

Slovenská technická univerzita

Fakulta informatiky a informačných technológií

Ilkovičova 2, 842 16 Bratislava 4

Vizualizácia informácií v obohatenej realite

Dokumentácia k riadeniu projektu

Tím č. 5 - ARVis

Bc. Duško Dogandžić

Bc. Dávid Durčák

Bc. Ján Handzuš

Bc. Patrik Hlaváč

Bc. Marek Jakab

Bc. Matej Marcoňák

Bc. Daniel Soós

Bc. Martina Trégerová

Vedúci projektu: Ing. Peter Kapec, PhD.

Predmet: Tímový projekt II

Kontakt: teamtp05@gmail.com

Akademický rok: 20213/2014, letný semester

Obsah

1 Úvod	1
2 Ponuka	2
2.1 Členovia tímu	2
2.2 Téma: Zábavný systém pre spolucestujúcich v automobile	3
2.2.1 Motivácia	3
2.2.2 Koncept riešenia	4
2.3 Téma: Vizualizácia informácií v obohatenej realite	4
2.3.1 Motivácia	4
2.3.2 Koncept riešenia	5
2.4 Príloha A - zoznam preferencií	6
3 Úlohy členov tímu	7
3.1 Manažérske úlohy členov	7
3.2 Krátkodobé úlohy členov	7
3.3 Podiel práce členov na jednotlivých častiach dokumentácie	8
3.3.1 Dokumentácia k inžinierskemu dielu	8
3.3.2 Dokumentácia k riadeniu	9
4 Plán projektu	10
5 Metodiky použité pri vývoji	11
5.1 Metodika prípravy úloh na realizáciu	11
5.1.1 Slovník pojmov	11
5.1.2 Procesy	12
5.2 Metodika manažmentu verzií	15
5.2.1 Zoznam nadväzujúcich metodík	15
5.2.2 Slovník pojmov	15
5.2.3 Role a ich zodpovednosti	16
5.2.4 Procesy manažmentu verzií	16
5.3 Metodika programového kódu	20
5.3.1 Pravidlá pre tvorbu modulov	20
5.3.2 Pravidlá pre tvorbu tried	21
5.3.3 Pravidlá pre používanie vlákien	22
5.3.4 Všeobecné pravidlá	23

5.3.5	Konvencia pomenovaní názvov	24
5.3.6	Všeobecné konvencie	26
5.4	Metodika prehliadky kódu	28
5.4.1	Slovník pojmov	28
5.4.2	Zoznam nadväzujúcich metodík	28
5.4.3	Role a ich úlohy	29
5.4.4	Metodika prehliadky kódu	29
5.4.5	Konvencie pri prehliadke zdrojového kódu	33
5.5	Metodika pre písanie komentárov a dokumentovanie zdrojového kódu	34
5.5.1	Role a povinnosti	34
5.5.2	Základné pravidlá pre písanie komentárov	35
5.5.3	Procesy	38
5.6	Metodika evidencie úloh	40
5.6.1	Slovník pojmov	40
5.6.2	Role a povinnosti	41
5.6.3	Procesy a evidencie	41
5.7	Metodika testovania softvéru	45
5.7.1	SCRUM	45
5.7.2	Role a ich povinnosti	46
5.7.3	Unit testovanie	46
5.7.4	Integračné testovanie	47
5.7.5	Testovanie systému	47
5.7.6	Akceptačné testovanie	48
5.8	Metodika dokumentácie knižníc	49
5.8.1	Roly	49
5.8.2	Procesy	49
6	Manažment kvality a monitorovanie projektu	55
6.1	Monitorovanie postupu projektu	55
6.2	Monitorovanie kvality projektu	55
7	Manažment podpory vývoja a integrácie	56
7.1	Manažment pre podporu verzií	56
7.2	Nasadenie aplikácie a podpora pri vývoji	56

8	Manažment vývoja	58
8.1	Rozdelenie práce	58
8.2	Prostriedky použité pri vývoji	59
9	Manažment rizík	60
9.1	Nedostatočná analýza pôvodného programu	61
9.2	Strata člena tímu	61
9.3	Problémy funkcionality na rôznych platformách	62
9.4	Zlé rozlíšenie ovládania gestami	62
9.5	Zlá voľba technológie pre snímanie prostredí	63
9.6	Hardvérové chyby a poškodenia	63
9.7	Zlá prepojitelnosť funkcionality s pridaným ovládaním	64
9.8	Slabá optimalizácia a pomalý chod programu	64
10	Manažment rozvrhu a plánovania	65
10.1	Postup plánovania šprintu	65
10.2	Používateľské príbehy	65
10.3	Všeobecný rozvrh zimného semestra	66
10.4	Vyhodnotenie plánov	66
11	Manažment komunikácie	67
11.1	Stretnutia	67
11.2	Formálna komunikácia	67
11.3	Neformálna komunikácia	68
12	Manažment tvorby dokumentácie	69
A	Zápisnice zo stretnutí	A-1
A.1	Zápis zo stretnutia č. 1 tímu č. 5	A-2
A.2	Zápis zo stretnutia č. 2 tímu č. 5	A-4
A.3	Zápis zo stretnutia č. 3 tímu č. 5	A-7
A.4	Zápis zo stretnutia č. 4 tímu č. 5	A-10
A.5	Zápis zo stretnutia č. 5 tímu č. 5	A-13
A.6	Zápis zo stretnutia č. 6 tímu č. 5	A-16
A.7	Zápis zo stretnutia č. 7 tímu č. 5	A-20
A.8	Zápis zo stretnutia č. 8 tímu č. 5	A-23
A.9	Zápis zo stretnutia č. 9 tímu č. 5	A-26

A.10	Zápis zo stretnutia č. 10 tímu č. 5	A-29
A.11	Zápis zo stretnutia č. 11 tímu č. 5	A-32
B	Reporty z letného semestra	B-1
B.1	Report č. 1 v letnom semestri	B-2
B.2	Report č. 2 v letnom semestri	B-9
B.3	Report č. 3 v letnom semestri	B-12
B.4	Report č. 4 v letnom semestri	B-17
B.5	Report č. 5 v letnom semestri	B-21
B.6	Report č. 6 v letnom semestri	B-26
C	Obsah elektronického média	C-1
D	Preberací protokol	D-1

1 Úvod

Tento dokument predstavuje dokumentáciu riadenia k projektu Vizualizácia informácií v obohatenej realite v rámci predmetu Tímový projekt na Fakulte informatiky a informačných technológií na Slovenskej technickej univerzite v Bratislave.

Cieľom projektu je preniesť známe metódy vizualizácie informácií do prostredia obohatenej reality a hľadať nové metódy zobrazenia a interakcie. Projekt má skôr výskumný charakter.

V druhej kapitole je ponuka tímu na získanie tímového projektu spolu s motiváciou a konceptom riešenia. Tretia kapitola opisuje úlohy členov v tíme, takisto ako aj podiel autorstva na dvoch dokumentáciach.

Štvrtá kapitola obsahuje plán projektu. Ďalšie kapitoly opisujú metodiky použité pri vývoji softvéru a manažment v tíme. Každý typ manažmentu je obsiahnutý v samostatnej kapitole, kým metodikám patrí jedna kapitola.

V prílohe A sa nachádzajú všetky zápisy zo stretnutí v zimnom semestri a príloha B obsahuje vygenerované reporty v letnom semestri zo systému, ktorý má na starosť manažment úloh. Obsah elektronického média je zdokumentovaný v prílohe C a preberací protokol sa nachádza tiež v prílohovej časti.

2 Ponuka

2.1 Členovia tímu

Bc. Duško Dogandžić

Absolvent trojročného bakalárskeho štúdia v Užiciach, odbor business informatika. V rámci bakalárskej práce vypracoval informačný systém pre fakultu. Má skúsenosti s C, PHP, CSS, HTML, MySQL a e-business. Počas štúdií pracoval ako asistent profesora grafického designu a webdesignu. Po ukončení bakalárskeho štúdia pracoval 1 rok ako grafický dizajnér. Na fakulte by sa rád venoval problematike vývoja softvéru.

Bc. Dávid Durčák

Absolvent študijného odboru Informatika na FIIT STU, vypracoval bakalársku prácu na tému z oblasti evolučných algoritmov. V budúcnosti sa chce venovať počítačovej grafike a spracovaniu obrazu. Má skúsenosti s prácou v jazykoch a prostrediach C/C++, Java, Qt, OpenGL, MySQL, MS Visual Studio, Eclipse.

Bc. Ján Handzuš

Bakalárske štúdium absolvoval na FIIT STU. Jeho bakalársky projekt sa zaoberal zobrazením a zachytením svetelých polí na mobilnom zariadení pri tomto projekte používal OpenGL a Objective-C. Ďalej sa zaoberal polohovými senzormi na mobilnom zariadení iPhone ako akcelerometer, megnetometer a gyroskop a ich kombináciu pomocou frameworku Core Motion. Ovláda programovacie jazyky Java, Objective-C, C, Lisp a Prolog.

Bc. Patrik Hlaváč

Absolvent bakalárskeho štúdia v odbore Informatika na FIIT STU. Práca spočívala v PHP+SQL a tiež tvorbe interface. Zaoberá sa tvorbou webových systémov. V tomto smere výrazne odľahčí ostatných členov tímu a umožní lepšie sústredenie sa na svoje zameranie. Taktiež ovláda programovacie jazyky C/C++, Java. Má skúsenosti s počítačovou grafikou v súvislosti s knižnicou OpenGL.

Bc. Marek Jakab

Bakalárske štúdium absolvoval na FIIT STU. Zaujíma sa o oblasť počítačového videnia a grafiky. Jeho bakalárska práca sa týkala vytvárania obohatenej reality na mobilnom telefóne s využitím rozpoznávania planárnych objektov metódou lokálnych deskriptorov. Túto prácu úspešne prezentoval aj na dvoch konferenciách IITSRC 2013 a CESC. Programuje

v C/C++, Java a skúsenosti má aj s vytváraním aplikácií pre mobilnú platformu android, OpenCV, OpenGL, MySQL.

Bc. Matej Marcoňák

Absolvoval bakalárske štúdium FIIT STU v odbore Informatika. V bakalárskom štúdiu sa zameriaval na prácu s databázami a spracovanie dát. Počas štúdia získal skúsenosti v programovacích jazykoch C,C++,C# a Java a tiež základy práce s OpenGL.

Bc. Daniel Soós

Absolvoval bakalárske štúdium na FIIT STU. Jeho bakalárska práca bola zameraná na evolučné algoritmy a výstupom projektu bolo nájdenie takého podporného mechanizmu pri simulovaní procesu zberu potravy v živote sociálnych živočíchov, ktorý prispieva k optimalizácii. Táto práca bola úspešne prezentovaná aj na konferencii IITSRC 2013. Má skúsenosti s jazykmi C/C++, Java, PHP, MySQL.

Bc. Martina Trégerová

Absolvovala trojročné bakalárske štúdium na FIIT STU, vypracovala bakalársku prácu s problematikou zameranou na počítačovú grafiku a 3D grafiku. Má skúsenosti s OpenGL, C/C++, Java, MySQL, Eclipse, Qt, Lisp a Prolog. Zaujíma sa o oblasť počítačového videnia a grafiky a tiež by sa rada informovala o problematike programovania na Android v mobile.

2.2 Téma: Zábavný systém pre spolucestujúcich v automobile

2.2.1 Motivácia

Počítačové videnie patrí medzi najzaujímavejšie odbory v súčasnosti. Téma nás zaujala hlavne kvôli možnosti pracovať s obohatenou realitou, ktorú ako členovia tímu považujeme za veľmi modernú a atraktívnu s perspektívou pokračovania vývoja v budúcnosti.

Možnosť zlepšiť inak nudný zážitok v podobe cesty autom pre spolucestujúcich je pre nás príjemnou výzvou. Táto téma zároveň poskytuje dostatok priestoru pre realizáciu vlastných nápadov a dostupné senzory na mobilných telefónoch nám umocňujú výsledný efekt obohatenej reality.

V rámci tímu sa značný počet členov zaujíma o problematiku počítačového videnia, obohatenej reality a počítačovej grafiky. Marek Jakab, Ján Handzuš a Martina Trégerová sa v rámci bakalárskej práce venovali oblasti počítačového videnia a počítačovej grafiky, pričom sa zaoberali obohatenou realitou, prácou s mobilnými telefónmi a 3D grafikou,

preto majú pre túto tému vhodné predpoklady. Ostatní členovia absolvovali predmet Počítačová grafika I. v rámci bakalárskeho štúdia, a teda majú skúsenosti s prácou v OpenGL a programovaním v jazyku C/C++, prípadne majú iné pracovné skúsenosti.

Členovia tímu sa zaujímajú o oblasť vizualizácie a grafiky a na základe tohto záujmu si zvolili súvisiace predmety, ako napr.: Počítačové videnie alebo Vizualizácia dát. Dôležité sú aj ďalšie schopnosti našich členov z iných oblastí, v ktorých pôsobili, ako napríklad databázové systémy alebo webová tvorba. To môže byť nápomocné pri navrhovaní samotnej aplikácie, ako aj pri tvorbe pútavej webovej stránky, kde môžeme naše prípadné úspechy prezentovať.

2.2.2 Koncept riešenia

Mobilné zariadenie pripojené do daného systému v automobile nám ponúka ďalšie možnosti využitia obohatenej reality v automobile. Okrem základného rozpoznávania okolia prostredníctvom kamery a vykresľovania na projekčnú fóliu tak môžeme doplniť systém o informácie z mobilného telefónu. Príkladom môže byť senzor GPS, ktorý nám poskytne informácie o polohe a pomocou nich vieme jednoduchšie vyhľadať informácie o okolí, prípadne zúžiť okruh prehľadávania pri detekcii objektov. Okrem GPS môžeme využiť ďalšie senzory pre interakciu človeka so systémom a dať tak používateľovi priestor pre vlastnú realizáciu a výrazne zvýšiť pocit obohatenej reality. Vznikajú tak vhodné podmienky pre interaktívne menu, prípadne minihry pre deti a iné.

Hry môžu mať formu náučnú, kde napríklad hádame informácie o detegovanom objekte, alebo priradíme korešpondujúce objekty. Je možné ich riešiť aj zábavnou formou, ako môže byť napríklad vyhýbanie sa rôznym prekážkam, alebo vytvorenia puzzle z aktuálneho obrazu. Tiež by mohlo zaujať spestrenie cestovania statického charakteru vo forme kvízu o okolitom svete, alebo jednoducho získavanie náučných informácií.

Okrem senzorov mobilného telefónu je možné využiť ako ďalší vstup aj gestá rúk prípadne pohyb hlavy. V tomto prípade by okrem kamery, ktorá by sledovala obraz za oknom auta, pribudla kamera pre sledovanie používateľa.

2.3 Téma: Vizualizácia informácií v obohatenej realite

2.3.1 Motivácia

Spojenie obohatenej reality s vizualizáciou dát predstavuje zaujímavý prínos v oblasti počítačového videnia a počítačovej grafiky, kde sa pútavou formou prezentujú dôležité dáta a stávajú sa tak pre človeka zaujímavejšími.

Tento projekt pre nás predstavuje atraktívnu výzvu a veríme, že sa na ňom môžeme veľa naučiť. Podobných projektov v súčasnosti nie je veľa a preto sa nám naskytuje pomerne veľké množstvo realizácií, vlastných nápadov ako aj typov dát, ktoré môžeme vizualizovať.

V rámci tímu sa značný počet členov zaujíma o problematiku počítačového videnia, obohatenej reality a počítačovej grafiky. Marek Jakab, Ján Handzuš a Martina Trégerová sa v rámci bakalárskej práce venovali oblasti počítačového videnia a počítačovej grafiky, pričom sa zaoberali obohatenou realitou, prácou s mobilnými telefónmi a 3D grafikou, preto majú pre túto tému vhodné predpoklady. Ostatní členovia absolvovali predmet Počítačová grafika I. v rámci bakalárskeho štúdia, a teda majú skúsenosti s prácou v OpenGL a programovaním v jazyku C/C++, prípadne majú iné pracovné skúsenosti.

Členovia tímu sa zaujímajú o oblasť vizualizácie a grafiky a na základe tohto záujmu si zvolili súvisiace predmety, ako napr.: Počítačové videnie alebo Vizualizácia dát. Dôležité sú aj ďalšie schopnosti našich členov z iných oblastí, v ktorých pôsobili, ako napríklad databázové systémy alebo webová tvorba.

2.3.2 Koncept riešenia

K prvotným spôsobom vykresľovania dát a vytvorenia prvku obohatenej reality patrí použitie značiek. Od tohto by sme ale radi odpúťali a vytvorili 3D model grafu bez ich využitia. Manipulácia s dátami by mohla prebiehať pomocou rôznych gest rúk. Hlavným cieľom by v tomto prípade bolo sprehl'adnenie celej vizualizácie, kde môže človek upriamiť pozornosť na vybranú časť jeho záujmu. Existuje veľké množstvo typov grafov, ktoré je vhodné pri danom projekte zobrazovať a poskytovali by dobrú informatívnu hodnotu.

Prvá časť bude venovaná problému zobrazovania 3D objektov s použitím informácií z kamery, zistenie polohy dosky stola prípadne ďalších objektov nachádzajúcich sa v okolí, ktoré môžu určitým spôsobom zasahovať do vykresľovania dát. Je to možné dosiahnuť použitím knižnice OpenCV, alebo formou počiatočnej inicializácie, kde sa vymedzí priestor, prípadne použitím Kinectu, ktorý nám navyše poskytuje informáciu o hĺbke. Vyriešenie tohto problému je kľúčové pre vytvorenie ilúzie reálneho objektu, ktorý je súčasťou prostredia.

Ďalší priestor bude venovaný práve vykresleniu dát, výberu vhodného spôsobu ich zobrazenia, kde bude snaha dostať čo najväčšie množstvo informácií pri zachovaní prehľadnosti zobrazovaných 3D modelov. Ďalej sa chceme venovať rôznym možnostiam manipulácie - možnosť intuitívneho manipulovania s dátami používateľom výrazne prispieva k čitateľnosti daných dát a môže sa tak sústrediť na časti, ktoré ho najviac zaujímajú.

2.4 Príloha A - zoznam preferencií

1. Zábavný systém pre spolucestujúcich v automobile
2. Vizualizácia informácií v obohatenej realite
3. Digital SweatShop
4. Sledovanie pohľadu pri používaní aplikácií
5. Webový komunitný systém otázok a odpovedí
6. Interaktívne hry na mobile s multimedialným obsahom
7. Trojdimenzionálne UML
8. Prehliadka kódov v tímových projektoch
9. Monitor programátora v IDE
10. Virtuálna FIIT na mobile
11. Distribuované počítanie na FIIT
12. Robotický futbal
13. Analýza výsledkov výskumu
14. Rečové poruchy

3 Úlohy členov tímu

3.1 Manažérske úlohy členov

Člen tímu	Rola v tíme
Bc. Duško Dogandžić	manažér testovania
Bc. Dávid Durčák	manažér podpory vývoja
Bc. Ján Handzuš	manažér rozvrhu a plánovania
Bc. Patrik Hlaváč	manažér rizík a obstarávania
Bc. Marek Jakab	manažér vývoja
Bc. Matej Marcoňák	manažér kvality a monitorovania projektu
Bc. Daniel Soós	manažér dokumentácie
Bc. Martina Trégerová	vedúca tímu, manažér komunikácie

3.2 Krátkodobé úlohy členov

- Bc. Duško Dogandžić
 - Testovanie
- Bc. Dávid Durčák
 - Ovládanie pomocou značiek
 - Obraz z kamery na pozadí
- Bc. Ján Handzuš
 - Textúry pre uzly
 - Zmena pozadia
- Bc. Patrik Hlaváč
 - Rozpoznávanie ruky
 - Generovanie reportov a webstránka
 - Import textúr
- Bc. Marek Jakab
 - Import Kinect Speech

- Rozpoznávanie hlasových povelov
- Vytvorenie SizedQueue
- Bc. Matej Marcoňák
 - Rozpoznávanie ruky a funkcionality myši
 - Import Kinect funkcionality do programu
 - Refaktoring a podpora vývoja
- Bc. Daniel Soós
 - Rozpoznávanie ruky
 - Integrácia dokumentácie
- Bc. Martina Trégerová
 - Reálna scéna
 - Rozpoznávanie markerov

3.3 Podiel práce členov na jednotlivých častiach dokumentácie

3.3.1 Dokumentácia k inžinierskemu dielu

- Bc. Duško Dogandžić
 - Kapitoly 2.1.1, 2.5.6, 3.4.2
- Bc. Dávid Durčák
 - Kapitoly 2.3.3, 2.4.4, 2.4.2, 2.5.2, 3.1.2, 3.2.1, 3.2.2.5, 3.2.2.6, 3.2.3
- Bc. Ján Handzuš
 - Kapitoly 2.2.3, 2.2.4, 2.3.5, 2.4.4, 2.5.4, 3.2.3.6
- Bc. Patrik Hlaváč
 - Kapitoly 2.1.3, 3.2.2.1
- Bc. Marek Jakab
 - Kapitoly 2.1.1, 2.2.5, 2.3.1, 2.4.1, 2.5.1, 3.2.2.3, 3.2.2.4, 3.3.2

- Bc. Matej Marcoňák
 - Kapitoly 2.1.2, 2.2.6, 2.3.2, 2.4.3, 2.5.3, 3.1.1, 3.2.2.2
- Bc. Daniel Soós
 - Kapitoly 2.2.2, 2.4.5, 3.2.2.1, Prílohy A, B
- Bc. Martina Trégerová
 - Kapitoly 2.2.1, 2.4.4, 2.5.5, 3.3.1

3.3.2 Dokumentácia k riadeniu

Kapitola	Autor
Úvod	Bc. Daniel Soós
Ponuka	všetci
Úlohy členov tímu	Bc. Daniel Soós
Plán projektu	Bc. Daniel Soós
Metodika prípravy úloh na realizáciu	Bc. Ján Handzuš
Metodika manažmentu verzií	Bc. Matej Marcoňák
Metodika programového kódu	Bc. Dávid Durčák
Metodika prehliadky kódu	Bc. Marek Jakab
Metodika dokumentácie zdrojového kódu	Bc. Martina Trégerová
Metodika evidencie úloh	Bc. Patrik Hlaváč
Metodika testovania softvéru	Bc. Duško Dogandžić
Metodika dokumentácie knižníc	Bc. Daniel Soós
Manažment kvality a monitorovania projektu	Bc. Matej Marcoňák
Manažment podpory vývoja a integrácie	Bc. Dávid Durčák
Manažment vývoja	Bc. Marek Jakab
Manažment rizík	Bc. Patrik Hlaváč
Manažment rozvrhu a plánovania	Bc. Ján Handzuš
Manažment komunikácie	Bc. Martina Trégerová
Manažment tvorby dokumentácie	Bc. Daniel Soós
Príloha A - zápisnice zo stretnutí	všetci
Návod na inštaláciu	Bc. Dávid Durčák Bc. Ján Handzuš Bc. Marek Jakab
Príloha B	Bc. Patrik Hlaváč
Prílohy C, D	Bc. Daniel Soós

4 Plán projektu

Práca na tomto projekte je realizovaná metódou SCRUM, ktorý je inkrementálny a iteratívny agilný prístup na vývoj softvérov. Základnou jednotkou tejto metódy sú šprinty - vopred definované časové intervaly s presne identifikovanými úlohami, ktoré sa majú v tomto intervale vykonať. V našom projekte sú naplánované dvojtýždňové šprinty. Vyhodnotenie predošlých a identifikácia nových úloh prebieha primárne na tímových stretnutiach. Letný semester je rozdelený na šesť šprintov. V tabuľke 1 vidíme náš semestrový plán.

