SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE

Fakulta informatiky a informačných technológií

VIZUALIZÁCIA INFORMÁCIÍ V OBOHATENEJ REALITE

Dokumentácia k riadení 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

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 dokumentácií. 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 súčastnosti 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 skusenosti 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, vedúci tímu.
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

- s 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, inštalačná príručka
- formátovanie dokumentov

BC. FILIP JURČACKO

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

BC. DALIBOR MÉSZÁROS

- dok. k riadenie: metodika komunikácie, manažment vývoja
- dok k inž. dielu: inštalačná 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 rekapituluje, čo sa nám počas šprintu podarilo, čo nešlo podľa našich predstáv a čo by bolo treba zlepšiť do budúcna.

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 ním vytvorené úlohy v JIRE, ktoré sa následné riešia v ďalšom šprinte.

2.4 Manažment prehliadky kódu

Keď je kód otestovaný jeho autorom, ako členovia tímu sa dohodneme medzi sebou kto bude ten kód prehliadať. 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 úlohy

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 o výbere úloh 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 retrospektívnu za posledným šprintom a zhodnotili čo sme robili zle, ale hlavne čo a ako môžeme zlepšiť do budúceho šprintu. Priebežný status úloh exportujme 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 funkcionalita, 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, prideľovanie 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 revidačnej 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 funkcionalitu 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 schopný 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.

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 cene produkoval konzistentný a čitateľný kód pre ostatných členov, pripadne 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 skupín – documents, bugs, warnings, epics. 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 poukazuje na dôležitosť sledovania pasívnych komunikačných kanálov, ktoré si musia ustrážiť jednotlivý členovia tímu osamote (napr. pridelený branch na GitHub-e, pridelená úloha v Jire). Členovia tímu sú tiež povinný mať všeobecný prehľad, ktorý je poskytnutý skupinovou 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 kód 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 striktne 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

- urobiť prehľad kódu, snažiť sa ho pochopiť
  - v prípade akýchkoľvek otázok, spýtať sa autora kódu ako to myslel
  - opraviť zjavné chyby
- skontrolovať ci je kód okomentovaný
  - ci je kód okomentovaný v dostatočnej miere
  - doplniť komentáre keď treba, dopísať nové
- skontrolovať formátovanie
  - ci sú dodržané pravidlá napísania kódu v jazyku c++
- preveriť kód nástrojmi cppcheck, cpplint
- urobiť kontrolu warningov pri kompilácii
- overiť funkčnosť nového kódu v programe
- keď všetko bude overené, treba vystaviť pull request

A.b Metodika manažmentu verzií

Používame vetvy: Master, Develop, Feature, Hotfix

Master - hlavný projekt

Develop - branchnutá z mastra, kazdý ľprint ma vlastnú Develop vetvu, na konci šprintu sa mergne späť do mastra

Feature - branchnutá z developu, každý nový kus funkcionality (task v Jire), ktorý sa kódi musí mat vlastnú Feature vetvu, po dokončení a validácii kódu sa mergne späť do Developu, NEINTERAGUJE S MASTER VETVOU

Hotfix - vetva na rýchly fix priamo z mastera, merguje sa do mastera AJ developu, navyšuje aktuálnu verziu

Vždy mergujeme cez Shell a s prepínačom „--no-ff“

Cheat sheet so vsetkymi základnými commandmi:

https://training.github.com/kit/downloads/github-git-cheat-sheet.pdf

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 mergu.

Commit messages:

v Commit messages používame tagy na začiatok:

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

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

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

[REF] - pre refaktoring

[FMT] - formátovanie textu, úprava

[TEST] - pre testy

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

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 branch -d "feature/meno-feature"
```

//Deleted branch meno_feature (was 05e9557).

```
$ git push origin develop
```

Tvorba hotfix branch-u:

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

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

```
$ ./bump-version.sh {cislo_verzie}
```

//Navýšenie verzie

```
$ git commit -a -m "Bumped version number to {cislo_verzie}"
```

Uzatvorenie Hotfix branchu:

//----------MERGE DO MASTRA----------

```
$ git checkout master
```

//Switched to branch 'master'

```
$ git merge --no-ff hotfix/nazov-co-fixujem
$ git tag -a {cislo_verzie}
```

//----------MERGE DO DEVELOP----------

