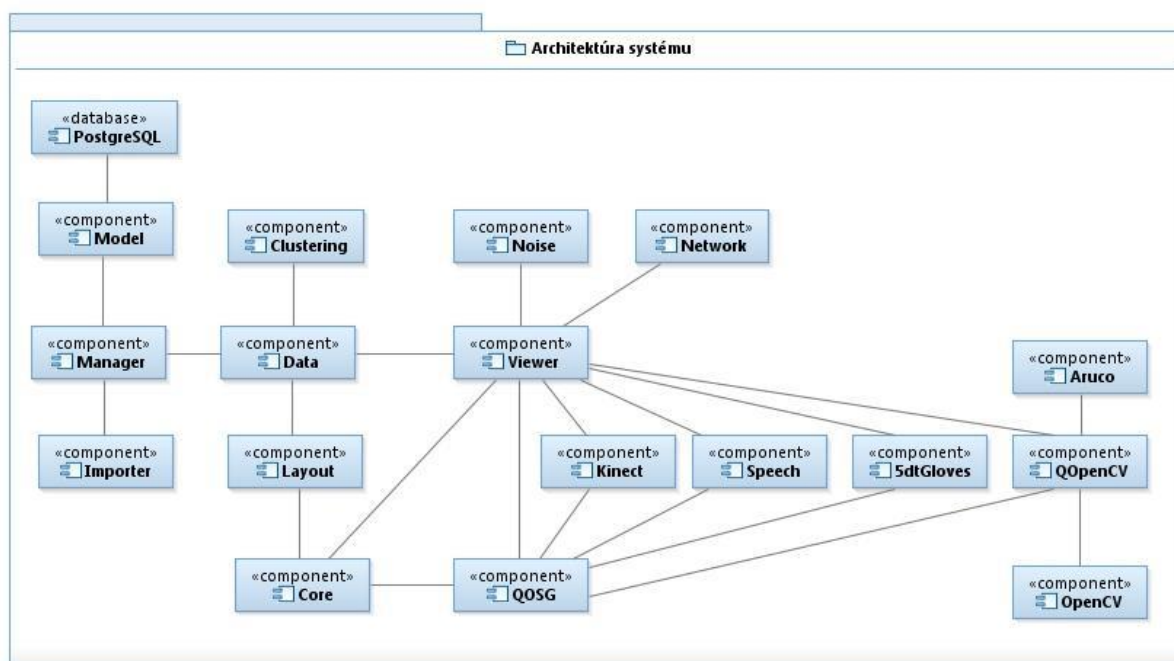
Vyriešením predošlej problematiky sa môže vyskúšať prepojenie spomenutých geografických objektov a grafom volaní, ktorý sa môže vytvoriť z podobných modulov. Výsledkom má byť komplexná grafová štruktúra znázorňujúca jednotlivé moduly, graf volaní v moduloch a prepojenia medzi modulmi. Komplikované abstraktné dátové štruktúry používané v komplexných heterogénnych architektúrach zvyčajne nie sú prehľadné a so zvyšovaním komplexnosti sa prehľad iba zhoršuje. Touto metódou by bolo možné zobraziť vnútornú štruktúru softvéru a odhaliť relevantné dáta, prípadne nekorektné operácie s nimi.

## 2 Celkový pohľad na systém

V tejto kapitole sa nachádza celkový pohľad na systém z hľadiska architektúry, dátového modelu, diagramu tried a jednotlivých modulov projektu.

### 2.1 Architektúra

Na Obrázku 1 je zobrazená štruktúra projektu, ktorá bola štartovacím bodom pre náš projekt. Pracujeme na rozšírení projektu o nové moduly ako je 3D myš, pomocou ktorej sa zefektívni ovládanie programu, teda manipulácia s grafmi vo virtuálnom priestore. Následne sa pracovalo na úpravách existujúcich komponentoch, aby sme pripravili softvér na implementáciu nových častí. Vyriešili sme chyby, ktoré nedovoľovali využívanie celého potenciálu softvéru 3DSoftViz.

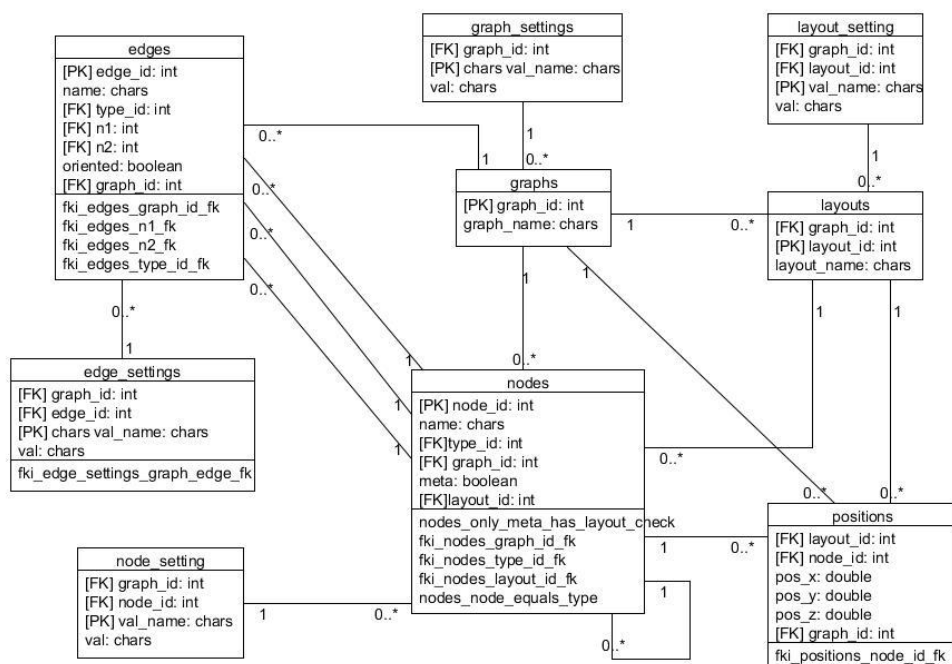


Obrázok 1 - architektúra softvéru

### 2.2 Dátový model

Dátový model databázy sme prebrali od predchádzajúcich prác, zobrazený je na Obrázku 2.