Tabuľka 1: *Harmonogram nášho tímu v letnom semestri*

týždeň v semestri	plán
1.	inicializácia semestra a plán pre šprint č.6-1L
2.	práca na príbehoch šprintu č.6-1L
3.	ukončenie šprintu č.6-1L
4.	práca na príbehoch šprintu č.7-2L
5.	ukončenie šprintu č.7-2L
6.	práca na príbehoch šprintu č.8-3L
7.	ukončenie šprintu č.8-3L
8.	práca na príbehoch šprintu č.9-4L
9.	ukončenie šprintu č.9-4L
10.	práca na príbehoch šprintu č.10-5L
11.	ukončenie šprintu č.10-5L
12.	práca a ukončenie šprintu č.11-6L

5 Metodiky použité pri vývoji

5.1 Metodika prípravy úloh na realizáciu

Táto metodika sa zaoberá procesmi presného určovania obtiažnosti a času potrebného na vykonanie jednotlivých úloh v Scrum metodike riadenia projektu. Náš projekt je špecifický tým, že počet úloh približne zodpovedá počtu členov v tíme. Dôvodom je hlavne fakt, že väčšina úloh je príliš komplexná a neumožňuje asynchrónnu spoluprácu viacerých členov tímu na realizácii jednej úlohy. Spomenutá metodika dolnej úrovne preberá výstup metodiky identifikácia úloh, ktorá je na rovnakej úrovni. Táto metodika spadá do témy riadenia, jej úlohou je jednoznačne pripraviť dáta, ktoré budú neskôršie použité pri analyzovaní stavu úloh. A táto analýza posluží pre plánovanie ďalšieho Sprintu a jej dáta umožnia zefektívniť fungovanie tímu.

5.1.1 Slovník pojmov

- **Scrum master** – člen tímu, ktorý hlavne zodpovedá za to ako sa vymieňajú informácie v tíme.
- **Sprint** – jeden pevne daný časový úsek vo vývoji softvéru, na konci ktorého musia byť ukončené všetky špecifikované ciele. V našom prípade tento časový úsek má trvanie dva týždne.
- **Sprint planning meeting** – stretnutie na začiatku Sprintu v trvaní od dvoch do troch hodín. Hlavnou úlohou je naplánovanie cieľov pre ďalší Sprint a vyhodnotenie predchádzajúcej práce.

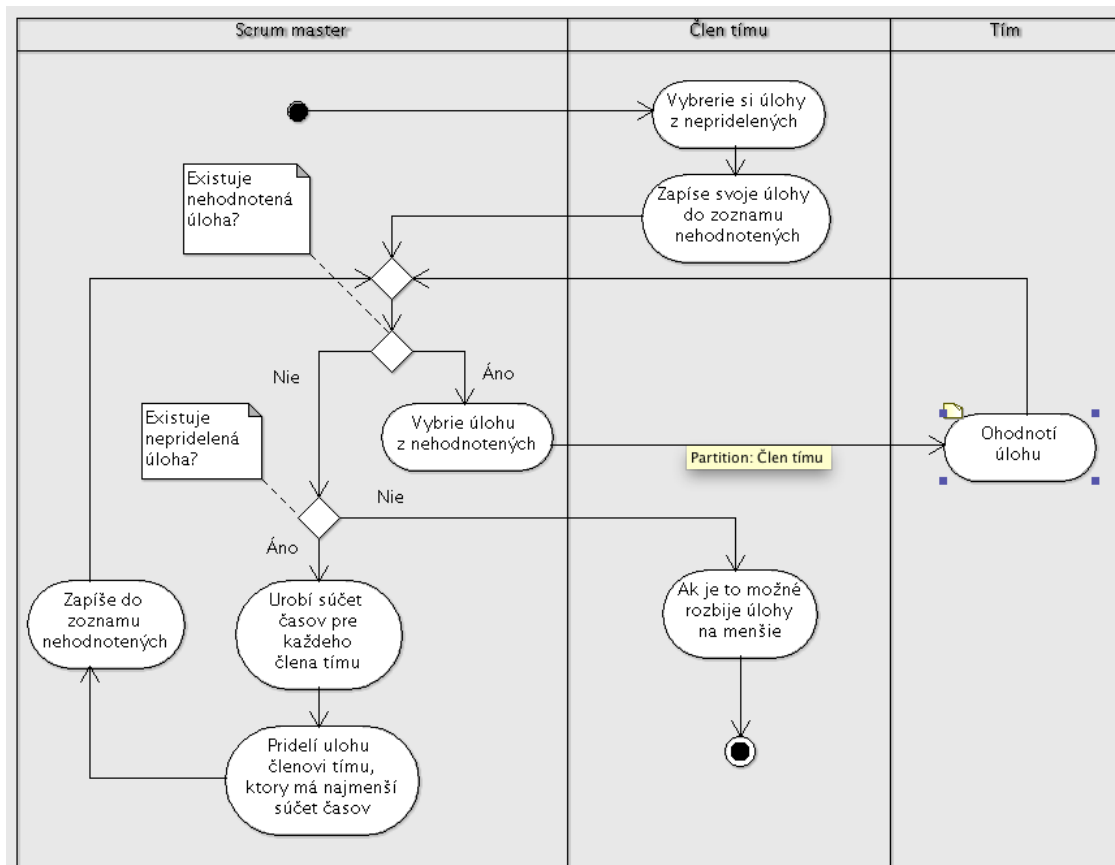


Obr. 1: Scrum Board

- **Scrum board** - tabuľka, ktorá obsahuje informácie o všetkých cieľoch a ich prioritách, ktoré má splniť výsledný produkt. Taktiež ciele, ktoré sa majú dosiahnuť v aktuálnom Sprinte a úlohy pre ich dosiahnutie (Obr. 1). Je to hlavný nástroj metodiky plánovania.
- **Product owner** - člen tímu ktorý, má predstavu o konečnom produkte a cieľoch, ktoré má splniť. Zodpovedá hlavne za priority všetkých cieľov, ktoré sú potrebné pri plánovaní Sprintu. Tento člen tímu však nehodnotí jednotlivé úlohy, ale môže pomôcť pri procese hodnotenia.

5.1.2 Procesy

V tejto časti sú definované procesy a ich výstupné informácie ako aj osoby zodpovedne za riadenie týchto procesov a role ostatných členov tímu, ktorí sa na procesoch hodnotenie podieľajú (Obr. 2).



Obr. 2: Diagram aktivít metodiky prípravy úloh na realizáciu

5.1.2.1 Pridelenie úloh

Iniciátor: Scrum master

Vstup: Je vytvorený zoznam úloh podľa cieľov pre ďalší Sprint

Výstup: Existuje zoznam pridelených a ohodnotených úloh

Kedy: Počas Sprint planning meetingu

Hlavný tok:

1. Každý člen tímu si vyberie úlohy.
2. Aktivuje sa proces "Ohodnotenie úloh tímom".
3. Ak neostala žiadna nepridelená úloha, proces sa ukončí, inak sa pokračuje v alternatívnom toku 1.

Alternatívny tok: tok sa aktivuje, ak zoznam nehodnotených úloh je prázdny a existuje nepridelená úloha.

1. Scrum master urobí súčet všetkých časov úloh pre každého člena tímu.
2. Vyberie úlohu zo zoznamu nepridelených úloh a pridelí ju členovi tímu, ktorý má najmenší súčet všetkých časov.
3. Pokračuje sa v bode 2. hlavného toku.

5.1.2.2 Ohodnotenie úloh tímom (Planning poker)

Iniciátor: Scrum master

Vstup: Existuje zoznam pridelených úloh podľa cieľov na ďalší Sprint

Výstup: Každá úloha má pridelené číslo podľa obtiažnosti

Kedy: Počas Sprint planning meetingu

Hlavný tok:

1. Scrum master rozdá každému členovi tímu kartovú sadu, kde na každej karte je jedno číslo z fibonacciho postupnosti od 0.5 po 100.
2. Scrum master vyberie a prečíta nahlas nehodnotenú úlohu, ktorá je pridelená.
3. Každý člen tímu nezávisle na ostatných vyberie kartu s číslom podľa toho ako danú úlohu hodnotí podľa obtiažnosti. Čím vyššie číslo karty, tým väčšia obtiažnosť úlohy. Kartu však zatiaľ neukazuje ostatným členom tímu.

4. Scrum master sa presvedčí, či má, každý vybratú kartu. Ak áno, na jeho pokyn všetci ukážu svoju kartu.
5. Scrum master skontroluje hodnoty od každého člena tímu. Ak medzi najvyššou a najnižšou hodnotou je rozdiel väčší ako päť, pokračuje sa v alternatívnom toku 1.
6. Scrum master zapíše hodnotu, ktorú si zvolil člen tímu zodpovedný za jej realizáciu.
7. Scrum master vypočíta a zapíše čas potrebný na realizáciu úlohy. Vzorec na výpočet času $C = P/2$:
 - C.....Celkový čas v hodinách potrebný na realizáciu úlohy.
 - P.....Hodnota obtiažnosti.
8. Ak je zoznam nehodnotených úloh prázdny proces sa končí, inak sa pokračuje krokom 2. hlavného toku.

Alternatívny tok: tok sa aktivuje v prípade, ak je päťbodový rozdiel medzi najnižšou a najvyššou hodnotou obtiažnosti.

1. Každý člen tímu uvedie dôvod jeho hodnotenia a vydiskutuje ho z ostatnými členmi tímu. Člen tímu môže kedykoľvek zmeniť svoje hodnotenie počas diskusie. Možné dôvody vysokého rozdielu v hodnotení:
 - Nejasnosti požiadaviek cieľa, ktorý ma splňať produkt. V tomto prípade bude požiadany Product owner o upresnenie jeho požiadaviek.
 - Člen tímu nemá skúsenosti s danou technológiou alebo oblasťou.
 - Člen tímu precenil alebo podcenil úlohu. Tu pomôže diskusia v tíme, ktorá objasní rozsah úlohy.
2. Pokračuje sa v bode 6. Hlavného toku.

5.1.2.3 Rozbitie úlohy jednotlivcom

Každý člen tímu, ktorý má pridelenú a ohodnotenú úlohu, ktorých čas na splnenie presahuje päť hodín sa pokúsi rozbiť úlohu na menšie časti. Pokiaľ je to možné, tak na úlohy, ktorých čas realizácie nepresahuje dve hodiny. Tento proces má za úlohu zlepšenie sledovania prác na projekte počas Sprintu.

5.2 Metodika manažmentu verzií

Pri práci na projekte v rámci vývojového tímu, ktorý má viac ako dvoch členov, je žiaduce používanie nástrojov pre správu verzií. Jedným z takýchto nástrojov je Git, ktorý ponúka riešenie problémov pre efektívnejšiu spoluprácu v tíme vzhľadom na vývoj softvéru.

Účelom tohto dokumentu je opis procesov a postupnosť krokov pri manažmente verzií zdrojového kódu. V rámci tímu je táto metodika určená pre všetkých členov.

5.2.1 Zoznam nadväzujúcich metódik

V tejto metodiky sa budeme odkazovať na postupy, ktoré s procesmi manažmentu verzií úzko spolupracujú. Tieto procesy sú:

- prehliadky kódov,
- manažment úloh,
- manažment písania zdrojových kódov,
- manažment testovania.

5.2.2 Slovník pojmov

- **Git** - distribuovaný systém pre správu verzií.
- **Git Bash** - konzolové prostredie pre prácu s Gitom pod systémom Windows.
- **Repozitár** - základ systému pre správu verzií, ktoré slúži ako úložisko pre údaje súvisiace s projektom. Rozlišuje sa na dva základe typy úložísk, centrálna a lokálna.
- **Hlavná vetva** - v tomto projekte rozlišujeme dve hlavné vetvy:
 - *master* – hlavná vetva, ktorá obsahuje kód, ktorý je pripravený na produkciu,
 - *develop* – vetva, o ktorú sa operia vývoj. Obsahuje funkcionality, ktorá sa pripravuje na vydanie projektu.

V rámci projektu tieto vetvy existuje počas celého vývoja.

- **Vedľajšia vetva** - vetva, ktorá vzniká od hlavnej vetvy pre potreby projektu z hľadiska pridávania novej funkcionality (features vetvy), opravy chýb (hotfixes vetvy) a prípravy na produkciu (release vetvy), ktorá sa dočasnú funkcionality.
- **Commit** - zoznam zmien, ktoré nastali v lokálnom repozitáre.

- **Správa** - informácie o commite napísané tvorcom commitu.

5.2.3 Role a ich zodpovednosti

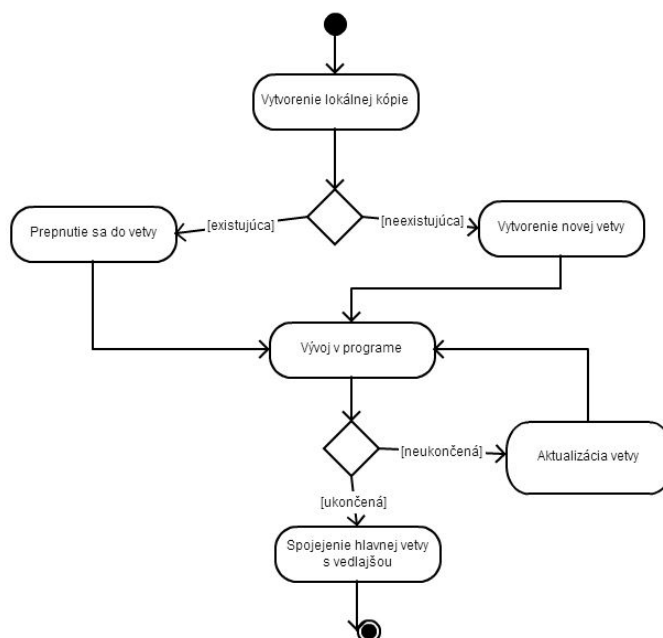
Rola	Zodpovednosť
Programátor/Vývojár	- Vytváranie novej funkcionality - Oprava chýb - Komentovanie kódu
Správca repozitára	- Správa centrálného repozitára

5.2.4 Procesy manažmentu verzii

Pri procese správy verzii vo vývoja softvéru sú identifikované tieto základné procesy:

- 5.2.4.1 vytvorenie novej vedľajšej vetvy,
- 5.2.4.2 pridávanie zmien vo vetve,
- 5.2.4.3 spájanie vetiev.

Tieto jednotlivé základné procesy je možné rozdeliť na menšie podprocesy, ktoré sú následne opísané v príslušných častiach procesu. Na Obr. 3 sú znázornené tieto procesy.



Obr. 3: Diagram aktivít pre časť manažmentu verzii

Jednotlivé postupy sú postavené na nástrojoch terminál (Linux/Mac) a Git Bash (Windows), v ktorých sa môžu spúšťať git príkazy.

5.2.4.1 Vytvorenie novej vedľajšej vetvy

Vstup: Úloha, ktorá vznikla v rámci procesu rozdeľovania úloh

Výstup: Pripravené lokálne úložisko s príslušnou vetvou

Podmienkou vývoja novej vetvy je mať nastavené lokálne úložisko. V prípade, že tomu tak nie je, vývojár si vytvorí lokálnu kópiu z centrálného úložiska príkazom:

```
git -b <názov vetvy> clone <url adresa projektu> <názov na uloženie>
```

Pričom názov vetvy a názov na uloženie nie sú povinné.

Nová vedľajšia vetva je vytvorená na základe jednej z hlavných vetiev, preto je potrebná aktualizácia týchto vetiev na lokálnom úložisku, kde v konkrétne v tomto prípade máme dve hlavné vetvy, master a develop.

```
git checkout <názov vetvy>
git pull origin <názov vetvy>
```

Ďalším krokom je proces *vytvárania novej vetvy*, kde názov pozostáva z troch častí, ktoré sú oddelené pomocou pomlčky. Prvá časť označuje typ vedľajšej vetvy (features, hotfix, ...), druhá obsahuje popis úlohy (faceDetection, gestureDetection, ...) a posledná časť bude zodpovedná osoba za vetvu (Matej, Marek, ...), napr. *features-faceDetection-Marek*. Následne sa vetva vytvorí pomocou príkazu:

```
git checkout <názov hlavnej vetvy>
git branch <názov vetvy>
git checkout <názov vetvy>
git push origin <názov vetvy>
```

V prípade, že existuje podobná vetva pre takúto funkcionálnu, rozhodujeme sa, či úloha, ktorá sa má vykonávať je rozdielna od existujúcej a vytvoríme novú vetvu s modifikovaním názvom alebo sa vývojár *prepne do tejto vetvy*.

```
git checkout <názov vetvy>
git pull origin <názov vetvy>
```

5.2.4.2 Pridávanie zmien vo vetve

Vstup: Pripravené lokálne úložisko

Výstup: Zmeny v softvéri, ktoré aktivujú ďalšie procesy (prehliadky kódov, testovanie)

Po vytvorení novej vetvy programátor začne pracovať na projekte. Následne vykonané zmeny na lokálnom úložisku je potrebné zverejniť pre ostatných programátorov na centralizovanom úložisku. Zmeny vykonané na lokálnom úložisku sa zobrazia pomocou príkazu:

```
git checkout <názov vetvy>
git status
```

Tieto zmeny je vhodné rozdeliť medzi viacero commitov na základe logických zmien v programe. V prípade menšieho počtu zmien je vhodné použiť príkaz:

```
git add -p
```

Príkaz zobrazuje jednotlivé zmeny s možnosťou výberu. Tento príkaz sa nehodí pri veľkých zmenách, pričom prechádzanie jednotlivých zmien by bolo príliš zdĺhavé. V tom prípade sa môže použiť viackrát príkaz:

```
git add <názov súboru>
```

Týmto príkazom sa môže postupne *pridávať súbory*, poprípade použiť príkaz, ktorým sa pridávajú naraz všetky súbory.

```
git add -A
```

Po vybraní zmien sú tieto *odoslané do lokálneho úložiska príkazom:*

```
git commit
```

Pri tomto príkaze sa zobrazí predvolený textový editor, v ktorom vypíšeme commit správu. Táto správa ma byť stručná a výstižne opisovať opis zmien, ktoré commit obsahuje. Je to znázornené v tejto tabuľke:

```
Sumár commitu
<prázdny riadok>
Opis problému/funkcionality, ktorý commit obsahuje.
<prázdny riadok>
poznámky v odrážkach
```

Po vytvorení commitov je následne potrebné tieto zmeny odoslať do centrálného repozitára.

```
git push origin <názov vetvy>
```

Vedľajšie vetvy je potrebné udržiavať aktuálne vzhľadom na vetvu, od ktorej boli vytvorené. Tento proces je opísaný v časti 5.2.4.3.

5.2.4.3 Spájanie vetiev

Vstup: Dve vetvy, ktoré spolu súvisia

Výstup: Aktualizovaná vetva na základe druhej vetvy

V priebehu vývoja je potrebné *udržiavať vedľajšiu vetvu aktuálnu* na základe vetvy, z ktorej je vytvorená a v určitom momente je potrebné *spojenie vedľajšej vetvy s hlavnou* na základe procesov testovanie, prehliadky kódov a zadanej úlohy. Tento proces prebieha príkazmi:

```
git checkout <názov vetvy>  
git pull origin <názov spájanej vetvy>  
git push origin <názov vetvy>
```

V priebehu aktualizácie a spájania týchto dvoch vetiev môže prísť ku konfliktom, ktoré je následne nutné vyriešiť pred aktualizáciou centrálného repozitára. V prípade riešenia konfliktov je vhodné si vypísať zoznam nespojených súborov príkazom:

```
git diff --name-only --diff-filter=U
```

V prípade potreby je možné použiť aj iné filtre pre tento príkaz. Po vyriešení konfliktov sú tieto zmeny následne pridané v novom commite do centrálného repozitára.

```
git add -A  
git commit  
git push origin <názov vetvy>
```


5.3 Metodika programového kódu

Tento dokument obsahuje metodiku programového kódu pre účely použitia v tímovom projekte tímu 05 ARVis. Pravidlá a postupy opísané v tomto dokumente vychádzajú zo všeobecne odporúčaných konvencií jazyka C++, pričom ale dodržiajú a dopĺňajú pravidlá určené predchádzajúcimi tímami tak, aby bola zachovaná konzistencia v projekte. Opísané su pravidlá pre tvorbu modulov, tried a vlákien, konvencie pre pomenovania a všeobecné pravidlá a konvencie.

Dodržiavanie týchto pravidiel je záväzné pre všetkých členov tímu, ktorí sa podieľajú na tvorbe a úprave programového kódu. Vystupujú v roliach developerov, tj. programátorov a hlavného architekta.

Tento dokument sa nevenuje opisu postupu písaniu komentárov a tvorby dokumentácie, a ani tvorbe testov, nakoľko tieto sú opísané v samostatne v Metodike pre písanie komentárov a dokumentovanie zdrojového kódu a Metodike testovania.

5.3.1 Pravidlá pre tvorbu modulov

Program už v súčasnej verzii pozostáva z množstva modulov, preto pridávanie ďalšej funkcionality, od ktorej nie sú závislé súčasné moduly, bude realizované taktiež formou modulov. Zabezpečí sa tým nízka vzájomná závislosť a možnosť vytvárania modulov ako knižníc, čo sa prejaví rýchlejšou kompiláciou. Tieto pravidlá sú záväzné hlavne pre hlavného architekta. Pravidlá:

- Hlavičkové súbory modulu pre každý modul sú v samostatnom adresári s názvom *< nazov modulu >*. Tento adresár je umiestnený v adresári *include*".
- Zdrojové súbory modulu pre každý modul sú v samostatnom adresári s názvom *< nazov modulu >*. Tento adresár je umiestnený v adresári *"src"*.
- V rámci modulu sa zavedie jednotný namespace s názvom modulu. Všetky triedy a ostatné typy v hlavičkových súboroch sú deklarované pod týmto namespace. V zdrojových súboroch je kvôli lepšej prehľadnosti povolené použiť *using namespace < nazov modulu >* pre modul v ktorom sa nachádzame. Pre ostatné moduly je povinné explicitne ich názov uvádzať. Príklad: *Layout :: ShapeGetter :: getInstance()*
- V prípade potreby je možné rozdeliť modul na samostatné submoduly. Toto sa vykoná analogicky ako v prípade modulov. Každý submodul sa oddelí kvôli prehľadnosti do samostatného adresára a vytvorí sa ďalší namespace.

- Program môže obsahovať voliteľné moduly, ktorých funkčnosť je určitým spôsobom podmienená, napr. modul ovládania grafu pomocou gest je podmienený dostupnosťou Kinectu. Je potrebné tieto moduly navrhovať tak, aby boli zvyšné časti od nich nezávislé. Tento fakt opíšeme v dokumentácii, avšak kvôli budúcim zmenám tento fakt nezohľadňujeme v názve modulu, ani ho nijako špeciálne neoddelujeme.

5.3.2 Pravidlá pre tvorbu tried

V projekte sa vytvárané triedy delia na viacero typov. Prvé delenie je na dátové triedy a funkčné triedy.

- Účelom dátovej triedy je ukladanie dát z tabuľky v databáze, a preto sa jej názov musí zhodovať s názvom tabuľky. Táto trieda môže obsahovať len premenné pre ukladanie dát zo stĺpcov, pričom ich názvy a poradie sa zhodujú. A ďalej obsahuje konštruktor, deštruktor a príslušné metódy pre získanie a nastavenie premenných - get a set metódy.
- Účelom funkčnej triedy je poskytovanie potrebnej funkcionality, a preto má obsahovať hlavne metódy a minimálny počet potrebných premenných. Pri týchto triedach je potrebné sledovať metriky o dĺžke celej triedy, počtu a dĺžke jednotlivých metód a faktore vetvenia. V prípade výrazného prekročenia odporúčaných hodnôt je nutné ich rozdeliť na menšie celky.

Pri vytváraní aplikácie s GUI rozhraním je ju potrebné vytvárať podľa vzoru MVC s dodržaním nasledovných pravidiel:

- Controller triedy obsahujú inštancie potrebných model a view tried a vytvárajú spojenie medzi nimi, čiže sa v nich volajú funkcie *connect()* pre spájanie signálov a slotov. Inú funkcionality tieto triedy neobsahujú. Do ich názvu pridáme sufix "Controller".
- View triedy sú všetky GUI triedy pre interakciu s používateľom. Vytvárame ich dedením od Qt GUI tried. Pri deklarácií, ale aj pri vytváraní objektu je kvôli prehľadnosti potrebné do ich názvu pridať sufix o aký prvok ide, napr. Button. Modul obsahujúci view triedy musí mať navyše prefix "Q".

Pri vytváraní objektov je nutné dodržať stromovú štruktúru. Každý objekt musí mať definovaný vlastný rodičovský objekt.

- Model triedy nemajú obmedzenia pre názov, musia byť však nezávislé od view tried.

