Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológií

Ilkovičova2, 842 16 Bratislava 4

AugReality

Dokumentácia projektu - riadenie

Vedúci tímu: Ing. Kapec Peter Phd.

Členovia tímu: Bc. Filip Škultéty,

Bc. Peter Belai,

Bc. Martin Mokrý,

Bc. Lukáš Hagara,

Bc. Patrik Berger,

Bc. Marek Roštár,

Bc. Michal Galbavý

Akademický rok: 2016/2017

Obsah

1. Úvod

Kapitola riadenie projektu opisuje všetky procesy, ktoré boli použité pre efektívnejšiu prácu tímu Aug_RealityKings počas tímového projektu. Obsahuje využívané metodiky, pravidlá, postupy, ktorými sme sa riadili. Z dôvodu, že na projekte pokračujeme po minuloročných tímoch/študentoch niektoré časti riadenia sme prebrali, niektoré upravili/doplnili a niektoré vytvorili nové.

V kapitole 1.1 je zoznam kapitol s uvedením autora, ktorý danú kapitolu vypracoval. Určenie a popis rolí sa nachádza v kapitole 0. Každému členovi boli určené viaceré role a zodpovednosti. V kapitole 2 je opis aplikovaných metodík, ktorými sa tím riadil. 3. kapitola sumarizuje doterajšie šprinty a úlohy. V 4. kapitole je stručný opis používaných metodík, ktorými sme sa riadili. V 5. kapitole je globálna retrospektíva zimného semestra, ktorá sumarizuje doterajšie šprinty.

1.1. Podiel práce na dokumentácií k riadeniu projektu

Kapitola	**Autor/autori**
Úvod	Filip Škultéty
Role členov	Filip Škultéty
Aplikácie manažmentov	Filip Škultéty, Martin Mokrý
Sumarizácia šprintov	Filip Škultéty, Marek Roštár
Používané metodiky: TFS	Filip Škultéty
Používané metodiky: Astyle	Martin Mokrý
Používané metodiky: Cpplint	Martin Mokrý
Používané metodiky: Cppcheck	Martin Mokrý
Používané metodiky: Logovanie	Peter Belai
Používané metodiky: Testovanie	Lukáš Hagara
Používané metodiky: GitFlow	Patrik Berger
Globálna retrospektíva zimného semestra	Celý tím
Export úloh počas šprintov	Filip Škultéty
Tvorba stránky	Martin Mokrý

Percentuálny podiel práce šprint č.1

Meno	Počet hodín [h]	Percentuálny Podiel [%]
Filip Škultéty	8	15.09
Peter Belai	6	11.32
Martin Mokrý	10	18.87
Lukáš Hagara	8	15.09
Patrik Berger	9	16.98
Marek Roštár	12	22.64
Michal Galbavý	0	0

Percentuálny podiel práce šprint č.2

Meno	Počet hodín [h]	Percentuálny Podiel [%]
Filip Škultéty	13	16.25
Peter Belai	8	10.00
Martin Mokrý	20	25.00
Lukáš Hagara	14	17.50
Patrik Berger	15	18.75
Marek Roštár	10	12.50
Michal Galbavý	0	0

Percentuálny podiel práce šprint č.3

Meno	Počet hodín [h]	Percentuálny Podiel [%]
Filip Škultéty	8	7.21
Peter Belai	35	31.53
Martin Mokrý	26	23.42
Lukáš Hagara	15	13.51
Patrik Berger	20	18.02
Marek Roštár	7	6.31
Michal Galbavý	0	0

Percentuálny podiel práce šprint č.4

Meno	Počet hodín [h]	Percentuálny Podiel [%]
Filip Škultéty	10	11.49
Peter Belai	14	16.09
Martin Mokrý	24	27.59
Lukáš Hagara	21	24.14
Patrik Berger	10	11.49
Marek Roštár	8	9.20
Michal Galbavý	0	0

1.2. Role členov tímu a podiel práce

Členovia tímu nadobúdali rôzne role počas semestra, dynamicky pribúdali povinnosti a zodpovednosti členov.

Marek Roštár

Marek je linux integrátor a tester. Keďže projekt je multiplatformový a Marek je jediný člen tímu, ktorý pracuje s operačným systémom Linux, nikto iný z tímu sa nehodil na túto pozíciu viac než on. Úloha linuxového integrátora zahŕňa: testovanie kompatibility implementovaných častí, podpora ovládačov.

Peter Belai

Peter je študentský vedúci tímu. Počas ZS sa venoval tvorbe dátových sad pre ostatných študentov fakulty, ktorý pracujú/budú pracovať s vizualizáciou dát. Dátové sady získaval pomocou zariadenia kinect 1 a snaží sa riešiť získavanie dát zo zariadenia kinect 2.

Filip Škultéty

Filip je manažér dokumentácie. Stará sa o vytváranie exportov zo softvéru na manažment riadenia tímu a generovanie technickej dokumentácie.

Martin Mokrý

Martin je zodpovedný za spravovanie webovej stránky, aktualizovanie obsahu, nahrávania nových dokumentov. Počas šprintov 1-3 sa venoval štruktúre programu a tvorbe analýzy nových častí.

Patrik Berger

Patrik spravuje spoločný git repozitár. Spolu s Martinom Mokrým navrhujú architektúru nových častí systému, konkrétne zobrazenie modelov rúk v scéne. Spravovaním úloh na TFS prispieva k manažmentu úloh.

Lukáš Hagara

Lukáš je manažér vývoja a testovania.

Michal Galbavý

Opustil tím v treťom týždni zimného semestra. Nevykonával žiadnu rolu.

2. Aplikácie manažmentov

V kapitole je bližší opis aplikovania manažmentov.

Manažment komunikácie

Na komunikáciu v tíme používame:

- Stretnutie tímu podľa rozvrhu - v ZS semestri stretnutia prebiehali každý štvrtok od 10:00 do 13:00.
- Slack - Mimo tímových stretnutí komunikujeme pomocou nástroja Slack, v ktorom sme vytvorili samostatné sekcie:
    - #announcments – dôležité oznamy,
    - #dokumentácia – výstupy analýz, návrhy modelov tried, dokumentácia k riadeniu a ing. Dielu a zápisnice,
    - #manažment – organizačné záležitosti (napr. sumarizácia odpracovaných hodín, náhradné termíny stretnutí),
    - #stranka – tvorba a aktualizácia webovej stránky,
    - #vyvoj – zdieľanie tutoriálov, knižníc a diskusia k pull requestom.

Tvorba dokumentácie

Dôležitou súčasťou softvéru je dokumentácia, ktorú treba neustále dopĺňať a aktualizovať. Metóda SCRUM, ktorou sa riadime, vyžaduje taktiež tvorbu dokumentácie. Počas projektu sme vytvorili alebo dopĺňali nasledovné dokumenty:

- Motivačný dokument
- Zápisnice zo stretnutí
- Exporty zo softvéru na manažment úloh, konkrétne TFS
- Retrospektívy na konci šprintov
- Dokumentácia k riadeniu
- Dokumentácia k inžinierskemu dielu
- Inštalačná príručka
- Technické dokumentácie (sphinx, doxygen, latex)

Manažment verzií

Pred implementáciou novej funkcionality, si vytvárame novú vetvu produktu. Vetvenie zmien, funkcionalít umožňuje vrátenie k predchádzajúcim verziám a opravu chýb. Pre správu verzií používame Git. Pri vývoji kladieme dôraz aby boli zmeny komitované do prislúchajúcej vetvy. Keď je funkcionalita hotová, vytvárame pull request, po ktorom členovia tímu testujú a vyjadrujú sa k novej funkcionalite a navrhujú schválenie alebo prerobenie.

Proces plánovania úloh

Riadime sa metodikou SCRUM, v ktorej šprint trvá 2 týždne. Na začiatku, respektíve na konci predchádzajúceho šprintu sa plánujú úlohy, ktoré sa budú v danom šprinte vykonávať. Plánovanie úloh zahrňuje ohodnotenie úloh, určenie podmienok, za ktorých sa úloha vyhodnotí ako dokončená a priradenie priorít úlohám. Po vytvorení a nahodení úloh do TFS, prebieha prideľovanie úloh, kedy si každý člen vyberie úlohu/úlohy, za ktorú bude zodpovedný. Na stretnutí po prvom týždni šprintu sa diskutuje pokrok v danom šprinte a riešia sa vyskytnuté problémy. Na zaznamenávanie stavu úloh

používame nástroj Team Foundation Server 2015(skrátene TFS). Bližšie informácie o používaní nástroja TFS sa nachádzajú v metodike TFS.

Proces hodnotenia úloh

Úlohy hodnotíme na začiatku šprintu. Úlohy hodnotíme hodnotami 1, 2, 3, 5, 8, 13, 20 ,40, 100 a ?. Ak má úloha hodnotenie viac ako 13, rozdelí sa na viacero úloh, aby hodnotenie bolo menšie 13. Ak tím vyhodnotí úlohu rôznymi hodnotami, prebehne diskusia, kde členovia vyjadria, prečo ohodnotili úlohu daným číslom. Cyklus sa opakuje pokým sa tím nezhodne v hlasovaní. Následne vlastník produktu stanoví priority úlohám.