```
$ git checkout develop
```

//Switched to branch 'develop'

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

//Deleted branch hotfix/nazov-co-fixujem (was abbe5d6).

A.c 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.

Komentovanie zdrojového kódu Používame dva typy komentárov:

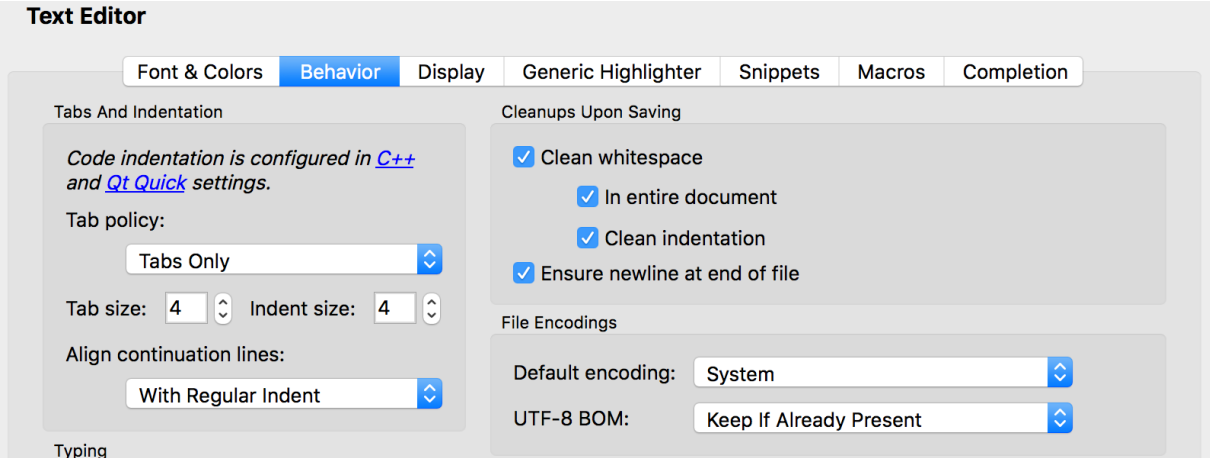
• jednoriadkový, na jeden riadok do 80 znakov, stručne opisuje časť kódu,

• blokový komentár, opisuje väčšiu časť kódu, zväčša metódu alebo triedu.

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 vývojové 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

Zlé praktiky:

- nikdy nepoužívať using namespace a tiež keyword using.
- nepoužívať 0 (nulu) ako NULL pointer, vždy použiť nullptr (ak kompilator podporuje C++11), inak NULL.
- otvorenie pull-requestu bez predchádajúcej aplikácie metodiky pre prehliadku kódu, warningy počas kompilácie.
- pri testovaní pointrov, použite (!myPtr) alebo (myPtr); nepoužívajte myPtr != nullptr.
- neporovnávajte 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,
    - inizializovať v takom poradí v akom sú zapísané v triede, resp. usporiadať od najväčších (poiner, 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.d Metodika vytvárania úloh v nástroji na evidenciu úloh

Metodika je určená predovšetkým hlavnému administrátorovi nástroja AJira, 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 produktového backlogu do backlogu šprintu. 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.e 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 <QtTest/QtTest>
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 umožnuje 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 funkcionalitu používateľských scenárov, pridanej funkcionality.

Príprava testu:

Vstup: požiadavka na funkcionalitu, používateľský scenár

Výstup: testovací scenár

Zodpovedný: Product Owner

Akceptačný test sa najprv vytvorí, v podobe scenára ktorý napĺňa požiadavku na funkcionalitu. 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:

ID	1	**Názov**	Navigácia v 3D priestore prostredníctvom 3d myši	
Účel	Overenie funkčnosti 3D myši			
Vstupné podmienky			3D myš je pripojená k počítaču	
Výstupné podmienky				
Krok	**Akcia**		**Očakávaná reakcia**	**Skutočná reakcia**
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	

Vykonanie 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 ním vytvorené úlohy v JIRE, ktoré sa následné riešia v ďalšom šprinte.

Príloha B Zápisnice

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 co sluzi a co momentalne dokaze,
  - Predstavenie vizie.
- Diskusia o:
  - SCRUMe,
  - O riadeni a nastrojoch riadenia,
  - O tom ako budeme komunikovat mimo strenuti.

Úlohy na budúce stretnutie:

ID	Riešiteľ	Opis úlohy
-	Volba "sefa" timu	Tim si zvoli zodpovedneho clena za tim, ktory bude riesit pripadne problemy.
-	Vyber hlavneho architekta	Hlavny architekt projektu, mal by mat najlepsi prehlad o vsetkych castiach systemu.
-	Vyber nastroja na task mgmt	Trello, Jira, Redmine alebo nieco ine?
-	Navrhnut web timu	Webova stranka 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:

ID	Riešiteľ	Opis úlohy	Stav
-	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:

ID	Riešiteľ	Opis úlohy
-	Tím	Výber frameworku na BDD/TDD development.
-	Tím	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:

ID	Riešiteľ	Opis úlohy	Stav
-	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:

ID	Riešiteľ	Opis úlohy
-	Cimerman	Gitflow - naštudovať
-	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	dokumentacia - vytvoriť metodiku
-		Zjednotiť metodiky: ako programovat v cpp, gitflow, formatovanie kodu, ...
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:

ID	Riešiteľ	Opis úlohy	Stav
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 inštalačných problémov, kompilácia projektu.

Úlohy na budúce stretnutie:

ID	Riešiteľ	Opis úlohy
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

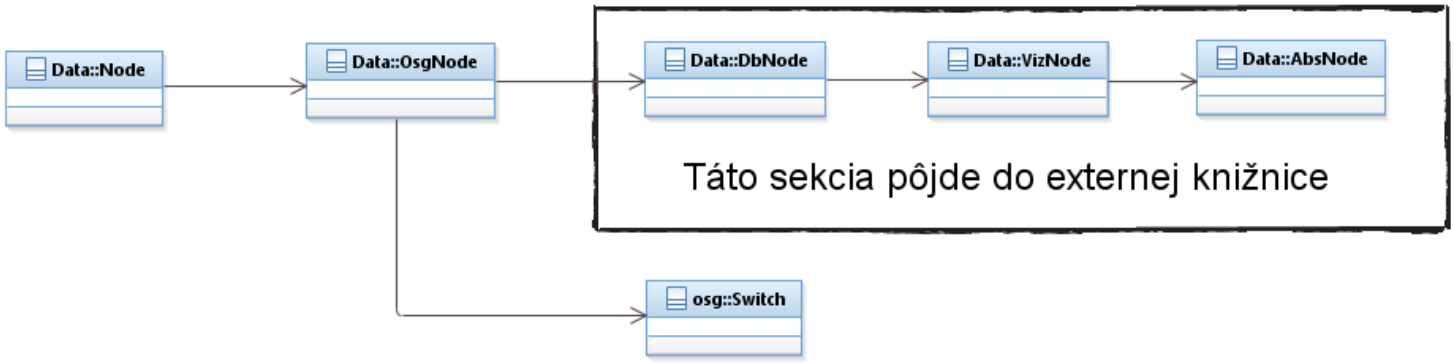
Stav úloh z minulého stretnutia:

ID	Riešiteľ	Opis úlohy	Stav
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	Nahrať dokumenty na webstránku	done

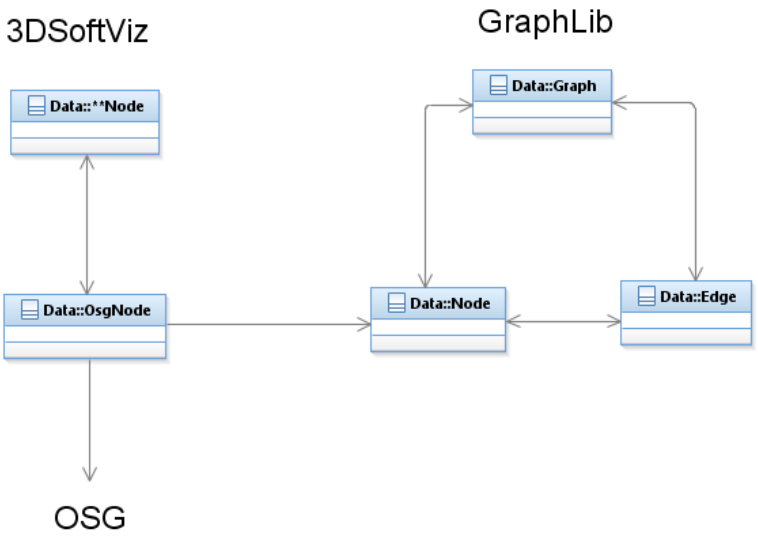
Priebeh stretnutia:

- 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 problematiku 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 modifkovať 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 namespacu "OsgModeling" je problém, že "Cube", "Sphere", "SpikySphere" sú mimo svojho namespacu
- "HistoryBuffer" by sa mal nachádzať v namespacu "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	Vytvorit metodiku - ako programovat 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

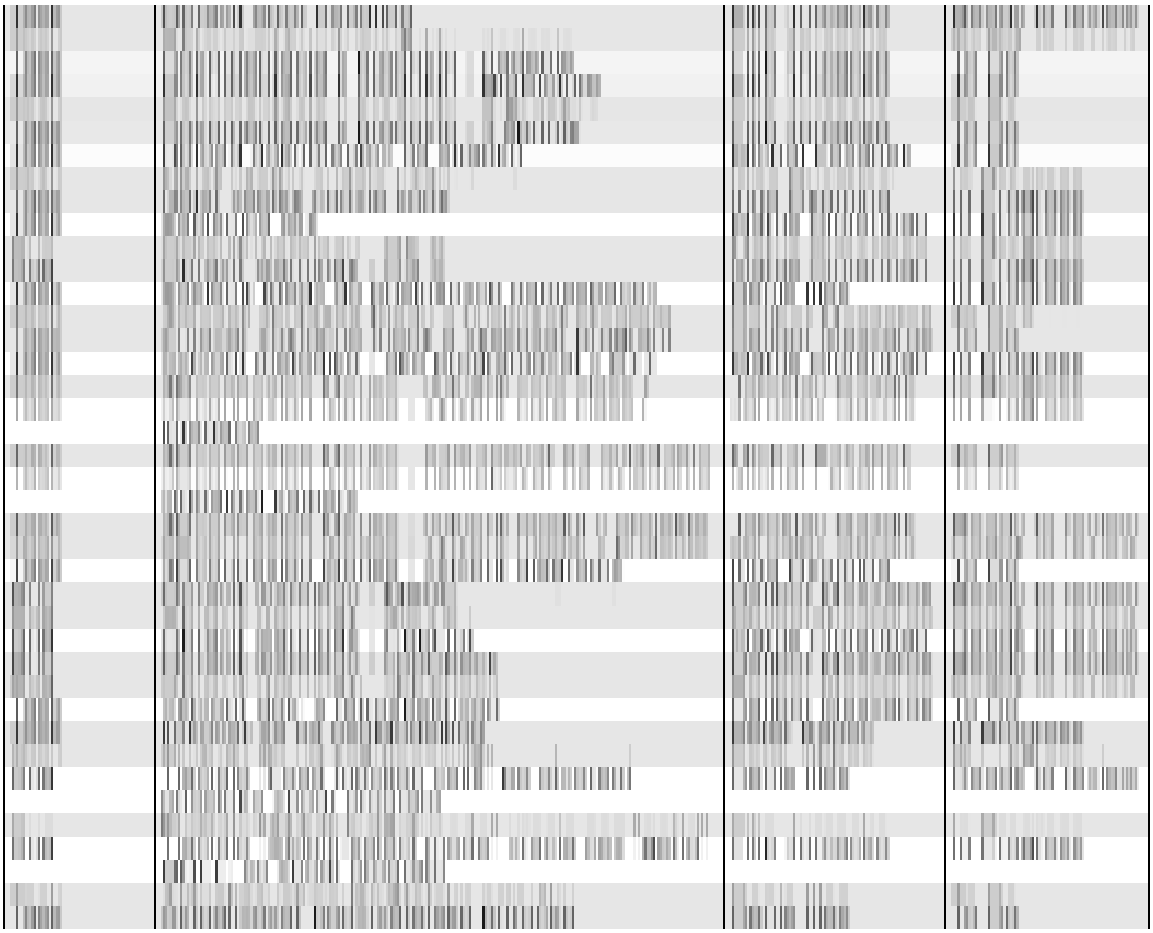
Stav úloh z minulého stretnutia:

ID	Riešiteľ	Stav	Opis úlohy
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	Vytvorit metodiku - ako programovat v C++
79	Dyomina	Done	Vytvoriť CodeReview metodiku
41	Meszaros	Done	Vytvoriť dok. s retrospektívou
13	Illes	Ready to review	Tlačidlo "Load function calls" po otvorení priečinka zhodí program
12	Jurcacko	Merge od Garaja	Tlačidlo "&Multi-select mode" vracia cez "getId" NULL, čo zhodí program
57	Cimerman	Ready to review	Opraviť warningy - CppLint
56	Meszaros	Ready to review	Opraviť warningy - cppCheck
58	Meszaros	Ready to review	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:
  - Mergovat aspoň deň pred stretnutím, resp. pred koncom šprintu.
  - Na stretnutí by už malo byť všetko mergnute 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 blockerov. 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 napísana 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:



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:

ID	Riešiteľ	Opis úlohy	Stav
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 modeling	ready to review
12	Jurčacko	Tlačidlo “Multi-select Node” vracia NULL, čo zhodí program	in progress
85	Illés	Začleniť triedy do príslušných namespace-ov	in progress
86	Cimerman	Opraviť warningy - OSX	in progress
69	Dyomina	3DConnexion mouse vytvoriť diagramy	in progress
82	Gazdík	3DConnexion mouse vytvoriť core class/interface	in progress
72	Fašánek	3DConnexion mouse nama-povať na ovládanie virtuálnej	to do

		kamery	

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