5.3.3 Pravidlá pre používanie vlákien

Vykonávanie funkcií náročných na výpočet a dlho trvajúce procesy je potrebné oddeliť od hlavného vlákna pre používateľské rozhranie do osobitných vlákien. Pri vytváraní vlákien pre tento projekt je nutné dodržať nasledovné pravidlá:

- Vytvorí sa nová trieda, ktorej názov sa doplní na konci o slovo "Thread". Táto trieda bude dediť od triedy QThread a prostredníctvom nej sa bude pristupovať k funkcionalite, ktorá sa má oddeliť. Iné spôsoby tvorby vlákien sú zakázané, aby bola zachovaná platformová nezávislosť a konzistencia. Implementovať sa musí minimálne funkcia `run()`. V prípade nekonečne opakovaného procesu je ju potrebné rozšíriť podľa nasledovnej kostry:

```
class <nazov> : public QThread {
    Q_OBJECT
public:
    <nazov>();
    void stop();
protected:
    void run();
private:
    volatile bool stopped;
};

<nazov>:: <nazov>() {
    stopped = false;
}

void <nazov>::run() {
    while (!stopped){
        <požadovana funkcionalita>
    }
    stopped = false;
}

void <nazov>::stop() {
    stopped = true;
}
```

- Po vytvorení objektu vlákna sa jeho vykonávanie začne volaním `< thread > .start()`
- Pre bezpečné ukončenie vlákna je potrebné pre objekt vlákna zavolať obe funkcie `stop()` alebo `quit()` a následne `< thread > .wait()`. Je zakázané použitie funkcie `terminane()`.

- Pre informovanie o stave vlákna je povolené použiť len funkciu *isRunning()* a zachytávanie signálov *finished()*, *started()*, *terminated()* a vlastných definovaných signálov.
- Pre komunikáciu s hlavným vláknom je zakázané použiť iné spôsoby ako mechanizmus signálov a slotov.
- Pre komunikáciu len medzi vedľajšími vláknami a pre riešenie konfliktov je navyše povolené použiť zdieľané premenné s mutexami, semaforami a zdieľanými premennými s využitím nasledovných tried: *QMutex*, *QMutexLocker*, *QReadWriteLock*, *QReadLocker*, *QWriteLocker*, *QSemaphore*, *QThreadStorage < T >* a *QWaitCondition*.
- Implementáciu vlákien je povinné doplniť o logovanie. Minimálne o začatí a ukončení vykonávania vlákna a voliteľne o uspatí a zobudení s využitím funkcie *sleep()* a pri komunikácií - posielaní a zachytávaní signálov.
- Pri vytváraní objektov odvodených od *QObject* a ich používaní v rámci vlákien, musia byť na viac dodržané tieto 3 pravidlá:
 1. Objekty, ktoré sú deťmi iného objektu musia byť vytvorené v rovnakom vlákne ako tento rodičovský objekt.
 2. Všetky objekty vytvorené vo vlákne musia byť vymazané pred vymazaním objektu tohto vlákna.
 3. Objekt musí byť vymazaný vo vlákne, v ktorom bol vytvorený.

5.3.4 Všeobecné pravidlá

- Je odporúčané používanie návrhových vzorov. Pri ich používaní je odporúčané doplniť názvy tried podľa príslušného vzoru, príkladom sú sufixy Adapter, Factory, Visitor.
- Kvôli podpore viacjazyčnosti je nutné všetky reťazce, ktoré sa zobrazujú v používateľskom rozhraní pri implementácii obaliť makrom *tr(< retazec >)*.
- Je odporúčané použitie Qt kontajnerových tried pred STL triedami kvôli funkcii implicitného zdieľania. Je bezpečná aj v rámci vlákien.
- Je nutné udržiavať jednotne naformátovaný zdrojový kód, pričom je odporúčané použiť automatické formátovanie programu QtCreator pre označený text skratkou *Ctrl + I*.

- Názov súboru musí byť rovnaký ako názvom triedy, kt. je v ňom definovaná. Hlavičkový súbor musí byť hneď na začiatku ochránený pred opakovaným includovaním. Všetky definície by musia byť v zdrojovom súbore.
- Ostatné použité súbory je potrebné do projektu začleniť pomocou resource súboru *resource.rc*. Neplatí to ale pre konfiguračné súbory a súbory, ktoré je potrebné môcť upravovať. Tieto musia byť v adresári "resources".

5.3.5 Konvencia pomenovaní názvov

1. V názvoch reprezentujúcich typy (triedy, vlastne typy pomocou type def,..) musí každé slovo začínať veľkým písmenom, ostatné písmena sú malé

```
Line , SavingsAccount
```

2. Názov premenných začína s malým písmenom a každé ďalšie slovo s veľkým písmenom

```
line , savingsAccount
```

3. Konštanty je nutné vytvárať ako vymenovaný typ. Je zakázané vytvárať ich pomocou *#define*. V názve majú všetky písmena veľké. Konštanty môžu mať spoločný prefix.

```
enum NoiseQuality {
    QUALITY_FAST = 0 }
```

4. Triedy dediace od výnimiek by mali mať sufix Exception

```
class AccessException
```

5. Názvy metód a funkcií musia byť slovesá a začínajú malým písmenom, každé ďalšie slovo začína veľkým písmenom.

```
getName() , computeTotalWidth()
```

6. Názvy reprezentujúce typy v templatách sú veľké písmená.

```
template <class C, class D> ...
```

7. Skratky v názvoch od druhého znaku pokračujú malým písmenom.

```
exportHtmlSource (); // NOT: exportHTMLSource ();
```

8. Všeobecné premenné by mali mať rovnaký názov ako ich typ

```
void setTopic (Topic* topic) // NOT: not value, aTopic, t
connect (Database* database) // NOT: db, oracleDB
Point startingPoint, centerPoint;
```

9. Názvy premenných z dlhšou platnosťou (dlhší skope) by mali mať dlhšie názvy, premenné s kratšou platnosťou kratšie názvy.
10. Vyhybať sa opakovaniu názvov typov, keď sú implicitné.

```
line.getLength (); // NOT: line.getLineLength ();
```

11. Môžeme použiť slovo compute, find, initialize v názvoch metód, ktoré majú príslušnú funkciu.

```
valueSet->computeAverage ();
matrix->computeInverse ();
vertex.findNearestVertex ();
matrix.findMinElement ();
printer.initializeFontSet ();
```

12. Množné číslo je dovolené používať len pri poliach a kolekciách.

```
vector<Point> points; int values [];
```

13. Je nutné použiť jednotný prefix "n" pri metódach a premenných a vracajúcich alebo reprezentujúcich počet.

```
nPoints, nLines
```

14. Sufix "No" musia mať premenné reprezentujúce číslo alebo poradie objektu.

```
tableNo, employeeNo
```

15. Názvy iterátorov môžu byť malé písmená. V zložitejších cykloch to musia byť slová.

```
for (vector<MyClass>::iterator i = list.begin(); i != list.end();>
    >> i++) {
    Element element = *i; ...
}
```

16. Premenné a metódy a vracajúce bool musia mať prefix is, has, can, should.

```
isSet , isVisible , isFinished , isFound , isOpen
bool hasLicense ();
bool canEvaluate ();
bool shouldSort ();
```

17. Je nutné dodržiavať komplementárne názvy pri komplementárnych operáciách.

```
get/set , add/remove , create/destroy , start/stop , insert/delete ,
increment/decrement , old/new , begin/end , first/last , up/down , min>
>> /max,
next/previous , old/new , open/close , show/hide , suspend/resume , >>
>> etc .
```

18. Neoznačovať explicitne pointre a referencie.

```
Line *line ; // NOT: Line* pLine ;
// NOT: Line* linePtr ;
```

19. Nepoužívať negované názvy.

```
bool isFound ; // NOT: isNotFound
```

5.3.6 Všeobecné konvencie

1. Je zakázané používanie globálnych premenných.
2. Musí sa používať angličtina.
3. Je nutné používať zapuzdrenie.
4. Je zakázané používanie "magických čísel" ako 0 a 1. Musia sa použiť konštanty alebo hodnoty z konfiguračného súboru.
5. Typová konverzia musí byť explicitná.

```
floatValue = static_cast<float>(intValue) ;
```

6. V definícii cyklu musí byť len iteračná premenná a ihneď inicializovaná. Vyhnúť sa do-while cyklom a používaniu break a continue.
7. Nepoužívať metódy v podmienkach.

```
File *fileHandle = open(fileName, "w");  
if (!fileHandle)  
// NOT: if (!(fileHandle = open(fileName, "w")))
```

8. Desatinné čísla vždy písať s bodkou.

```
double total = 0.0;      // NOT: double total = 0;  
double speed = 3.0e8;   // NOT: double speed = 3e8;
```


5.4 Metodika prehliadky kódu

Pri práci na väčšom projekte v tíme sa stretávame s problémom písania zdrojových kódov. Je potrebné udržať konzistenciu kódu, čo je pri práci viacerých ľudí na projekte problematické. Preto sa pri väčších projektoch zavádzajú takzvané konvencie písania zdrojových kódov. Dodržanie týchto konvencií však nie je ľahké a preto je pre každý väčší projekt nevyhnutná prehliadka a následná revízia kódu. Touto prehliadkou dosiahneme celkovú lepšiu a rýchlejšiu prácu na projekte pri programovaní.

Účelom tejto metodiky je zabezpečenie konzistencie, prehľadnosti a čitateľnosti kódu. Popisujeme postup práce po odovzdávaní vytvoreného zdrojového kódu. Zameriavame sa na kroky potrebné pre vykonanie prehliadky kódu aby mohla byť následná revízia rýchla a jednoduchá. Proces prehliadky kódu zavádzame v nástroji github pre správu a manažment verzií zdrojového kódu.

5.4.1 Slovník pojmov

- **Github** – nástroj pre manažment verzií zdrojového kódu. Dostupný na stránke www.github.com. Okrem toho podporuje funkcie ako komentovanie kódu a hlásenie bugov, ktoré sa využijú pri prehliadke kódu.
- **Branch** – vetva. Kópia momentálneho stavu zdrojového kódu na Githube. Vytvára sa aby mohli developeri pracovať na vlastnej funkcionalite programu a nenastali konflikty keď pracujú naraz s rovnakým kódom.
- **Commit** – zverejnenie časti vypracovaného kódu vo vlastnej branchi na Githube. Väčšinou ide o ucelenú časť kódu s pridanou funkcionalitou.
- **Task** - úloha, pridelená členovi tímu pre vypracovanie v danom šprinte.
- **Feedback** - spätná väzba. V prípade prehliadky kódu sú to komentáre ku jednotlivým častiam kódu.
- **Issue** - Vytvorená úloha pre členov tímu, ktorí sú priradení pre vykonanie prehliadky zdrojového kódu.

5.4.2 Zoznam nadväzujúcich metódik

2.1. Metodika písania programového kódu

2.2. Metodika písania komentárov

2.3. Metodika manažmentu verzií zdrojového kódu pomocou nástroja Git

2.4. Metodika testovania

5.4.3 Role a ich úlohy

Rola	Úloha
Developer	<ul style="list-style-type: none"> - Vytvoriť funkcionality programu a zverejniť ju prostredníctvom zdrojových kódov na stránke pre manažovanie kódu (Github) - Oprava nahlásených chýb podľa metodiky písania programového kódu (2.1) - Nahlásiť dokončenie funkcionality GitHub Masterovi
Reviewer	<ul style="list-style-type: none"> - Prezrieť zdrojový kód na githube a podľa metodiky konvencie písania programového kódu - V prípade zisteného problému vytvoriť komentár k danému riadku kódu - Rozhodovanie sa o správnosti kódu a jeho celkové prijatie/odmietnutie.
GitHub Master	<ul style="list-style-type: none"> - Vykonáva merge vetvy vypracovanej developerom s hlavnou develop vetvou - Možnosť konečného prijatia funkcionality pre následné testovanie alebo odmietnutia pre prepracovanie.
Člen tímu	<ul style="list-style-type: none"> - V rámci prehliadky kódu sa môže stať developerom alebo reviewerom

5.4.4 Metodika prehliadky kódu

Po ukončení vývoja funkcionality programu nastáva vykonanie commitu a odovzdanie vytvorenej funkcionality programu na Github. Tento proces prebieha vo vlastnej vetve vytvorenej pre daný task, ktorý zastrešuje ucelenú funkcionality programu podľa metodiky pre prácu s githubom (2.3). Pred spojením danej vetvy s hlavnou develop vetvou je však potrebný proces prehliadky a revízie kódu, po ktorom môže nasledovať proces testovania (2.4).

5.4.4.1 Ukončenie funkcionality a commit na Github

Vstup: Zdrojový kód

Výstup: Zdrojový kód dostupný na Githube

Zúčastnený: Developer

Poradie	Názov	Kapitola
1.	Ukončenie funkcionality a commit na Github	5.4.4.1
2.	Vytvorenie issue pre prehliadku	5.4.4.2
3.	Vykonanie prehliadky kódu	5.4.4.3
4.	Uzavretie issue pre prehliadku	5.4.4.4
5.	Rozhodnutie o úprave kódu	5.4.4.5

Pri vytváraní funkcionality pre systém sa riadime metodikou pre github(2.3), kde sa vytvárajú branchy pre jednotlivé funkcionality na krátky čas. Toto je dôvod, prečo nemusí prehliadka kódu prebehnúť zvlášť pre každý github commit, ale je možné zahájiť proces prehliadky kódu až po dokončení funkcionality.

5.4.4.2 Vytvorenie issue pre prehliadku

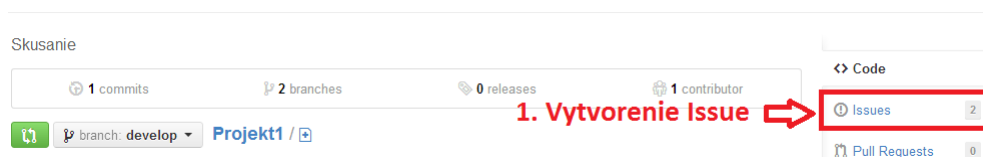
Vstup: Zadanie údajov pre issue

Výstup: Vytvorená issue pridelená reviewerom

Zúčastnený: Developer/Github master, Členovia tímu

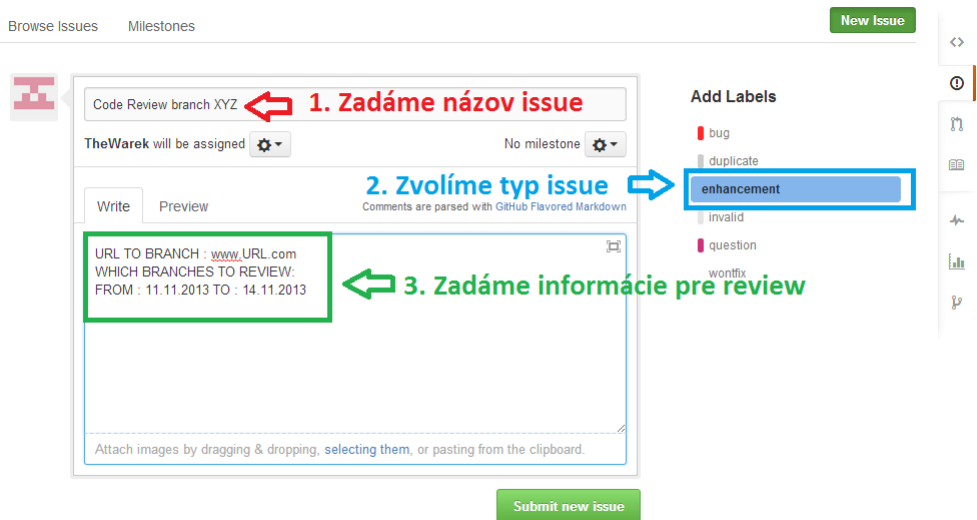
Developer alebo github master majú právo po ukončení funkcionality vytvoriť úlohu pre 2 členov tímu - reviewerov. Odporúčanie je spojiť reviewerov zároveň s testovaním, kde je podľa metodiky testovania (2.4) nutné funkcionality otestovať na rôznych platformách. Vytvorenie issue prebieha nasledovne:

1. Vo vetve, ktorú chceme prehliadnúť spustíme issue.

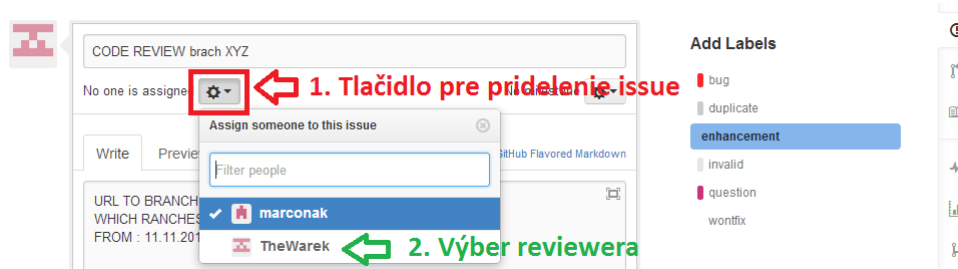


2. Nastavíme parametre pre issue:

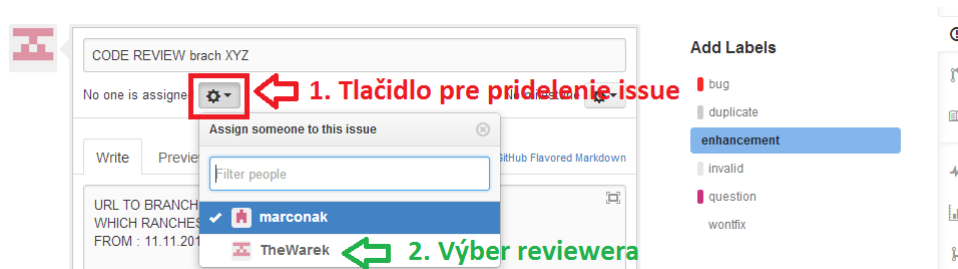
- (a) Názov: "Code Review branchä pridáme názov samotnej vetvy.
- (b) Zvolíme typ issue. Pre prehliadku kódu to je: "enhancement".
- (c) Zadáme parametre pre reviewera podľa obrázku. Nesmie chýbať odkaz na branch. Predpokladá sa prehliadka celého branchu. V prípade prehliadky jednotlivých commitov ich konkretizujeme.



3. Zvolíme komu chceme issue priradiť.



4. Potvrdíme vytvorenie tlačidlom "Submit new issue". Tento proces opakujeme aj pre druhého reviewera.



Účastníci budú upozornení e-mailom a môžu začať vykonávať prehliadku kódu.

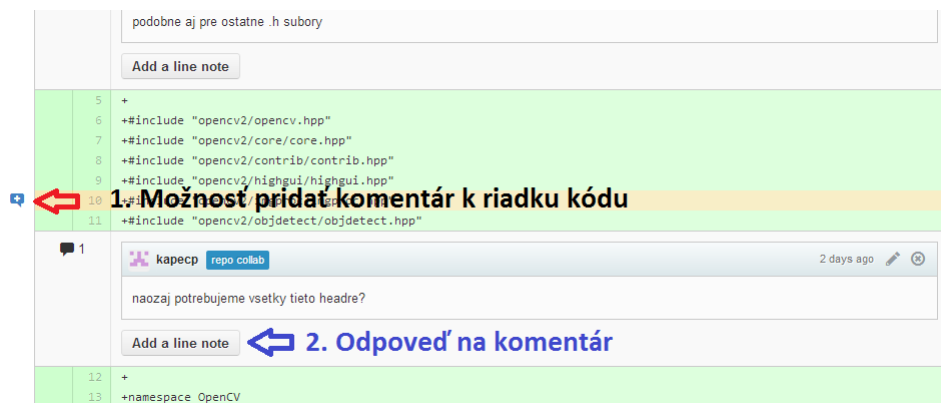
5.4.4.3 Vykonanie prehliadky kódu

Vstup: Pridanie issue pre reviewera

Výstup: Vykonaná preliadka kódu

Zúčastnený: Reviewer

Reviewer podľa pokynov vykoná prehliadku kódu. Účelom je mať prehľadný zdrojový kód podľa metodiky písania programového kódu (2.1). Konvencie pre reviewera pri prehliadke kódu môžeme nájsť v prílohe. V prípade konfliktu reviewer opíše prostredníctvom github komentáru ku riadku kódu problém a zároveň ponúkne riešenie. Revieweri medzi sebou môžu komunikovať prostredníctvom odpovede na komentár.



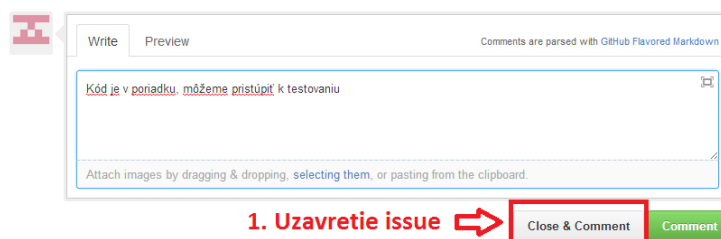
5.4.4.4 Uzavretie issue pre prehliadku

Vstup: Vykonaná prehliadka kódu

Výstup: Celkový komentár ku kódu a uzavretie issue

Zúčastnený: Reviewer

Po vykonaní prehliadky napíše reviewer komentár k issue, kde vykoná rozhodnutie o tom, či daný kód odsúhlasí a po jeho menšej úprave ho posunie ho k fáze testovania, alebo ho zamietne a odporučí prepracovať.



5.4.4.5 Rozhodnutie o úprave kódu

Vstup: Hodnotenie od reviewerov o zdrojovom kóde

Výstup: Rozhodnutie o úprave/oprave kódu

Zúčastnený: Github master, Developer

Po skončení revízie od oboch reviewerov vykoná github master rozhodnutie o tom, či sa daná funkcionálnosť posunie ďalej na testovanie, alebo sa vráti developerovi na opätovné prepracovanie.

5.4.5 Konvencie pri prehliadke zdrojového kódu

Jednotlivé body revízie úzko súvisia s metodikou písania programového kódu (2.1).

1. Skontrolovať hlavičkové súbory
 - (a) Umiestnenie súboru
 - (b) Include, skontrolovať či sa tam nenachádzajú nepotrebné súbory
 - (c) Pozrieť sa na funkcie get a set pre potrebné privátne premenné
 - (d) Použitie namespace
2. Skontrolovať .cpp súbory
 - (a) Umiestnenie súboru
 - (b) Include na hlavičkový súbor
 - (c) Skontrolovať názvy premenných
 - (d) Skontrolovať thready podľa stanovenej konvencie
 - (e) Skontrolovať komentáre ku jednotlivým funkciám
3. Skontrolovať CMakeLists a import jednotlivých súborov
 - (a) Nepoužívať "require" pri importe knižníc, riešiť ich ako optional prvky
 - (b) Skontrolovať resources v prípade, že ich funkcionálnosť vyžaduje
4. Skontrolovať používanie angličtiny

5.5 Metodika pre písanie komentárov a dokumentovanie zdrojového kódu

V rámci nášho projektu rozvíjame už existujúci program, ktorý je značne rozsiahly, čo sa týka zdrojových kódov. Aby sme sa v tomto už existujúcom projekte vyznali, a zároveň aby sme sa dokázali jednoducho orientovať v našich vlastných zdrojových kódoch potrebujeme definovať metodiku pre písanie zdrojových kódov. Pre zjednodušenie dokumentovania zdrojových kódov a vytvoreného programu na technickej úrovni budeme v rámci projektu používať systém Doxygen, ktorý umožňuje generovanie štrukturovaných dokumentov vo forme pdf alebo html.

Doxygen je systém vytvorený pre generovanie dokumentácie z komentárov umiestnených v zdrojovom kóde. Požíva pritom špecifické označenia, dokáže spracovať značky špecifické pre latex a html a umiestniť ich do dokumentácie. Okrem toho, keďže už máme existujúci projekt, ktorý je sčasti okomentovaný, zaujíma nás aj jeho vlastnosť, že klasické komentáre ignoruje, spracúva iba komentáre, ktoré sú umiestnené v špecificky označených blokoch. Dokáže však z existujúceho "neokomentovaného" zdrojového kódu vygenerovať dokument, ktorý popisuje základné informácie o triedach a o štruktúre zdrojových kódov. Vďaka tomuto nemusíme meniť, mazať ani nijako upravovať už existujúce komentáre a môžeme pokojne pokračovať s implementáciou softvéru a popisovaniu ho komentármi, ktoré doxygen už dokáže spracovať.

Táto metodika sa nevenuje testovaniu ani code review, keďže pre obidva procesy sú v rámci tímu zdokumentované v iných metodikách. Okrajovo však tieto procesy spomenieme a preto sme definovali rolu testera a code review-era.