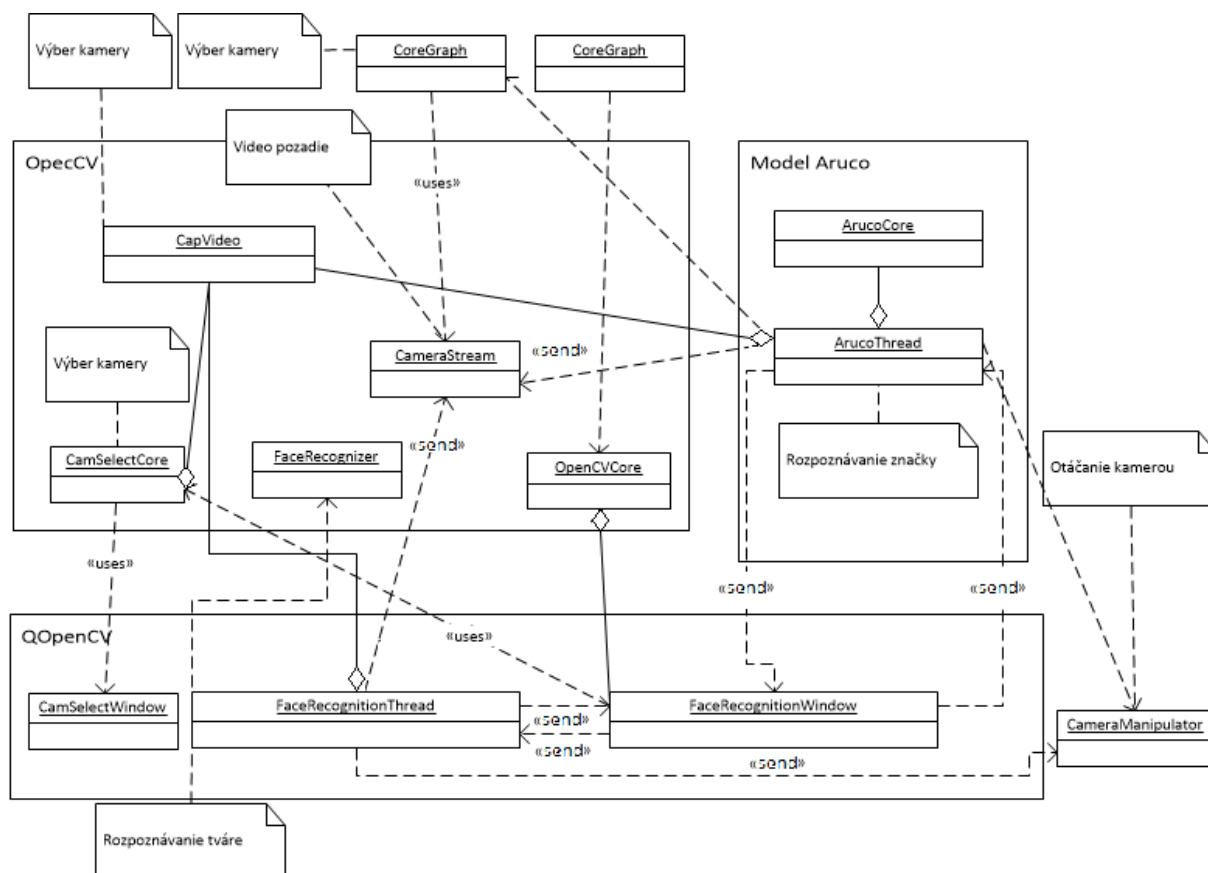


Obrázok 2 - dátový model

Stručný opis vybraných entít:

- **Graphs** – predstavuje jednotlivé záznamy grafov
- **Layouts**– obsahuje layout pre daný graf. Layout predstavuje konkrétne rozloženie množiny uzlov a hrán grafu v priestore.
- **Nodes**– obsahuje uzly a typy v grafe. Graf je vytvorený z množiny uzlov poprepájaných hranami, pričom každý uzol má svoj typ.
- **Positions** – obsahuje súradnice uzlov v priestore, pričom pozostáva z troch súradníc, ktoré sú naviazané na konkrétny uzol a layout.
- **Edges** – má uložené hrany spájajúce uzly v grafe. Hrana môže byť orientovaná neorientovaná a má začiatkový a koncový uzol a tiež typ.

## 2.3 Diagram tried



Obrázok 3 - diagram tried

Diagram tried je rozdelený do troch veľkých celkov:

- OpenCV - hlavný celok, ktorý obsahuje základné triedy (core) pre celú funkčnosť projektu
- Model Aruco - triedy obsahujúce funkčnosť pre rozpoznávanie značiek, bežia vo vlastnom vlákne
- QOpenCv - obsahuje triedy pre vlákno na rozoznávanie tváre, spolu so samostatným vykresľovaním

## 3 Prehľad modulov

### 3.1 Implementované moduly

**Core** – obsahuje jadro systému, inicializuje základné časti systému.

**Data** - dátový modul pre opis štruktúry grafu, obsahujúci triedy reprezentujúce jednotlivé prvky grafu (graph, node, edge, type, layout, ...).

**Importer** - modul pre parsovanie vstupných súborov vo formátoch GraphML, RSF a GXL.

**Layout** – modul, ktorý má na starosti rozmiestňovanie uzlov v 3D priestore, taktiež obsahuje implementácie layout algoritmu a triedy pre pridávanie ohraničení rozmiestnenia.

**Manager** - modul pre prácu s grafom.

**Math** - model pre rozšírenie práce s kamerou.

**Model** – modul pre komunikáciu systému s databázou. Funkcionalitou je mapovanie objektov do databázy, vytvorenie spojenia s databázou a základne funkcie výberu a uloženia grafu. Taktiež umožňuje uloženie uzlov aj s ich atribútmi a viacero rozmiestnení pre 1 graf.

**Network** - modul pre podporu kolaboratívnej práce nad grafom. Poskytuje klient/server funkcionality.

**Noise** - modul pre vytvorenie generovaného 3D priestoru pre pozadie.

**OsgBrowser** - modul zahŕňa viazanie udalostí pre jednotlivé klávesy a akcie myši medzi rozhraniami Qt a OpenSceneGraph a vizualizáciu načítaných grafov.

**QOSG** – modul pre prácu s grafickými prvkami softvéru. Má na starosti vytvorenie hlavného okna a prácu s pomocnými oknami a widgetami.

**Util** - zabezpečuje konfiguráciu nastavení aplikácie a funkcie pre vyčistenie pamäte.

**Viewer** - modul zabezpečuje pohyb v 3D priestore a prácu s kamerou. V module sa tiež pripravuje graf a jeho pre zobrazenie a vytvorenie 3D kocky pre pozadie.

**Kinect** – modul pre komunikáciu a ovládanie zariadenia Kinect. Medzi jeho funkcionality patrí získavanie informácií a ich nasledovné spracovanie. Obsahuje detegovanie gest, ktoré nahrádzajú ovládanie myšou, otáčanie a pohyb grafu a gestá pre ďalšie ovládanie.