Manažment kvality

Nás vlastník produktu je nás vedúci tímu, ktorému na stretnutí prezentujeme dosiahnutý progres a výstupy úloh. Vyjadrenia, poznámky a pripomienky zapisuje zapisovateľ stretnutia. Pre zachovanie kvality kódu používame nástroje Cpplint, Cppcheck a na jednotné formátovanie zdrojového kódu používame Astyle. Pred dokončením úlohy, je potrebné opraviť upozornenia a chybové hlášky z vyššie spomenutých programov, aby sa mohla úloha vyhodnotiť ako dokončená.

Správa webovej stránky

Na začiatku bolo potrebné vytvoriť tímovú webovú stránku, ktorá sa každý týždeň aktualizuje v závislosti od vytvorených dokumentov. Počas semestra je web správca zodpovedný za udržiavanie obsahu stránky. Keď člen tímu vytvorí dokument, ktorý je určený na publikovanie, pošle tento dokument správcovi webu.

Proces testovania

Testovanie je v našom projekte dôležitou súčasťou. Po programovaní, refaktorizácií , musí prejsť projekt procesom testovania. Počas programovania projektu sa vytvorili dvojice, v ktorých si členovia navzájom kontrolujú a testujú program. Každá nová funkcionalita musí byť otestovaná na Linux aj Windows.

3. Sumarizácie šprintov

3.1. Začiatočná fáza

V začiatočnej fáze sme sa oboznamovali s projektov. Aktuálny stav projektu je výsledkom viacerých tímových, diplomových, bakalárskych prác a preto sme úvodne chvíle venovali zorientovaniu sa v stave projekte. V druhom týždni ZS sme sa venovali inštalácií potrebných programov, ovládačov a vykonali sme prvé kompilácie a spustenie programu. Začali sme používať nástroj na manažment úloh – TFS.

3.2. Šprint č. 1

Na začiatku šprintu č. 1 sme diskutovali s vedúcim tímu o prvých úlohách. Prvý šprint sa niesol v znamení metodík- študovanie, upravovanie a vytváranie nových metodík, ktoré nám budú pomáhať efektívne fungovať ako tím. Medzi hlavné úlohy patrilo vytvorenie metodiky ku kvalite, gitflow, TFS, code review, logovaniu, testovaniu. V prvom šprinte prebiehalo aj testovanie existujúcich funkcionalít systému. Vlastníkovi produktu sme ukázali výsledky úloh, ku ktorým sa vyjadril.

Retrospektíva

Na záver šprintu sme diskutovali, ktoré veci treba zlepšiť. Problémom prvého šprintu bolo, že sme sa neriadili všetkými metodikami, lebo sme nemali zosynchronizované dokumenty s metodikami.

3.3. Šprint č. 2

Počas druhého šprintu sme sa hlbšie zoznamovali so zdrojovým kódom projektu. Opravovali sa upozornenia z kompilátora, cppchcek a cpplint, ostatné úlohy sú znázornené na [**Error! Reference s ource not found.**]. Na konci šprintu sme predviedli vlastníkovi produktu aktuálny stav a výstupy jednotlivých úloh.

Retrospektíva

Počas retrospektívy šprintu č.2 sme spoločne identifikovali nasledovné nedostatky, ktoré sa budeme snažiť v budúcich šprintoch zlepšiť:

- Pri oprave upozornení z kompilátora, cppchek a cpplint dvojice nemali zosynchronizované vetvy. Keďže všetky tri úlohy boli navzájom naviazané, synchronizácia by výrazne zefektívnila overovanie opráv upozornení.
- Počas plánovania úloh sme zabudli ohodnotiť úlohy, čo spôsobilo, že sa nám nepodarilo vygenerovať burndown chart pre šprint 2.

3.4. Šprint č. 3

Počas tretieho šprintu sme robili analýzu možností pre markerless kižnicu, vytvárali sme dokumentáciu, vytvárali sme model ruky pre Leap a začali sme s integráciou Kinectu2. Ostatné úlohy sú zobrazené na . Na konci šprintu sme predviedli vlastníkovi produktu aktuálny stav a výstupy jednotlivých úloh.

Retrospektíva

Počas šprintu č. 3 sme identifikovali nasledovné nedostatky, ktoré sa budeme snažiť v budúcich šprintoch zlepšiť:

- Súbory nevytvárať v Qtcreatore a includovať ich tak ako sa volajú so správnymi veľkými a malými písmenami.
- Pull requesty by sa mali riešiť rýchlejšie, kedže v tomto bode sa niektoré neriešili niekoľko dní.
- Spustiť default build pred pull requestom, na kontrolu errorov a warningov.

- Preberať cudzie úlohy čo sa nestíhajú.

3.5. Šprint č. 4

Počas tretieho šprintu sme riešili ďalej zobrazenie modelu ruky pre Leap a integráciu Kinectu2. Tiež sme zisťovali vhodnosť a pridávali do projektu knižnice VISP a PCL. Ostatné úlohy sú zobrazené na . Na konci šprintu sme predviedli vlastníkovi produktu aktuálny stav a výstupy jednotlivých úloh.

Retrospektíva

Počas šprintu č. 4 sme identifikovali nasledovné nedostatky, ktoré sa budeme snažiť v budúcich šprintoch zlepšiť:

- Nedostatočné dodržiavanie metodík.
- Komunikácia ohľadom TP aj na facebooku, čo spôsobuje zlú dohľadateľnosť informácii.

3.6. Šprint č.5

Ukončovali sme práce na záverečných úlohach zimného semestra. Prebiehalo hromadné mergovanie funckionalít z bakalárskych, diplomových prác.

3.7. Šprint č. 6

Začiatok letného semestra odštartoval šprint č.6. Povedali sme si ciele na letný semester. Rozdelili sme si úlohy a dohodli sa manažmente počas letného semestra. Odúshlasili sme zrušenie zápisníc zo stretnutí. Začali sme pracovať na prenositeľnej verzii programu, aby sa dal spustiť na tablete bez nutnosti kompilácie.

3.8. Šprint č. 7

Pokúsime sa zlepšiť:

- Includovanie headrov
- Dlhá schvalovacia doba pull requestov
- Dlhá schvalovacia doba pull requestov

3.9. Šprint č. 8

Neidentifikovali sme žiadne dobré a zlé vlastnosti počas tohto šprintu.

3.10. Šprint č. 9

Veci, ktoré sme nerobili správne:

- Dlhá schvalovacia doba pull requestov
- Nedostatočná dokumentácia zdrojového kódu
- Meškanie členov na tímové stretnutie

Veci, ktoré sme robili správne:

- Organizácia náhradných tímových stretnutí

3.11. Šprint č. 10

Veci, ktoré sme robili správne:

- Rozdelovanie úloh
- Komunikácia v tíme

Veci, ktoré sme nerobili správne:

- Nedostatočná dokumentácia zdrojového kódu
- Meškanie členov na tímové stretnutie
- Používanie nevhodných namespacov

Veci, ktoré sme robili správne:

- Organizácia tímových stretnutí
- Komunikácia v tíme

4. Používané metodiky

Počas projektu používame nasledovné metodiky:

- Ako programovať v C++ - Ako písať headre a cpp súbory
- CppCheck – Analýza cpp kódu
- Cpplint – c++ style kontrola
- Astyle – formátovanie a kontrola c++ kódu
- CodeReview – Kontrola vytvoreného kódu
- GitFlow – práca s gitom (vetvy, pull requesty, commit správy).
- Tvorba a údržba UML diagramov prostredníctvom PlantUML
- Sphinx – práca s nástrojom sphinx (tvorba html, latex dokumentácie)
- TFS – manažment úloh
- Logovanie s knižnicou EasyLogging++ - logovanie záznamov, chýb, pomocné výpisy
- Písanie BDD testov

5. Globálne retrospektívy

5.1. Globálna retrospektíva ZS

Prvé týždne semestra sme sa oboznamovali s projektom a pripravovali si softvérové prostredie pre ďalší vývoj. Po inštalovaní softvérov sme sa začali riadiť existujúcimi metodikami a chýbajúce metodiky (k riadeniu úloh, cpplint, cppcheck, style, logovanie, písanie testov) sme vytvorili. Počas semestra sa vyskytli problémy s nedodržiavaním metodík, hlavne s gitflow metodikou. V neskorších fázach ZS, sme sa na základe vyskytnutých problémov dohodli, že na písomnú komunikáciu ohľadom projektu budeme používať výlučne slack (žiadne iné komunikačné kanály). Ďalej sme sa dohodli na vytváraní .cpp /.h mimo IDE (nie je schopné vytvárať názvy súborov vhodné pre triedy) a potrebe rýchlejšej reakcie na pull requesty.

5.2. Globálna retrospektíva LS

Veci, ktoré sme nerobili správne:

- Includovanie headrov
- Dlhá schvalovacia doba pull requestov
- Nedostatočná dokumentácia zdrojového kódu
- Meškanie členov na tímové stretnutie
- Používanie nevhodných namespacov

Veci, ktoré sme robili správne:

- Rozdeľovanie úloh

- Plnenie plánu
- Organizácia tímových stretnutí
- Komunikácia v tíme

6. Motivačný dokument

6.1. Tím

Náš tím sa skladá zo 7 ľudí. Sú to Michal Galbavý, Peter Belai, Patrik Berger, Marek

Roštár, Lukáš Hagara, Martin Mokrý a Filip Škultéty. Všetci sme absolventi bakalárskeho štúdia na FIIT STU. Vďaka tomu sme nadobudli znalosti z rôznych technológií, ako je napr. Java, C. Časť tímu má taktiež skúsenosti s funkcionálnym programovaním alebo spracovaním obrazu a počítačovej grafiky. Väčšina členov tímu má taktiež skúsenosti, či už zo školských projektov, bakalárskych prác alebo zo zamestnania, s webovými technológiami HTML, CSS, JavaScript a PHP. Dvaja členovia z tímu majú taktiež skúsenosti s webovým frameworkom django. Ako tím máme široký záber rôznych technológií, ktoré sme nadobudli pri štúdiu alebo pri práci, takže členovia tímu ovládajú mnoho rozličných technológií. Medzi ne patria napr. jazyky python, R, C++, Lisp, Prolog, RESTové služby, XML, XSLT, UML, xPath a framework Spring. Na inžinierskom štúdiu sme si, k nami najviac preferovaným témam, zvolili predmety: vyhľadávanie informácií, vizualizácia dát a objavovanie znalostí.