- Súbory 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

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	*Unassigned*	Done			implement
Task	Nastaviť editor v QtCreator	*Unassigned*	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	*Unassigned*	Done			implement
Task	Vyskúšať nástroje pre C++	*Unassigned*	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 zápisnice z 2-4w	Matus Cimerman	Done			other
Task	Šablóna pre zápisnice zo strenutí	Jaroslav Gazdik	Done	Vytvorit gdoc sablonu pre zapisnice zo strenuti - pre web.		other
Task	Aktualizovať knižnice v projekte	*Unassigned*	Done	Knižnice Aruco, OpenSceneGraph 3.4		implement

Task	Nastaviť JIRU podľa potreby	Denis Illes	Done			other
Task	Upraviť inštalačný manuál	*Unassigned*	Done			other
Task	Nahrať dokumenty na webstránku	Michal Fasanek	Done			other
Task	Dokončiť webstránku	Michal Fasanek	Done			other

C.b Úlohy z druhého šprintu

Issue Type	Summary	Assignee	Status	Description	Epic Name	Labels
Task	Zapisnica w8	Michal Fasanek	Done	V prilohe export taskov z Jiry. w8-1_status - status uloh z minuleho tyzdna w8_todo-inprogress - zvysne ulohy na buduci tyzden		other
Task	Gitflow metodiku upraviť	Jaroslav Gazdik	Done	Pill requesty a nič viac...		other
Task	Dokumentácia k produktu - riade-nie	Filip Jurcacko	To Do	BIELIKOVÁ http://www2.fiit.stuba.sk/~bielik/courses/tp-slov/tp-main.html		other
Task	Dokumentácia k produktu - BigPic-ture	Filip Jurcacko	To Do	BIELIKOVÁ http://www2.fiit.stuba.sk/~bielik/courses/tp-slov/tp-main.html		other
Task	Dokumentácia k produktu - 3 šprinty	Filip Jurcacko	To Do	do 18.11. v AIS		other
Task	Úprava CodeRe-view metodiky	Irina Dy-omina	Done	do 19.11 do AIS - BIELIKOVÁ		other
Task	Upload doku-mentov na stránku	Michal Fasanek	Done			other
Task	Zápisnica W7	Matus Ci-merman	Done			other

Task	Vytvorit metodiku - ako programovat 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, * cpplint, * cppcheck, * spojit s txt na drope.		other
Task	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.		implement
Task	3Dconnexion Mouse - vytvoriť 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 - vytvoriť diagramy	Irina Dyomina	In Progress	Vytvoriť 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 Fasanek	Done	V prilohe export taskov z Jiry. w8-1_status - status uloh z minuleho tyzdna w8_todo-inprogress - zvysne ulohy na buduci tyzden		other
Task	Opravit error handling	Filip Jurcacko	Done	Na viacerych miestach v kode je do funkcii posielany argument bool* error ktory sa vnutri funkcie nastavy v pripade chyby Avsak casto po vykonani funkcii bud nieje skontrolovany vobec alebo je viacnasobne prepisany viacerymi volaniami funkcii nez je skontrolovany. Fix it.		implement
Task	Gitflow metodiku upraviť	Jaroslav Gazdik	Done	Pill requesty a nič viac...		other
Task	Dokumentácia k produktu - riadenie	Filip Jurcacko	To Do	BIELIKOVÁ http://www2.fiit.stuba.sk/~bielik/courses/tp-slov/tp-main.html		other
Task	Dokumentácia k produktu - BigPicture	Filip Jurcacko	To Do	BIELIKOVÁ http://www2.fiit.stuba.sk/~bielik/courses/tp-slov/tp-main.html		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 Fasanek	Done			other