**Speech** - implementuje funkcionality rozpoznávania hlasu.

**OpenCV** - zabezpečuje rozpoznávanie tváre na obraze z kamery a poskytuje funkcionality pre správu kamier.

**QOpenCV** – obsahuje okno pre ovládanie rozpoznávania tváre, značky a ovládanie video pozadia

**Aruco** – obsahuje funkcionality, ktorá vie rozpoznávať značky z kamery použitím knižnice Aruco.

**5DTGloves** – zabezpečuje detegovanie gesta ruky a vykonávanie korešpondujúcich akcií.

**Leap senzor** – deteguje dve ruky používateľa v 3D priestore a sleduje pohyby rúk až na úrovni článkov prstov.

### 3.2 *Plánované moduly*

3D myš (SpacePilot Pro, SpaceNavigator for Notebooks od spoločnosti 3DConnexion)

Equalizer (CAVE, stereoskopické zobrazenie)

Stereoskopické okuliare (PNY 3D Vision PRO od spoločnosti Nvidia)

AR okuliare (STAR 1200XLD od spoločnosti VUZIX)

V momentálnom stave sú do nášho projektu pridané knižnice 3D myšky a naprogramované základné triedy, ktoré budú poskytovať funkcionality pre 3D myš. Vo veľkej miere naše úlohy doteraz spočívali v úpravách zdrojového kódu, opravách warningov a chýb, s ktorými sme sa úspešne popasovali.

## 4 Moduly systému

### 4.1 *Mouse3d*

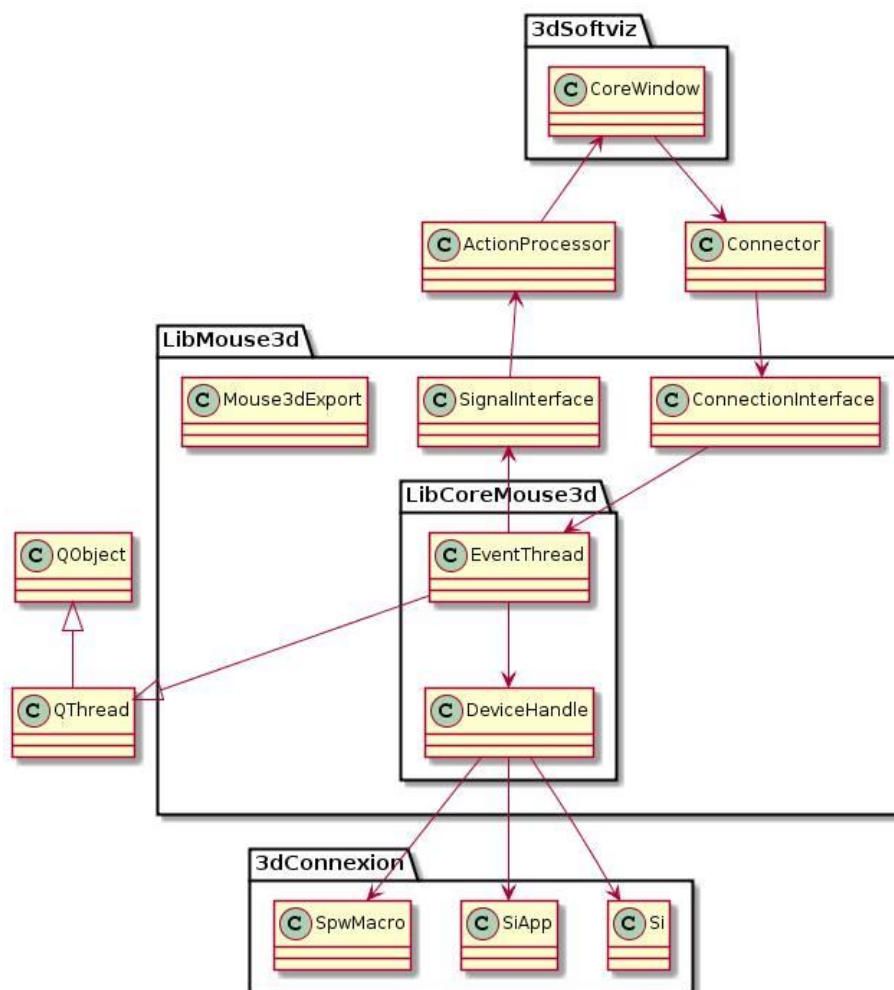
K dispozícii máme dva modely počítačových 3D myší. Kompaktný Space Navigator for notebooks je pre svoje malé rozmery ľahko prenosný. Na druhej strane, Space Pilot Pro je mohutnejšia myš obohatená o niekoľko tlačidiel navyše. Pre obe zariadenia slúži rovnaké SDK.

#### 4.1.1 *Analýza modulu Mouse3d*

Mouse3d je samostatný implementačný modul v projekte 3dSoftviz zodpovedný za pripojenie funkcionality 3d myši. Pri využívaní priestorových údajov je podpora efektívnej manipulácie s 3d modelmi nevyhnutná pre projekt. Modul tvorí rozhranie pre pripojenie myši nezávislé na konkrétnom produkte. Jeho účelom je zachytávanie a spracúvanie signálov z 3d myši a ich interpretácia pre 3dSoftviz. Spolu s modulom je vytváraná knižnica pre prácu so SpaceNavigator for Notebooks a Space Pilot Pro myšou od spoločnosti 3dConnexion. Knižnica LibMouse3d slúži ako prostriedok na komunikáciu modulu 3dMouse s funkcionalitou SpaceNavigator. V rámci vývoja nám spoločnosť na požiadavku poskytla SDK pre tento hardvér.

#### 4.1.2 *Návrh modulu Mouse3d*

Pri návrhu sme sa inšpirovali predošlými funkčnými modelmi. Identifikovali sme skupinu tried Mouse3d, ktoré budú zabezpečovať rozhranie medzi projektom 3dSoftviz a súborom 3d myši. Toto je nevyhnutné kvôli tomu, aby mohla byť zo súborov 3d myši vytvorená samostatná knižnica. Skupina tried v balíku Mouse3d slúži na spracovanie signálu získaného zo špeciálnej myšky do použiteľnej podoby pre náš projekt. LibMouse3d obsahuje rozhrania, ktoré oddeľujú aplikačnú logiku myšky od samotného softvéru. V balíku sú umiestnené aj triedy slúžiace na namapovanie ovládania pomocou 3D myšky – LibCoreMouse3d. Usporiadanie takýmto spôsobom je prospešné pre projekt z hľadiska modularity. Takto sa vytvorí modul Mouse3d, ktorý môže byť zaradený medzi ostatné. Z adresára LibMouse3d sa v projekte vytvára samostatne spustiteľná knižnica.