6.2. Motivácia - Extrakcia dát z webu [WebExtraction]

Túto tému sme si zvolili ako prioritnú z viacerých dôvodov. Jedným z nich je možnosť preukázania nášho riešenia nad štátnymi dátami. Tie sú pre náš tím zaujímavé, pretože sa vďaka nim dá odhaliť mnoho rôznych súvislostí. S tým má už jeden člen tímu, Peter Belai, skúsenosti. Tie nadobudol pri hackatone, ktorý bol zameraný práve na štátne dáta. Tento hackaton taktiež vyhral. Ďalšou skúsenosťou je práca pre Alianciu Fair Play, kde extrahoval štrukturované dáta zo štátnych webov. Okrem týchto skúseností je pre nás motivácia naučiť sa pracovať s cloudovými technológiami. V neposlednej rade by sme radi pracovali na tejto téme z dôvodu dobrých recenzií na vedúceho pre danú tému. Do projektu by sme mohli priniesť viaceré skúsenosti. Časť tímu nadobudla skúsenosti s xPathom a XML schemou, či už z predmetu webové publikovanie alebo z pracovných skúseností, kde sa upravovali XML súbory v docx súboroch. Mimo iného má veľká časť tímu skúsenosti s webovými technológiami. Tieto sme nadobudli tiež z práce, kde sme vytvárali webové stránky, napr. aj e-Shopy, alebo aj zo školských projektov. S ruby on rails sme sa doposiaľ nestretli, ale sme otvorení novým príležitostiam naučiť sa túto technológiu

6.3. Motivácia - Inteligentný sklad [SmartStore]

Jedna z charakteristických vlastností tejto témy, ktorá nás zaujala je práca so skutočným zákazníkom. Komunikácia s ním, spracovanie požiadaviek a možnosť voľne zvoliť webové technológie je to čo nás primárne oslovilo. Taktiež vidíme možnosť využiť metody strojového učenia, ktoré sú z nášho pohľadu veľmi zaujímavou oblasťou informatiky.

Členovia nášho tímu majú skúsenosti s tvorbou rozsiahlych webových aplikácií, vytváraním e-shopov a v rámci bakalárskych prác sa vo väčšom rozsahu venovali genetickým algoritmom, heuristikám a klasifikácii. Niektorí členovia majú zapísaný predmet Objavovanie znalostí, ktorý by mohol byť

prínosom pri riešení tejto témy. Po krátkom brainstormingu sme identifikovali ďalšie vlastnosti tovaru, ktoré by bolo vhodné sledovať:

- rok vydania tovaru (nové vydanie, staré tituly),
- úspešnosť autora knihy, na základe predchádzajúcich diel,
- žáner knihy,
- zľava na daný tovar,
- predajnosť cez sviatky, víkendy, prac. dní a ročné obdobia,
- udalosti v živote autora.

Ako sme vyššie spomínali systém by fungoval s využitím strojového učenia. Toto strojové učenie by sme obohatili o rôzne heuristiky. Tieto by mohli byť následne upravované genetickým algoritmom, ktorý by prispôsoboval hodnoty parametrov. V riešení by sme chceli optimalizovať inteligentný sklad v 3 úrovniach:

- Tovar v sklade (optimálny počet produktov v sklade)
- Efektívne doplnenie tovarov do jedného skladu(správne naplánovať počet kusov tovaru, dátum, využitie dodávacieho vozidla a pod.)
- Efektívne doplnenie tovarov do viacerých skladov

6.4. Motivácia - Vizualizácia informácií v obohatenej realite [AugReality]

Táto téma sa radí medzi naše prioritné z viacerých dôvodov. Zaujíma nás uľahčovanie hľadania súvislostí s využitím rozličných metod vizualizácie. Túto tému vnímame ako výzvu, ktorá spočíva v práci s neprebádanými technológiami. Veríme v potenciál objavenia nových prístupov pri práci s vizualizačnými technológiami. Časť nášho tímu bola motivovaná k výberu tejto témy prednáškou docenta Kapca o vizualizácií na predmete ICP. Viacerí členovia absolvovali počas bakalárskeho štúdia predmet PPGSO a tak majú skúsenosti s programovacím jazykom C++ a knižnicou OpenGL. Ďalší z členov pracoval pri bakalárskej práci s komerčnými senzormi. Následne máme členov, ktorí majú skúsenosti s alternatívnymi spôsobmi ovládania počítača. Týmito spôsobmi sú konkrétne ovládanie hlasom a mysľou. Veríme, že by sme ovládanie gestami mohli týmto spôsobom doplniť a potenciálne aj vylepšiť

Poradie nami zvolených tém:

1. WebExtraction

2. SmartStore

3. AugReality

4. EduSim

5. CodeCrutches

6. DronSim V2

7. Story Teller

8. DeepSearch

9. VirtualFEI

10. 3D futbal

7. Metodiky

V projekte nadväzujeme na prácu predchádzajúcich študentov, preto sme niektoré metodiky prebrali z minulých rokov a niektoré sme si vytvorili alebo čiastočne upravili pre tohtoročné potreby.

7.1. Ako programovať v C++

Ako písať headre a cpp súbory

Includuje sa v tomto poradí (platí pre .h aj .cpp):

- headre z projektu
- headre z Qt a OSG
- systémové headre

v .cpp je prvý header prislusne .h-cko toho .cpp

Important: pravidlo: includujem najprv tie, ktoré môžu includnut čo najviac

Pravidla pre pisanie .h

• obsahujú iba definície metod, NIE implementaciu metod

• includujeme LEN čo je treba pre header, NIC naviac

• ak je v triede/metode použitý pointer typ, napr. Node* n;

tak stačí použiť doprednú deklaráciu, t.j. class Node;

a netreba includovať Node.h

Attention: toto nefunguje, keď:

• sa dedí trieda

• to nie je pointer, t.j. je to Node n;

• keď je použitý osg::ref_ptr pointer, resp. templaty

Pravidla pre písanie .cpp

ak je typ premennej použitý LEN v .cpp (typicky lokálna premenná v metode), tak príslušný header dávame len do .cpp

ZLE praktiky, resp. čo nerobiť

NIKDY v headroch a cpp súboroch nepoužívať “using namespace”

• nepoužívať “using” keyword

ak už je použité, tak to treba upraviť na :

```
namespace XYZ {
... implementacia ...
}
```

NEPOUŽÍVAŤ “0” (nulu) ako NULL pointer, ale

• použiť nullptr (ak kompilator podporuje C++11), pripadne použiť NULL

DOBRE praktiky, resp. čo robit

inicializovať VŠETKY atributy “pri” konštruktore cez “initialization list”

• najme pointre

– skontrolovať ak je “new” v konštruktore, tak MUSI byt “delete” v deštruktore (neplatí pre: osg::ref_ptr)

• inicializovať v takom poradí v akom sú zapísané v triede:

– usporiadať atributy od “najväčších” (napr. pointer, trieda, double) po “najmenšie” (int, char, bool)

aj keď sa bude mixovať public/private

“std::cout”

• pokiaľ je v kóde, ktorý používa Qt, tak prerobiť na qDebug

• resp. nájsť a používať nejakú externú knižnicu na logovanie

Ak sa použije cudzí existujúci kód tak, že sa priamo jeho zdrojáky pridajú do existujúceho projektu

• tak treba aby zostal pôvodný a úpravy sa riešili napr. v zdedenej triede

WARNING-y - opravujú sa všetky warningy v našom kóde (minimalizácia možných problémov)

pravidelne sa robí statická analýza kódu pomocou nástrojov cppcheck a cpplint

• vid. metodiky ccpchceck, cpplint

pravidelne sa zdrojový kód formátuje pomocou nastroja astyle pravidelne sa mergujú všetky prace.

7.2. CppCheck

http://cppcheck.sourceforge.net

Použite cppcheck v 1.70

Ako používať

1. v Edit->Preferences nastaviť: General: check “Force checking all #ifdef configurations” Paths: pridať cestu do “RealityNotFound/include” Advanced: check “Show inconclusive errors” !!! toto hlási dosť false-positive, ALE občas nájde dôležité veci !!!

2. Check->Directory a vybrat “RealityNotFound/src”

poznámky k reportom: !!! ak je niečo nejasne, treba sa opýtať !!!

!!! ignorovať hlášky pre externý kód: !!! noiseutils, qtcolorpicker

Dôležité hlášky typu:

Technically the member function XYZ can be const.

• najmä funkcie typu “getter” mozu byt const, napr.:

```
int  getX ()  const {
return 1;
}
```

Technically the member variable XYZ can be const.

• treba skontrolovať!!!