Task	Nájsť všetky cudzie kódy	Irina Dyomina	Done	identifikovať všetky cudzie kódy, a nájsť odkiaľ boli zobraté napr: foldre: src > Noise (hlavička zobrazuje že kód je prevzatý) Viewer > TreeItem, TreeModel QOSG > qtcolorpicker OSGQTBrowser > WebViewImage LuaGraph >LuaGprahTreeItem		implement
Task	Zápisnica W7	Matus Cimerman	Done			other
Bug	Opraviť warningy - OS X	Matus Cimerman	In Progress			implement
Task	Začleniť triedy do príslušných namespacov	Denis Illes	Done	Triedy ako "Cube", "Sphere", "SpikySphere" sa nachádzajú mimo svojho namespacu (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	Vytvoriť 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 - 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.		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
Task	3Dconnexion Mouse - vytvoriť 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 - vytvoriť diagramy	Irina Dyomina	In Progress	Vytvoriť 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 odstranit z projektu subory v Math/- Bezier- Curve- Export - Model- Utilitiesa linkovat sa na osgModeling kniznicu, ktoru treba pridat 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 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

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.
- Prideľovanie 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 neboj 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 revidačnej 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, kedykoľvek nevieme jednoznačne určiť, ako postupovať v riešení úloh
- Nebudeme si odkladať riešenie úloh na poslednú chvíľu, aby sa stihli zrevidovať

D.c Po treťom 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.

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

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. Jednotlivý členovia sa musia zoznámiť s problematikou vizualizácie informácií. Na predmete tímový projekt sa bude vylepšovať a rozširovať existujúci rozsiahli projekt, ktorý má implementovanú funkcionalitu 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 spájaní 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 implementácií 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.

Najnáročnejšou úlohou bude správne nasadiť funkcionalitu 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 navrhnúť vnorené objekty, ktoré majú jednoznačne odzrkadľovať štruktúru softvéru.

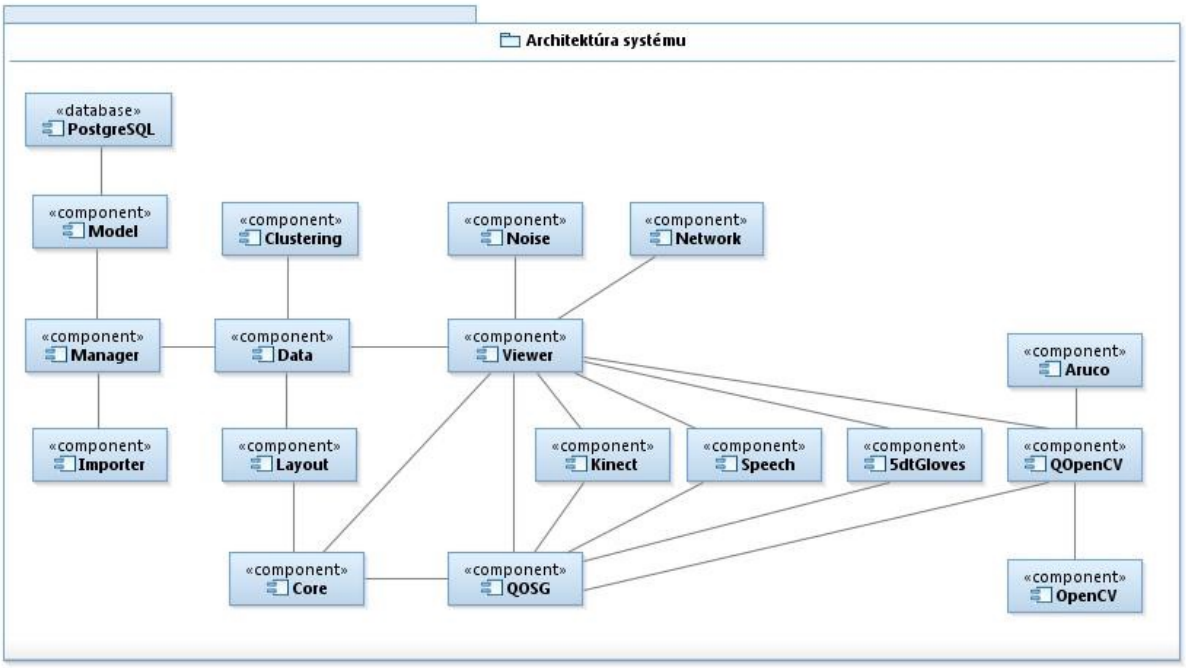
Vyriešením predošlej problematiky sa môže vyskúšať prepojenie spomenutých geografických objektov a grafom volaní, ktorý sa môže vytvoriť z podobných modulov. Výsledkom má byť komplexná grafová štruktúra znázorňujúca jednotlivé moduly, graf volaní v moduloch a prepojenia medzi modulmi. Komplikované abstraktné 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

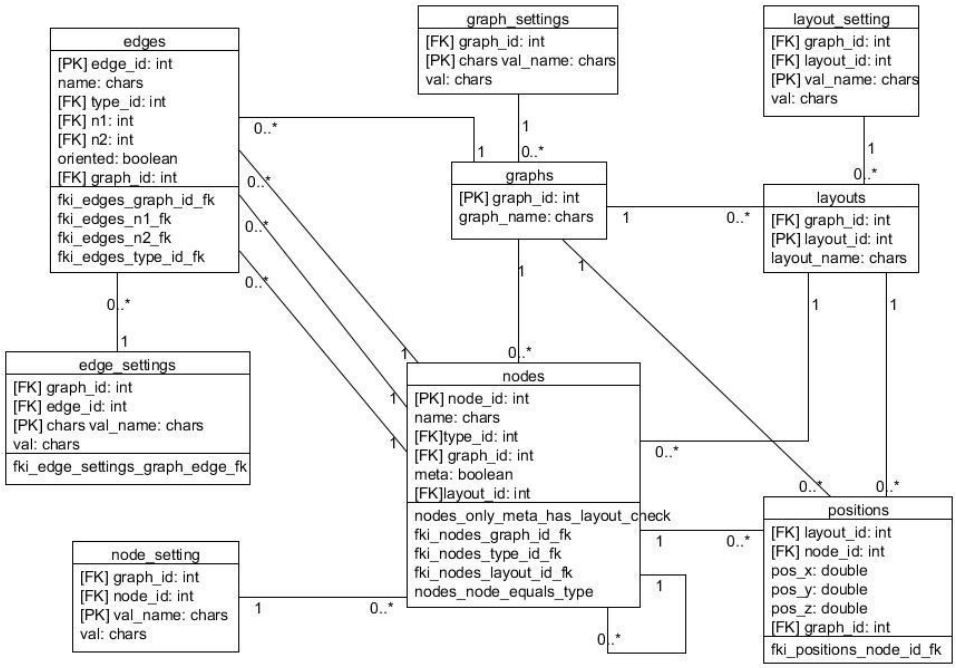
Na Obrázku 1 je zobrazená štruktúra projektu, ktorá bola štartovacím bodom pre náš projekt. Pracujeme na rozšírení projektu o nové moduly ako je 3D myš, pomocou ktorej sa zefektívni ovládanie programu, teda manipulácia s grafmi vo virtuálnom priestore. Následne sa pracovalo na úpravách existujúcich komponentoch, 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.