Obrázok 4 - triedy modulu 3dMouse

### 4.1.3 Implementácia

Implementovali sme moduly systému z diagramu na obrázku 4 a integrovali ich do projektu. Moduly Connector a ConnectionInterface prepájajú jadro aplikácie 3dSoftviz s modulom 3d myši. Modul DeviceHandle sa stará o správnu inicializáciu 3d myši. Modul EventThread obsahuje hlavnú slučku odchyťovania signálov z 3d myši. Modul SignalInterface filtruje signály z 3d myši a odosiela ich ActionProcessoru. Modul ActionProcessor mapuje pohyb kamery v aplikácií na pohyb 3d myši.

Pri implementácií sme narazili na nedostatok vo vývojárskom balíčku produktu, ktorý znemožňoval správnu detekciu produktu. Kvôli tomu v súčasnosti vývojársky tím

komunikuje s podporným tímom 3D myši. Napriek tomu sa nám podarilo napísať a vytvoriť všetky moduly systému okrem modulu SignalInterface, keďže s nesprávne detegovanou 3d myšou nebolo možné zachytávať jej signály. Všetky ostatné moduly boli napísané a obsahujú základnú funkcionality. Architektúra je však navrhnutá tak, aby bolo jednoduché ich rozšíriť a zmeniť.

## 4.2 *Equalizer*

Štandardný middleware na vytváranie a nasadenie aplikácií založených na OpenGL. Utlizuje grafickú procesorovú silu v multi GPU systémoch, centrálnu procesorovú silu v multi CPU systémoch a taktiež umožňuje škálovanie renderovania na viacerých počítačoch. Na ňom je postavená svetovo najvyspelejší vizualizačný systém – CAVE s ultra-high rozšírením, ktorú poháňajú 24 4K projektory.

### 4.2.1 *Analýza*

V našom projekte Equalizer posluži ako nástroj na vytvorenie 3 až 5 stennej CAVE. Na základe poskytnutých názorných ukážok zistíme správny spôsob implementácie. Pre spustenie nášho softvéru pomocou Equalizera potrebujeme vytvoriť konfiguračný súbor, v ktorom sa jasne zadefinujú nastavenia renderovania (počet monitorov/projektorov, mód – stereoskopický,...). Jeho úlohou je zobrazenie scény na príslušný monitor pre každú kameru vytvorenú vo virtuálnom priestore.

## 5 Inštalčná príručka

### 5.1 Návod na Windows

Tento návod bol úspešne otestovaný na operačnom systéme Windows 8.1 a Windows 10. Na inštaláciu potrebujeme klonovať projekt RealityNotFound<sup>1</sup> (tím č.4 klónuje 3dsoftviz<sup>2</sup>) z Githubu a stiahnuť potrebný softvér:

- CMake<sup>3</sup> (v3.3.2)
- OpenSceneGraph<sup>4</sup> (v3.4) – iba zdrojáky > treba buildnúť (cca 40-50min)
  - OpenSceneGraph<sup>5</sup> (v3.4) – buildnuté (17.10.2015)
- Kinect for Windows SDK<sup>6</sup>
- Microsoft VisualStudio 2010 SP1<sup>7</sup> (okrem Express edition)
- Qt<sup>8</sup> (v4.8.5)
  - QtCreator<sup>9</sup> (v3.2.1)
- OpenCV<sup>10</sup> (v2.4.10)
- Boost<sup>11</sup> (v1.57.0)
- RapidEE<sup>12</sup> - program na prácu s premennými
- Inštaláciu knižnice 3rd party dependencies<sup>13</sup>
- OpenNI2<sup>14</sup> a NiTE2 (nachádzajú sa v *OpenNI-Nitte.rar*)
- Debugging Tools for Windows (WinDbg) – Win8.1<sup>15</sup>, Win10<sup>16</sup>
- FreeGlut<sup>17</sup>

Postup inštalácie:

1. Nainštalovať CMake. (Cesta je v dokumente označená ako %CMAKE\_DIR%)
2. Nainštalovať Qt (%QT\_DIR%)
3. Nainštalovať QtCreator do zložky Qt

<sup>1</sup> Zdroj: <https://github.com/dominikhorniak/RealityNotFound>

<sup>2</sup> Zdroj: <https://github.com/cimox/3dsoftviz>

<sup>3</sup> Zdroj: <https://cmake.org/files/v3.3/cmake-3.3.2-win32-x86.exe>

<sup>4</sup> Zdroj: [http://trac.openscenegraph.org/downloads/developer\\_releases/OpenSceneGraph-3.4.0.zip](http://trac.openscenegraph.org/downloads/developer_releases/OpenSceneGraph-3.4.0.zip)

<sup>5</sup> Zdroj: [https://mega.nz/#!cIpy2TZQ!4k29x33yVvQDJRXT8B1J\\_3F-L1t30B3A2Z-uoQF9OeU](https://mega.nz/#!cIpy2TZQ!4k29x33yVvQDJRXT8B1J_3F-L1t30B3A2Z-uoQF9OeU)

<sup>6</sup> Zdroj: <http://www.microsoft.com/en-us/download/details.aspx?id=40278>

<sup>7</sup> Zdroj: <http://www.microsoft.com/en-us/download/details.aspx?id=23691>

<sup>8</sup> Zdroj: <http://ftp.fau.de/qtproject/archive/qt/4.8/4.8.5/qt-win-opensource-4.8.5-vs2010.exe>

<sup>9</sup> Zdroj: [http://ftp.fau.de/qtproject/official\\_releases/qtcreator/3.2/3.2.1/qt-creator-opensource-windows-x86-3.2.1.exe](http://ftp.fau.de/qtproject/official_releases/qtcreator/3.2/3.2.1/qt-creator-opensource-windows-x86-3.2.1.exe)

<sup>10</sup> Zdroj: <http://sourceforge.net/projects/opencvlibrary/files/opencv-win/2.4.10/opencv-2.4.10.exe/download>

<sup>11</sup> Zdroj: [http://sourceforge.net/projects/boost/files/boost/1.57.0/boost\\_1\\_57\\_0.zip/download](http://sourceforge.net/projects/boost/files/boost/1.57.0/boost_1_57_0.zip/download)

<sup>12</sup> Zdroj: [http://www.rapidee.com/download/RapidEE\\_setup.exe](http://www.rapidee.com/download/RapidEE_setup.exe)