The class ‘RestrictionsManager’ does not have a constructor.

• každá trieda by mala mat konštruktor (kompiler nám síce vytvorí default, ale... vid. nižšie)

‘class Type’ does not have a copy constructor which is recommended since the class contains a pointer to allocated memory.

• závisí, či sa “copy constructor” v kóde používa - nutná kontrola

The scope of the variable ‘gesto_hore’ can be reduced. Variable ‘gesto_hore’ is assigned a value that is never used.

• to je jasne

C-style pointer casting

• to je riešene aj cez cpplint v samostatnej hotfix branch

• v gcc da sa zapnúť -Wold-style-cast -> momentálne hlási veľmi veľa warningov

• je to pre čitateľnosť, ALE ma to svoje opodstatnenie

Member variable ‘Cube::at’ is not initialized in the constructor.

• všetky class variables by mali byt inicializovane v konštruktore cez “initialization list”

!!! najmä pointre !!! na NULL, resp. cez new (a delete v destructore)

Possible null pointer dereference: conn - otherwise it is redundant to check it against null. Possible leak in public function. The pointer ‘nodeTypeComboBox’ is not deallocated before it is allocated.

!!! treba skontrolovať - indikuje závažnú chybu !!!

Uninitialized variable: newGraph

!!! treba skontrolovať - indikuje závažnú chybu !!!

ostatne hlasky

• treba skontrolovať - a mali by sa opraviť \

7.3. CppLint

Použitý cpplint dostupny:
https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0CB8QFjAA&url=http%3A%2F%2styleguide.googlecode.com%2Fsvn%2Ftrunk%2Fcpplint%2Fcpplint.py&ei=Ii6pVO_bK8mAUZSVgSg&usg=AFQjCNGnSkFFrrX3TI Eh8vRmUDJQ&bvm=bv.82001339,d.d24

Je to Python script.

Ako používať (Linux)

pouzite testy su nastavene v CPPLINT.cfg dostupne testy sa vypisu: cpplint.py –filter=

spustit v include adresari: cpplint.py Aruco/* Core/* Data/* Importer/* Layout/* Math/* Network/* Network/executors/* OpenCV/* QOpenCV/* Speech/* Viewer/* Clustering/* Clustering/Figures/* Fglove/* Kinect/* Kinect/RansacSurface/* Manager/* Model/* OsgQtBrowser/* QOSG/* Util/*

2>&1 | tee report-include.txt

spustit v src adresari: cpplint.py Aruco/* Core/* Data/* Importer/* Layout/* Math/* Network/* Network/executors/* OpenCV/* QOpenCV/* Speech/* Viewer/* Clustering/* Clustering/Figures/* Fglove/* Kinect/* Kinect/RansacSurface/* Manager/* Model/* OsgQtBrowser/* QOSG/* Util/*

2>&1 | tee report-src.txt

Poznamky k reportom

!!! ak je niečo nejasne, treba sa opytat !!!

ignorovat hlášky pre externy kod: noiseutils, qtcolorpicker ciastocne pre cameramanipulator, QGraphicsViewAdapter

dolezite hlášky typu:

Constructors callable with one argument should be markeď explicit

• nastudovat, asi staci pridať keyword explicit

Is this a non-const reference? If so, make const or use a pointer

• nastudovat (ci to nieje false-positive)

Use int16/int64/etc, rather than the C type long

• nastudovat

Do not use namespace using-directives.

• odstranit “using namemspace” (okrem externeho/cudieho kódu) Consider using rand_r(...) instead of rand(...) for improved thread safety.

• keďze mame vlakna, asi by bolo vhodne. vid.: https://stackoverflow.com/questions/3973665/how-do-i-use-rand-r-and-how-do-i-use-it-in-a-thread-safe-way

Are you taking an address of a cast? This is dangerous: could be a temp var. Take the address before doing the cast, rather than after

!!! indikuje vazny problem !!!

flase-positive:

Clustering/Figures/Cube.h:28: Add #include <algorithm> for transform [build/include_what_you_use] [4]

Cube ma metodu tranform -> netreba include

3.2.2 ostatne hlasky:

• treba skontrolovať - a mali by sa opraviť (resp. už som ich opravil ;-)

7.4. Astyle

Ako pouzivat v QTCreatore (Windows)

1) Stiahni AStyle do priecinku s programmi tykajucim sa projektu $ASTYLE_PATH.

2) Pridaj cestu k astyle-u do systemovej premennej PATH: $ASTYLE_PATH/bin (napr. d:/timak/AStyle/bin)

3) Spusti QTCreator

4) Projects -> Build & Run -> Build 5) V Build Settings -> Edit Build configuration klikni Add -> clone selected a zadaj “style”

6) V Build steps rozklikni Details a v Targets oznac “style”. Ak su oznacene aj ine targety, tak ich je potrebne

odznacit(Vysledok: Make: jom.exe style). 7) Klikni na kladivko v lavom dolnom rohu (Build).

7.5. PlantUML

PlantUML je jednoduchý program na tvorbu UML diagramov prostredníctvom ich textového opisu. K samotnému programu prislúcha aj rozsiahla dokumentácia.

PlantUML je voľne dostupný na stiahnutie z oficiálnej stránky, prípadne je možné na otestovanie použiť aj jednoduchú web aplikáciu.

Pre plnohodnotné využitie je potrebné mať taktiež nainštalovaný Graphviz.

Tiež ponúka možnosť integrácie s viacerými textovými editormi a wiki stránkami.

Pravidlá pre tvorbu súborov

1. Každý diagram sa nachádza v samostatnom textovom súbore (koncovka .txt, resp .wsd pri použití integrácie so sublime text).
2. Vygenerovaný diagram má identický názov ako prislúchajúci textový súbor (koncovka .png).
3. Názvy súborov sú po anglicky.

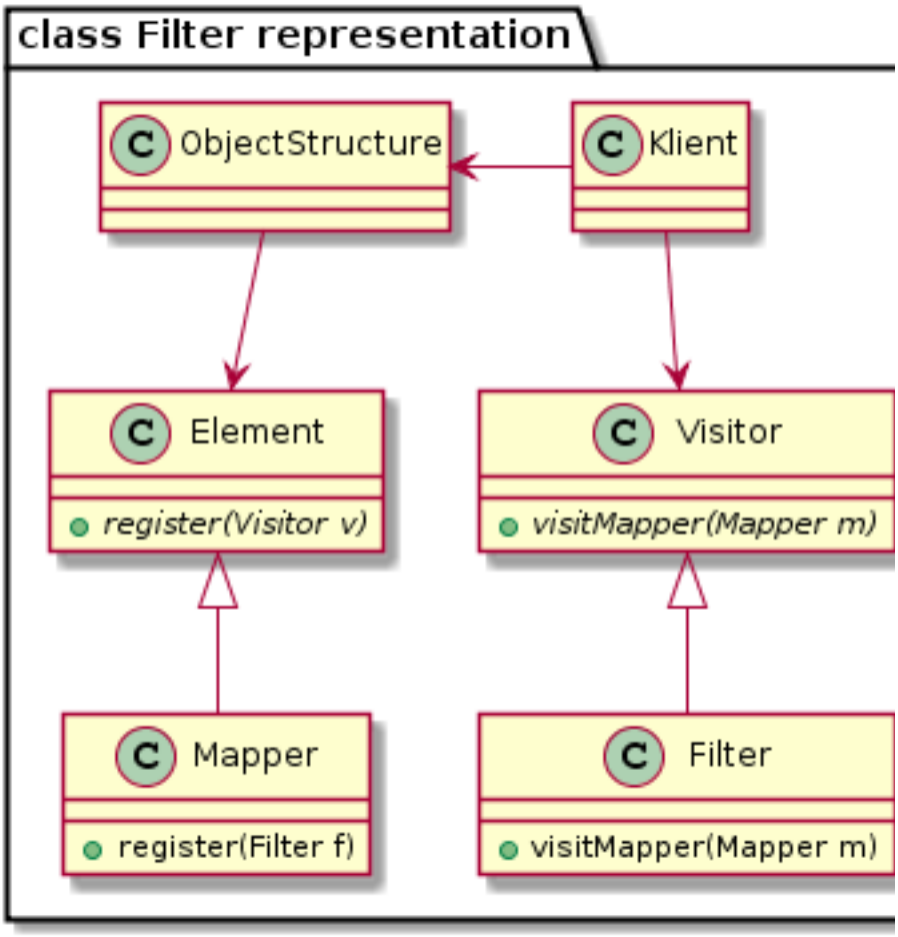
Odstránenie duplicity pomocou Preprocesoru

Pri písaní diagramov, ktoré obsahujú komplikované vzťahy medzi entitami môžeme naraziť na situáciu, kde budeme veľa krát za sebou písať ten istý názov triedy alebo metody. S využitím makier preprocessoru môžeme túto duplicitu ľahko odstrániť.

```
@startuml
'Bez proprocesoru
package "class Filter  representation" {
class  ObjectStructure class  Element {
+{abstract}register(Visitor  v)
}
class Mapper {
        +register(Filter f)

}

class Klient class  Visitor  {

        +{abstract}visitMapper(Mapper m)

}

class Filter {

        +visitMapper(Mapper  m)

}

ObjectStructure -down-> Element

Mapper  -up-|>  Element

ObjectStructure <-left-  Klient

Klient  -down-> Visitor

Filter  -up-|> Visitor

}

@enduml


@startuml

'S  preprocesorom

!define o(x) ObjectStructure

!define  e(x)  Element

!define  m(x) Mapper
```

```
!define k(x) Klient
!define v(x) Visitor
!define f(x) Filter

package "class Filter  representation" {
class o()
class e() {
+{abstract}register(v(x)  v)
class k()
class  v()  {
+{abstract}visitMapper(m(x) m)
}
class f() {
+visitMapper(m(x) m)
}
o()  -down-> e()
m() -up-|> e()
o() <-left- k()
k()  -down-> v()
f()  -up-|> v()
}
@enduml
```

V oboch prípadoch bude výsledok nasledovný:



V druhom prípade sa rozhodne menej napíšeme a máme možnosť meniť použité názvy tried na jednom mieste namiesto toho aby sme ich museli meniť všade. Stojí za poznámku, že každé definované makro musí mať parameter (v našom prípade x, z ktorého ajtak nečítame). Viac o Preprocesore na tejto stránke.