5.5.1 Role a povinnosti

Rola	Popis roly
Programátor	- Člen tímu, ktorý píše zdrojový kód a popri tom ho rozumne a zrozumiteľne komentuje.
Code reviewer	- Člen tímu, ktorý bol poverený kontrolou pridanej funkcionality a spôsobu akým bola naprogramovaná.
Tester	- Člen tímu, ktorý testuje funkčnosť programu.
Člen (každý)	- Vyberá si úlohy z Product Backlogu
Product owner	- Vedúci. V prípade, že sa rozhodne prezerat' kód, doplnit' doňho inú funkcionality alebo nájde chybu, ktorú tím nenašiel pri code review ani pri testovaní.

5.5.2 Základné pravidlá pre písanie komentárov

5.5.2.1 Popis riadku kódu (class members)

V prípade okomentovania jedného riadku kódu, kde sa jedná o jeden špecifický riadok v rámci metódy, prípadne popisu člena niektorej z tried v rámci definície triedy používame skrátenú notáciu pre komentár.

Komentár sa v tomto prípade môže zapísať pred daným riadkom kódu, alebo za ním v tom istom riadku. Popis v nasledujúcom riadku síce doxygen správne priradí k danej časti kódu, ktorú chceme okomentovať, avšak pre prehľadnosť zdrojového kódu je tento spôsob nepoužiteľný.

Komentár potom vyzerá nasledovne:

```
///Definicia pred parametrom.  
int param; ///Definicia po parametri.
```

5.5.2.2 Popis metódy

Metóda sa popisuje v rámci bloku komentáru, ktorý je definovaný nasledovne:

```
/**  
* @author Autor: Jozko Mrkvicka  
* @param x vstupny parameter typu integer  
* @param y druhy vstupny parameter typu integer  
* @return spocitane vracia sucet x+y  
* @brief Vrati sucet dvoch vstupnych parametrov  
*/
```

Pričom nutné súčasti popisu metódy sú:

- popis funkcionality metódy na najvyššej úrovni s čo najmenej technickými detailami, aby bolo okamžite jasné čo daná funkcia robí,
- vstupné parametre,
- návratová hodnota.

5.5.2.3 Popis triedy

Popis triedy sa taktiež nachádza v rámci bloku komentárov, pričom príklad časti takéhoto popisu je na obrázku.

Popis triedy pritom musí spĺňať nasledujúce body:

- popis triedy sa nachádza v hlavičkovom súbore (*.h),
- meno autora triedy,
- popis jednotlivých metód v triede,
- krátky popis premenných triedy.

```
/** @brief Trieda pouzita pre reprezentacne ucely
 *
 * @author Autor: Jozko Mrkvicka
 * @author Code review: Janko Kukurica
 * @date Oktober 2013
 */
class myClass{
    /** Zakladny konstruktor.
     */
    myClass(float x, float y, float z);
    /** Zakladny destruktork.
     */
    ~myClass();

public:
    /** @brief Vypise vsetkych privatnych clenov triedy.
     */
    void print();
    /**@brief Vypocita vzdialenost od stredu.
     */
    float getDistance(float x, float y, float z);

private:
    float _x0;/**<x-koordinat*/
    float _y0;/**<y-koordinat*/
    float _z0;/**<z-koordinat*/
};
```

Pri popisovaní metód a premenných triedy sa využívajú pravidlá definované v podkapitolách vyššie venovaných tejto časti komentárov.

5.5.2.4 "To do"

V prípade zistenej funkcionality, ktorú treba doplniť sa k funkcii alebo triede, kde daná funkcionalita chýba doplní popis budúcej funkcionality, pričom táto je označená značkou "@todo".

Systém doxygen potom sám vytvorí prehľadný zoznam jednotlivých položiek, ktoré treba dokončiť čím zjednodušuje hľadanie danej časti kódu, ktorú je potrebné upraviť alebo rozšíriť.

Todo List

```
Member edu::uiowa::icts::NewClass.main (String[] args)  
Make it do something.
```

Tieto bloky komentárov musia obsahovať minimálne:

- autora požiadavky,
- definovanú funkcionality,
- dôvod, prečo je táto funkcionalita potrebná.

5.5.2.5 "Chyby"

Rovnako ako zoznam budúcej funkcionality dokáže doxygen spracovávať aj chyby zistené a popísané v rámci zdrojového kódu. Tieto chyby sa potom popíšu v rámci bloku kódu prislúchajúcemu k danej funkcii, ktorá je chybná pričom opis je identifikovaný značkou "@bug".

Bug List

```
Member edu::uiowa::icts::NewClass.main (String[] args)  
To be Microsoft Certified, must never deallocate  
memory.
```

Tieto bloky komentára musia obsahovať minimálne:

- autora komentára,
- popis nájdenej chyby.

Blok môže obsahovať aj informáciu o tom, kedy sa chyba prejavila, prípadne ako sa prejavila.

5.5.3 Procesy

5.5.3.1 Vytvorenie novej triedy (metódy)

Účastníci: Programátor

V rámci hlavičkového súboru sa píše detailný popis triedy, čo trieda robí a k čomu slúži, definujú sa metódy a premenné tejto triedy. Metódy triedy sa v hlavičke popisujú krátkym popisom, pričom je možné doplniť detailnejší opis v rámci samotného zdrojového kódu metódy. Pri vytvorení opisu triedy je povinné definovať meno autora triedy, pri popisoch metód v rámci triedy toto nie je nutné. Pri veľmi jednoduchých metódach, jednoriadkových metódach alebo pri "getter", "setter" metódach sa meno autora nepíše nikdy. Pri metódach, ktoré majú určitú rozsiahlejšiu funkcionálnosť sa meno autora môže avšak nemusí písať.

5.5.3.2 Pridanie poznámky pri code review

Účastníci: Programátor, Code reviewer, Product owner, Tester

V prípade, že pri code review bol člen tímu, ktorý bol poverený kontrolou tímu nútený prezerat' zdrojový kód vo svojom vývojom prostredí, je možné aby v kóde nechal poznámku k zdrojovému kódu. V tomto prípade napíše blok komentáru, kde ako autora označí seba a dopíše poznámku k časti kódu. Potom tento komentár vyzerá nasledovne:

```
/**
 *@author Code reviewer: Janko mrkvicka
 *@note Tato metoda by sa dala prerobit tak aby sa vykonavala rychlejsie.
 */
```

Pridanie takejto poznámky je samozrejme možné aj mimo code review napríklad pri testovaní alebo pri práci na inej časti kódu.

5.5.3.3 Zapísanie plánovanej funkcionality

Účastníci: Programátor, Product owner

V rámci práce na systéme môže nastať situácia, kedy jeden z programátorov vytvorí funkcionality, ktorá nadväzuje na inú, takú na ktorej pracoval alebo pracuje iný člen tímu. V prípade, že programátor potrebuje túto funkcionality rozšíriť tak napíše do zdrojového kódu blok kódu, kde uvedie svoje meno a popis budúcej funkcionality.

Taktiež sa môže stať, že product owner považuje niektorú funkcionality za neúplnú, prípadne ju chce z akéhokoľvek dôvodu rozšíriť. V tomto prípade rovnako vytvorí blok kódu, kde definuje túto funkcionality.

Tento blok kódu pritom bude spĺňať požiadavky definované v základných pravidlách v podkapitole "To do", ktorá popisuje tieto bloky komentárov.

5.5.3.4 Zapísanie zistenej chyby

Účastníci: Programátor

Zistenie chyby môže nastať v troch prípadoch.

V prvom prípade sa nájde daná chyba pri code review. V tomto prípade sa postupuje podľa metodiky pre code review, avšak je možné zapísať chybu aj do zdrojového kódu prostredníctvom bloku komentára.

V druhom prípade sa jedná o testovanie, kde sa rovnako ako v predošlom prípade postupuje podľa metodiky pre testovanie avšak je možné dopísať informácie o chybe aj do zdrojového kódu.

V treťom prípade sa chyba nájde neskôr, po kontrole a testovaní zdrojového kódu. Môže sa stať, že chyba, ktorá nebola doteraz zjavná sa prejaví napríklad pri implementácii novej funkcionality. V tomto prípade sa táto chyba zapíše do komentárového bloku. Tento blok spĺňa požiadavky vyššie definované v podkapitole chyby.

5.6 Metodika evidencie úloh

Zavedenie jednotného postupu pre evidenciu úloh vychádza z "granularity" úloh projektu, a teda celkovo veľkého množstva plánovaných úloh pre tím, ako aj jednotlivcov. Potreba evidencie vzniká zároveň s plánovaním, pričom sú vzájomne prepojené. Ohraničujúce sú procesy tvorenia a ukončenia úloh. Metodika pre evidenciu úloh sa teda zaoberá procesmi zahŕňajúcimi zapisovanie, kontrolu a aktualizáciu údajov. Konkrétna metodika sa opiera o využitie aplikácie pre evidenciu úloh v tíme. Webová aplikácia Redmine je rozšírená o niektoré moduly pre názorné grafické sledovanie vývoja. Toto prostredie je dostupné pre všetkých členov zúčastnených na projekte.

Aby bola metodika použiteľná, musí sa ňou riadiť celý tím, pričom v stratégií vývoja SCRUM, je rovnako dôležitá pre každého člena počas celého obdobia projektu.

5.6.1 Slovník pojmov

- **Redmine** - webová aplikácia pre evidenciu úloh a sledovanie priebehu projektu, poskytuje Burndown chart a pod. SCRUM nástroje.
- **Scrum** - flexibilná stratégia na tvorbu produktov v tíme.
- **Scrum Master** - člen tímu zodpovedný za priebeh šprintov podľa SCRUM. V rámci projektu tieto vetvy existuje počas celého vývoja.
- **Product backlog** - ohodnotený zoznam požiadaviek na systém, čakajúci na realizáciu.
- **Sprint** - dané časové obdobie, v ktorom prebieha vývoj.
- **Sprint backlog** - zoznam požiadaviek určený k splneniu do konca obdobia vývoja.
- **Tabuľa úloh** - fyzická tabuľa s lepiacimi papiermi s označením úlohy a zodpovedného člena.

5.6.2 Role a povinnosti

Rola	Povinnosti
Moderátor (Scrum Master)	- Vedie časť stretnutia s hlasovaním a označovaním nových úloh zložitou - Používa pre dočasný záznam nástennú tabuľu
Zapisovateľ	- Počas moderátorovho vstupu zapisuje a eviduje informácie v Redmine Robí papierovú evidenciu na Tabuli úloh, pre kontrolu a optický prehľad
Člen s úlohou	- Spravuje svoje pridelené úlohy v Redmine
Člen (každý)	- Oboznamuje ostatných s výsledkami pri určených stretnutiach - Vyberá si úlohy z Product Backlogu - Hlasuje o zložitosti a dobe trvania - Aktualizuje stavy svojich úloh
Product owner	- Vyberá úlohy, ktoré treba splniť - Pomáha určovať nové požiadavky do Product Backlogu

5.6.3 Procesy a evidencie

Kompletná evidencia pre projekt sa skladá z jednotlivých procesov, rovnakých pre každú úlohu. V týchto procesoch sú zahrnuté viaceré z definovaných rolí, pričom každá osoba môže byť zahrnutá vo viacerých roliach.

Poradie	Názov	Kapitola
1.	Pripravenie prostredia pre šprint	5.6.3.1
2.	Zaevidovanie novej úlohy	5.6.3.2
3.	Editovanie úlohy	5.6.3.3
4.1.	Aktualizácia stráveného času	5.6.3.4
4.2.	Aktualizácia vypracovania úlohy	5.6.3.4
5.	Zmena stavu	5.6.3.5
6.	Evidovanie ukončenia úlohy	5.6.3.6

5.6.3.1 Pripravenie prostredia pre šprint

Vstup: Správna inštalácia aplikácie Redmine a pluginu AgileDwarf

Výstup: Možnosť evidencie úloh projektu

Zúčastnený: Scrum Master

Pokiaľ je vytvorený projekt a zadaný používateľ, je možné prejsť na kapitolu 5.6.3.2. Aby mohol systém fungovať interaktívne pre každého používateľa, je nutné zadať jednotlivé práva používateľov a zadanie projektov. Tomu predchádza vytvorenie projektu (napr. "Letný semester"). Používatelia sa samostatne registrujú s menom vo forme "Meno.Priezvisko". V nastaveniach projektu sa potom dajú pridať používatelia pre prácu v danom projekte. Scrum master ich pridáva so statusom "redaktor", aby vedeli upravovať a dopĺňať informácie, nie však robiť zásadné zmeny, ako mazanie alebo menenie nadržaného šprintu. Pripravenie Tabule úloh, vertikálne rozčlenenie na vytvorené, rozpracované a ukončené úlohy.

5.6.3.2 Zaevidovanie novej úlohy

Vstup: Naplánovanie novej úlohy

Výstup: Pridanie úlohy a detailov do aplikácie Redmine, nový lístok na Tabuli úloh

Zúčastnený: Moderátor, Zapisovateľ

Počas stretnutia sa na základe informácií od Product Ownera zostavuje zoznam úloh na riešenie. Zapisovateľ v aplikácii Redmine v časti "Nová úloha" vytvorí novú úlohu a zadá jej parametre, dĺžku trvania a prioritu. Pokiaľ úloha bola už v minulosti riešená a ešte nebola dokončená, zadá stav "In Progress". Pokiaľ je dohodnuté, že sa úloha bude riešiť v danom šprinte, môže sa nastaviť cieľová verzia (výber daného šprintu). Zapisovateľ vytvorí lístok s názvom úlohy a menom študenta a pridá ho do kategórie na tabuľu.

5.6.3.3 Editovanie úlohy

Vstup: Vznik zmien na základe plánovania, úloha existujúca v systéme Redmine

Výstup: Zmeny v nastavení pôvodnej úlohy v systéme, zmenený lístok na Tabuli

Zúčastnený: Moderátor, Zapisovateľ

Úloha môže byť editovaná za rôznymi účelmi. Jedným z nich je pridelenie úlohy konkrétnemu členovi, alebo rôzne zmeny v názvoch, alebo zaradenia v šprintoch. Všetky zmeny sa vykonávajú v rovnakom prostredí ako pri vytváraní novej úlohy.

5.6.3.4 Aktualizácia

Aktualizácia stráveného času

Vstup: Bol strávený čas na danej úlohe

Výstup: Pridanie nových údajov ku úlohe, zmena informácií sa prejaví na celkovom grafickom zobrazení, lepší prehľad v projekte

Zúčastnený: Člen s úlohou

Pridanie informácie o strávenom čase ovplyvňuje Burndown graf, ktorý opisuje stav riešenia úloh v závislosti od plnenia časového plánu.

Aktualizácia vypracovania úlohy

Vstup: Nutná práca s kladnými výsledkami

Výstup: Pridanie percentuálneho ohodnotenia aktuálneho stavu úlohy z pohľadu kompletného vyriešenia

Zúčastnený: Člen s úlohou

Pridávanie týchto informácií je podmienené aspoň čiastočným výstupom práce na projekte, pretože priamo ovplyvňuje ďalší odhad zložitosti.

1. AKTUALIZÁCIA STRÁVENÉHO ČASU

2. AKTUALIZÁCIA VYPRACOVANIA ÚLOHY

Obr. 4: Možnosti editovania formulárov

5.6.3.5 Zmena stavu

Vstup: Dosiahnutie alebo nedosiahnutie cieľov, potrebná zmena v riešení úlohy

Výstup: Úlohy v stave "feedback" alebo "resolved" značia ostatným určeným členom, v

akom štádiu sa nachádza, prípadne aby skontrolovali a odsúhlasili ukončenie

Zúčastnený: Člen s úlohou, Scrum Master

Zmenou stavu dáva člen jasnejšie najavo zmeny v priebehu riešenia. Zároveň tieto zmeny môže využiť na získanie spätnej väzby ostatných členov alebo tiež odsúhlasenia ukončenia práce na úlohe. V časti „Tasks“ sa nachádza formulárové prostredie ako na obrázku č. 1. V prípade potreby komentáru od spolupracovníkov, zadá stav "Feedback", ináč zadá stav "Resolved". Pokiaľ bola úloha úplne splnená, zadá stav 100%, aj v prípade, že doba vypracovania bola kratšia alebo dlhšia.

5.6.3.6 Evidovanie ukončenia úlohy

Vstup: Úlohy už v neriešenom stave (feedback, resolved)

Výstup: Ukončenie priebehu úlohy v systéme, zmena poradia lístkov na Tabuli, priradenie nových úloh

Zúčastnený: Scrum Master

Scrum Master potvrdí členom dostatočné vypracovanie ich úlohy a priradí status "closed", alebo znovu priradí úlohe status "new" v prípade, že úloha nebola vypracovaná dostatočne. Takto núti člena pracovať na úlohe znova.

5.7 Metodika testovania softvéru

V tejto časti je opísaná metodika testovania softvéru v projekte tímu č. 5 ARVis. Pravidlá a konvencie obsiahnuté v tomto dokumente sú definované na základe agilnej metodiky SCRUM. SCRUM procedúry slúžia ako návod pre naše testovacie procedúry.

5.7.1 SCRUM

SCRUM framework pozostáva z iterácií, v ktorých sa vyvíja softvér agilne. Niektoré pojmy a špecifiká v tejto metodike hrajú dôležitú rolu aj v testovaní softvéru.

Product Backlog

Product owner má celkový prehľad o projekte a on definuje používateľské príbehy, ktoré sú často zároveň vlastnosťami softvéru. Takým spôsobom, akým definuje používateľské príbehy a ich priority, takisto definuje aj to, čo sa má a do akej miery testovať.

Sprint Planning Meeting

Počas stretnutia si členovia tímu rozdelia úlohy v rámci šprintu. Akonáhle si člen priradil úlohu, má myslieť aj na testovanie funkcionality, ktorú pridáva do softvéru.

Sprint

Počas šprintov sa vyvíja a testuje softvér. Tester a vývojári spolupracujú vo vypracovaní testovacích úloh. Unit testy sú zadané a vykonané. Testy na funkcionality sú taktiež potrebné, aby nedošlo ku konfliktu v porovnaní s požiadavkami vedúceho. Po každej reintegrácii jedného komponentu je potrebné vykonať zadané testy, aby bol komponent kompatibilný s ostatnými časťami softvéru.

Review Meeting

Kým sa spraví prehľadka aktuálnej pridanej funkcionality, testery by mali byť schopní identifikovať problémy s novou verziou a definovať ich pre novú iteráciu, t.j. definovať testy.

Retrospective Meeting

Retrospektíva celej iterácie nám dáva celkový pohľad o tom, ako by sa mohla zefektívniť produktivita pre ďalšie iterácie.

5.7.2 Role a ich povinnosti

- Testerí počas stretnutí Sprint Planning Meeting poskytujú informácie o testovacom prostredí a o prípadoch testovania.
- Tester má za úlohu analyzovať všetky úlohy a pripraviť komplexné, resp. automatizované prostredie na testovanie.
- Musia sa definovať akceptačné kritéria pre používateľské príbehy na základe používateľskej skúsenosti a výkonnostných otázok. Definované akceptačné kritéria sa môžu líšiť oproti tomu, ako si to predstavuje vedúci, takže validácia je potrebná.
- Ak tím a vedúci poskytnú spätnú väzbu, akceptačné kritéria sú nanovo definované a vytvoria sa testovacie prípady.
- Ak sa používajú automatizované testovacie frameworky, testerí vyprodukovujú unit testy predtým, než vývojári ukončili svoja programátorské úlohy.
- Akonáhle je hotová implementácia jedného používateľského príbehu, testovacie prípady by mali byť vykonané - takýmto spôsobom dôjde k poznaniu chýb a limitácií softvéru.
- Prieskum by sa mal konať po každom šprinte s cieľom lepšieho nahliadnutia a inšpekcie softvéru.
- Testerí s predstihom zdôraznia všetky identifikované výstupy počas Review Meetings, aby ostatní členovia tímu neboli týmto zmätení v ďalšom priebehu stretnutia.

5.7.3 Unit testovanie

Unit testy sa konajú v každom šprinte s cieľom testovania funkcionality každého komponentu. Pričom pre každý jeden komponent sa napíše testovací scenár a ten je následne nasadený na ten komponent, aby sa zhodnotila jeho funkčnosť. Motivácia unit testov spočíva v tom, aby sa zistilo, nakoľko daný komponent korešponduje s biznis logikou.

Unit testy píše vývojári a testerí. Vytvorenie takýchto testov môže byť časovo náročné a aby sa optimalizovala efektívna práca na projekte, časti s jednoduchým zdrojovým kódom nebudú testované ako napr. *getters* a *setters*.

V agilnom vývoji je ťažké definovať unit testy na dlhšiu dobu, takže sa píše v každej iterácii (t.j. v každom šprinte). Môže sa však stať, že sa napíšu aj vopred - v prípade, že požiadavky na systém sú jednoznačné a tím ich jasne identifikoval. Unit test obsahuje:

- logiku, ktorá definuje ciele daného komponentu, ktorý sa testuje,
- komponent, ktorý sa testuje,
- zdrojový kód, ktorý testuje,
- správu o stave testovania, ktorá je spätne dostupná.

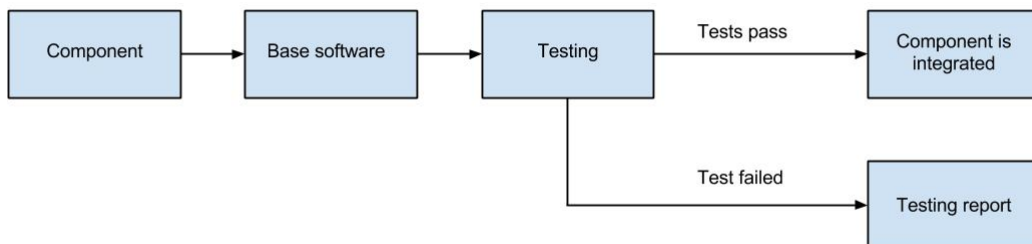
Unit testy by sa mali rozvíjať spolu so softvérom - refaktoring unit testov sa koná v každom šprinte, ak to je žiadúce.



5.7.4 Integračné testovanie

Integračné testovanie sa koná pre každý jeden komponent zvlášť. Je to dôležité preto, aby sme sa ubezpečili v tom, že poznáme všetky zmeny, ktoré daný komponent spôsobil počas jeho implementácie. V prípade, že boli testy úspešné, sa komponent integruje do bázy softvéru a tým vzniká nová verzia bázy softvéru.

Komponent sa do softvéru neintegruje, ak testy poukázali na chyby. V tomto prípade sa vykoná správa o teste, ktorá obsahuje detaily problému a rovno sa táto správa doručí programátorom, alebo sa problém - podľa dohody v tíme - pridá do backlogu a ku konkrétnemu riešeniu dôjde až niekedy v priebehu nasledujúcich šprintov.



5.7.5 Testovanie systému

Testovanie tohto typu sa vykonáva na konci šprintu a jeho cieľom je merať do akej miery sa stretli požiadavky na systém s odovzdaným prototypom. Testovať je potrebné na všetkých

platformách. Možné výkonnostné problémy by sa mali identifikovať týmto testovaním a testy by mali vyriešiť otázku, či je daný prototyp vhodný na odovzdanie vedúcemu.

5.7.6 Akceptačné testovanie

Akceptačné testovanie sa uskutočňuje počas Review Meetings. Sú to formálne testy s cieľom testovania tých charakteristík, o ktoré žiadal klient (v tomto prípade Product Owner). Táto osoba by mala poskytnúť následne spätnú väzbu o tom, ako je spokojný s vytvoreným prototypom a zároveň by mal definovať jeho problémy. Na základe tohto by sa určili ďalšie úlohy na refaktorizáciu a vyriešenie problémov.

5.8 Metodika dokumentácie knižníc

Nie všetky knižnice, s ktorými sa v tomto tímovom projekte pracuje, majú dokumentáciu k svojmu aplikačnému programovaciemu rozhraniu (*API*) výhodne rozpracovanú pre ciele tohto projektu, ako to býva zvykom pre známejšie *API* (napr. v *OpenCV* alebo *OpenSceneGraph* dokumentácii sa ťažko prehl'adáva hĺbkovo a všeobecne je pre ciele tohto projektu príliš robustne dokumentovaná). V tejto metodike sú opísané pravidlá a procesy pre vytvorenie pomocnej (skrátenej) dokumentácie k takýmto *API*. Nástrojom pre tvorbu takejto dokumentácie je typografický systém *LaTeX*, kde sa jednotlivé návestia prehl'adne zobrazujú vo výstupnom dokumente.