<sup>13</sup> Zdroj: <http://ulozto.sk/xaZJArtg/vc10-zip>

<sup>14</sup> Zdroj: <http://ulozto.to/xpz9zqAi/openni-nitte-rar>

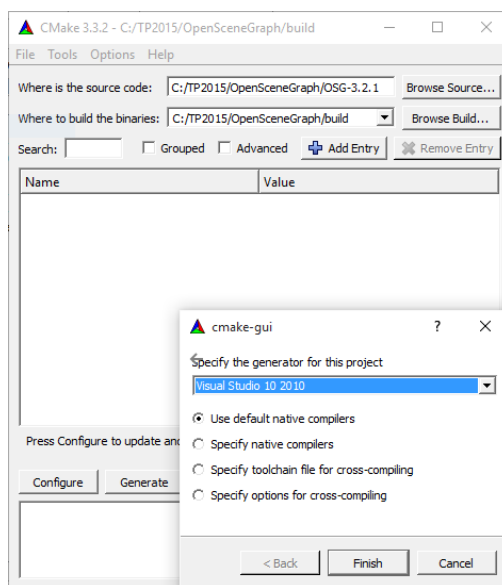
<sup>15</sup> Zdroj: <http://download.microsoft.com/download/B/0/C/B0C80BA3-8AD6-4958-810B-6882485230B5/standalonesdk/sdksetup.exe>

<sup>16</sup> Zdroj: <http://download.microsoft.com/download/E/1/F/E1F1E61E-F3C6-4420-A916-FB7C47FBC89E/standalonesdk/sdksetup.exe>

<sup>17</sup> Zdroj: [https://mega.nz/#!VYAVBKwb!l-SFHJnPq3JcbRG7nzkmwv2ov3SxHZia\\_o16dFx\\_duA](https://mega.nz/#!VYAVBKwb!l-SFHJnPq3JcbRG7nzkmwv2ov3SxHZia_o16dFx_duA)



4. Vytvoriť zložku OpenSceneGraph (%OSG\_DIR%) a OpenSceneGraph\ThirdParty
5. Rozbaliť 3rd Party knižnice (VC10) do %OSG\_DIR%\ThirdParty\
6. V prípade stiahnutia zbuildovaných súborov OSG (mega.nz)
  - a. Rozbaliť zložky *build* a *install* do %OSG\_DIR%
  - b. Pokračovať krokom číslo 8.
7. V prípade stiahnutia iba zdrojových súborov OSG (oficiálna stránka)
  - a. Rozbaliť OSG\_3.4 do %OSG\_DIR%
  - b. Vytvoriť zložku *build* a *install* v %OSG\_DIR%
  - c. Premenovať súbory:
    - %OSG\_DIR%\ThirdParty\VC10\x86\include\GL\glut.h > glut.h.bak
    - %OSG\_DIR%\ThirdParty\VC10\x86\lib\glut32.lib > glut32.lib.bak
    - %OSG\_DIR%\ThirdParty\VC10\x86\lib\glut32D.lib > glut32D.lib.bak
  - d. Spustiť CMake (*cmake-gui.exe*)
    - source code > %OSG\_DIR%\OSG\_3.4
    - binaries > %OSG\_DIR%\build
    - stlačiť Configure (VS2010 kompilátor)
    - stlačiť Generate
      - ak došlo k erroru: File > Delete Cache a skúsiť znovu

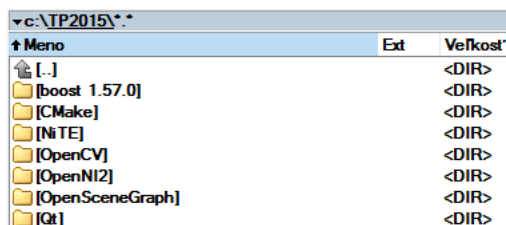


Obrázok 5 - CMake pre OSG

- e. Nájst' súbor *OpenSceneGraph.sln* v %OSG\_DIR%\build
- f. Otvoriť súbor vo VS2010
- g. Nastaviť Solution Configuration na *Debug*
- h. Nájst' projekt ALL\_BUILD > pravý klik > build
- i. Po skončení nájsť projekt INSTALL > pravý klik > build
- j. Nastaviť Solution Configuration na *Release*
- k. Nájst' projekt ALL\_BUILD > pravý klik > build
- l. Po skončení nájsť projekt INSTALL > pravý klik > build

8. Nainštalovať OpenCV (%OPENCV\_DIR%)
9. Rozbaliť Boost (%BOOST\_DIR%)

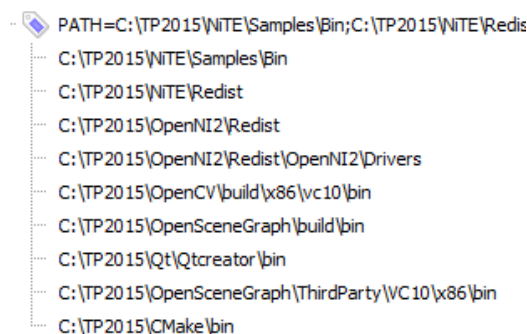
Ideálne je mať všetko na spoločnom mieste kvôli prehľadnosti, napr.



Obrázok 6 - Nainštalovaný SW

10. Nainštalovať a otvoriť RapidEE, v ktorom sa vykonajú tieto zmeny:

- a. do PATH pridať premenné:
  - %CMAKE\_DIR%\bin
  - %QT\_DIR%\bin
  - %QT\_DIR%\Qtcreator\bin
  - %OSG\_DIR%\build\bin
  - %OSG\_DIR%\ThirdParty\VC10\x86\bin
  - %OPENCV\_DIR%\build\x86\vc10\bin



Obrázok 7 - PATH premenná

\*pozn. premenné NiTE a OpenNI sa nastavujú až v ďalšej kapitole

- b. Vytvoriť premennú CMAKE\_INCLUDE\_PATH a pridať:
  - %OSG\_DIR%\install\include
  - %OSG\_DIR%\ThirdParty\VC10\x86\include
  - %OPENCV\_DIR%\build\include

```

CMAKE_INCLUDE_PATH=C:\TP2015\NITE\include;C:\TP2015
C:\TP2015\NITE\include
C:\TP2015\OpenNI2\include
C:\TP2015\OpenCV\build\include
C:\TP2015\OpenSceneGraph\ThirdParty\VC10\x86\include
C:\TP2015\OpenSceneGraph\install\include