Použitie aliasov v sekvenčnom diagrame

V sekvenčných diagramoch odporúčame pri definovaní volaní medzi objektami používať aliasy (skratky). Ich princíp je analogický s predchádzajúcim makrom avšak sú ešte o niečo prehľadnejšie. Aliasy nie sú však podporované v class diagrame.

```
@startuml

participant  Client as c participant Server as s

title sd Basic Server call activate  c

c -> s:  sendMessage("You are  awesome!")

activate s

s --> c: result =  "ok" deactivate s
```

```
deactivate c

@enduml
```

Výsledok:

Užitočnosť týchto skratiek (a makier) pochopiteľne narastá s narastajúcou komplexnosťou daného diagramu.

Pravidlá pre súborovú štruktúru

Samotné UML diagramy je potrebné rozdeliť do prehľadnej súborovej štruktúry:

- projekt (názov projektu, napr. 3dsoftviz)
  - doc (inštalačná dokumentácia, vygenerovaná dokumentácia atd'.)
  - uml
    - structural
      - class diagrams (korešpondujúce s reálnym kódom)
      - component diagrams
    - behavioral
      - activity diagrams
      - use-case diagrams
      - sequence diagrams
      - state diagrams

7.6. GitFlow

Vetvy

- Master - hlavný projekt

• Develop - branchnutá z mastra, každý šprint má vlastnú Develop vetvu, na konci šprintu sa mergne späť do mastra

• Feature - branchnutá z developu, každý novy kus funkcionality (task v TFS), ktorý sa kódi musí mať 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 prepínacom –no-ff

Cheat sheet so všetkymi základnymi príkazmi:

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 mergovat.

Pull request sa robí z GUI GitHubu (pravý horný roh), alebo $ git request-pull {meno_commitu} {URL} (doporučujem robit cez GUI)

Po odsuhlásení 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 novu funkcionalitu, subor, ...

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

• [REF] - pre refactoring

• [FMT] - formátovanie textu, úprava

• [TEST] - pre testy

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

Useful commands

• $ git submodule update –init –recursive

– update submodulov (dependencies)

• $ git checkout -f meno_branch

– checkout branche aj napriek lokalnym zmenam, budu zahodene

• $ git status

– vypise všetky vykonane zmeny

• $ git stash / $ git stash pop

– ulozi stav projektu do stashu, z ktoreho sa da potom tento stav pop-nut, dobre na prenos zmien medzi vetvami

Tvorba feature branch-u

• $ 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, čo som spravil”

Uzatvorenie Hotfix branchu:

• $ git checkout develop //Switched to branch ‘develop’

• $ git merge –no-ff “hotfix/nazov-co-fixujem”

7.7. Sphinx dokumentácia

Pre prácu so Sphinxom treba mať nainštalovaný Python.

Python ponuka verzie 2.x a 3.x. Sphinx 1.3 môže bežať pod Python 2.6, 2.7, 3.3, 3.4, ale odporúčaná verzia je 2.7.

Pre stiahnutie a inštalovanie externých knižníc pre Python existuje príkaz pip. Príkaz už sa nachádza v oficiálnych verziách Pythonu 3.4.0 alebo 2.7.9.

Ak sa príkaz nenainštaloval automaticky, treba ho stiahnut zo stránky https://bootstrap.pypa.io/get-pip.py a niekam uložiť. V príkazovom riadku treba prejsť do adresára s get-pip.py a spustiť nasledovný príkaz:

```
python get-pip.py
```

Sphinx

• Prejsť do priečinku s dokumentáciou (tam kde index.rst sa nachádza) a pomocou príkazu pip nainštalovať Sphinx:

```
pip install  sphinx
```

• Ak treba vytvorit novu dokumentaciu, pre nastavenie zdrojoveho adresara a vytvorenie potrebnych suborov na prácu so Sphinx treba spustit prikaz:

```
sphinx-quickstart
```

a odpovedat na otazky. Vyberte si všetky predvolene odpovede a po vyzve zadajte nazov, autorov a verziu projektu.

• Tymto prikazom budu vygenerovane subory Makefile, make.bat a conf.py.in.

  – Všetky konfiguracie dokumentacie su v conf.py.in.

Upozornenie: Sphinx-quickstart a vytvaranie tychto suborov generuju novu dokumentaciu! Ak subory index.rst, Makefile, make.bat a conf.py.in uz existovali, tak sa prepisu!

• Sphinx dokumentacia generuje vystup v roznych formatoch zo suborov .rst.

HTML dokumentacia

• Subor make.bat povoli vygenerovat dokumentaciu v tom formate, ktory potrebujete

• Pre generovanie HTML dokumentacie treba v prikazovom riadku prejst do priecinku s ReST subormi a make.bat suborom a spustit prikaz

```
make   html
```

• Inak generovanie dokumentacie sa da spustit pomocou CMake v QtCreatore.

PDF dokumentacia

Pre generovanie PDF dokumentacie potrebujeme najprv vytvorit Latex dokumentaciu.

Note: Pre prácu s Latex treba mat TeXlive.

Prikazom

```
make   latex
```

vygeneruje sa Latex dokumentacia, ktora nasledne sa moze konvertovat do PDF pomocou programu TeXstudio.

PDF dokumentacia generuje len pomocou prikazoveho riadku a externeho programu, neda sa spustat cez CMake!

PlantUML

Pre prácu s PlantUML nastrojmi v Sphinx treba:

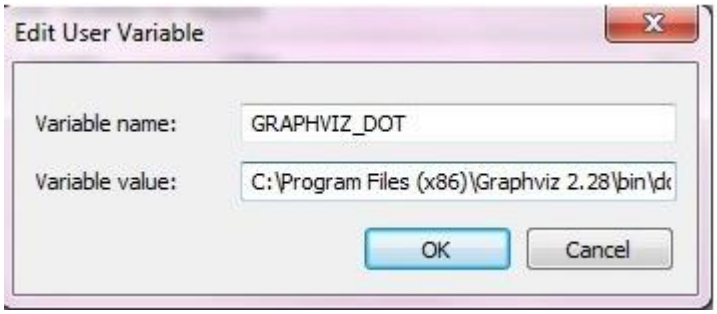
• nainstalovat Javu

• pridať Javu do premennych prostredi (environment variable)

• nainstalovat Graphviz

– odporúčaná verzia je 2.28

• pridať Graphviz do premennych. Hodnota premennej ma byt do dot.exe:



• pridať Graphviz do extensions v conf.py.in:

extensions = ['sphinx.ext.graphviz']

• nainstalovat sphinxcontrib-plantuml zo stranky alebo prikazom

pip install sphinxcontrib-plantuml

• pridať pantuml do extensions v conf.py.in:

extensions = ['sphinxcontrib.plantuml']

• stiahnut plantuml.jar

• pridať do conf.py.in prikaz:

plantuml = 'java -jar /cesta/do/plantuml.jar'

Attention: Dolezite je zmenit tuto cestu na spravnu, aku mate aktualnu na Vasom pocitaci!

• pridavat UML do dokumentacii je mozne pomocou

```
.. uml::

!include /cesta/do/subor.wsd(txt)
```

alebo

```
@startuml
```

PlantUML kod

@enduml

7.8. TFS

Metodika, ktorá určuje pravidlá a postupy na manažment úloh v nástroji TFS.

Pridanie novej úlohy - všeobecne

• Zadať ohodnotenie úlohy, priority, opis, Zadať keďy je úloha hotová.

• Zadať odhadovaný čas dokončenia.

• Opis musí byť podrobný, aby každému členovi bolo jasné, čo má vykonať po pridelení úlohy.

Rozdeľovanie úloh

• Každý si vyberie (potiahne) úlohu/úlohy, ktorá/é majú najvyššiu prioritu.

• Ak ostanú nepridelené úlohy, študentský vedúci tímu pridelí členom zvyšné úlohy.

TFS

• Adresa TFS: https://tfs.fiit.stuba.sk:8443/tfs/ (Potrebné sa prihlásiť 2x)
• Projekt: AugReality / AugReality Team

Pridanie nových úloh – TFS

• Úlohy podobného charakteru priradíme do spoločného backlog itemu.