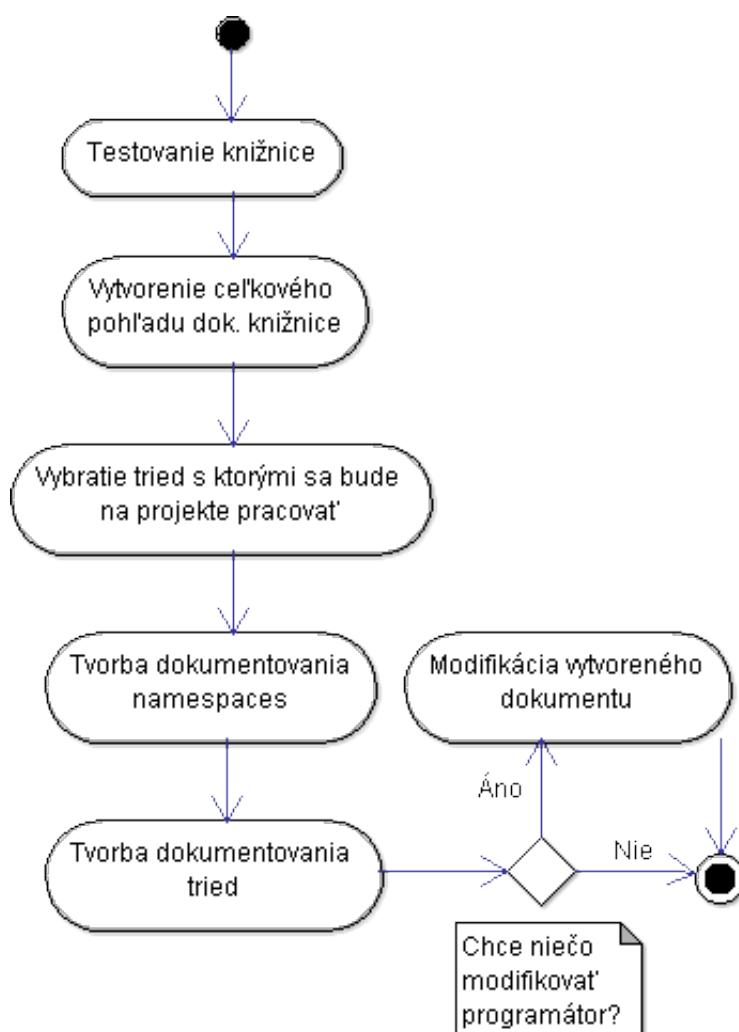
Táto metodika do malej miery nadväzuje na metodiku dokumentácie zdrojového kódu, keďže v tých procesoch vystupuje ako nástroj *Doxygen*, a všeobecné konvencie dokumentovania pomocou tohto nástroja sú dodržané aj v tejto metodike.

5.8.1 Roly

Rola	Procesy
Manažér podpory vývoja	Vytvorenie celkového pohľadu dokumentácie knižnice
Analytik	Tvorba dokumentovania <i>namespaces</i> Tvorba dokumentovania tried
Programátor	Modifikácia vytvoreného dokumentu

5.8.2 Procesy

Obrázok 5 znázorňuje všetky identifikované procesy metodiky v podobe diagramu aktivít. Okrem toho sú zobrazené aj "vstupné podmienky" pre prvý, druhý a tretí proces tiež ako aktivity. Analytik musí identifikovať triedy a funkcie knižníc, ktoré budú alebo môžu byť potrebné pre prácu na projekte, kým manažér podpory vývoja ručí za úspešnú inštaláciu a spustenie knižničných funkcií.



Obr. 5: Procesy dokumentácie knižníc

5.8.2.1 Vytvorenie celkového pohľadu dokumentácie knižnice

Vytvorenie celkového pohľadu (*Overview* alebo *Getting Started*) je v API dokumentáciach základnou praktikou. Zachytáva proces inštalácie a projektové nastavenia pre prácu so samotným rozhraním. Prípadné rozšírenie tejto sekcie je možné zdokumentovaním všeobecných konvencií danej knižnice.

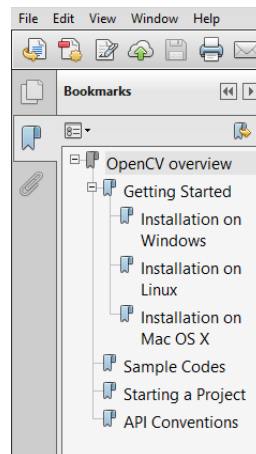
Vstup: Úspešná inštalácia a testovanie knižnice členom tímu

Výstup: Dokumentácia s opísaným Getting Started

Zodpovedný: Manažér podpory vývoja

Krok	Popis
1.1	Opis priebehu inštalácie po bodoch pre všetky potrebné platformy
1.2	Stručný opis súborov (hlavne príkladových zdrojových kódov)
1.3	Opis vytvorenia projektu s opisom projektových nastavení
1.4	Opis všeobecných konvencií knižničných tried a funkcií (opcionálne)

Tabuľka opisuje podrobnejší opis procesu tvorby časti Getting Started. Dôležitým pravidlom pre tvorcu tejto časti dokumentu je, aby každú časť separoval ako samostatný úsek v TeX (*section, subsection*). Takýmto spôsobom sa dosiahne podobný efekt v štrukturovaní dokumentácie ako v klasických API dokumentáciách.



Obr. 6: Ukážka návěstí v Readeri

5.8.2.2 Tvorba dokumentovania namespaces

Každá knižnica obsahuje minimálne jeden namespace, tento proces zaručuje dokumentáciu o kompletom prehľade tried, premenných a ďalších dátových typov v rámci namespaceu.

Vstup: Podrobná analýza knižnice

Výstup: Prehľad o namespaces v dokumentácii

Zodpovedný: Analytik

Jazyk TeX zahŕňa v sebe možnosť vygenerovania kódových ukážok (s podporou syntaxe C++), takže opis jednotlivých tried a premenných vie byť prehľadný. Pravidlom

Krok	Popis
2.1	Štrukturované vymenovanie namespaces v knižnici
2.2	Opis vyskytujúcich sa tried a premenných

je aj, aby sa tvorca dokumentu odvolával referenciami na jednotlivé triedy a ostatné typy, keďže tretí proces je práve o opise týchto entít.

5.8.2.3 Tvorba dokumentovania tried

Analytik sa rozhoduje na základe vykonanej analýzy, že ktoré triedy považuje za relevantné v rámci riešenia projektu a takisto spraví aj rez medzi všetkými členmi triedy, aby dokumentácia obsahovala len dôležité informácie.

Vstup: Podrobná analýza knižnice

Výstup: Prehľad o triedach v dokumentácii

Zodpovedný: Analytik

Krok	Popis
3.1	Štrukturované vymenovanie tried s referenciami
3.2	Štrukturovaný opis vymenovaných tried
3.3	Štrukturované vymenovanie členov tried podľa ich typu

Prvým krokom je vymenovanie tried s referenciami. Je to zoznam všetkých identifikovaných tried, ktoré považuje analytik za potrebné pre prácu na projekte. Kliknutím na jednotlivé triedy by sa mal ukázať detailnejší popis tried zahŕňajúci všetky základné informácie (opäť ako zoznam) - konštruktory, deštruktory a ďalšie funkcie spoločne so stručným opisom, prípadne aj príbuzenské triedy. Všetky vymenované funkcie sú referenciami v TeX dokumente na ich detailnejší popis, ktorý sa nachádza v tom istom odseku, ale až po vymenovaní všetkých funkcií.

Ukážka 1: *Formátovanie funkcie v triede*

```
<Type><Namespace >:: <Class >:: <Function > ( <ParamName> & <>
  » ParamName > )
<Description >
```

Parameters

<ParamName> <Description >

Returns <Description >

Vzorový popis funkcie je znázornený v ukážke 1, kde <Type> znamená dátový typ, <Class> triedu, <Function> meno funkcie, <ParamName> vstupné parametre funkcie a <Description> stručný opis. Pre lepšiu orientáciu treba kľúčové slová ako *Returns* a *Parameters* písať polotučným písmom.

Posledným krokom je vymenovanie všetkých členov tried, ktorí sa nachádzajú vo spomenutých triedach. Patria sem nielen konštruktory, deštruktory a ostatné funkcie triedy, ale aj jednotlivé premenné. Jedná sa o zoznam všetkých členov, zároveň tento zoznam slúži ako referencia na podrobný opis členov. V prípade funkcií sa odvoláva na rovnaké návestie ako keď používateľ klikol na funkciu v opise triedy, kým zoznam premenných sa nachádza v ďalšom odseku.

Ukážka 2: *Formátovanie premennej v triede*

<Type><Namespace >:: <Class >:: <VariableName >

<Description >

Vzorový popis premennej je znázornený v ukážke 2, kde <VariableName> je meno premennej.

3 Classes

3.1 Class List

3.1.1 Class 1

List of Functions

Detailed Description of Functions

3.1.2 Class 2

3.2 Class Members

3.2.1 Constructors

3.2.2 Destructors

3.2.3 Functions

3.2.4 Variables

Obr. 7: Ukážka štruktúrovania tried v dokumentácii

5.8.2.4 Modifikácia vytvoreného dokumentu

Programátor má možnosť meniť obsah dokumentu v prípade, ak identifikoval chýbajúce triedy (funkcie, premenné), ktorých znalosť bola potrebná v projekte. V rámci tohto procesu koná podľa rovnakej metodiky ako analytik v treťom procese.

Vstup: Identifikácia chýbajúcich knižničných informácií

Výstup: Modifikovaná dokumentácia

Zodpovedný: Programátor

Krok	Popis
4.1	Pridanie záznamu do dokumentácie na základe procesu 5.8.2.3

6 Manažment kvality a monitorovanie projektu

Jednou zo základných manažérskych pozícií v tíme je manažér kvality a monitorovania. V našom tíme sa povinnosti tohto manažéra delia na dva druhy monitorovania : monitorovanie projektu ako súbor procesov a monitorovanie projektu z hľadiska kvality.

6.1 Monitorovanie postupu projektu

V rámci vývoja projektu sa riadime agilným vývojom. Tento typ vývoja sa odohráva inkrementálne a iteračne, pričom v našom prípade jeden časový úsek (šprint) trvá dva týždne. V tomto jednom časovom úseku manažér sleduje postup práce jednotlivých členov tímu. Pričom sa zameriava na obsah backlogu z hľadiska aktuálnosti jeho obsahu, následne sleduje rozdelenie úloh, či všetci majú pridelené úlohy a pohľad z rovnomernosti pridelenia úloh. Následne v jednom časovom úseku sleduje postup práce jednotlivých členov, procesy v plnení úlohy, viditeľnosti progresu úlohy a schopnosť splnenia pridelených termínov.

Na sledovanie jednotlivých akcií slúži nástroj na manažment úloh, v našom prípade Redmine. Tento nástroj sme obohatili o rozšírenia na prácu s *Backlogom*, plánovania šprintu a sledovania progresu *Run Charts*. Na základe týchto rozšírení a možností nástroja Redmine sme schopní monitorovania spomenutých častí v procese postupu projektu.

6.2 Monitorovanie kvality projektu

V procese vývoja projektu sa sleduje kvalita procesov a ich výstupu. Jednou z hlavných úloh je monitorovanie používania jednotlivých metodík, ktoré boli vytvorené jednotlivými manažérmi v tíme. Jednou z najviac monitorovaných oblastí je vytváranie zdrojového kódu a jeho dokumentácia. Táto oblasť je sledovaná zo všeobecného hľadiska dodržiavania vytvorených konvencií pri písaní zdrojového kódu, ale aj monitorovanie multiplatformového vývoja v rámci používaných knižníc a vlastností jednotlivých operačných programov.

7 Manažment podpory vývoja a integrácie

Pri práci na väčších projektoch, akým je aj tímový projekt, je nevyhnutné zaoberať sa manažmentom podpory vývoja. Jeho cieľom je systematické zavádzanie takých postupov a metód, aby bol čo v najväčšej miere podporený samotný vývoj a jeho zjednodušenie ako aj vývoj v tíme.

V tejto časti je opísaná práca s podpornými nástrojmi a spôsob akým bolo v našom tíme riadené nasadenie aplikácie a potrebných knižníc na začiatku a počas vývoja.

7.1 Manažment pre podporu verzií

Nakoľko už v rámci vývoja v predchádzajúcich rokoch bol pre riadenie verzií používaný nástroj Git a všetky jednotlivé repozitáre boli umiestnené na serveri GitHub, dospelo sa k rozhodnutiu v tomto pokračovať. Toto so sebou prináša niekoľko výhod.

Jednoduchý prístup ku všetkým doterajším verziám aplikácie od všetkých jej tvorcov, čo bolo dôležité na začiatku, keď bolo potrebné zjednotiť ich a vyriešiť konflikty, aby sa mohlo začať pracovať s verziou, ktorá ponúka najviac funkcionality. Taktiež aj možnosť vytvárania a spravovania viacerých vetiev pre paralelnú prácu na projekte.

Ako primárny nástroj sa začala používať aplikácia Git, pomocou ktorej sú spravované jednotlivé verzie aplikácie. Z pôvodného repozitára vedúceho projektu bol ako vetva založený vlastný repozitár s názvom nášho tímu ARVis. Takto bolo umožnené samostatne spravovať repozitár len pre náš projekt a v budúcnosti jeho výsledok začleniť späť do hlavného repozitára.

V rámci prípravy boli vypracované viaceré analýzy Gitu, pluginov pre GitHub a analýzy pre možné zlepšenie práce s repozitárom, čoho výsledkom bola hlavne priložená Metodika manažmentu verzií. Touto bol zavedený postup GitFlow, čoho výsledkom má byť sprehľadnenie riadenia jednotlivých vetiev. Pre jeho podporu boli navrhnuté aj niektoré pluginy, avšak z analýzy vyplývalo, že ich nie je nutné používať.

7.2 Nasadenie aplikácie a podpora pri vývoji

Aplikácia, na ktorej vývoji pracujeme, je vyvíjaná pre viacero platforiem, a taktiež používa niekoľko ďalších knižníc a frameworkov (Qt, OpenSceneGraph, OsgThirdParty, LibNoise), pričom je plánované rozšíriť ju o ďalšie (OsgArt, OpenNI pre Kinect, OpenCV, prípadne iné). Z týchto faktov vyplýva aj netriviálny postup pre prípravu a jej nasadenie, kým je možné začať s jej ďalším vývojom. Preto je potrebné zaoberať sa týmto pri manažmente

podpory vývoja.

Aby bola zabezpečená kontrola multiplatformovosti vývoja, každý člen tímu si zvolil platformu, pre ktorú chce vyvíjať, avšak s tým, že v konečnom dôsledku boli takmer rovnomerne pokryté všetky 3 hlavné platformy. Tieto postupy boli v predchádzajúcich používateľských príručkách vypracované nedotačne, preto pre ďalších členov tímu aj budúcich vývojárov boli vypracované nové podrobné postupy pre všetky platformy. Tieto boli začlenené do dokumentácie.

Ďalej boli vypracované analýzy pre ďalšie knižnice, na ktorých začlenení a testovaní ešte pracujú poverení členovia tímu. Títo majú za úlohu taktiež prípravu konkrétnych návodov pre ich správne nastavenie.

Pre podporu tvorby dokumentácie bola vypracovaná Metodika pre písanie komentárov a dokumentovanie zdrojového kódu a zavedenie používania Doxygen aplikácie. Toto prispeje k ľahšej tvorbe kvalitnejšej dokumentácie, čím dôjde k podpore porozumenia aplikácii a tým aj samotného vývoja.

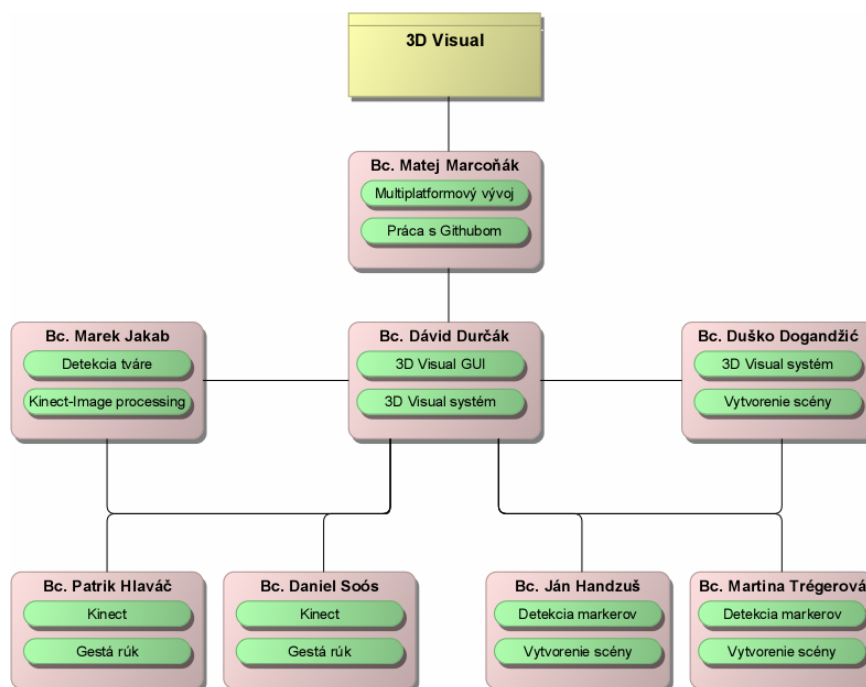
8 Manažment vývoja

Vzhľadom na fakt, že cieľom tímového projektu je doplniť funkcionality do existujúceho programu, je potrebné sa s aktuálnym systémom oboznámiť. Aktuálne systém ponúka prácu a manipuláciu s grafmi vo virtuálnom priestore. V tejto časti je opísané rozdelenie práce jednotlivých členov tímu na vývoji danej aplikácie a prostriedky, ktoré boli počas vývoja použité.

Hlavným cieľom tímového projektu je do vyvíjaného softvéru doplniť prvky obohatenej reality. V rámci vytvorenia obohatenej reality sa virtuálna scéna okolia grafu nahradí scénou získanou snímaním priestoru pomocou kamery, prípadne senzoru Kinect. Získaná hĺbková mapa zároveň môže poslúžiť pre získanie ohraničení v priestore scény. Manipulácia s grafom sa presunie od používania vstupno-výstupných zariadení ku gestám rúk a umocní sa tak pocit obohatenej reality.

8.1 Rozdelenie práce

Pre vývoj a nasadenie spomenutých funkcionalít sme vytvorili rozdelenie, ktoré je znázornené na Obr. 8.



Obr. 8: Rozdelenie práce v tíme

8.2 Prostriedky použité pri vývoji

Aby mohol nastať vývoj všetkých spomenutých funkcionalít bolo potrebné vykonať analýzu dostupných knižníc. Analýza a prieskumné prototypovanie bolo vykonané na nasledujúcich knižniciach:

- OpenSceneGraph
- OpenCV
- OpenNI
- Nite
- Microsoft Kinect SDK
- NUI
- OpenKinect
- ARToolkit
- Aruco
- OSGart

Funkcionalita jednotlivých knižníc sa plánuje vytvárať v samostatných moduloch, aby bolo možné dané funkcionality ľahko vypnúť a pracovať v prípade požiadavky používateľ a aj bez nich.

Pre tvorbu softvéru v tímovom projekte bola vytvorená metodika pre písanie programového kódu, ktorou sa členovia tímu musia pri vývoji držať. Aby sme zabezpečili prehľadnosť a súdržnosť vytvoreného kódu, prebehne pre každý vytvorený kód jeho prehliadka, podľa metodiky prehliadky kódu. Zároveň sa každá funkcionalita otestuje na rôznych platformách pre potreby multiplatformového vývoja.

9 Manažment rizík

V tejto časti sú opísané riziká, ktorá sa počas vývoja softvéru identifikovali v rámci tímového projektu.

Dokončenie, ale aj samotná úspešnosť projektu závisí od problémov, ktoré nastanú pri vývoji. Čím menej problémov vznikne a čím menej ovplyvnia vývoj, tým je projekt úspešnejší. Vznik a ošetrovanie problému sa dá ovplyvniť skorým určením rizík a ich permanentným sledovaním.

Identifikácia rizík prebieha zároveň s plánovaním nových úloh a získavaním bližšieho pohľadu na budúcnosť projektu. Za sledovanie identifikovaného problému je zodpovedný člen, o ktorom sa predpokladá, že v danej oblasti pracuje najintenzívnejšie. Je snaha pristupovať ku rizikám kontinuálne, pričom v niektorých prípadoch môže nastať prehliadnutie situácie a potreba riešenia problému pri nastaní.

Pre zvýšenie konzistentnosti vlastností rizík, určujeme niektorým atribútom hodnoty z pripravenej množiny. Jedná sa o atribúty *Pravdepodobnosť* (nastania udalosti), *Dopad* (veľkosť škody) a *Frekvencia* (ako často sa očakáva riziko). Množina obsahuje hodnoty *nízka/y*, *stredná/y* a *vysoká/y*.

Zoznam rizík identifikovaných na stretnutiach:

- nedostatočná analýza pôvodného programu,
- strata člena tímu,
- problémy funkcionality na rôznych platformách,
- zlé rozlíšenie ovládania gestami,
- zlá voľba technológie pre snímanie prostredí,
- hardvérové chyby a poškodenia,
- zlá prepojitelnosť funkcionality s pridaným ovládaním,
- slabá optimalizácia a pomalý chod programu.

Zoznam opodstatnených rizík:

- v zimnom semestri žiadne.

Riziká neidentifikované včas:

- v zimnom semestri žiadne.

9.1 Nedostatočná analýza pôvodného programu

Názov rizika	Nedostatočná analýza pôvodného programu		
Opis	Pri slabom oboznámení sa tímu s pôvodným riešením projektu a s aktuálnym stavom, je pravdepodobné nastanie chýb v implementácii.		Identifikované: 15.10.2013 Zodpovedný: Dávid Durčák
Pravdepodobnosť: nízka	Dopad:	stredný	Frekvencia: nízka
Následky	Tím sa musí zaoberať novými riešeniami situácie, čo spomalí celkové napredovanie a zmení výsledné riešenie.		
Prevenca	Dôkladná analýza pôvodného projektu.		
Náprava	Zmena priorít riešenia úloh, investovanie ďalšieho času.		

9.2 Strata člena tímu

Názov rizika	Strata člena tímu		
Opis	Nedostatočný čas venovaný projektu, slabá adaptácia alebo tiež osobné dôvody člena.		Identifikované: 21.10.2013 Zodpovedný: Martina Trégerová
Pravdepodobnosť: nízka	Dopad:	stredný	Frekvencia: nízka
Následky	Možná strata kľúčových odborníkov, nemožnosť dokončiť projekt v danom rozsahu.		
Prevenca	Dbať na dostatočnú informovanosť v tíme, časté spoločné Scrum Retrospektívy.		
Náprava	Prerozdelenie úloh ostatných členov, časové vynahradenie práce pôvodných členov.		

9.3 Problémy funkcionality na rôznych platformách

Názov rizika	Problémy funkcionality na rôznych platformách		
Opis	Zadanie zahŕňa tvorbu multiplatformového programu. Niektoré z modulov, nemusia byť podporované na rôznych systémoch.		Identifikované: Zodpovedný: 21.10.2013 Matej Marcoňák
Pravdepodobnosť: stredná	Dopad:	stredný	Frekvencia: stredná
Následky	Možné nutné obmedzenie funkcionality.		
Prevenia	Dostatočná analýza pred začiatkom práce.		
Náprava	Hľadanie funkčných riešení a zmeny aj v pôvodnom kóde.		

9.4 Zlé rozlíšenie ovládania gestami

Názov rizika	Zlé rozlíšenie ovládania gestami		
Opis	Zadanie zahŕňa aj ovládanie grafov pomocou gest. Problém zlého snímania Kinectom môže vyústiť do nepoužiteľnosti programu.		Identifikované: Zodpovedný: 4.11.2013 Patrik Hlaváč
Pravdepodobnosť: nízka	Dopad:	vyšoký	Frekvencia: stredná
Následky	Nesplnenie požadovanej funkcionality		
Prevenia	Analýza dostupných riešení a výber riešenia s dobrou perspektívou		
Náprava	Implementácia iného druhu ovládania (mobil, hlas)		

9.5 Zlá voľba technológie pre snímanie prostredí

Názov rizika	Zlá voľba technológie pre snímanie prostredí		
Opis	Prostredie a pohyb používateľ a je možné monitorovať viacerými spôsobmi. Nie všetky poskytujú potrebné výsledky.	Identifikované: Zodpovedný:	4.11.2013 Marek Jakab
Pravdepodobnosť: nízka	Dopad:	vysoký	Frekvencia: nízka
Následky	Nefunkčnosť aplikácie, zložitá implementácia.		
Prevenčia	Dobrá a včasná analýza dostupných riešení.		
Náprava	Zameranie sa na tvorbu modulu iným spôsobom.		