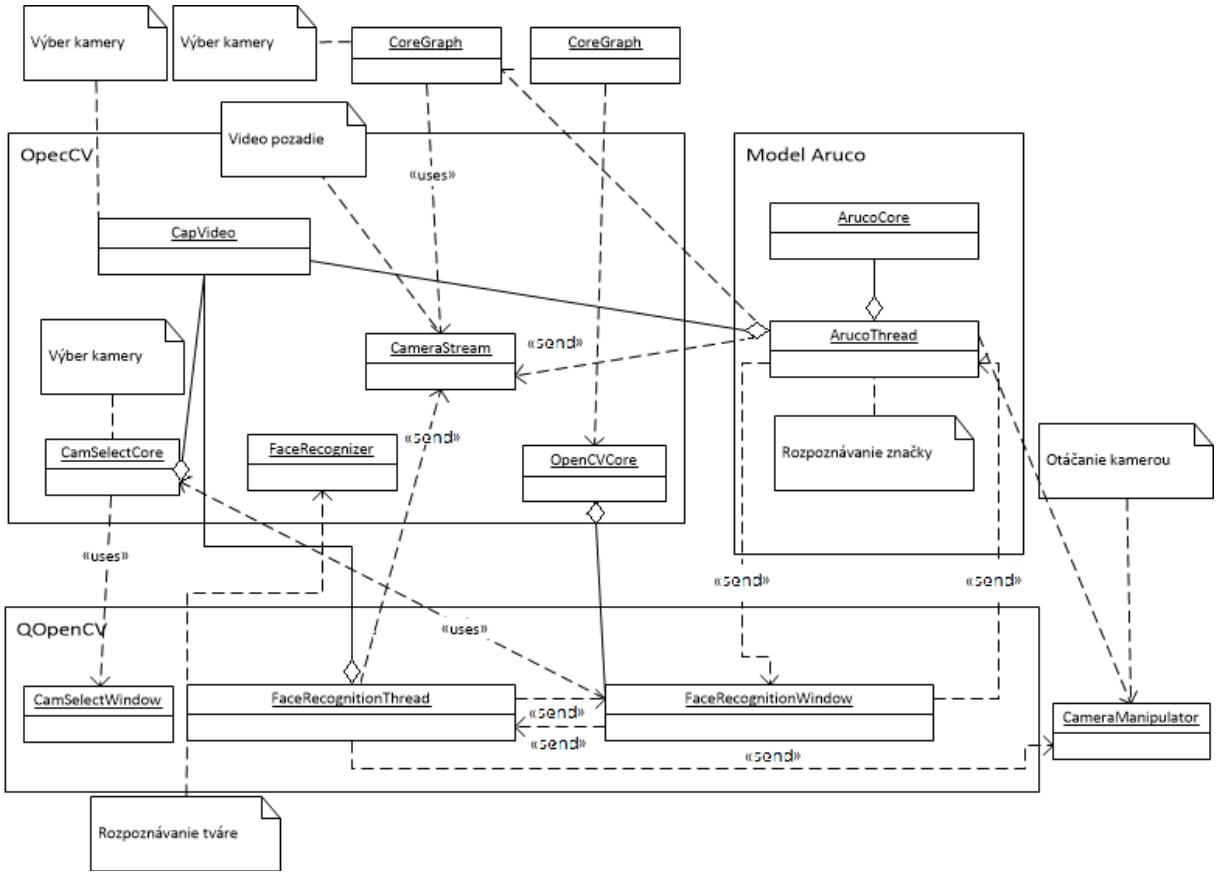


Obrázok 2 - dátový model

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 začiatočný 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 funkcionalitu projektu
- Model Aruco - triedy obsahujúce funkcionalitu 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 funkcionalitu.

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 funkcionalitu 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 funkcionalitu rozpoznávania hlasu.

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

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

Aruco – obsahuje funkcionalitu, 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 úroveň článkov prstov.

3.2 Plánované moduly

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

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

AR okuliare (STAR 1200XLD od spoločnosti VUZIX)

V momentálnom stave projektu sú do nášho projektu pridané knižnice 3D myšky, funkcionalitu s ním spojenú plánujeme implementovať v nasledujúcich šprintoch. Vo veľkej miere naše úlohy spočívali v úpravách zdrojového kódu, opravách warningov a chýb, s ktorými sme sa úspešne popasovali.

4 Moduly systému

4.1 3dMouse

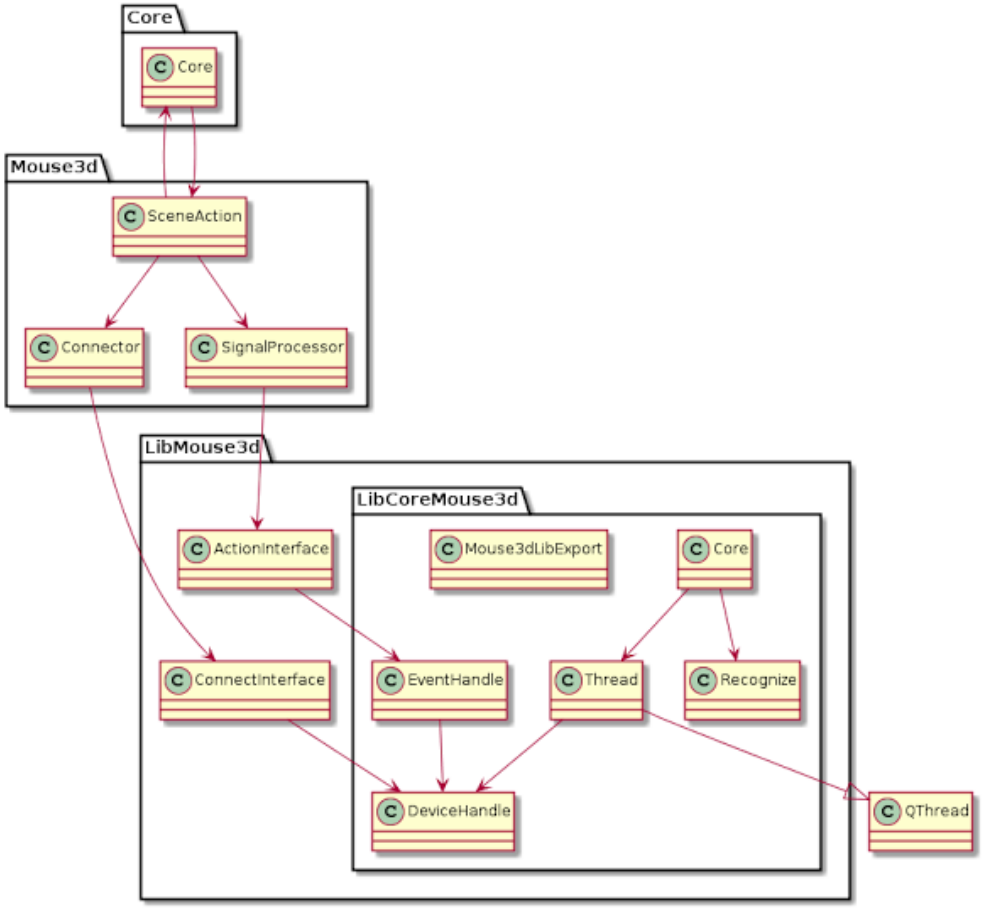
K dispozícii máme dve modely počítačových 3D myší. Kompaktný Space Navigator for notebooks je pre svoje malé rozmery ľahko prenosný. Na druhej strane, Space Pilot Pro je mohutnejšia myš obohatená o niekoľko tlačidiel naviac. Pre obe zariadenia slúži rovnaké SDK.

4.1.1 Analýza modulu 3dMouse

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 3dMouse

Pri návrhu sme sa inšpirovali predošlými funkčnými modelmi. Identifikovali sme Core triedu, ktorá bude bezprostredne pracovať s poskytnutým SDK pre 3D myš. 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 3dMouse, ktorý môže byť zaradený medzi ostatné.



Obrázok 4 - triedy modulu 3dMouse

4.1.3 Implementácia

4.1.4 Testovanie

5 Inštalačná 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[1] (tím č.4 klónuje *3dsoftviz*[2]) z Githubu a stiahnuť potrebný softvér:

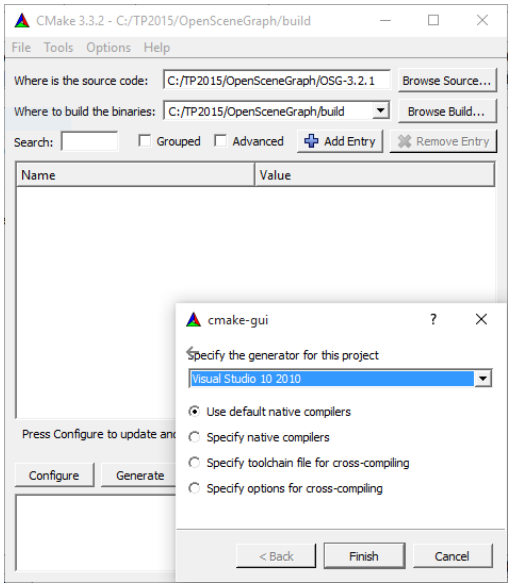
- CMake[3] (v3.3.2)
- OpenSceneGraph[4] (v3.4) – iba zdrojáky > treba buildnúť (cca 40-50min)
  - OpenSceneGraph[5] (v3.4) – buildnuté (17.10.2015)
- Kinect for Windows SDK[6]
- Microsoft VisualStudio 2010 SP1[7] (okrem Express edition)
- Qt[8] (v4.8.5)
  - QtCreator[9] (v3.2.1)
- OpenCV[10] (v2.4.10)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

Postup inštalácie:

1. Nainštalovať CMake. (Cesta je v dokumente označená ako *%CMAKE_DIR%*)
2. Nainštalovať Qt *(%QT_DIR%*)
3. Nainštalovať QtCreator do zložky Qt

[1] Zdroj: https://github.com/dominikhorniak/RealityNotFound
[2] Zdroj: https://github.com/cimox/3dsoftviz
[3] Zdroj: https://cmake.org/files/v3.3/cmake-3.3.2-win32-x86.exe
[4] Zdroj: http://trac.openscenegraph.org/downloads/developer_releases/OpenSceneGraph-3.4.0.zip
[5] Zdroj: https://mega.nz/#!cIpy2TZQ!4k29x33yVvQDJRXT8B1J_3F-L1t30B3A2Z-uoQF9OeU
[6] Zdroj: http://www.microsoft.com/en-us/download/details.aspx?id=40278