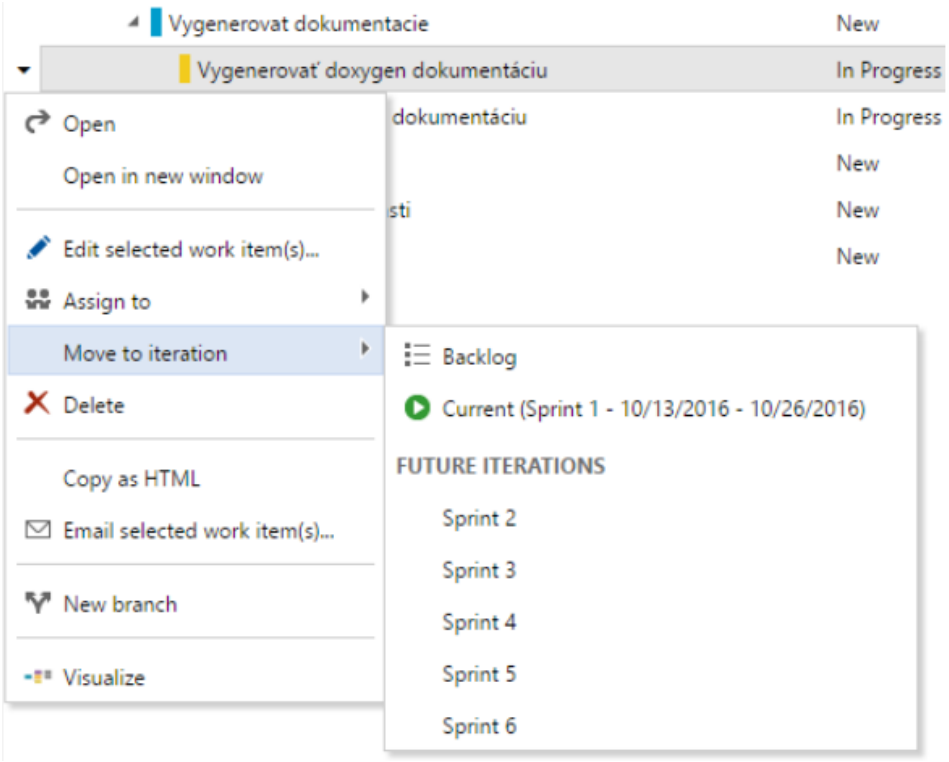
• Pri pridávaní úlohy sa automaticky nastaví stav ‘To Do’.

• V prípade objavenia chyby, je potrebné vytvoriť novú úlohu typu Bug (Chyba)

Úlohy (tasky) sa môžu nachádzať v troch stavoch:

- To Do
- In Progress
- Done

Nesplnené úlohy, ktoré sa nestihli dokončiť v danom šprinte, presunieme do nasledujúceho šprintu.



Obrázok 7.8.1 Presun úlohy do ďalšieho šprintu

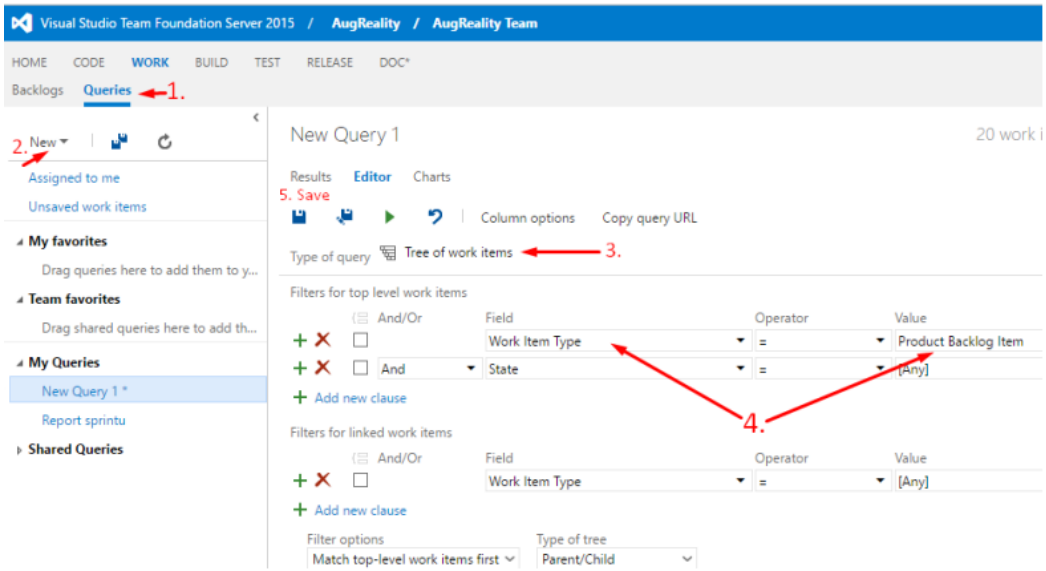
Vytváranie exportov z TFS

Navod na stranke: https://msdn.microsoft.com/en-us/library/dd286627(v=vs.110).aspx

URL: https://tfs.fiit.stuba.sk:8443/tfs/

Login: ako do AIS-u

Pred prvým exportom je potrebné si vytvoriť query, ktoré vráti stav úloh v danom šprinte. Návod na vytvorenie query:



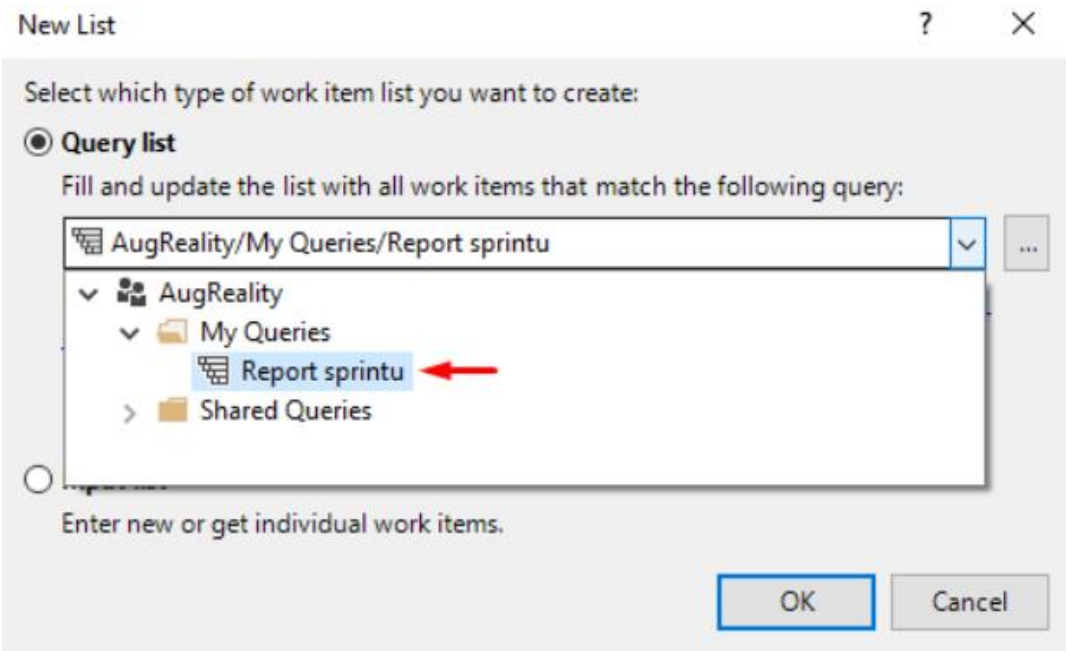
Obrázok 7.8.2 Vytvorenie query v TFS

Import do Excelu

• V hlavnom menu vybrať záložku TEAM (mala by byť vpravo hore).

• Klik na New List (umiestnená na ľavo pod záložkou File/Súbor).

• Vybrať novo vytvorené query:



Obrázok 7.8.3 Výber query v excely

7.9. Logovanie s knižnicou Easylogging++

Úrovne logovania

Easylogging++ poskytuje tieto úrovne logovania:

- Global - Generická úroveň, reprezentujúca všetky ostatné ˇ
- Trace - Zachytávame informácie, vhodné na back-trackovanie rôznych problémov.
- Debug - Zaznamenávame informácie, vhodné pri vývoji aplikácie.
- Fatal - Používame, keď" nastane chyba, ktorá pravdepodobne ukončí program. ˇ
- Error - Používame ,keď nastane závažnejšia chyba ale program bude naďalej pracovať.
- Warning - Používame keď nastane chyba ale program bude naďalej pracovať.
- Info - Používame na zachytenie priebehu aplikácie.
- Verbose - Nepoužívame.
- Unknown - Nepoužívame.

Čo a ako treba logovať

• Začiatok každej metody ˇ

• Všetky možné errory / warningy

Ako treba logovať:

• Začiatok každej metody logujeme v tvare :

function(){

LOG(INFO) << "MENO_BALIKU/MENO_TRIEDY/MENO_FUNKCIE(PARAMETRE)"

}

Pozor, parametre používať iba v prípade, že je to vhodné. Všetky možné errory / warningy logujeme v tvare:

*LOG(WARNING/ERROR/FATAL) << "MENO_BALIKU/MENO_TRIEDY/MENO_*

→FUNKCIE(PARAMETRE)"

V tomto prípade sa snažíme zalogovať aj všetky potrebné parametre, ktoré spôsobili warning / error.