```

Obrázok 8 - CMAKE\_INCLUDE\_PATH premenná

\*pozn. premenné NiTE a OpenNI sa nastavujú až v ďalšej kapitole

- c. Vytvoriť premennú CMAKE\_LIBRARY\_PATH a pridať:
- %OSG\_DIR%\build\lib
  - %OSG\_DIR%\install\lib
  - %OSG\_DIR%\ThirdParty\VC10\x86\lib
  - %OPENCV\_DIR%\build\x86\vc10\lib

```

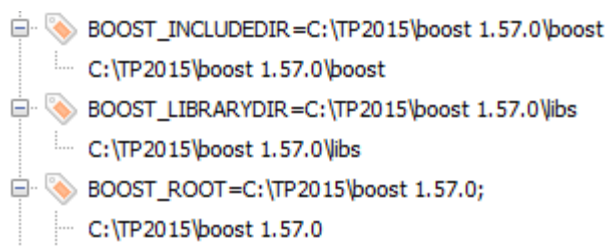
CMAKE_LIBRARY_PATH=C:\TP2015\OpenSceneGraph\in
C:\TP2015\OpenSceneGraph\install\lib
C:\TP2015\NITE\Lib
C:\TP2015\NITE
C:\TP2015\OpenNI2\Lib
C:\TP2015\OpenNI2\Redist\OpenNI2\Drivers
C:\TP2015\OpenNI2\Redist
C:\TP2015\OpenNI2\Driver
C:\TP2015\OpenCV\build\x86\vc10\lib
C:\TP2015\OpenSceneGraph\build\lib
C:\TP2015\OpenSceneGraph\ThirdParty\VC10\x86\lib

```

Obrázok 9 - CMAKE\_LIBRARY\_PATH premenná

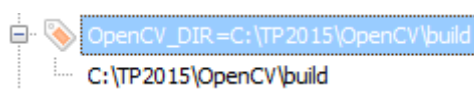
\*pozn. premenné NiTE a OpenNI sa nastavujú až v ďalšej kapitole

- d. Vytvoriť premennú BOOST\_INCLUDEDIR a pridať:
- %BOOST\_DIR%\boost
- e. Vytvoriť premennú BOOST\_LIBRARYDIR a pridať:
- %BOOST\_DIR%\libs
- f. Vytvoriť premennú BOOST\_ROOT a pridať:
- %BOOST\_DIR%



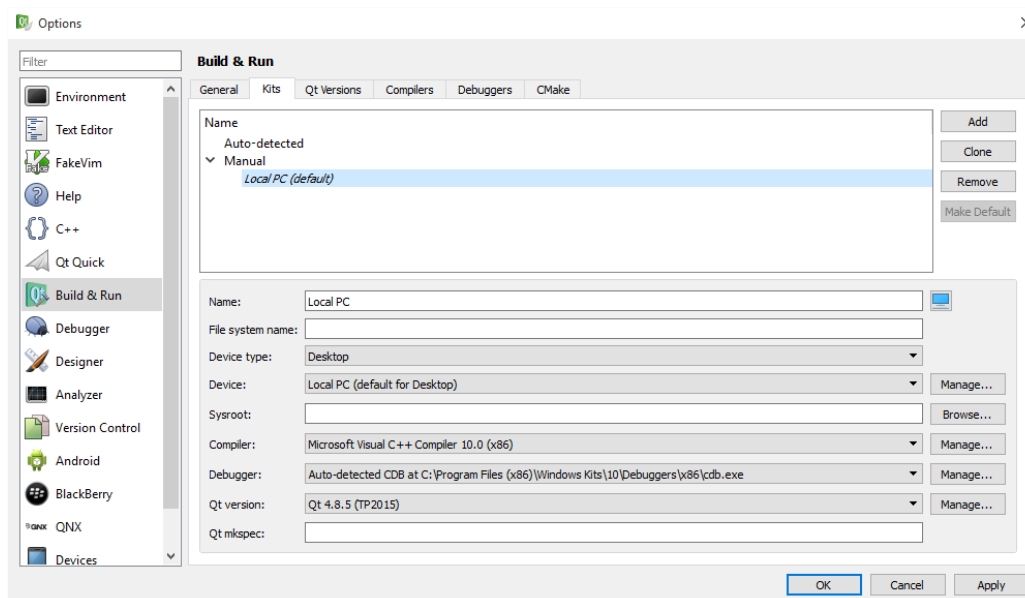
Obrázok 10 - BOOST premenne

- g. Vytvoriť premennú OPENCV\_DIR a pridať:
- %OPENCV\_DIR%\build



Obrázok 11 - OPENCV\_DIR premenná

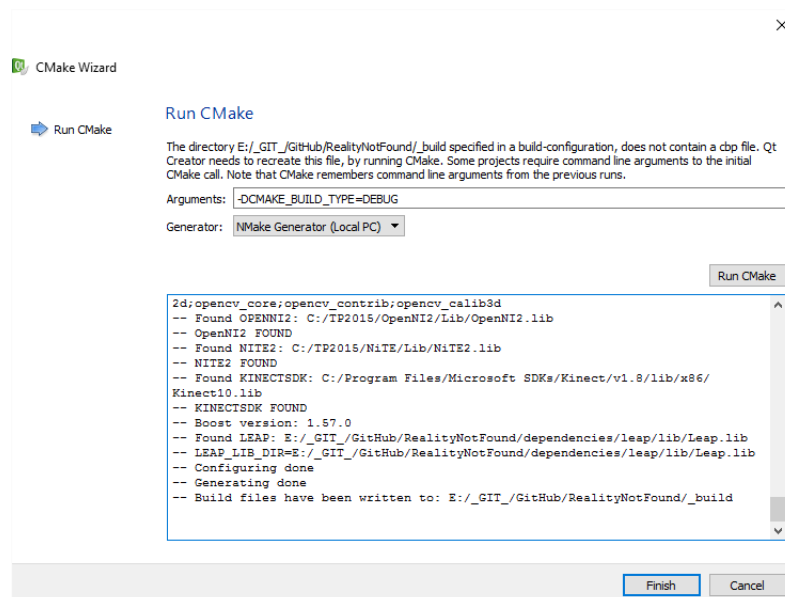
11. Naklónovať projekt RealityNotFound cez git shell (%RealityNotFound%)
12. Vytvoriť v priečinku %RealityNotFound% priečinky `_build` a `_install`
13. Spustiť QtCreator. Tools > Options... > Build and Run:
  - a. záložka CMake – zadať cestu %CMAKE\_DIR%\bin\cmake.exe
  - b. záložka Compilers – ak existuje VS2010 tak sú autodetected
  - c. záložka Qt Versions – zadať cestu %QT\_DIR%\bin\qmake.exe
  - d. záložka Kits – vytvoriť nový a vybrať hodnoty nasledovne:



Obrázok 12 - QtC Kits nastavenia

\*pozn. debugger sa nastavuje až v ďalšej kapitole

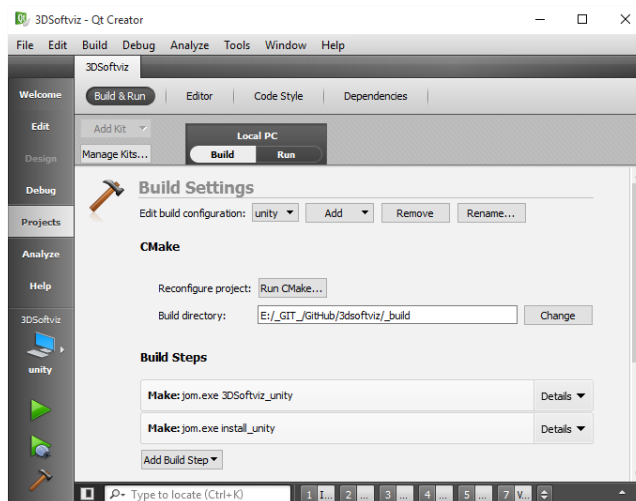
- e. záložka General – nastaviť default build directory: %RealityNotFound%\\_build
- f. Potvrdiť – OK
14. File > Open File or Project... > vybrať *CMakeLists.txt* z %RealityNotFound%
15. Zadať do poľa Arguments:
  - a. -DCMAKE\_BUILD\_TYPE=DEBUG alebo
  - b. -DCMAKE\_BUILD\_TYPE=RELEASE
16. Vybrať z nastavený generátor – NMake Generator (*názov kitu*)
  - a. ak sa generátor nedá vybrať, je dôležité zmazať súbor *CMakeLists.txt.user.2.5pre1* z %RealityNotFound%
17. Stlačiť Run CMake



Obrázok 13 - QtC CMake wizard

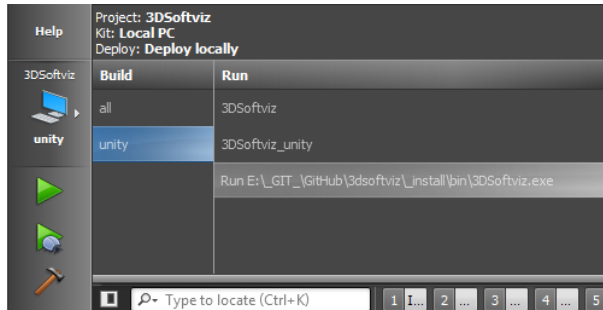
- a. ak došlo k erroru – chýbajúce moduly:
  - cez git shell prejsť do %RealityNotFound%
  - uistiť sa, že je nastavený na *master*
    - na prepnutie *git checkout master*
  - *git submodule update –init*
- b. ak došlo k erroru – cesta nebola nájdená
  - skontrolovať nastavenia premenných v RapidEE
18. Ukončiť – Finish
19. Vybrať Projects > Build & Run > Build, v časti *Edit build configuration* kliknúť na Add > Clone selected, nazvať napr. „unity“

20. Prejsť na vytvorený build config. „unity“, v časti *Build Steps* otvoriť *Details* a zaškrtnúť pri build step *jom.exe* možnosť *3DSoftviz\_unity*
21. Pridať *Add Build Step* > *Make* > *install\_unity*



Obrázok 14 - QtC build project

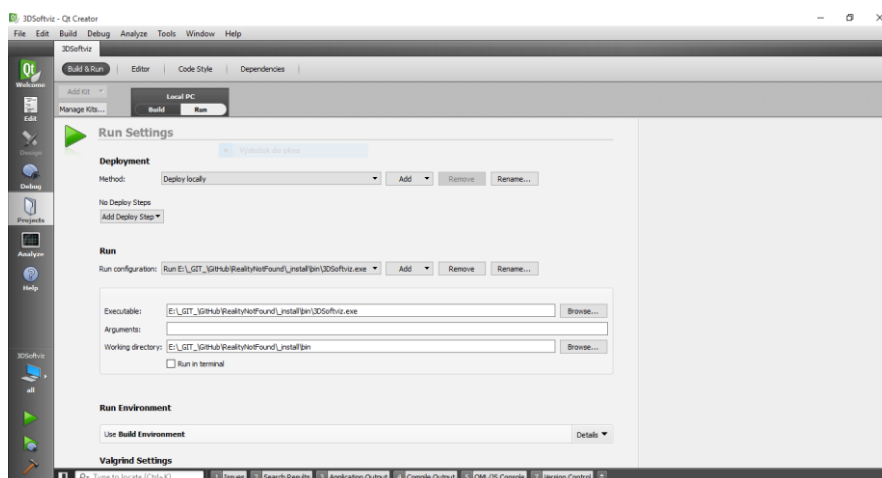
22. Skontrolovať nastavenie build config. – unity



Obrázok 15 - QtC build config

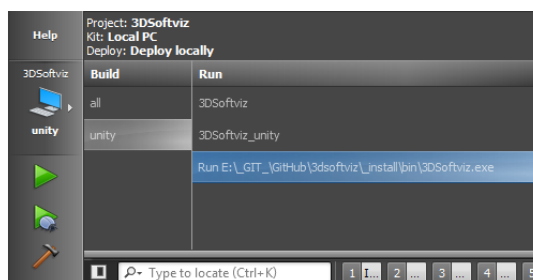
23. Prejsť na pôvodnú „all“ konfiguráciu, v časti *Build steps* pridať *Add Build Step* > *Make* > *install*
24. Stačiť *Build* (kladivko vľavo dole)
  - a. ak došlo k erroru – *-Zm###*
    - *CMakeLists.txt* > riadok 128 zmeniť */Zm216* na */Zm240*
    - spustiť *CMake* odznovu
25. Po úspešnom zbuildovaní vybrať *Projects* > *Build & Run* > *Run*, v časti *Run* pridať *Add* > *Custom Executable* a nastaviť:
  - executable: *%RealityNotFound%\\_install\bin\3DSoftviz.exe*