9.6 Hardvérové chyby a poškodenia

Názov rizika	Hardvérové chyby a poškodenia		
Opis	Program pracuje s viacerými periférnymi zariadeniami. Ich bezchybné fungovanie zaručuje správny priebeh plánu.	Identifikované: Zodpovedný:	4.11.2013 Duško Dogandžić
Pravdepodobnosť: stredná	Dopad:	vysoký	Frekvencia: nízka
Následky	Nemožnosť dokončiť projekt.		
Prevenčia	Dbanie na bezpečnosť a pozornosť pri práci. Zamedzenie manuálnemu poškodeniu.		
Náprava	Získanie nového zariadenia.		

9.7 Zlá prepojitelnosť funkcionality s pridaným ovládaním

Názov rizika	Zlá prepojitelnosť funkcionality s pridaným ovládaním		
Opis	Zlý prístup ku sledovaniu používateľa a neskoré (prípadne žiadne) reagovanie na sledované správanie.		Identifikované: 11.11.2013 Zodpovedný: Daniel Soós
Pravdepodobnosť: nízka	Dopad:	stredný	Frekvencia: nízka
Následky	Nefunkčnosť ovládania aplikácie, nutné hľadanie nových riešení.		
Prevenia	Testovanie od začiatku vývoja nového ovládania.		
Náprava	Hľadanie a implementovanie náhradných riešení.		

9.8 Slabá optimalizácia a pomalý chod programu

Názov rizika	Slabá optimalizácia a pomalý chod programu		
Opis	Množstvo rozličných modulov môže spôsobiť vysokú záťaž zariadenia. To sa môže prejavovať v spomalenom reagovaní alebo nefunkčnosti.		Identifikované: 11.11.2013 Zodpovedný: Ján Handzuš
Pravdepodobnosť: stredná	Dopad:	nízky	Frekvencia: stredná
Následky	Pomalé testovanie, nižšia použiteľnosť.		
Prevenia	Časté kontroly stavu programu počas vývoja.		
Náprava	Analyzovanie chýb a spoločné hľadanie riešení.		

10 Manažment rozvrhu a plánovania

Keďže vyvíjame náš softvér pomocou metodiky Scrum, nie je možné zostaviť podrobný rozvrh na celý semester. Preto táto časť bude priebežne dopĺňaná počas semestra.

Každé dva týždne pri plánovaní nového šprintu sme si definovali úlohy a osoby, ktoré sú zodpovedné za ich realizáciu. Väčšinou počet úloh zodpovedá počtu členov v tíme. Avšak na niektorých úlohách môže pracovať viac členov tímu súčasne. Za dokončenie takejto úlohy v šprinte zodpovedá každý, komu bola takáto úloha pridelená. Pokiaľ je to možné, každý kto má pridelenú úlohu, si ju rozdelí na menšie podúlohy a zaeviduje do systému. Na evidenciu úloh a na sledovanie stavu úloh v našom tímovom projekte používame Redmine, ktorý je nainštalovaný a spustený na školskom serveri.

10.1 Postup plánovania šprintu

V tejto časti sa nachádza opis priebehu plánovania nového šprintu. V našom prípade osobu vlastníka produktu zastupuje pedagogický vedúci tímu.

1. Dopredu sa určí človek, ktorý zapisuje priebeh stretnutia a osoba, ktorá vedie stretnutie. Osoba, ktorá zapisovala predchádzajúce stretnutie, spravidla vedie aktuálne stretnutie.
2. Ak sa nejedná o prvý šprint, tak sa zhodnotí stav úloh z predchádzajúceho šprintu. Prípadné nedokončené úlohy prechádzajú automaticky do ďalšieho šprintu.
3. Vlastník produktu predloží svoje požiadavky a ich priority na základe, ktorých tím zostaví zoznam cieľov na ďalší šprint.
4. Ak je to možné, ciele sa rozdelia na menšie úlohy a tím následne ohodnotí náročnosť jednotlivých úloh a priradí ich podľa metodiky prípravy úloh na realizáciu, ktorá sa nachádza v časti 5.1.

10.2 Používateľské príbehy

Na prvom stretnutí s vlastníkom produktu sa identifikovali tieto požiadavky:

1. spustiteľnosť na všetkých platformách,
2. umiestnenie grafu v priestore pomocou značiek,
3. manipulácia grafu pomocou značiek,

4. umiestnenie grafu v priestore pomocou rozpoznávania priestoru.
5. manipulácia grafu gestami,
6. prispôsobenie grafu podľa pozície tváre tak aby vznikol 3D efekt.

Jednotlivé požiadavky zákazníka sme rozdelili medzi členov tímu tak, aby každý mal približne rovnako veľa práce na projekte.

10.3 Všeobecný rozvrh zimného semestra

Na začiatku semestra sme si zostavili všeobecný plán na semester, ktorý sme si upresňovali na stretnutiach.

Číslo šprintu	Dátum začiatku	Dátum konca	Úlohy
1.	7.10.2013	21.10.2013	Analýza
2.	21.10.2013	4.11.2013	Analýza a refaktorizácia
3.	4.11.2013	18.11.2013	Implementácia, testovanie a nasadenie
4.	18.11.2013	2.12.2013	Implementácia, testovanie a nasadenie
5.	2.12.2013	16.12.2013	Implementácia, testovanie a nasadenie

10.4 Vyhodnotenie plánov

Keďže žiadny člen tímu zatiaľ nepracoval metodikou Scrum, tak sme sa na začiatku dopustili niekoľkých chýb pri plánovaní. Medzi tieto chyby patrilo používanie zlej metodiky ohodnocovania náročnosti, čo spôsobovalo podceňovanie úloh a následne naplánované úlohy neboli ukončené načas a prechádzali do ďalšieho šprintu. Dalším problémom bol fakt, že naplánovaná analýza a refaktorizácia kódu nám zabrali viac času ako sme plánovali. Aj s týchto príčin sme nestihli ukončiť všetky úlohy, ktoré sme začali riešiť počas semestra. Preto sme sa ako tím dohodli, že v medzisesestrálnom období sa dokončia a otestujú úlohy, ktoré sa nestihli ukončiť.

11 Manažment komunikácie

Práca na projekte v rámci väčšieho tímu vyžaduje dostatočne organizovanú komunikáciu, pričom je dôležitá hlavne osobná komunikácia jednotlivých členov tímu. Táto bola riešená hlavne v rámci stretnutí. Mimo stretnutí sa potom jednotliví členovia tímu, ktorí spolu pracovali stretávali osobne, avšak celý tím komunikoval či už formálne alebo neformálne hlavne pomocou on-line prístupov.

V rámci stretnutí prebiehala hlavne formálna komunikácia s vedúcim, pričom následne bez vedúceho tím pokračoval v stretnutí a rozhodoval o ďalšej časti práce na projekte.

11.1 Stretnutia

Keďže tím pracoval metódou SCRUM, na začiatku každého stretnutia sme zhodnotili doterajšiu prácu v rámci šprintu, každý člen tímu sa vyjadril k tomu, čo počas predošlého týždňa spravil, s čím mal problém a čo ešte plánuje spraviť budúci týždeň.

Ďalej sa riešila komunikácia s vedúcim tímu, teda s product ownerom, ktorý zastupoval stakeholderov a určoval priority jednotlivých úloh zapísaných v backlogu, pričom tiež definoval nové úlohy.

Z úloh zapísaných v backlogu sa následne tím rozhodoval, ktoré úlohy sa budú riešiť v rámci nasledujúceho šprintu, pričom sa riadil podľa priorít definovaných vedúcim tímu. Úlohy, ktoré si členovia tímu rozdelili medzi sebou a rozhodli sa ich vyriešiť v priebehu nasledujúceho šprintu boli následne pridané do systému Redmine a určené na riešenie v nasledujúcom šprinte.

11.2 Formálna komunikácia

Formálna komunikácia, ktorá prebiehala v rámci stretnutí s vedúcim, je zdokumentovaná v zápisniciach tímu z jednotlivých stretnutí. Mimo stretnutí prebiehala komunikácia hlavne on-line.

Hlavný zdroj on-line komunikácie je prostredníctvom Google groups, kde tím používa špecifickú notáciu pre sprehl'adnenie jednotlivých príspevkov. Táto notácia bola definovaná na jednom z prvých stretnutí. V prípade, že niekto chce kontaktovať tím, mailovú adresu tímu možno nájsť na tímovej webstránke.

Tretím zdrojom formálnej komunikácie tímu je systém Redmine, kam jednotliví členovia zaznamenávajú čas, ktorý venovali práci na úlohe, ktorú majú definovanú v rámci systému a tiež nakoľko pokročili v danej úlohe.

11.3 Neformálna komunikácia

Okrem formálnej prebieha v rámci tímu aj neformálna komunikácia. Táto prebieha osobne v rámci stretnutí bez vedúceho. Tieto neformálne stretnutia nie sú nijak definované, ich výsledkom je zväčša rozdelenie úloh v rámci tímu a definovanie úloh v systéme Redmine. Okrem osobnej prebieha aj on-line komunikácia.

Hlavný zdroj on-line komunikácie je systém Skype, kde sa v rámci skupinového rozhovoru dokáže celý tím veľmi rýchlo dohodnúť na aktuálnych alebo akútnych problémoch, ktoré treba vyriešiť. Táto komunikácia je braná ako neformálna a často sa stáva, že jednotlivé správy sú neprehľadné, preto sme maximalizovali používanie mailing listu hlavne na dlhodobjšie riešenie niektorých problémov a prehľadný spôsob informovania kolegov v tíme.

Okrem Skype sa využívajú aj iné druhy komunikácie, napr. email, chat, instant messaging alebo aj osobná komunikácia s jednotlivými kolegami. Prítom sa väčšinou stretnú iba niektorí členovia tímu - hlavne tí, ktorí sa snažia vyriešiť spolu jednu časť úlohy.

12 Manažment tvorby dokumentácie

Dokumentácia sa vytvára priebežným spôsobom a finálne dokumenty sú sformované v nástroji LaTeX. Časti, z ktorých dokumentácie (riadenia a inžinierskeho diela) pozostávajú zdieľa tím na repositári Dropbox. Obrázky sa ukladajú vo formáte .png a textové dokumenty môžu byť .doc, .tex a .pdf súbory. Manažér dokumentácie vykonáva nasledujúce úlohy počas tvorby dokumentácie:

- vytvorí šablónu k dokumentáciám,
- integruje jednotlivé dokumenty do finálneho dokumentu,
- korekcia, pravopis a štylizovanie,
- komunikácia s ostatnými členmi ohľadom chýbajúcich elementov v dokumentácii.

Po vykonaní týchto úloh môže manažér finalizovať dokumentácie, pričom vytvorí všetky dopĺňajúce kapitoly ako napr. semestrový plán, úlohy členov v tíme, preberací protokol atď. Manažér dokumentácie zároveň zodpovedá za odovzdávanie dokumentov vedúcemu.

A Zázpisnice zo stretnutí

Táto kapitola sa zaoberá záznamami zo stretnutí tímu č. 5 ARVis. Každý záznam obsahuje tabuľky, kde sa vyhodnocujú, resp. vypíšu úlohy pre jednotlivé stretnutia. Takýmto spôsobom je prehľadne zdokumentované, že aké úlohy mal daný člen v konkrétnych týždňoch. Zázpisnice zároveň obsahujú stručný obsah týchto stretnutí.

A.1 Zázpis zo stretnutia č. 1 tímu č. 5

Téma stretnutia:	Úvodné stretnutie – úvodná analýza a rozdelenie úloh		
Dátum stretnutia:	30.09.2013		
Čas stretnutia:	18:00 – 19:40		
Miesto stretnutia:	Laboratórium počítačového videnia a počítačovej grafiky (UAI FIIT)		
Účastníci:	Bc. Duško Dogandžić	Bc. Dávid Durčák	Bc. Ján Handzuš
	Bc. Patrik Hlaváč	Bc. Marek Jakab	Bc. Matej Marcoňák
	Bc. Daniel Soós	Bc. Martina Trégerová	
	Ing. Peter Kapec, PhD.		
Zapisovateľ:	Bc. Daniel Soós		
Vedúci stretnutia:	Ing. Peter Kapec, PhD.		

Priebeh stretnutia:

- Úvod do problematiky:
 - diskusia k predmetu
 - priblíženie podporných technológií na riešenie projektu
 - prvotný náčrt termínov.
- Prehľad podporných nástrojov:
 - Git repozitáre - Github, GitFlow, HubFlow
 - Redmine – nástroj podporujúci aj metodiku SCRUM
 - Qt – framework použiteľný na viacerých platformách
 - API OpenSceneGraph – nie je potrebné explicitne využívať OpenGL
 - osgART knižnica – poskytnutie knižnice ARToolKit.
- Hlavné témy stretnutia:
 - opis potenciálneho riešenia projektu – výhoda spomenutých technológií, spôsob ukladania 3D grafov
 - prvý prototyp – základ so značkami, nastavba už existujúceho riešenia
 - diskusia o výstupnom riešení projektu – k dispozícii by mala byť špeciálna fólia na premietanie
 - prihláška na TP Cup.
- Rozdelenie úloh v tíme.
- Ukážka školského projektu zaoberajúca sa danou témou na dodatočnom stretnutí.

- Úlohy do nasledujúceho stretnutia:

ID	Popis úlohy	Zodpovedný	Termín začiatku	Termín ukončenia	Stav
0.1	Návrh tímovej webstránky	Patrik	30.09.2013	07.10.2013	Riešená
0.2	Preskúmanie oblasti GitFlow a HubFlow	Matej	30.09.2013	07.10.2013	Riešená
0.3	Skúmanie podpory Qt a OpenSceneGraph na Windows platformách	Dávid, Marek	30.09.2013	07.10.2013	Riešená
0.4	Vytvorenie šablón pre dokumenty	Daniel	30.09.2013	07.10.2013	Riešená
0.5	Vytvorenie tímového plagátu	Patrik	30.09.2013	07.10.2013	Riešená
0.6	Rozbehnutie aplikácie Redmine	Martina	30.09.2013	14.10.2013	Riešená
0.7	Analýza softvéru 3DVisual	Dávid, Ján, Marek, Martina, Duško	30.09.2013	14.10.2013	Riešená
0.8	Prihláška na TP Cup	Marek	30.09.2013	28.10.2013	Riešená

Zhodnotenie stretnutia:

Na stretnutí sme sa dozvedeli, ako by mal vyzerat' výstup projektu a bližšie sme sa oboznámili so základnou problematikou.

A.2 Zázpis zo stretnutia č. 2 tímu č. 5

Téma stretnutia:	Naplánovanie prvého šprintu, ukážka korektných nastavení pre prácu s vývojovým prostredím		
Aktuálny šprint:	Prvý šprint		
Dátum stretnutia:	07.10.2013		
Čas stretnutia:	18:40 – 20:50		
Miesto stretnutia:	Laboratórium počítačového videnia a počítačovej grafiky (UAI FIIT)		
Účastníci:	Bc. Duško Dogandžić	Bc. Dávid Durčák	Bc. Ján Handzuš
	Bc. Patrik Hlaváč	Bc. Marek Jakab	Bc. Matej Marcoňák
	Bc. Daniel Soós	Bc. Martina Trégerová	
	Ing. Peter Kapec, PhD.		
Zapisovateľ:	Bc. Daniel Soós		
Vedúci stretnutia:	Ing. Peter Kapec, PhD.		

Priebeh stretnutia:

- Vyhodnotenie úloh z predošlého stretnutia:
 - zodpovední sa vyjadrili k svojim úlohám
 - prediskutovanie problémových oblastí – funkčnosť a podpora vývojových nástrojov na rôznych platformách, Redmine
 - plánovanie prvého šprintu aj na základe vyhodnotenia doterajších úloh.
- Hlavné témy stretnutia:
 - analyzovať a vyhodnotiť potrebu nástrojov ako Redmine a GitHub (resp. iné, ak sa tak rozhodne); v prípade pozitívneho vyhodnotenia ich inštalácia na server – Redmine by mal mať aj git repozitár
 - code review – analýza existujúcich nástrojov; Redmine a GitHub problém pravdepodobne nerieši, Doxygen vhodný na generovanie dokumentácie C++ zdrojového kódu
 - dohoda o termínoch nasledujúcich šprintov a ich náplň
 - analýza (testovanie) nových prostriedkov chatu – Skype, ani mailing list optimálnym riešením nie sú pri práci v tíme; Slack, Hipchat a pod. možno áno.
- Rozchodenie projektu na jednom počítači:
 - úspešné na platforme Linux
 - potrebné aj pre ostatné platformy – dokumentácia návodov k tomu.

- **Vyhodnotenie úloh z predchádzajúceho stretnutia:**

ID	Popis úlohy	Zodpovedný	Termín začiatku	Termín ukončenia	Stav
0.1	Návrh tímovej webstránky	Patrik	30.09.2013	07.10.2013	Dokončená
0.2	Preskúmanie oblasti GitFlow a HubFlow	Matej	30.09.2013	07.10.2013	Dokončená
0.3	Skúmanie podpory Qt a OpenSceneGraph na Windows platformách	Dávid, Marek	30.09.2013	07.10.2013	Dokončená
0.4	Vytvorenie šablón pre dokumenty	Daniel	30.09.2013	07.10.2013	Dokončená
0.5	Vytvorenie tímového plagátu	Patrik	30.09.2013	07.10.2013	Dokončená
0.6	Rozbehnutie aplikácie Redmine	Martina	30.09.2013	14.10.2013	Riešená
0.7	Analýza softvéru 3DVisual	Dávid, Ján, Marek, Martina	30.09.2013	14.10.2013	Riešená
0.8	Prihláška na TP Cup	Marek	30.09.2013	28.10.2013	Riešená

- **Úlohy do prvého šprintu:**

ID	Popis úlohy	Zodpovedný	Termín začiatku	Termín ukončenia	Stav
1.1	Skompilovanie a spustenie predchádzajúcich častí programu na vlastnom stroji	všetci	07.10.2013	21.10.2013	Riešená
1.2	Analýza predošlých tímových projektov s touto témou	všetci	07.10.2013	21.10.2013	Riešená

1.3	Analýza chatov podporujúcich komunikáciu v tíme	Daniel, Ján	07.10.2013	21.10.2013	Riešená
1.4	Vytvorenie základnej funkcionality webstránky	Patrik	07.10.2013	21.10.2013	Riešená
1.5	Analýza metodík na SCRUM (Redmine)	Martina	07.10.2013	21.10.2013	Riešená
1.6	Inštalácia potrebných nástrojov pre prácu na projekte na vlastnom serveri	Duško, Matej, Patrik	07.10.2013	21.10.2013	Riešená

Zhodnotenie stretnutia:

Na stretnutí sme definovali, ako má vyzerat' priebeh a časová následnosť šprintov. Zároveň nám vedúci prezentoval správne nastavenie vývojového prostredia a spustenie programu, s ktorým sa bude pracovať počas dvoch semestrov.

A.3 Zázpis zo stretnutia č. 3 tímu č. 5

Téma stretnutia:	Vyhodnotenie vykonaných úloh z prvého šprintu a bližšia špecifikácia ostatných, ešte nevyriešených úloh v tomto šprinte		
Aktuálny šprint:	Druhý šprint		
Dátum stretnutia:	14.10.2013		
Čas stretnutia:	16:20 – 18:50		
Miesto stretnutia:	Laboratórium počítačového videnia a počítačovej grafiky (UAI FIIT)		
Účastníci:	Bc. Duško Dogandžić	Bc. Dávid Durčák	Bc. Ján Handzuš
	Bc. Patrik Hlaváč	Bc. Marek Jakab	Bc. Matej Marcoňák
	Bc. Daniel Soós	Bc. Martina Trégerová	
	Ing. Peter Kapec, PhD.		
Zapisovateľ:	Bc. Matej Marcoňák		
Vedúci stretnutia:	Ing. Peter Kapec, PhD.		

Priebeh stretnutia:

- Vyhodnotenie úloh z predošlého stretnutia:
 - Ján a Daniel analyzovali dostupné chatové aplikácie; rozhodlo sa, že pre podporu komunikácie bude vhodnejšie využívať mailing list s vybranou metodikou a na projektový manažment Redmine
 - Daniel navrhoval zvoliť a zdokumentovať metodiky pre vývoj projektu (tak manažmentské ako aj technické), vedúci pripomenul, že je možnosť inšpirovať sa riešeniami minulých projektov, alebo predmetom MIS/MSI
 - Matej analyzoval Git – webové rozhranie pre prácu s ním je možným riešením, GitFlow na Windows by nemal byť problémom, naopak, Git je distribuovaný systém a je otázne, nakoľko sa dá separovať voči GitFlow
 - tím sa spoločne s vedúcim rozhodol, že je nepotrebné spájať Git s Redmine
 - Patrik prezentoval základnú funkcionálnosť webstránky – schválená vedúcim
 - každý úspešne rozchodil projekt 3Dvisual na vlastnom stroji, okrem Jana, ktorému sa to ešte nepodarilo na Mac OS X.
- Hlavné témy stretnutia:
 - code review – Redmine síce obsahuje plugin aj k tomu, ale optimálne je minimalizovať jeho funkcionálnosť čisto na projektový manažment (t.j. plánovanie šprintov), skôr to spojiť s GitHubom
 - OpenSceneGraph – podporuje ho aj diskusné fórum, je žiadúce prehľadávať v ňom, ak sa naskytnú problémy

- vedúci na položenú otázku vysvetlil, ako by mal vyzerat' druhý šprint (naplánovaný od piateho týždňa semestra) – doimplementácia pôvodného riešenia, skompletizovaná analýza všetkých technológií (vrátane aj hardvérov) a predošlého riešenia
- vedúci objasnil, že by nemalo záležat' na tom, či riešiteľ pracuje pod QtCreatorom alebo vo Visual Studiu, dokonca to môže pomáhať pri implementovaní – rôzne warning hlášky pod iným implementačným prostredím
- Doxygen – existujúci plugin pre „všetky“ implementačné prostredia na všelijakých platformách; jednoduché vygenerovanie komentárov k zdrojovým kódom, práca prakticky pre jedného člena tímu
- plánovanie nových úloh do prvého šprintu – niektoré už tím definoval na neformálnom stretnutí, iné pribudli explicitne na základe tohto stretnutia po dohode s vedúcim projektu.

• **Vyhodnotenie úloh z predchádzajúceho stretnutia:**

ID	Popis úlohy	Zodpovedný	Termín začiatku	Termín ukončenia	Stav
1.1	Skompilovanie a spustenie predchádzajúcich častí programu na vlastnom stroji	všetci	07.10.2013	21.10.2013	Riešená
1.2	Analýza predošlých tímových projektov s touto témou	všetci	07.10.2013	21.10.2013	Riešená
1.3	Analýza chatov podporujúcich komunikáciu v tíme	Daniel, Ján	07.10.2013	21.10.2013	Dokončená
1.4	Vytvorenie základnej funkcionality webstránky	Patrik	07.10.2013	21.10.2013	Dokončená
1.5	Analýza metodík na SCRUM (Redmine)	Martina	07.10.2013	21.10.2013	Dokončená
1.6	Inštalácia potrebných nástrojov pre prácu na projekte na vlastnom serveri	Duško, Matej, Patrik	07.10.2013	21.10.2013	Dokončená

- **Nové úlohy do prvého šprintu:**

ID	Popis úlohy	Zodpovedný	Termín začiatku	Termín ukončenia	Stav
1.7	Git metodika a poskytnutie stručného návodu na používanie Gitu	Matej	14.10.2013	21.10.2013	Riešená
1.8	Analýza zdrojového kódu predošlého riešenia a tvorba metodiky k písaniu zdrojového kódu	Dávid	14.10.2013	21.10.2013	Riešená
1.9	Metodika pre prácu s Redmine a nasadenie pluginu SCRUM na server	Patrik	14.10.2013	21.10.2013	Riešená
1.10	Analýza Doxygenu	Martina	14.10.2013	21.10.2013	Riešená
1.11	Analýza OpenCV	Marek	14.10.2013	21.10.2013	Riešená
1.12	Analýza osgArt	Marek, Matej	14.10.2013	21.10.2013	Riešená
1.13	Analýza OpenSceneGraph	Duško	14.10.2013	21.10.2013	Riešená
1.14	Testovanie verzií potrebných knižníc	Ján	14.10.2013	21.10.2013	Riešená
1.15	Analýza ovládania programu 3DVisual	Daniel	14.10.2013	21.10.2013	Riešená

Zhodnotenie stretnutia:

Na stretnutí sme vyhodnotili doterajší priebeh prvého šprintu a na základe nášho stavu riešenia sme naplánovali nové úlohy pre ukončenie tohto šprintu.