7.10. Metodika na písanie BDD testov

1. Vytvorme normálny .cpp súbor a v hlavičke uvedieme:

```
#include <igloo/igloo_alt.h>

using namespace igloo;
```

2. Najskôr popíšeme, čo chceme testovať pomocou Describe-u:

```
Describe(a_foo_bar) {

...

}
```

3. V tele Describe-u uvedieme funkcie, ktorými budeme testovať spravanie. Funkcie pomenujme tak, aby boli samoopisné:

```
It( foo_should_be_bar ) {

...

}
```

4. V tele It-u už píšeme klasické asserty, napr:

```
Assert::That( 2+4, Is().EqualTo(6) );

Assert::That( "FIIT", Is().EqualTo("LIFE") );
```

Spustenie testov

1. Testy (všetky *.cpp) pre váš modul umiestňujte do priečinka:

```
~root/tests/Foo
```

Testy fungujú tak, že si buildnete vlastný .exe súbor, ktorý odtestuje danú funkcionalitu. Test si buildnete nasledovne:

- otvorte hlavný CMakeLists.txt
- ctrl+f -> “BDD Igloo tests”

2. Teraz si potrebujete nakopírovať nejaký kód, minimálne by ste mali mat:

```
file( GLOB_RECURSE SRC_FOO_TESTS "tests/Foo/*.cpp" )

add_executable( test_foo_module ${TEST_RUNNER} ${SRC_FOO_TESTS} )
```

add_dependencies(test_foo_module igloo)

target_compile_options(test_foo_module PUBLIC ${FLAGS_FOR_DEBUG})

Dole pod testami (alebo ctrl+f -> “run all tests”) pridajte ešte jeden riadok kódu:

add_test(testing_foo_module test_foo_module)

3. Teraz musíte zmeny uloziť a nechať zbehnúť CMake

• pravým na project -> Run CMake

• v QTCreatori zmeňte build/run target na Tests/test_foo_module

4. Pokiaľ už máte v priečinku tests/foo testy mali by sa buildnut/zbehnúť

8. Export evidencie úloh

V kapitole sú zobrazené exporty úloh.

8.1. Šprint č. 1, týždeň 1

Project: AugReality **Server:** tfs.fiit.stuba.sk\StudentsProjects **Query:** Report sprintu **List type:** Tree

ID	Work Item Type	Title 1	Title 2	Assigned To	State	Tags
4539	Product Backlog Item	Sprava metodik			New	
4540	Task		Metodika ku kvalite	Bc. Martin Mokry	Done	
4541	Task		Metodika gitflow	Bc. Patrik Berger	Done	
4542	Task		Metodika k TFS	Bc. Filip Skultety	Done	
4543	Task		Metodika ku code review	Bc. Patrik Berger	Done	
4544	Task		Metodika k logovaniu	Bc. Peter Belai	Done	
4545	Task		Metodika k testovaniu		To Do	
4546	Product Backlog Item	Otestovať existujuce casti			New	
4547	Task		Otestovať vuzix okuliare	Bc. Marek Rostar	To Do	
4548	Task		Otestovať web kameru, face detection	Bc. Marek Rostar	Done	
4549	Task		Otestovať kinect	Bc. Peter Belai	To Do	
4550	Task		Otestovať načítanie zo súborov	Bc. Lukas Hagara	In Progress	
4551	Task		Otestovať otáčanie grafov	Bc. Martin Mokry	To Do	
4552	Task		Otestovať funkčnosť aplikácie	Bc. Lukas Hagara	In Progress	
4555	Product Backlog Item	Fixnut problemy			New	
4556	Task		Opravit warning	Bc. Patrik Berger	Done	
4560	Task		Dokončiť build	Bc. Marek Rostar	Done	
4557	Product Backlog Item	Vygenerovat dokumentacie			New	
4558	Task		Vygenerovať doxygen dokumentáciu	Bc. Filip Skultety	To Do	
4559	Task		Vygenerovať sphinx dokumentáciu	Bc. Filip Skultety	In Progress	

Obr. 1 Export úloh šprint 1 týždeň 1

8.2. Šprint č. 1, týždeň 2

Project: AugReality **Server:** tfs.fiit.stuba.sk\StudentsProjects **Query:** Report sprintu **List type:** Tree

ID	Work Item Type	Title 1	Title 2	Assigned To	State
4540	Task		Metodika ku kvalite	Bc. Martin Mokry	Done
4541	Task		Metodika gitflow	Bc. Patrik Berger	Done
4542	Task		Metodika k TFS	Bc. Filip Skultety	Done
4543	Task		Metodika ku code review	Bc. Patrik Berger	Done
4544	Task		Metodika k logovaniu	Bc. Peter Belai	Done
4545	Task		Metodika k testovaniu	Bc. Lukas Hagara	Done
4547	Task		Otestovať vuzix okuliare	Bc. Marek Rostar	Done
4548	Task		Otestovať web kameru, face detection	Bc. Marek Rostar	Done
4549	Task		Otestovať kinect	Bc. Peter Belai	Done
4550	Task		Otestovať načítanie zo súborov	Bc. Lukas Hagara	Done
4551	Task		Otestovať otáčanie grafov	Bc. Martin Mokry	Done
4552	Task		Otestovať funkčnosť aplikácie	Bc. Lukas Hagara	Done
4556	Task		Opravit warning	Bc. Patrik Berger	Done
4560	Task		Dokončiť build	Bc. Marek Rostar	Done
4558	Task		Vygenerovať doxygen dokumentáciu	Bc. Filip Skultety	In Progress
4559	Task		Vygenerovať sphinx dokumentáciu	Bc. Filip Skultety	Done

Obr. 2 Export úloh šprint 1 týždeň 2

8.3. Šprint č. 2, týždeň 1

Project: AugReality **Server:** tfs.fiit.stuba.sk\StudentsProjects **Query:** Report Sprint 2 **List type:** Flat

ID	Work	Title	Assigned To	State	Tags
4558	Task	Vygenerovať doxygen dokumentáciu	Bc. Filip Skultety	Done	
4812	Task	Oprava warningov z kompilatora Windows	Bc. Patrik Berger	In Progress	
4813	Task	Oprava warningov z kompilatora Linux	Bc. Marek Rostar	In Progress	
4814	Task	Oprava warningov cppcheck	Bc. Filip Skultety	In Progress	
4815	Task	Oprava warningov cpplint	Bc. Lukas Hagara	In Progress	
4816	Task	3d mys zmena pouzitia	Bc. Marek Rostar	Done	
4820	Task	Analyza kniznic pre podporu K2 s freenect,openni	Bc. Peter Belai	In Progress	
4822	Task	Ozivit tlacitko kinect snapshot	Bc. Peter Belai	Done	
4823	Task	Opis Kinect use casov	Bc. Lukas Hagara	Done	
4824	Task	Vytvorit dataset via K1	Bc. Peter Belai	Done	
4840	Task	Zdokumentovat štruktúru grafu sceny	Bc. Martin Mokry	Done	
4852	Task	Vytvorenie diagramu tried 1	Bc. Patrik Berger	Done	
4853	Task	Vytvorenie diagramu tried 2	Bc. Martin Mokry	Done	

Obr. 3 Export úloh šprint 1 týždeň 1

8.4. Šprint č. 2, týždeň 2

Project: AugReality **Server:** tfs.fiit.stuba.sk\StudentsProjects **Query:** Report Sprint 2 **List type:** Flat

ID	Work	Title	Assigned To	State
4814	Task	Oprava warningov cppcheck	Bc. Filip Skultety	In Progress
4558	Task	Vygenerovať doxygen dokumentáciu	Bc. Filip Skultety	Done
4812	Task	Oprava warningov z kompilatora Windows	Bc. Patrik Berger	Done
4813	Task	Oprava warningov z kompilatora Linux	Bc. Marek Rostar	Done
4815	Task	Oprava warningov cpplint	Bc. Lukas Hagara	Done
4816	Task	3d mys zmena pouzitia	Bc. Marek Rostar	Done
4820	Task	Analyza kniznic pre podporu K2 s freenect,openni	Bc. Peter Belai	Done
4822	Task	Ozivit tlacitko kinect snapshot	Bc. Peter Belai	Done
4823	Task	Opis Kinect use casov	Bc. Lukas Hagara	Done
4824	Task	Vytvorit dataset via K1	Bc. Peter Belai	Done
4840	Task	Zdokumentovat štruktúru grafu sceny	Bc. Martin Mokry	Done
4852	Task	Vytvorenie diagramu tried 1	Bc. Patrik Berger	Done
4853	Task	Vytvorenie diagramu tried 2	Bc. Martin Mokry	Done

Obr. 4Export úloh šprint 2 týždeň 2

8.5. Šprint č. 3, týždeň 1

Project: AugReality **Server:** tfs.fiit.stuba.sk\StudentsProjects **Query:** Report Sprint 3 **List type:** Tree