[7] Zdroj: http://www.microsoft.com/en-us/download/details.aspx?id=23691
[8] Zdroj: http://ftp.fau.de/qtproject/archive/qt/4.8/4.8.5/qt-win-opensource-4.8.5-vs2010.exe
[9] Zdroj: http://ftp.fau.de/qtproject/official_releases/qtcreator/3.2/3.2.1/qt-creator-opensource-windows-x86-3.2.1.exe
[10] Zdroj: http://sourceforge.net/projects/opencvlibrary/files/opencv-win/2.4.10/opencv-2.4.10.exe/download
[11] Zdroj: http://sourceforge.net/projects/boost/files/boost/1.57.0/boost_1_57_0.zip/download
[12] Zdroj: http://www.rapidee.com/download/RapidEE_setup.exe
[13] Zdroj: http://ulozto.sk/xaZJArtg/vc10-zip
[14] Zdroj: http://uloz.to/xpz9zqAi/openni-nitte-rar
[15] Zdroj: http://download.microsoft.com/download/B/0/C/B0C80BA3-8AD6-4958-810B-6882485230B5/standalonesdk/sdksetup.exe
[16] Zdroj: http://download.microsoft.com/download/E/1/F/E1F1E61E-F3C6-4420-A916-FB7C47FBC89E/standalonesdk/sdksetup.exe
[17] Zdroj: https://mega.nz/#!VYAVBKwb!1-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

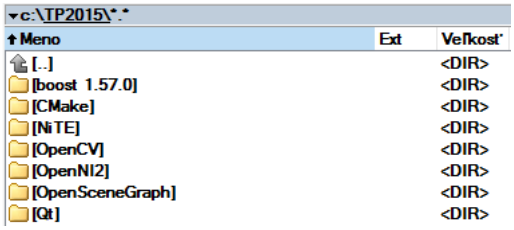


Obrázok 5 - CMake pre OSG

    e. Nájsť súbor *OpenSceneGraph.sln* v %OSG_DIR%\build
    f. Otvoriť súbor vo VS2010
    g. Nastaviť Solution Configuration na *Debug*
    h. Nájsť projekt ALL_BUILD > pravý klik > build
    i. Po skončení nájsť projekt INSTALL > pravý klik > build
    j. Nastaviť Solution Configuration na *Release*
    k. Nájsť projekt ALL_BUILD > pravý klik > build
    l. Po skončení nájsť projekt INSTALL > pravý klik > build

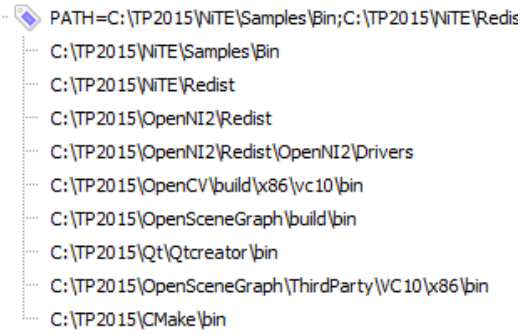
8. Rozbaliť FreeGlut do %OSG_DIR%\ThirdParty\VC10\x86\ (prepísať súbory)
9. Nainštalovať OpenCV (*%OPENCV_DIR%*)
10. Rozbaliť Boost (*%BOOST_DIR%*)

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



Obrázok 6 - Nainštalovaný SW

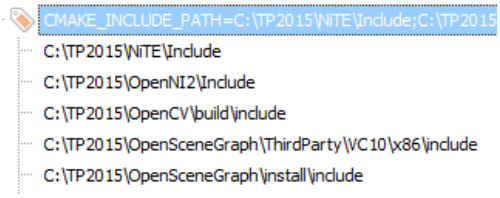
11. Nainštalovať a otvoriť RapidEE, v ktorom sa vykonajú tieto zmeny:
    a. do PATH pridať premenné:
        - %CMAKE_DIR%\bin
        - %QT_DIR%\bin
        - %QT_DIR%\Qtcreator\bin
        - %OSG_DIR%\build\bin
        - %OSG_DIR%\ThirdParty\VC10\x86\bin
        - %OPENCV_DIR%\build\x86\vc10\bin



Obrázok 7 - PATH premenná

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

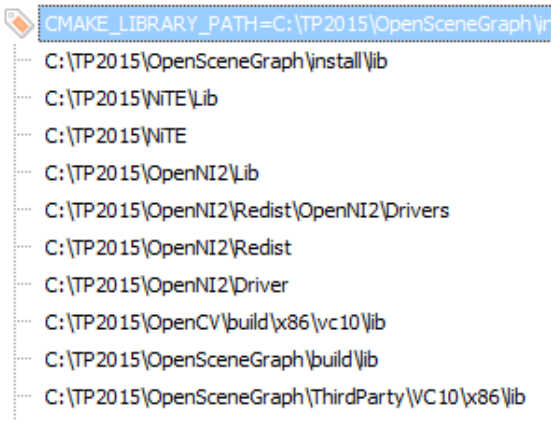
    b. Vytvoriť premennú CMAKE_INCLUDE_PATH a pridať:
        - %OSG_DIR%\install\include
        - %OSG_DIR%\ThirdParty\VC10\x86\include
        - %OPENCV_DIR%\build\include



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



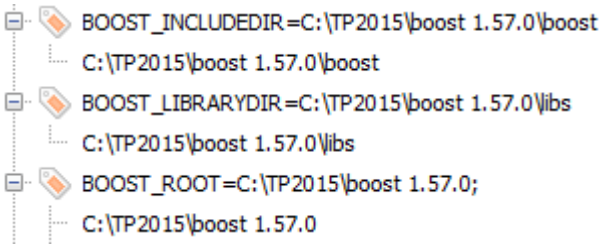
Obrázok 9 - CMAKE_LIBRARY_PATH premenná

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

d. Vytvoriť premennú BOOST_INCLUDEDIR a pridať:
   - %BOOST_DIR%\boost

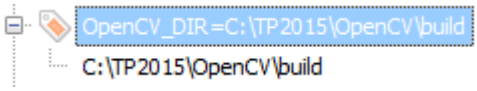
e. Vytvoriť premennú BOOST_LYBRARYDIR a pridať:
   - %BOOST_DIR%\libs

f. Vytvoriť premennú BOOST_ROOT a pridať:
   - %BOOST_DIR%



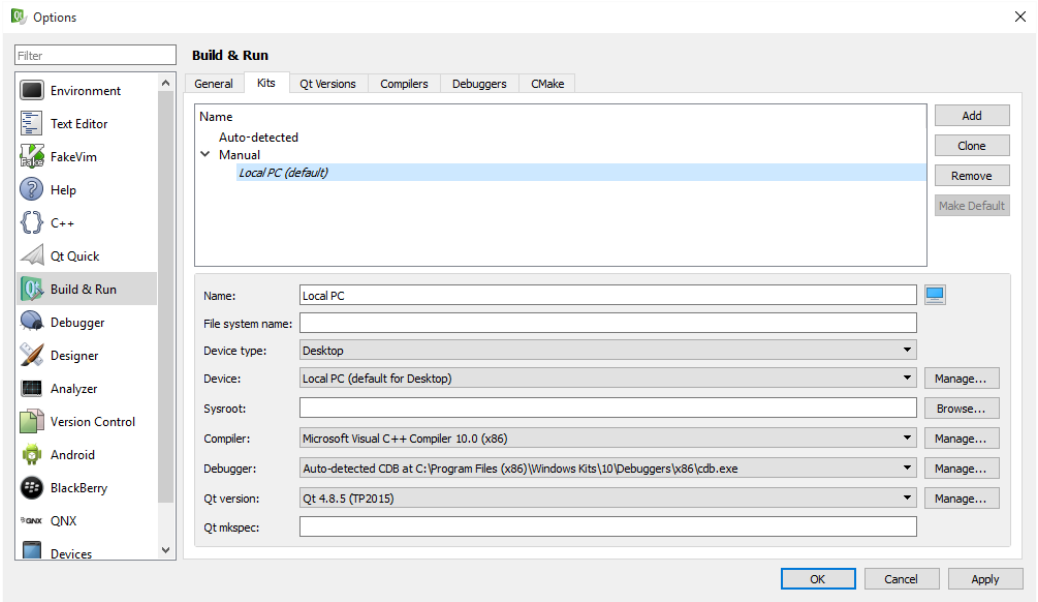
Obrázok 10 - BOOST premmené

g. Vytvoriť premennú OPENCV_DIR a pridať:
   - %OPENCV_DIR%\build



Obrázok 11 - OPENCV_DIR premenná

12. Naklónovať projekt RealityNotFound cez git shell (*%RealityNotFound%*)