A.4 Zázpis zo stretnutia č. 4 tímu č. 5

Téma stretnutia:	Vyhodnotenie prvého šprintu, identifikácia product backlogov, naplánovanie druhého šprintu		
Dátum stretnutia:	21.10.2013		
Čas stretnutia:	18:00 – 19:45		
Miesto stretnutia:	Laboratórium počítačového videnia a počítačovej grafiky (UAI FIIT)		
Účastníci:	Bc. Duško Dogandžić	Bc. Dávid Durčák	Bc. Ján Handzuš
	Bc. Patrik Hlaváč	Bc. Marek Jakab	Bc. Matej Marcoňák
	Bc. Daniel Soós	Ing. Peter Kapec, PhD.	
Zapisovateľ:	Bc. Marek Jakab		
Vedúci stretnutia:	Ing. Peter Kapec, PhD.		

Priebeh stretnutia:

- Vyhodnotenie prvého šprintu:
 - Matej zhrnul, že Redmine je už v plnej prevádzke, Patrik dodáva zistené nedostatky nainštalovaného pluginu na SCRUM
 - Dávid, Duško a Daniel sa stručne vyjadrili k svojim úlohám, analýzy a dokumenty k tomu sú na Dropboxe (OpenSceneGraph, ovládanie programu, metodika na písanie zdrojového kódu)
 - Ján konzultoval s vedúcim projektu rozbehaný projekt 3DVisual – chýbajúce png pri inštalácii na OS X
 - Matej o poznatkoch Git: používatelia Linux operačných systémov využijú plugin na GitFlow, Windows používatelia pracujú tiež s metodikou GitFlow, ale len s použitím GitHub for Win
 - vedúci projektu pripomenul, že sa spraví fork v GitHub až potom, čo sa opraví chyba v kompilácii v jednom z branchov – chyba vytvorená v niektorom z predchádzajúcich projektov
 - zároveň bolo spomenuté, ako vhodne vytvárať commity v Git – stručný a výstižný názov, následne referencia na typ commitu (výkričník pre opravu bugu atď.)
 - Marek sa vyjadril podrobnejšie k analýze OpenCV – rozpoznávanie tváre cez deskriptory atď.
 - Matej s Marekom mali za úlohu analyzovať osgART – v roku 2007 sa s tým prestalo, ale na GitHub prispievajú, naposledy pridali aj plugin na OpenCV.
- Hlavné témy stretnutia:
 - oprava chýb v predchádzajúcom projekte, aby sa mohol uskutočniť fork na GitHub a tým pádom mohla začať aj vývojárska práca na projekte

- spustenie videa o metóde SCRUM pre celý tím – termíny úloh riadiť podobne ako vo videu
- product backlog – v ideálnom prípade by časom prichádzal hlavný architekt tímu s vlastnými nápadmi, aby rozšíril požiadavky product ownera
- identifikácia product backlogov a ich rozbitie na menšie časti – sledovanie používateľa označené vedúcim ako prioritný backlog, t.j. jeho zakomponovanie do druhého šprintu je žiadúce, ďalšími backlogmi sú napr. rozpoznanie markeru, zobrazenie grafu, zmena pozadia
- význam projektového denníka – písať si dátumy a strávené časy nad každou vykonanou aktivitou v rámci tímového projektu; Redmine je určený primárne na konkrétne úlohy v rámci šprintov.

• **Vyhodnotenie úloh z predchádzajúceho šprintu:**

ID	Popis úlohy	Zodpovedný	Termín začiatku	Termín ukončenia	Stav
1.7	Git metodika a poskytnutie stručného návodu na používanie Gitu	Matej	14.10.2013	21.10.2013	Dokončená
1.8	Analýza zdrojového kódu predošlého riešenia a tvorba metodiky k písaniu zdrojového kódu	Dávid	14.10.2013	21.10.2013	Dokončená
1.9	Metodika pre prácu s Redmine a nasadenie pluginu SCRUM na server	Patrik	14.10.2013	21.10.2013	Dokončená
1.10	Analýza Doxygenu	Martina	14.10.2013	21.10.2013	Dokončená
1.11	Analýza OpenCV	Marek	14.10.2013	21.10.2013	Dokončená
1.12	Analýza osgArt	Marek, Matej	14.10.2013	21.10.2013	Dokončená
1.13	Analýza OpenSceneGraph	Duško	14.10.2013	21.10.2013	Dokončená
1.14	Testovanie verzií potrebných knižníc	Ján	14.10.2013	21.10.2013	Dokončená

- Úlohy do druhého šprintu:

ID	Popis úlohy	Zodpovedný	Termín začiatku	Termín ukončenia	Stav
2.1	Fixácia kompilačných chýb v predchádzajúcom programe, ktorý sa nachádza na GitHub	Dávid, Ján, Matej	22.10.2013	05.11.2013	Riešená
2.2	Nasadenie Gitu na vlastný stroj a osvojenie si manuálov k tomu	všetci	22.10.2013	05.11.2013	Riešená
2.3	Pokročilá analýza osgART, skúmanie využitia knižníc v existujúcom riešení	Marek, Matej	22.10.2013	05.11.2013	Riešená
2.4	Prihláška na TP Cup	Marek	22.10.2013	05.11.2013	Riešená

Zhodnotenie stretnutia:

Na stretnutí sme skompletizovali výsledky prvého šprintu a identifikovali základné product backlogy. Prvé úlohy do druhého šprintu boli rozdelené.

A.5 Zázpis zo stretnutia č. 5 tímu č. 5

Téma stretnutia:	Ďalšie plánovanie druhého šprintu, identifikovanie budúceho stavu pre začiatok tretieho šprintu		
Dátum stretnutia:	28.10.2013		
Čas stretnutia:	18:00 – 19:40		
Miesto stretnutia:	Laboratórium počítačového videnia a počítačovej grafiky (UAI FIIT)		
Účastníci:	Bc. Duško Dogandžić	Bc. Dávid Durčák	Bc. Ján Handzuš
	Bc. Patrik Hlaváč	Bc. Marek Jakab	Bc. Matej Marcoňák
	Bc. Daniel Soós	Bc. Martina Trégerová	
	Ing. Peter Kapec, PhD.		
Zapisovateľ:	Bc. Patrik Hlaváč		
Vedúci stretnutia:	Ing. Peter Kapec, PhD.		

Priebeh stretnutia:

- Hlavné témy stretnutia:
 - Marek zahajoval stretnutie; analýza OpenCV skompletizovaná, kompilačné chyby v programe v stave riešenia
 - Martina o analýze Doxygen: automatická tvorba HTML dokumentov, generovanie dokumentácie, písanie komentárov ručne podľa metodiky, nie automaticky
 - Marek navrhuje pracovať nad fungujúcou verziou a riešiť simultánne rozpoznávanie tváre popri oprave kompilačných chýb
 - Marek potvrdil vedúcemu odovzdanie prihlášky na TP Cup
 - Matej a Dávid pripomenuli, že riešia fixáciu kompilačných chýb v starej verzii
 - vedúci povedal, že mieni zmeniť cmake kvôli používaniu externých knižníc – osgART, prípadne Kinect môže byť problematické, tento Cmakelist.txt by mal byť lepší ako ten starý; zároveň oznámil tímu, že na jeho notebooku šlo skompilovať program
 - vedúci taktiež objasnil, že k osgART pravdepodobne bude potrebné buildovať knižnicu, kým k OpenCV existujú aj binárky
 - Ján pospájajal návody inštalovania programu na rôzne platformy.
- Ďalšie témy súvisiace s druhým šprintom:
 - vedúci sa s tímom dohodol, že analýza sa skompletizuje v rámci aktuálneho šprintu a ďalej už bude riešiť primárne implementačné, resp. návrhovo implementačné časti projektu

- vedúci zdôraznil, že je vhodné postupovať pri analýzach jednotlivých technológií tak, aby analyzujúci člen tímu aj reálne využil tieto vedomosti
- Dávid a Matej konkretizovali chybné časti programu s vedúcim
- vedúci o 3DVisual: niekto má zobrať ako úlohu ovládanie zdrojového kódu – nie do detailov, ale je prospešné vedieť poskytnúť informácie ostatným členom ohľadom tried, modulov, funkcionalít v kóde aspoň všeobecne
- tak ako je potrebné dokončiť analýzu, je treba to mať už aj zdokumentované – prvé odovzdávanie sa blíži
- tím by mal zistiť chod importovania ARToolkit, a tiež prehliť vedomosti o OSGart, či jeho pridaná hodnota je dostatočná pre tento projekt
- iní by si mali naštudovať Kinect – aká je jeho podpora v C++, resp. jeho spolupráca s knižnicami
- plán do budúcnosti: do konca semestra minimálne vyriešiť rozpoznávanie tváre a detekciu značky.

• **Vyhodnotenie úloh z predchádzajúceho šprintu:**

ID	Popis úlohy	Riešiteľ	Termín začiatku	Termín ukončenia	Stav
2.1	Fixácia kompilačných chýb v predchádzajúcom programe, ktorý sa nachádza na GitHub	Dávid, Ján, Matej	22.10.2013	05.11.2013	Riešená
2.2	Nasadenie Gitu na vlastný stroj a osvojenie si manuálov k tomu	všetci	22.10.2013	05.11.2013	Dokončená
2.3	Pokročilá analýza osgART, skúmanie využitia knižníc v existujúcom riešení	Marek, Matej	22.10.2013	05.11.2013	Riešená
2.4	Prihláška na TP Cup	Marek	22.10.2013	05.11.2013	Dokončená

- **Nové úlohy do druhého šprintu:**

ID	Popis úlohy	Riešiteľ	Termín začiatku	Termín ukončenia	Stav
2.5	Rozpoznanie tváre	Marek	28.10.2013	05.11.2013	Riešená
2.6	Globálny pohľad na program	Dávid	28.10.2013	05.11.2013	Riešená
2.7	Analýza Kinectu a spolupracujúcich knižníc	Daniel, Patrik	28.10.2013	05.11.2013	Riešená
2.8	Merge projektu v Git	Dávid, Matej	28.10.2013	05.11.2013	Riešená
2.9	Analýza osgART	Ján, Martina	28.10.2013	05.11.2013	Riešená
2.10	Úprava 3Dvisual pre novšie verzie knižníc	Ján	28.10.2013	05.11.2013	Riešená
2.11	Prezentácia SCRUM (SCRUM panel)	Marek, Martina, Patrik	28.10.2013	05.11.2013	Riešená
2.12	Spojenie analýz do dokumentácie	Daniel	28.10.2013	05.11.2013	Riešená
2.13	Skompletizovanie Git metodiky	Matej	28.10.2013	05.11.2013	Riešená
2.14	Aktualizácia webstránky	Patrik	28.10.2013	05.11.2013	Riešená

Zhodnotenie stretnutia:

Na stretnutí sme konzultovali čiastočné výsledky druhého šprintu a rozdelili sme si ďalšie, nové úlohy v rámci šprintu.

A.6 Zázpis zo stretnutia č. 6 tímu č. 5

Téma stretnutia:	Vyhodnotenie druhého šprintu, premyslenie product backlogov a na základe toho identifikácia úloh do tretieho šprintu		
Dátum stretnutia:	04.11.2013		
Čas stretnutia:	18:00 – 19:35		
Miesto stretnutia:	Laboratórium počítačového videnia a počítačovej grafiky (UAI FIIT)		
Účastníci:	Bc. Duško Dogandžić	Bc. Dávid Durčák	Bc. Ján Handzuš
	Bc. Patrik Hlaváč	Bc. Marek Jakab	Bc. Matej Marcoňák
	Bc. Daniel Soós	Bc. Martina Trégerová	
	Ing. Peter Kapec, PhD.		
Zapisovateľ:	Bc. Martina Trégerová		
Vedúci stretnutia:	Ing. Peter Kapec, PhD.		

Priebeh stretnutia:

- Hlavné témy stretnutia:
 - Patrik zahájil stretnutie, vyjadril sa k vyriešeným úlohám v druhom šprinte
 - Marek zmienil, že jeho práca v druhom šprinte prebehla rýchlejšie ako to očakával, a rád by už jeho riešenie dopracoval do pôvodného programu; navrhuje spraviť už merge
 - vedúci sa vyjadril k zmenám, ktoré spravil v branch vetve – Matej sumarizoval, že na Linux ide bezproblémovo, Visual Studio kompilátor však mal problémy na Windows, tento problém sa riešil na stretnutí do veľkej miery: prilinkovanie libnoise knižnice je tým pádom aj úloha do šprintu
 - vedúci zároveň informoval tím, že problémy s PNG sa dajú riešiť v Qt ako resources
 - vedúci našiel ďalšiu chybu v pôvodnom riešení: program sa snaží pripojiť na databázu aj v prípade, keď by to už nemal robiť, navyše tá istá funkcia rieši načítanie grafu a pripájanie na databázu
 - ďalším problémom v budúcnosti môže byť neefektívny kód v programe; vybrať čo najviac include, pokiaľ možné – lepšie je používať preddeklarované premenné
 - refaktor kódu ako takého je tiež žiadúce, pokúsiť sa prepojiť program s viacerými dll a identifikovať nové podmoduly, následne ich aj tak zmeniť v kóde – všetky tieto nedostatky v programe však sú úlohy, ktoré si netreba dávať ako elitne prioritné, ale riešiť by sa mali
 - ďalšie pripomienky k analýzam: ARToolkit je stále problematický, treba sa pozrieť aj na alternatívy, ale tam asi cesta nevedie, keďže niečo iné spojiť s osgART pôjde pravdepodobne ťažko.

- **Ďalšie témy a plánovanie tretieho šprintu:**
 - na základe predošlého šprintu a dávnejšie definovaných product backlogov sa identifikovali nové úlohy, tie boli znázornené aj na tabuli
 - keďže boli úlohy v druhom šprinte, ktoré potrebujú pokračovanie, nakoľko ich zložitost' presahuje možnosti dvojtýždňového šprintu, tie ostali s rovnakým pomenovaním aj v treťom šprinte
 - všetky úlohy boli ohodnotené metódou „planning poker“ a následne si každý vybral úlohu, ktorú bude riešiť v príslušnom šprinte.

- **Retrospektíva po druhom šprinte:**
 - tím sa sústredil hlavne na identifikáciu tých procesov (postupov), v ktorých by rád pokračoval, resp. ktoré by začal alebo zahodil, t.j. používala sa primárne technika podobná k *The Wheel*
 - tím sa dohodol, že by sa malo začať implementovať viac, hlavne sa koncentrovať na novovytvorené časti
 - SCRUM kartačky a ohodnocovanie používateľských príbehov (prípadne z toho vyvedených úloh, ktorých nie je v tomto projekte príliš veľa) tiež patrí do skupiny *start doing*
 - prestať by sa malo analýzou – priamo súvisí s tým, že sa hlavný dôraz kladie na implementáciu
 - pokračovať v stretávkach, resp. častejšie meetingy (*more of*)
 - podľa techniky *Mad, Sad, Glad* tím identifikoval progres v analýzach ako Glad, kým multiplatformový vývoj (jeho následkom je viacero testovaní) ako Sad.

- **Vyhodnotenie ostávajúcích úloh v druhom šprinte:**

ID	Popis úlohy	Riešiteľ	Termín začiatku	Termín ukončenia	Stav
2.1	Fixácia kompilačných chýb v predchádzajúcom programe, ktorý sa nachádza na GitHub	Dávid, Ján, Matej	22.10.2013	05.11.2013	Dokončená
2.3	Pokročilá analýza osgART, skúmanie využitia knižníc v existujúcom riešení	Marek, Matej	22.10.2013	05.11.2013	Dokončená
2.5	Rozpoznanie tváre	Marek	28.10.2013	05.11.2013	Dokončená
2.6	Globálny pohľad na program	Dávid	28.10.2013	05.11.2013	Dokončená

2.7	Analýza Kinectu a spolupracujúcich knižníc	Daniel, Patrik	28.10.2013	05.11.2013	Dokončená
2.8	Merge projektu v Git	Dávid, Matej	28.10.2013	05.11.2013	Dokončená
2.9	Analýza osgART	Ján, Martina	28.10.2013	05.11.2013	Dokončená
2.10	Úprava 3Dvisual pre novšie verzie knižníc	Ján	28.10.2013	05.11.2013	Dokončená
2.11	Prezentácia SCRUM (SCRUM panel)	Marek, Martina, Patrik	28.10.2013	05.11.2013	Dokončená
2.12	Spojenie analýz do dokumentácie	Daniel	28.10.2013	05.11.2013	Dokončená
2.13	Skompletizovanie Git metodiky	Matej	28.10.2013	05.11.2013	Dokončená
2.14	Aktualizácia webstránky	Patrik	28.10.2013	05.11.2013	Dokončená

• Úlohy do tretieho šprintu:

ID	Popis úlohy	Riešiteľ	Termín začiatku	Termín ukončenia	Stav
3.1	osgART – prieskumné prototypovanie	Martina	05.11.2013	18.11.2013	Riešená
3.2	Kinect – prieskumné prototypovanie	Daniel, Patrik	05.11.2013	18.11.2013	Riešená
3.3	Import rozpoznania tváre do programu	Marek	05.11.2013	18.11.2013	Riešená
3.4	Globálny pohľad na program	Dávid	05.11.2013	18.11.2013	Riešená
3.5	Zmena pozadia v programe	Duško	05.11.2013	18.11.2013	Riešená
3.6	Import OpenCV do programu	Marek	05.11.2013	18.11.2013	Riešená
3.7	Spôsob rozbitia programu na moduly	Matej	05.11.2013	18.11.2013	Riešená

3.8	Úprava 3Dvisual pre novšie verzie knižníc	Ján	05.11.2013	18.11.2013	Riešená
3.9	Oprava libnoise	Dávid	05.11.2013	18.11.2013	Riešená

Zhodnotenie stretnutia:

Na stretnutí sme vyhodnotili výsledný stav po druhom šprinte, vykonali k nemu retrospektívu a naplánovali sme si tretí šprint už aj s rozdelenými úlohami.

A.7 Zázpis zo stretnutia č. 7 tímu č. 5

Téma stretnutia:	Zhodnotenie doterajšieho priebehu šprintu a jeho ďalšie plánovanie		
Dátum stretnutia:	11.11.2013		
Čas stretnutia:	16:00 – 18:20		
Miesto stretnutia:	Laboratórium počítačového videnia a počítačovej grafiky (UAI FIIT)		
Účastníci:	Bc. Duško Dogandžić	Bc. Dávid Durčák	Bc. Ján Handzuš
	Bc. Patrik Hlaváč	Bc. Marek Jakab	Bc. Matej Marcoňák
	Bc. Daniel Soós	Bc. Martina Trégerová	
	Ing. Peter Kapec, PhD.		
Zapisovateľ:	Bc. Dávid Durčák		
Vedúci stretnutia:	Ing. Peter Kapec, PhD.		

Priebeh stretnutia:

- Hlavné témy stretnutia:
 - Martina zahájila stretnutie a vyjadrila sa postupne k úlohám, ktoré sa riešia v rámci tretieho šprintu
 - Marek sa vyjadril k importovaniu rozpoznania tváre do celkového programu – riešenie úlohy je už v pokročilom štádiu, implementácia nie je v kompletnom stave, ale už sa istá časť do programu importovala, zároveň sa odstránili v programe zbytočné hlavičky; podľa jeho odhadu by mal byť ďalší týždeň v rámci šprintu ideálny na vyriešenie úlohy
 - Martina a Ján o prieskumnom prototypovaní osgART a ARTToolkit: druhý menovaný je už zastaralý a má menšiu podporu, s rozbeháním osgART sa vyskytli problémy, ale ostáva hlavne sa dozvedieť, čo by nám v projekte priniesol osgART a čo Kinect – aké sú možnosti ich prepojenia, nakoľko ich využijeme; môže sa stať, že sa budeme primárne koncentrovať len na OpenCV a OpenNI
 - Daniel a Patrik nemali v týždni možnosť pracovať priamo s Kinectom, keďže sa zariadenie nenachádzalo v škole, ale počas stretnutia rozbehali príkladové zdrojové kódy pre Kinect, ktoré poskytovala knižnica OpenNI a NITE
 - Dávid preukázal progres v globálnom prehľade o programe, už je k dispozícii na Dropbox viacero dokumentov (diagramov), ktoré znázorňujú modularitu a organizovanie zdrojového kódu, ďalší týždeň by mal byť postačujúci na vyriešenie úlohy
 - Duško sa vyjadril, že zmenu pozadia v programe je možné vykonať a existuje k tomu aj funkcia, ktorá by to mala priamo riešiť – implementáciu si predstavuje tak, že táto funkcionálna by sa pre používateľa zobrazovala ako opcionálna

- Matej riešil rozbitie programu na moduly, sieťové nastavenia a databázu je vhodné odpojiť - vedúci schválil, že databáza sa nech ani nerieši, ideálnym výstupom pre tento proces by mohli byť aj dll súbory
 - Dávid o oprave libnoise: je to vyriešené, ešte sa tam niečo má zakomponovať a otestovať.
- **Ďalšie témy na stretnutí:**
 - vedúci projektu oboznámil tím s návrhom, ktorý mu prezentovala Martina ohľadom architektúry programu – po dohode s tímom sa bude pokračovať v trende, aký nastavil aj Marek pri detekcii tváre, t.j. najprv vyriešiť samostatne danú funkcionálnu a potom ho integrovať do celkového programu ako modul
 - vedúci sa detailnejšie vrátil k osgART a ARToolkit, či vôbec vie tím ako chce tieto knižnice využiť, po dohode to budú riešiť v tomto týždni ešte aj osobne, treba brať do ohľadu aj to, do akej miery sa dá implementovať požiadavky s OpenNI pre Kinect a, hlavne, ako môžu tieto knižnice spolupracovať
 - vedúci zároveň ešte pripomenul Dávidovu úlohu: do akej miery vie identifikovať konkrétne moduly v programe; následne nakreslili na tabuľku kostru programu – Dávidov návrh je, aby sa novovytváraná časť oddelila od starého riešenia aj v GUI toolbare
 - vedúci vyzdvihol Marekovo riešenie s rozpoznávaním tváre: je to vhodný postup; taktiež zdôraznil, že v zásade pre každý ďalší používateľský príbeh pôjdu vlastné namespacey – Marek dodal, že on sám vo svojom riešení využil dva namespacey
 - ďalšie identifikované príbehy do backlogu: rozpoznávanie gest a program aby nerobil skoky pri centralizovaní pohľadu, ale sa ten proces vizualizoval
 - skúšali sa príklady s Kinectom, vedúci pripomínal aj, že s osgART by sa tiež mali vyskúšať niektoré dostupné príklady
 - OpenCV popritom, že by nemal mať problémy so spoluprácou s OpenNI, taktiež môže aj slúžiť ako alternatíva v tom prípade, že Kinect nie je k dispozícii
 - Martina pripomenula dokumentáciu, ktorá by mala byť hotová do začiatku nového šprintu, keďže v budúcom týždni sa koná prvé odovzdávanie.
 - **Úlohy v tretom šprinte (posledný dodefinovaný po tomto stretnutí):**

ID	Popis úlohy	Riešiteľ	Termín začiatku	Termín ukončenia	Stav
3.1	osgART – prieskumné prototypovanie	Martina	05.11.2013	18.11.2013	Riešená
3.2	Kinect – prieskumné prototypovanie	Daniel, Patrik	05.11.2013	18.11.2013	Riešená

3.3	Import rozpoznania tváre do programu	Marek	05.11.2013	18.11.2013	Riešená
3.4	Globálny pohľad na program	Dávid	05.11.2013	18.11.2013	Riešená
3.5	Zmena pozadia v programe	Duško	05.11.2013	18.11.2013	Riešená
3.6	Import OpenCV do programu	Marek	05.11.2013	18.11.2013	Riešená
3.7	Spôsob rozbitia programu na moduly	Matej	05.11.2013	18.11.2013	Riešená
3.8	Úprava 3Dvisual pre novšie verzie knižníc	Ján	05.11.2013	18.11.2013	Riešená
3.9	Oprava libnoise	Dávid	05.11.2013	18.11.2013	Riešená
3.10	Integrovanie dokumentov do dokumentácií	Daniel	11.11.2013	18.11.2013	Riešená

Zhodnotenie stretnutia:

Na stretnutí sme diskutovali o aktuálnom stave tretieho šprintu a identifikovali sme aj nové príbehy do product backlogu, ktoré sa budú riešiť v nadchádzajúcich šprintoch.