ID	Work Item Type	Title 1	Title 2	Assigned To	State
4831	Product Backlog Item	Doplnit do projektu hole triedy - strukturu modelu			New
4931	Task		Doplnit do projektu hole triedy	Bc. Patrik Berger	Done
4833	Product Backlog Item	Vytvorenie zobrazenia ruk a jeho pridanie do grafu			New
5058	Task		Uprava vizualizacie - farby, velkost	Bc. Patrik Berger	Done
5063	Task		Implementacia logiky struktury - ru	Bc. Patrik Berger	Done
5064	Task		Implementacia logiky struktury - d	Bc. Patrik Berger	In Progress
5065	Task		Refaktoring kodu modelu.	Bc. Patrik Berger	Done
4838	Product Backlog Item	Analyza marker-less kniznic			New
4932	Task		Analyza marker-less kniznic	Bc. Lukas Hagara	Done
4916	Product Backlog Item	Rozsirit adapter leapu			New
4933	Task		Pozmenit strukturu ovladania	Bc. Martin Mokry	Done
5059	Task		Refaktoring kodu	Bc. Martin Mokry	In Progress
5060	Task		Pridat ovladanie pozicie klbov	Bc. Martin Mokry	Done
5061	Task		Pridat adekvatne rotacie kosti.	Bc. Martin Mokry	Done
5062	Task		Upravit skalovanie dlzky kosti.	Bc. Patrik Berger	Done
4919	Task	Mat kniznicu		Bc. Peter Belai	Done
4923	Product Backlog Item	Vytvorit dokumentaciu riadenia			New
4925	Task		Napisat uvod k riadeniu	Bc. Filip Skultety	Done
4926	Task		Napisat retrospektivu Sprint1,2	Bc. Filip Skultety	Done
4924	Product Backlog Item	Vytvorit dokumentaciu k ing. dielu			New
4927	Task		Napisat uvod k ing. dielu	Bc. Filip Skultety	Done
4928	Task		Spisat globalne ciele	Bc. Filip Skultety	Done
4929	Task		Vytvorit celkovy pohlad na system	Bc. Martin Mokry	Done
4934	Task		Popisat nami vytvarany modul 2	Bc. Martin Mokry	In Progress
4936	Task	Integreacia kniznice pod linuxom		Bc. Marek Rostar	Done
4998	Product Backlog Item	Osamostatnenie modulu LuaGraph			New
4999	Task		Odclenit kod modulu	Bc. Lukas Hagara	Done
5056	Product Backlog Item	Analyza QVR kniznice			New
5057	Task		Vytvorte podrobnu analyzu QVR kr	Bc. Lukas Hagara	Done

Obr. 5 Export úloh šprint 3 týždeň 1

8.6. Šprint č. 3, týždeň 2

Project: AugReality **Server:** tfs.fiit.stuba.sk\StudentsProjects **Query:** Report Sprint 3 **List type:** Tree

ID	Work Item Type	Title 1	Title 2	Assigned To	State
4814	Task	Oprava warningov cppcheck		Bc. Filip Skultety	In Progress
4831	Product Backlog Item	Doplnit do projektu hole triedy - strukturu modelu			New
4931	Task		Doplnit do projektu hole triedy	Bc. Patrik Berger	Done
4833	Product Backlog Item	Vytvorenie zobrazenia ruk a jeho pridanie do grafu			New
5058	Task		Uprava vizualizacie - farby, velkosti	Bc. Patrik Berger	Done
5063	Task		Implementacia logiky struktury - ruky zlozene z klbov	Bc. Patrik Berger	Done
5064	Task		Implementacia logiky struktury - doplnenie reprezentacie kosti	Bc. Patrik Berger	Done
5065	Task		Refaktoring kodu modelu.	Bc. Patrik Berger	Done
4838	Product Backlog Item	Analyza marker-less kniznic			New
4932	Task		Analyza marker-less kniznic	Bc. Lukas Hagara	Done
4916	Product Backlog Item	Rozsirit adapter leapu			New
4933	Task		Pozmenit strukturu ovladania	Bc. Martin Mokry	Done
5059	Task		Refaktoring kodu	Bc. Martin Mokry	In Progress
5060	Task		Pridat ovladanie pozicie klbov	Bc. Martin Mokry	Done
5061	Task		Pridat adekvatne rotacie kosti.	Bc. Martin Mokry	Done
5062	Task		Upravit skalovanie dlzky kosti.	Bc. Patrik Berger	Done
4919	Task	Mat kniznicu		Bc. Peter Belai	Done
4923	Product Backlog Item	Vytvorit dokumentaciu riadenia			New
4925	Task		Napisat uvod k riadeniu	Bc. Filip Skultety	Done
4926	Task		Napisat retrospektivu Sprint1,2	Bc. Filip Skultety	Done
4924	Product Backlog Item	Vytvorit dokumentaciu k ing. dielu			New
4927	Task		Napisat uvod k ing. dielu	Bc. Filip Skultety	Done
4928	Task		Spisat globalne ciele	Bc. Filip Skultety	Done
4929	Task		Vytvorit celkovy pohlad na system	Bc. Martin Mokry	Done
4934	Task		Popisat nami vytvarany modul 2	Bc. Martin Mokry	Done
4936	Task	Integreacia kniznice pod linuxom		Bc. Marek Rostar	Done
4998	Product Backlog Item	Osamostatnenie modulu LuaGraph			New
4999	Task		Odclenit kod modulu	Bc. Lukas Hagara	Done
5057	Task		Vytvorte podrobnu analyzu QVR kniznice	Bc. Lukas Hagara	Done

Obr. 6 Export úloh šprint 3 týždeň 2

8.7. Šprint č. 4, týždeň 1

Project: AugReality **Server:** tfs.fiit.stuba.sk\StudentsProjects **Query:** Report Sprint 4 **List type:** Tree

ID	Work Item Type	Title 1	Title 2	Assigned To	State
4814	Task	Oprava warningov cppcheck		Bc. Filip Skultety	In Progress
4918	Product Backlog Item	Chcem pomocout Kinect2 ziskat dataset.			New
4919	Task		Mat kniznicu	Bc. Peter Belai	Done
4920	Task		Zakomponovat kniznicu do projektu	Bc. Peter Belai	In Progress
4921	Task		Aplikacia by si mala vediet vybrat med	Bc. Peter Belai	To Do
4936	Task		Integreacia kniznice pod linuxom	Bc. Marek Rostar	Done
5077	Task		Doplnit instalacnu prirucku	Bc. Marek Rostar	Done
5078	Task		Data do nite2	Bc. Marek Rostar	Done
5067	Product Backlog Item	Doladenie ruk vo VR			New
5068	Task		Nastavit poziciu ruk ako slave kamery	Bc. Patrik Berger	In Progress
5069	Task		Refaktoring kodu 1	Bc. Patrik Berger	Done
5070	Task		Refaktoring kodu 2	Bc. Martin Mokry	Done
5071	Task		Skombinovat s grafom	Bc. Patrik Berger	To Do
5072	Product Backlog Item	Zobrazenie ruk v AR			New
5073	Task		Aruco	Bc. Martin Mokry	Done
5074	Task		Zobrazit ruky s vypnutym CameraMan	Bc. Martin Mokry	To Do
5075	Task		Pridat viedostream z leapu ako pozadi	Bc. Patrik Berger	To Do
5076	Task		Namapovanie virtualnych ruk na realne ruky		To Do
5079	Product Backlog Item	VISP			New
5080	Task		OpenCV - aktualizacia	Bc. Lukas Hagara	Done
5081	Task		Ziskat kniznicu VISP	Bc. Lukas Hagara	Done
5082	Task		Pridat binarky do projektu	Bc. Lukas Hagara	In Progress
5083	Product Backlog Item	PCL			New
5084	Task		Ziskat kniznicu PCL	Bc. Filip Skultety	Done
5085	Task		Pridat binarky do projektu	Bc. Filip Skultety	Done

Obr. 7 Export úloh šprint 4 týždeň 1

8.8. Šprint č. 4, týždeň 2

Project: AugReality **Server:** tfs.fiit.stuba.sk\StudentsProjects **Query:** Report Sprint 4 **List type:** Tree

ID	Work Item Type	Title 1	Title 2	Assigned To	State
4814	Task	Oprava warningov cppcheck		Bc. Filip Skultety	In Progress
4918	Product Backlog Item	Chcem pomocout Kinect2 ziskat dataset.			New
4919	Task		Mat kniznicu	Bc. Peter Belai	Done
4920	Task		Zakomponovat kniznicu do projektu	Bc. Peter Belai	In Progress
4921	Task		Aplikacia by si mala vediet vybrat med	Bc. Peter Belai	To Do
4936	Task		Integreacia kniznice pod linuxom	Bc. Marek Rostar	Done
5077	Task		Doplnit instalacnu prirucku	Bc. Marek Rostar	Done
5078	Task		Data do nite2	Bc. Marek Rostar	Done
5067	Product Backlog Item	Doladenie ruk vo VR			New
5068	Task		Nastavit poziciu ruk ako slave kamery	Bc. Patrik Berger	In Progress
5069	Task		Refaktoring kodu 1	Bc. Patrik Berger	Done
5070	Task		Refaktoring kodu 2	Bc. Martin Mokry	Done
5071	Task		Skombinovat s grafom	Bc. Patrik Berger	To Do
5072	Product Backlog Item	Zobrazenie ruk v AR			New
5073	Task		Aruco	Bc. Martin Mokry	Done
5074	Task		Zobrazit ruky s vypnutym CameraMan	Bc. Martin Mokry	To Do
5075	Task		Pridat viedostream z leapu ako pozadi	Bc. Patrik Berger	To Do
5076	Task		Namapovanie virtualnych ruk na realne ruky		To Do
5079	Product Backlog Item	VISP			New
5080	Task		OpenCV - aktualizacia	Bc. Lukas Hagara	Done
5081	Task		Ziskat kniznicu VISP	Bc. Lukas Hagara	Done
5082	Task		Pridat binarky do projektu	Bc. Lukas Hagara	In Progress
5083	Product Backlog Item	PCL			New
5084	Task		Ziskat kniznicu PCL	Bc. Filip Skultety	Done
5085	Task		Pridat binarky do projektu	Bc. Filip Skultety	Done

Obr. 8 Export úloh šprint 4 týždeň 2