13. Vytvoriť v priečinku %RealityNotFound% priečinky *_build* a *_install*
14. 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:

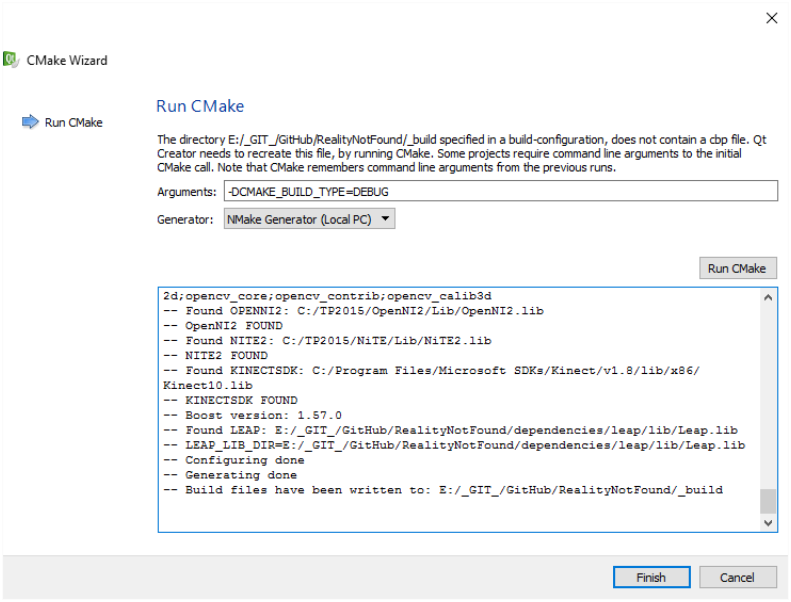


Obrázok 12 - QtC Kits nastavenia

*pozn. debugger sa nastavuje až v ďalšej kapitole

e. záložka General – nastaviť default build directory: %RealityNotFound%_build
f. Potvrdiť – OK

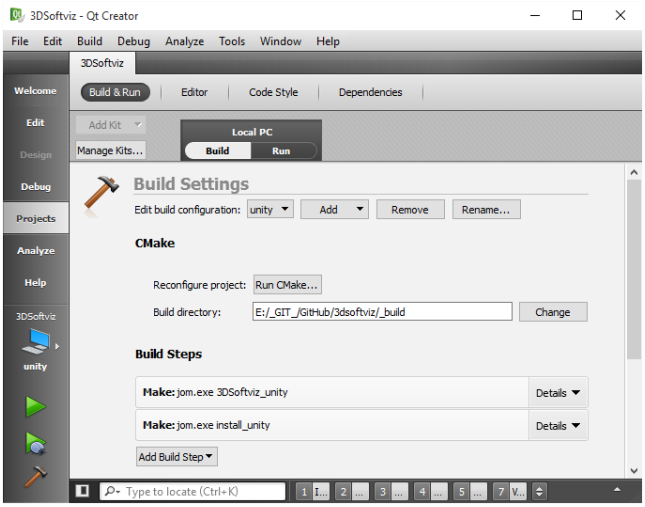
15. File > Open File or Project... > vybrať *CMakeLists.txt* z %RealityNotFound%
16. Zadať do poľa Arguments:
    a. -DCMAKE_BUILD_TYPE=DEBUG alebo
    b. -DCMAKE_BUILD_TYPE=RELEASE
17. Vybrať z nastavený generátor – NMake Generator (*názov kitu*)
    a. ak sa generátor nedá vybrať, je dôležité zmazať súbor *CMakeLists.txt.user.2.5pre1* z %RealityNotFound%
18. Stlačiť Run CMake



Obrázok 13 - QtC CMake wizard

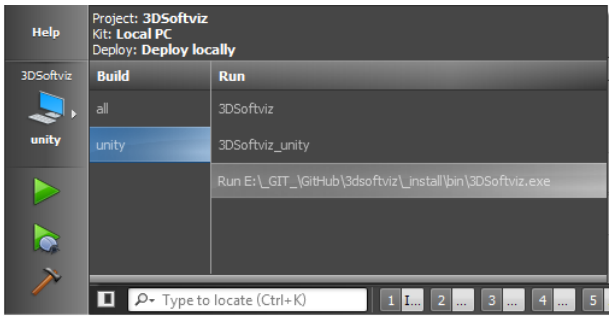
a. ak došlo k erroru – chýbajúce moduly:
    - cez git shell prejsť do %RealityNotFound%
    - uistiť sa, že je nastavený na *master*
        - na prepnutie *git checkout master*
    - *git submodule update –init*
b. ak došlo k erroru – cesta nebola nájdená
    - skontrolovať nastavenia premenných v RapidEE

19. Ukončiť – Finish
20. Vybrať Projects > Build & Run > Build, v časti *Edit build configuration* kliknúť na Add > Clone selected, nazvať napr. „unity“
21. Prejsť na vytvorený build config. „unity“, v časti *Build Steps* otvoriť *Details* a zaškrtnúť pri build step *jom.exe* možnosť *3DSoftviz_unity*
22. Pridať Add Build Step > Make > *install_unity*



Obrázok 14 - QtC build project

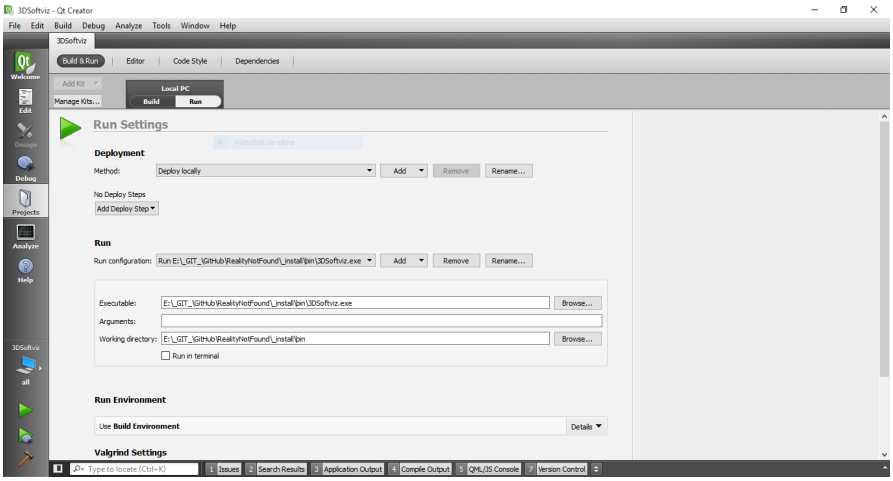
23. Skontrolovať nastavenie build config. – unity



Obrázok 15 - QtC build config

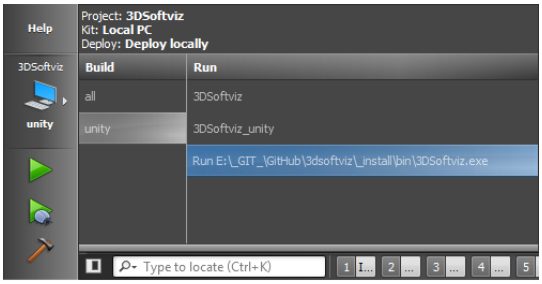
24. Stačiť Build (kladivko vľavo dole)
    a. ak došlo k erroru – *-Zm###*
        - CMakeLists.txt > riadok 128 zmeniť /Zm216 na /Zm240
        - spustiť CMake odznovu

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

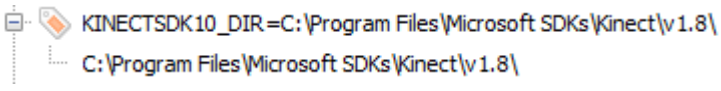


Obrázok 17 - QtC run config

27. Spustiť program pomocou zeleného tlačidla Run

5.2 Rozšírenie 3DSoftviz o Kinect

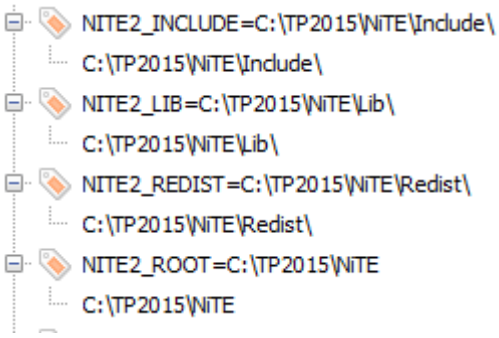
1. Nainštalovať Kinect for Windows
2. Skontrolovať v RapidEE či sa vytvorila premenná %KINECTSDK10_DIR%



Obrázok 18 - KINECTSDK10_DIR premenná

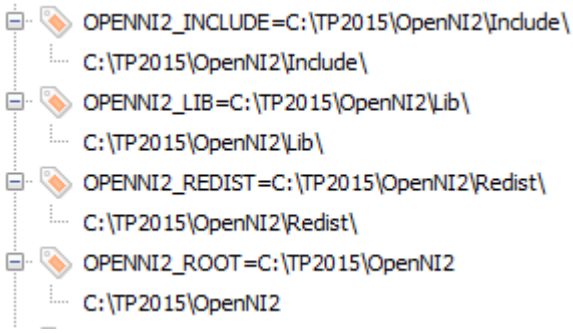
3. Nainštalovať OpenNI2 (*OpenNI-Windows-x86-2.2.msi*)
   - x86! – inak sa môžu vyskytnúť problémy s linkovaním
4. Skontrolovať v RapidEE či sa vytvorili premenné:
   a. %OPENNI2_INCLUDE%
   b. %OPENNI2_LIB%

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

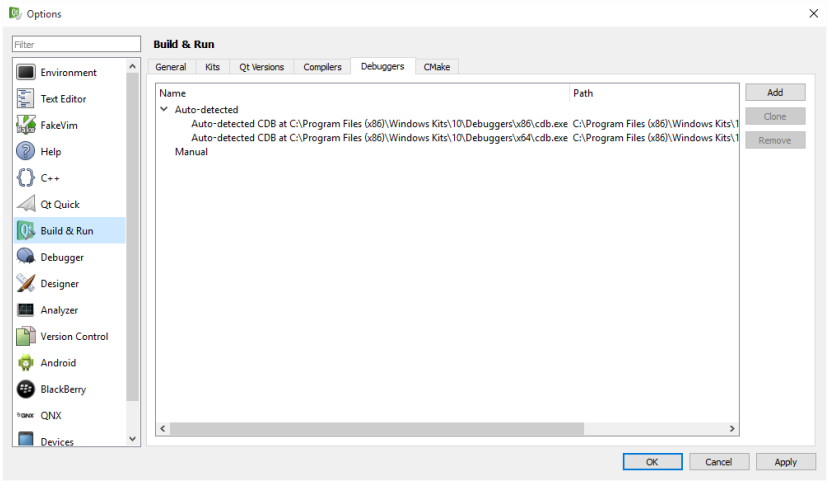
7. Pridať do premennej CMAKE_INCLUDE_PATH:
   a. %OPENNI2_INCLUDE%
   b. %NITE2_INCLUDE%
8. Pridať do premennej CMAKE_LIBRARY_PATH:
   a. %OPENNI2_ROOT%\Driver
   b. %OPENNI2_REDIST%
   c. %OPENNI2_REDIST%\OpenNI2\Drivers
   d. %OPENNI2_LIB%
   e. %NITE2_ROOT%\Samples\Bin\OpenNI2\Drivers
   f. %NITE2_LIB%
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 Návod na nastavenie debuggera 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)

5.4 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.