- working directory: %RealityNotFound%\\_install\bin\



Obrázok 16 - QtC run project

26. Skontrolovať nastavenie run config. – zadaná cesta

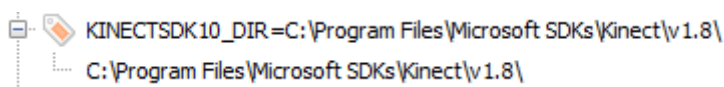


Obrázok 17 - QtC run config

27. Spustiť program pomocou zeleného tlačidla Run

## 5.2 Rozšírenie 3DSoftviz o Kinect

1. Nainštalovať Kinect for Windows
2. Skontrolovať v RapidEE či sa vytvorila premenná %KINECTSDK10\_DIR%

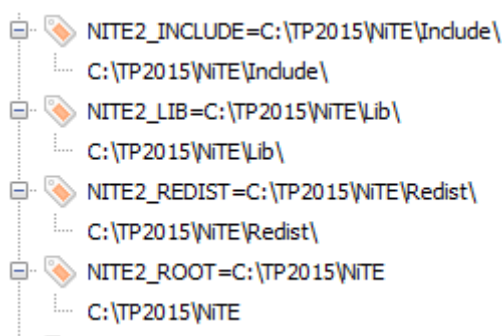


Obrázok 18 - KINECTSDK10\_DIR premenná

3. Nainštalovať OpenNI2 (*OpenNI-Windows-x86-2.2.msi*)

• x86! – inak sa môžu vyskytnúť problémy s linkovaním

4. Skontrolovať v RapidEE či sa vytvorili premenné:
  - a. %OPENNI2\_INCLUDE%
  - b. %OPENNI2\_LIB%
  - c. %OPENNI2\_REDIST%
  - d. %OPENNI2\_ROOT%

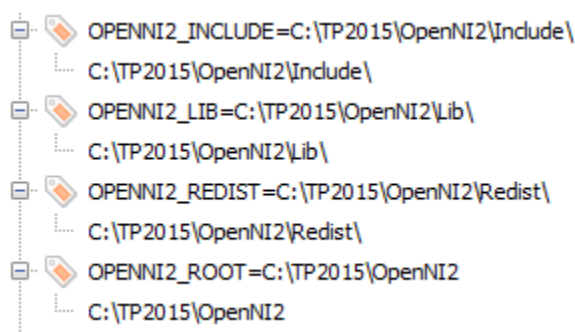


Obrázok 19 - NITE2 premenné

5. Nainštalovať NiTE2 (*NiTE-Windows-x86-2.2.msi*)

• x86! – inak sa môžu vyskytnúť problémy s linkovaním

6. Skontrolovať v RapidEE či sa vytvorili premenné:
  - a. %NITE2\_INCLUDE%
  - b. %NITE2\_LIB%
  - c. %NITE2\_REDIST%
  - d. %NITE2\_ROOT%



Obrázok 20 - OPENNI2 premenné

7. Pridať do premennej CMAKE\_INCLUDE\_PATH:
  - a. %OPENNI2\_INCLUDE%
  - b. %NITE2\_INCLUDE%
8. Pridať do premennej CMAKE\_LIBRARY\_PATH:
  - a. %OPENNI2\_ROOT%\Driver
  - b. %OPENNI2\_REDIST%



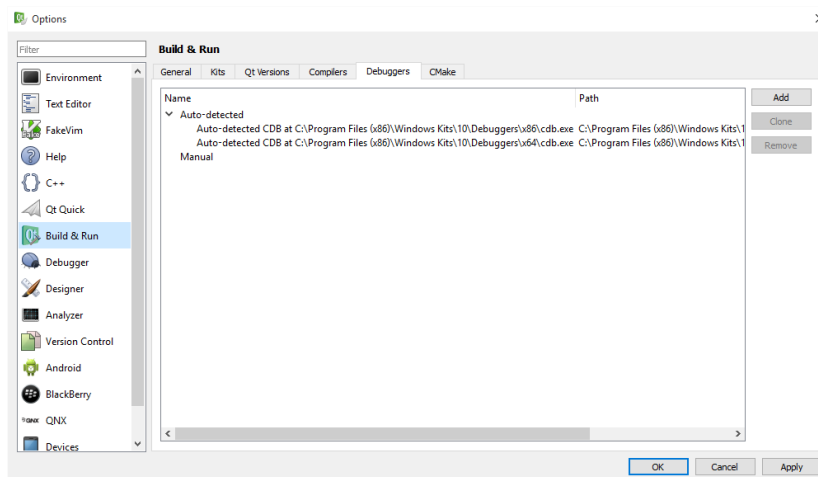
- c. %OPENNI2\_REDIST%\OpenNI2\Drivers
  - d. %OPENNI2\_LIB%
  - e. %NITE2\_ROOT%\Samples\Bin\OpenNI2\Drivers
  - f. %NITE2\_LIB%
9. Pridať do premennej PATH:
- a. %OPENNI2\_REDIST%\OpenNI2\Drivers
  - b. %OPENNI2\_REDIST%
  - c. %NITE2\_REDIST%
  - d. %NITE2\_ROOT%\Samples\Bin
10. Spustiť CMake a skontrolovať vo výpise:
- a. OpenNI2 FOUND
  - b. NITE2 FOUND
  - c. KINECTSDK FOUND

### 5.3 Rozšírenie 3DSoftviz o 3D myš

1. Nainštalovať 3D Connexion SDK(%MOUSE3D\_DIR%)
2. V RapidEE vytvoriť premennú MOUSE3D\_DIR a nastaviť na %MOUSE3D\_DIR%

### 5.4 Návod na nastavenie debugera v QtCreator

1. Nainštalovať WinDbg
2. Skontrolovať v QtCreator Tools > Options > Build & Run > záložka Debuggers či sú autodetected



Obrázok 21 - QtC debugger nastavenia

3. Pridať do QtCreator Tools > Options > Build & Run > záložka Kits pre vytvorený profil položku Debugger (x86)
4. Spustiť CMake (-DCMAKE\_BUILD\_TYPE=Debug)

5. Zvoliť možnosť Debug (vľavo dole medzi Run a Build)

## 5.5 Časté problémy

Pokiaľ sa vyskytnú problémy typu:

- nevie nájsť *glut.h*
- hlási chyby priamo v *glut.h*
- niečo iné s *glut.h*, alebo *glut.lib*, alebo *glut.dll*

Tak najjednoduchšie riešenie je premenovať

```
%OSG_DIR%\ThirdParty\VC10\x86\include\GL\glut.h > glut.h.bak
%OSG_DIR%\ThirdParty\VC10\x86\lib\glut32.lib > glut32.lib.bak
%OSG_DIR%\ThirdParty\VC10\x86\lib\glut32D.lib > glut32D.lib.bak
```

Pokiaľ sa projekt spustí iba cez

*-DCMAKE\_BUILD\_TYPE=Release*,

a cez *-DCMAKE\_BUILD\_TYPE=Debug*

hlási nullpoint exception pri načítavaní obrázkov na pozadie a program crashuje

Tak riešením je znovu stiahnuť/nakonfigurovať OSG podľa návodu.