A.8 Zápis zo stretnutia č. 8 tímu č. 5

Téma stretnutia:	Zhodnotenie tretieho sprintu a plánovanie štvrtého		
Dátum stretnutia:	18.11.2013		
Čas stretnutia:	18:00 – 21:10		
Miesto stretnutia:	Laboratórium počítačového videnia a počítačovej grafiky (UAI FIIT)		
Účastníci:	Bc. Duško Dogandžić	Bc. Dávid Durčák	Bc. Ján Handzuš
	Bc. Patrik Hlaváč	Bc. Marek Jakab	Bc. Matej Marcoňák
	Bc. Daniel Soós	Bc. Martina Trégerová	
	Ing. Peter Kapec, PhD.		
Zapisovateľ:	Bc. Ján Handzuš		
Vedúci stretnutia:	Ing. Peter Kapec, PhD.		

Priebeh stretnutia:

- Hlavné témy stretnutia:
 - Dávid zahájil stretnutie a postupne sa vyjadril k úlohám – riešitelia prebrali slovo od neho
 - Marek importoval spolu s OpenCV už aj konkrétnu detekciu tváre do softvéru – ďalej už mieni riešiť, ako sa dopracovať k 3D efektu v grafe
 - Matej sa vyjadril k modularite programu a s tým súvisiacou refaktorizáciou kódu: problémom je, že sú triedy až príliš previazané, čo nikdy nie je optimálne; vedúci navrhuje, aby sa aspoň jednoznačne oddeliteľné časti rozbili, dobrým kandidátom je sieťové pripojenie v programe
 - Dávid k tomu povedal, že triedy Data a Model by sa tiež mali dať oddeliť
 - Dávid skompletizoval úlohu o globálnom programe, momentálne identifikoval problémy s pochopením fungovania kamery v OSG – vedúci navrhol možné riešenie aj s OpenGL, prípadne s inými projekciami
 - Daniel a Patrik o OpenNI: verzia 2 vyzerá najslubnejšie pre prácu na projekte (Marek sa pridáva k názoru), kým priamo na stretnutí sa otestoval aj na Linuxe Kinect – drivery, ktoré poskytuje inštalácia OpenNI2 boli validné
 - vedúci zdôraznil, aby sa prišlo na to, aké výstupy vracia knižnica a následne ako tieto výstupy vie využiť napr. OSG
 - Martina a Ján zhodnotili, že osgART nie je vhodné využiť pre ciele tohto projektu, vedúci navrhol pozrieť sa na knižnicu ArUco – vyskúšať dostupné príklady a zistiť, ako sa dá získať translačná matica značky
 - Duško sa vyjadril k zmene pozadia – úloha vyriešená, ostáva doladiť menšie problémy, ktoré súvisia s chybami predošlej implementácie

- Ján vyriešil aktualizáciu OSG v programe – je to otestované a nachádza sa v develop vetve, pričom boli vymazané nepoužívané premenné a pridané ifdef bloky pre chod programu na OS X.
- Ďalšie témy na stretnutí:
 - vedúci upozornil, že v kóde by sa nemali nachádzať 0 pri priradeniach pre pointre, ale NULL
 - striktnější verzia kompilátora so zmenenými parametrami vo Visual Studio nefungovalo, takže takéto takéto podoba testovania kódu bude možné skôr len v Linux prostredí
 - dokumentácia ešte finišuje v deň stretnutia, keďže aj samotný koniec šprintu bol definovaný na pondelnajšie dni, takže vytlačená verzia bude dostavená vedúcemu v priebehu týždňa
- **Vyhodnotenie úloh z tretieho šprintu:**

ID	Popis úlohy	Riešiteľ	Termín začiatku	Termín ukončenia	Stav
3.1	osgART – prieskumné prototypovanie	Martina	05.11.2013	18.11.2013	Dokončená
3.2	Kinect – prieskumné prototypovanie	Daniel, Patrik	05.11.2013	18.11.2013	Dokončená
3.3	Import rozpoznania tváre do programu	Marek	05.11.2013	18.11.2013	Dokončená
3.4	Globálny pohľad na program	Dávid	05.11.2013	18.11.2013	Dokončená
3.5	Zmena pozadia v programe	Duško	05.11.2013	18.11.2013	Dokončená
3.6	Import OpenCV do programu	Marek	05.11.2013	18.11.2013	Dokončená
3.7	Spôsob rozbitia programu na moduly	Matej	05.11.2013	18.11.2013	Dokončená
3.8	Úprava 3Dvisual pre novšie verzie knižníc	Ján	05.11.2013	18.11.2013	Dokončená
3.9	Oprava libnoise	Dávid	05.11.2013	18.11.2013	Dokončená
3.10	Integrovanie dokumentácie	Daniel	11.11.2013	18.11.2013	Dokončená

- Úlohy do štvrtého šprintu:

ID	Popis úlohy	Riešiteľ	Termín začiatku	Termín ukončenia	Stav
4.1	Pohyb kamery podľa tváre	Marek	19.11.2013	02.12.2013	Riešená
4.2	Výpočet pozície očí	Marek	19.11.2013	02.12.2013	Riešená
4.3	Import rozpoznania tváre do programu	Marek	19.11.2013	02.12.2013	Riešená
4.4	Metóda pre výpočet novej pozície kamery podľa tváre	Dávid	19.11.2013	02.12.2013	Riešená
4.5	Nájdenie špeciálnej projekcie	Dávid	19.11.2013	02.12.2013	Riešená
4.6	Oprava chýb pre opcii apply	Duško	19.11.2013	02.12.2013	Riešená
4.7	Vytvoriť ikonu	Duško	19.11.2013	02.12.2013	Riešená
4.8	Vytvorenie modulov	Matej	19.11.2013	02.12.2013	Riešená
4.9	Úprava *.h súborov a CMakeLists	Matej	19.11.2013	02.12.2013	Riešená
4.10	Import ArUco a jeho testovanie	Martina	19.11.2013	02.12.2013	Riešená
4.11	ArUco a jeho prepojenie s OSG	Ján	19.11.2013	02.12.2013	Riešená
4.12	Zmena pozadia za reálnu scénu	Ján	19.11.2013	02.12.2013	Riešená
4.13	Import OpenNI do programu	Daniel	19.11.2013	02.12.2013	Riešená
4.14	Rozpoznávanie gest	Patrik	19.11.2013	02.12.2013	Riešená

Zhodnotenie stretnutia:

Na stretnutí sme vyhodnotili tretí šprint a naplánovali si úlohy na ďalší. Následne sme ohodnotili zložitost' úloh a priradili ich. Niektoré úlohy boli ešte v ten deň rozbité na ďalšie podúlohy, hlavne ArUco ako nová knižnica potrebuje rozsiahlejšiu analýzu a testovanie v tomto prostredí, tým pádom bolo možné vytvoriť menšie podúlohy – v Redmine je zobrazený už tento tvar.

A.9 Zázpis zo stretnutia č. 9 tímu č. 5

Téma stretnutia:	Revízia doterajšieho priebehu štvrtého šprintu		
Aktuálny šprint:	Štvrtý šprint		
Dátum stretnutia:	25.11.2013		
Čas stretnutia:	18:00 – 20:05		
Miesto stretnutia:	Laboratórium počítačového videnia a počítačovej grafiky (UAI FIIT)		
Účastníci:	Bc. Duško Dogandžić	Bc. Dávid Durčák	Bc. Ján Handzuš
	Bc. Patrik Hlaváč	Bc. Marek Jakab	Bc. Matej Marcoňák
	Bc. Daniel Soós	Bc. Martina Trégerová	
	Ing. Peter Kapec, PhD.		
Zapisovateľ:	Bc. Duško Dogandžić		
Vedúci stretnutia:	Ing. Peter Kapec, PhD.		

Priebeh stretnutia:

- Hlavné témy stretnutia:
 - Ján zahájil stretnutie – najprv sa vyjadril k vlastným úlohám,
 - testoval vetvu, ktorú vytvoril vedúci – neúspešne,
 - nainštaloval ArUco a prešiel všetky príklady, nie však každý zdrojový kód; v blízkej dobe nasleduje import ArUco do softvéru,
 - Marek a jeho úloha (pohyb kamery podľa tváre) je už v takom štádiu, že sa môže spraviť spojenie vetvy s develop vetvou, kým jeho vetva bude znovuvytvorená, aby sa pracovalo na úlohe ďalej, napr. je možné ešte vyriešiť vzdialenosť hlavy od kamery,
 - Dávid prezentoval vedúcemu implementáciu pohybu kamery podľa tváre, riešenie bolo schávelné vedúcim – v pokračovaní šprintu treba vyriešiť ešte zapracovanie hĺbky a korekciu projekcie,
 - vedúci ešte navrhol doplnenie osi o koordináty do scény, aby sa pri vývoji dalo lepšie orientovať pri posunoch,
 - Matej o úprave *.h súborov a CMakeLists: niektoré hlavičkové súbory sa ani nepoužívajú – napr. v triedach, ktoré riešia databázu,
 - vedúci k tomu dodal, že principiálne by mali byť všetky zahrnutia súborov v header súboroch, *getter* a *setter* funkcie môžu byť teoreticky aj implementované v týchto súboroch, všetko ostatné len deklarovať,
 - Matej ďalej povedal o úprave, že mu program padal počas testovania na jednej triede; niektoré nepoužívané časti kódu však zmazal alebo zakomentoval,
 - Daniel k importu OpenNI a NiTe: úprava CMakeLists ešte neprebehla, bola len vytvorená vlastná develop vetva pre tento proces – nie je to prípade OpenCV,

ktorá po inštalácii obsahuje aj súbor pre Cmake, bude treba importovať OpenNI aj NiTe spôsobom, že sa pre ne vytvorí CmakeLists súbor

- vedúci navrhol, že by potenciálne vedel upraviť Makefile pre knižnicu nachádzajúcu sa na GitHub repozitári OpenNI a taktiež povedal, že videl jedno riešenie v osgART príkladoch, kde bola vyriešená problematika importu OpenNI
- Patrik do svojho riešenia (detekcia ruky a gest) úspešne zahrnul príkladové zdrojové kódy, ktoré by mali byť súčasťou softvéru po importovaní OpenNI
- Duško začal riešiť svoje úlohy, ale ešte v nich príliš nepokročil.

- **Ďalšie témy na stretnutí:**

- vedúci prišiel s požiadavkou, aby sa do určitej miery upravilo aj grafické rozhranie programu,
- vedúci taktiež zdôraznil, že na začiatku vývoja je vhodné ešte častejšie merge, nakoľko používame nové knižnice v projekte, tým pádom je dobré ich mať čo najskôr v hlavnej develop vetve,
- vedúci ešte nemal čas prečítať si dokumentáciu, dostane sa k tomu v priebehu týždňa.

- **Úlohy do štvrtého šprintu:**

ID	Popis úlohy	Riešiteľ	Termín začiatku	Termín ukončenia	Stav
4.1	Pohyb kamery podľa tváre	Marek	19.11.2013	02.12.2013	Riešená
4.2	Výpočet pozície očí	Marek	19.11.2013	02.12.2013	Riešená
4.3	Import rozpoznania tváre do programu	Marek	19.11.2013	02.12.2013	Riešená
4.4	Metóda pre výpočet novej pozície kamery podľa tváre	Dávid	19.11.2013	02.12.2013	Riešená
4.5	Nájdenie špeciálnej projekcie	Dávid	19.11.2013	02.12.2013	Riešená
4.6	Oprava chýb pre opciiu apply	Duško	19.11.2013	02.12.2013	Riešená
4.7	Vytvoriť ikonu	Duško	19.11.2013	02.12.2013	Riešená
4.8	Vytvorenie modulov	Matej	19.11.2013	02.12.2013	Riešená

4.9	Úprava *.h súborov a CMakeLists	Matej	19.11.2013	02.12.2013	Riešená
4.10	Import ArUco a jeho testovanie	Martina	19.11.2013	02.12.2013	Riešená
4.11	ArUco a jeho prepojenie s OSG	Ján	19.11.2013	02.12.2013	Riešená
4.12	Zmena pozadia za reálnu scénu	Ján	19.11.2013	02.12.2013	Riešená
4.13	Import OpenNI do programu	Daniel	19.11.2013	02.12.2013	Riešená
4.14	Rozpoznávanie gest	Patrik	19.11.2013	02.12.2013	Riešená

Zhodnotenie stretnutia:

Na stretnutí sme hodnotili doterajší priebeh štvrtého šprintu a prezentovali vedúcemu implementované časti projektu.

A.10 Zázpis zo stretnutia č. 10 tímu č. 5

Téma stretnutia:	Vyhodnotenie štvrtého šprintu, naplánovanie finálneho v zimnom semestri		
Aktuálny šprint:	Piaty šprint		
Dátum stretnutia:	02.12.2013		
Čas stretnutia:	16:00 – 18:50		
Miesto stretnutia:	Laboratórium počítačového videnia a počítačovej grafiky (UAI FIIT)		
Účastníci:	Bc. Duško Dogandžić	Bc. Dávid Durčák	Bc. Ján Handzuš
	Bc. Patrik Hlaváč	Bc. Marek Jakab	Bc. Matej Marčoňák
	Bc. Daniel Soós	Bc. Martina Trégerová	
	Ing. Peter Kapec, PhD.		
Zapisovateľ:	Bc. Daniel Soós		
Vedúci stretnutia:	Ing. Peter Kapec, PhD.		

Priebeh stretnutia:

- Hlavné témy stretnutia:
 - Duško zahájil stretnutie a oslovil riešiteľov jednotlivých úloh,
 - Marek vyriešil pohyb kamery podľa tváre, vedúci však navrhuje, aby sa spájanie vetiev riešilo najprv cez pull requesty,
 - Dávid prezentoval vedúcemu zmenu projekcie, ktorá ešte nerieši vzdialenosť tváre
 - ďalej Dávid s Matejom a vedúcim hľadali chyby v CmakeLists ArUca; problém bol vyriešený a čaká sa na *commit* vedúceho na GitHub, takisto ako aj na Matejov, ktorý aktualizuje zmeny v CmakeLists,
 - Marek, Matej a Dávid riešili konflikty procesu spájania Marekovej vetvy do *develop* vetvy,
 - vedúci ešte navrhol doplnenie osi o koordináty do scény, aby sa pri vývoji dalo lepšie orientovať pri posunoch,
 - Duško vyriešil svoju úlohu vytvorenia ikonky, čiastočne aj opravu chyby pre opcii Apply v programe, konflikty riešil s vedúcim,
 - Matej preukázal progres v rozbití programu do modulov, ku koncu stretnutia s vedúcim vyriešili prvý faktický refaktoring kódu, t.j. vytvoril sa prvý .dll súbor,
 - Martina už testovala ArUco, program prezentovala vedúcemu, a diskutovali o napojení grafu na značku,
 - Ján si nainštaloval ArUco taktiež, ale okrem testovania funkcionality knižnice,
 - Patrik naďalej testoval Kinect, jeho program – riešený zatiaľ samostatne mimo 3DSoftviz – prezentoval vedúcemu, ktorý zároveň pripomenul, aby sa koncentrovalo už na napojenie knižníc na program, napr. riešenie cez Glut určite nevyužijeme v našom softvéri, keďže grafické rozhranie sa riešilo v Qt aj doteraz,

- Daniel ukázal vytvorené .cmake súbory, ktoré majú riešiť linkovanie knižničných súborov do projektu, na svojom lokálnom stroji zmenil aj CmakeLists – testoval to úspešne spoločne s vedúcim na stretnutí, zatiaľ sa však táto funkcionality plánuje riešiť výlučne na Windows.
- **Ďalšie témy na stretnutí:**
 - tím sa dohodol na tom, že úlohy v piatom týždni rozdelí ako štandardne na dva týždne, aj keď to presahuje rámec semestra – vyčlenia sa úlohy, ktoré plánuje tím doriešiť do odovzdávania dokumentácie,
 - nevyriešené úlohy sa naplánovali na ďalší šprint – tieto úlohy nemali dopad na riešenie ostatných a ich priorita nebola prvoradá.
- **Úlohy vo štvrtom šprinte:**

ID	Popis úlohy	Riešiteľ	Termín začiatku	Termín ukončenia	Stav
4.1	Pohyb kamery podľa tváre	Marek	19.11.2013	02.12.2013	Vyriešená
4.2	Výpočet pozície očí	Marek	19.11.2013	02.12.2013	Vyriešená
4.3	Import rozpoznania tváre do programu	Marek	19.11.2013	02.12.2013	Vyriešená
4.4	Metóda pre výpočet novej pozície kamery podľa tváre	Dávid	19.11.2013	02.12.2013	Vyriešená
4.5	Nájdenie špeciálnej projekcie	Dávid	19.11.2013	02.12.2013	Vyriešená
4.6	Oprava chýb pre opciu apply	Duško	19.11.2013	02.12.2013	Vyriešená
4.7	Vytvoriť ikonu	Duško	19.11.2013	02.12.2013	Vyriešená
4.8	Vytvorenie modulov	Matej	19.11.2013	02.12.2013	Vyriešená
4.9	Úprava *.h súborov a CMakeLists	Matej	19.11.2013	02.12.2013	Vyriešená
4.10	Import ArUco a jeho testovanie	Martina	19.11.2013	02.12.2013	Vyriešená
4.11	ArUco a jeho prepojenie s OSG	Ján	19.11.2013	02.12.2013	Vyriešená

4.12	Zmena pozadia za reálnu scénu	Ján	19.11.2013	02.12.2013	Nevyriešená
4.13	Import OpenNI do programu	Daniel	19.11.2013	02.12.2013	Nevyriešená
4.14	Rozpoznávanie gest	Patrik	19.11.2013	02.12.2013	Vyriešená

• **Úlohy do piateho šprintu:**

ID	Popis úlohy	Riešiteľ	Termín začiatku	Termín ukončenia	Stav
5.1	Vytváranie modulov	Matej	03.12.2013	16.12.2013	Riešená
5.2	Výpočet vzdialenosti tváre od kamery	Marek	03.12.2013	16.12.2013	Riešená
5.3	Projekcia podľa vzdialenosti	Dávid	03.12.2013	16.12.2013	Riešená
5.4	Import OpenNI2 a NiTe2	Daniel	03.12.2013	16.12.2013	Riešená
5.5	Import ArUco	Ján, Martina	03.12.2013	16.12.2013	Riešená
5.6	Statické testovanie	Duško	03.12.2013	16.12.2013	Riešená
5.7	Aktualizácia webstránky	Patrik	03.12.2013	16.12.2013	Riešená
5.8	Tvorba dokumentácií	Daniel	03.12.2013	16.12.2013	Riešená
5.9	Zmena pozadia za reálnu scénu	Ján	03.12.2013	16.12.2013	Riešená

Zhodnotenie stretnutia:

Na stretnutí sme vyhodnotili štvrtý šprint a naplánovali sme finálny pre zimný semester.

A.11 Zázpis zo stretnutia č. 11 tímu č. 5

Téma stretnutia:	Revízia k doterajšiemu priebehu piateho šprintu		
Aktuálny šprint:	Piaty šprint		
Dátum stretnutia:	09.12.2013		
Čas stretnutia:	17:00 – 18:40		
Miesto stretnutia:	Laboratórium počítačového videnia a počítačovej grafiky (UAI FIIT)		
Účastníci:	Bc. Duško Dogandžić	Bc. Dávid Durčák	Bc. Ján Handzuš
	Bc. Patrik Hlaváč	Bc. Marek Jakab	Bc. Daniel Soós
	Bc. Martina Trégerová		
	Ing. Peter Kapec, PhD.		
Zapisovateľ:	Bc. Daniel Soós		
Vedúci stretnutia:	Ing. Peter Kapec, PhD.		

Priebeh stretnutia:

- Hlavné témy stretnutia:
 - Dávid prezentoval svoju úlohu vedúcemu,
 - Marek sa zapojil do diskusie a informoval tím o návratových hodnotách OpenCV – štruktúra, kde sa udržiava obraz z kamery je k dispozícii,
 - tím sa zhodol na spôsobe, akým vyrieši spoluprácu tejto podúlohy s OpenCV,
 - Ján riešil zmenu pozadia, s vedúcim riešili konflikty a zároveň našli spôsob konvertovania OpenCV obrázku do osg,
 - Martina prezentovala svoj program s využitím ArUco,
 - Duško dlho konzultoval svoju úlohu s vedúcim – ukázal mu ako môže fungovať cppcheck v projekte, otestovali to a dohodli sa na tom, ako bude prebiehať testovanie týmto nástrojom,
 - Daniel oznámil, že riešil zatiaľ len dokumentáciu v priebehu tohto šprintu.
- Ďalšie témy na stretnutí:
 - vedúci priniesol vytlačenú dokumentáciu z minulého odovzdávania, aby sa to mohlo dať doplniť,
 - tím s vedúcim sa dohodol, že prvoradá je teraz vyriešiť dokumentáciu, samotný prototyp ako obsah elektronického média je možné odovzdať aj po odovzdaní dokumentácie, keď sa vyriešia všetky spojenia aktuálne riešených funkcionalít do *develop* vetvy,
 - vedúci a tím sa dohodli, že projekt a bude riešiť aj medzisemestrálne, ak bude potreba, pripadá aj možnosť stretnutia, hlavne je však dôležité používať aktívnejšie mailing list,

- tím si pripomenul prezentáciu v rámci predmetu Manažment v informačných systémoch: prešli sa jednotlivé body, ktoré sa týkajú prezentácie.

- **Úlohy v piatom šprinte:**

ID	Popis úlohy	Riešiteľ	Termín začiatku	Termín ukončenia	Stav
5.1	Vytváranie modulov	Matej	03.12.2013	16.12.2013	Riešená
5.2	Výpočet vzdialenosti tváre od kamery	Marek	03.12.2013	16.12.2013	Riešená
5.3	Projekcia podľa vzdialenosti	Dávid	03.12.2013	16.12.2013	Riešená
5.4	Import OpenNI2 a NiTe2	Daniel	03.12.2013	16.12.2013	Riešená
5.5	Import ArUco	Ján, Martina	03.12.2013	16.12.2013	Riešená
5.6	Statické testovanie	Duško	03.12.2013	16.12.2013	Riešená
5.7	Aktualizácia webstránky	Patrik	03.12.2013	16.12.2013	Riešená
5.8	Tvorba dokumentácií	Daniel	03.12.2013	16.12.2013	Riešená
5.9	Zmena pozadia za reálnu scénu	Ján	03.12.2013	16.12.2013	Riešená

Zhodnotenie stretnutia:





Na stretnutí sme vyhodnotili doterajší priebeh posledného šprintu v semestri a dohodli sa na pokračovaní tak tohto šprintu ako aj celkového projektu.

B Reporty z letného semestra

V letnom semestri nahradili zápisnice vygenerované reporty zo systému Redmine v tvare pdf súborov, ktoré zahŕňajú riešenie našich úloh pre toto obdobie. Príloha B zdokumentuje 6 reportov vytvorených za tento semester.

B.1 Report č. 1 v letnom semestri

Sprint1

ID	autor/názov/popis	od - do
 Issue: 151	[Marek.Jakab] Camera singleton Informacie o probleme: https://groups.google.com/forum/#!topic/teamtp05/ouoPuSTIH1k Vytvorenie singletonu pre pouzivanie kamery, vzhľadom na fakt, ze moze byt jej vyuzivanie vyžadovane viacero modulmi	[20. 02. 2014] ->
	20. 02. 2014 Informacie o probleme: https://groups.google.com/forum/#!topic/teamtp05/ouoPuSTIH1k Vytvorenie singletonu pre pouzivanie kamery, vzhľadom na fakt, ze moze byt jej vyuzivanie vyžadovane viacero modulmi	
	03. 03. 2014 https://github.com/marconak/Arvis/commit/92a96172b42be08f5f1c612634b6a926d0eda477 Zatial predbezna verzia. Nacr napadu. Singleton je spravny, zistene pocu kamier momentalne cez otvorenie a zatvorenie (inymi slovami, skusanie kolko zariadeni otvori) = OpenCV neponuka priamo riesenie tohto problemu Dalsi problem pri zozname kamier a ich ID je pri odpojeni kamery. Vtedy sa IDcka s najvacsou pravdepodobnostou reorganizuju. Nacr GUI, treba este pridat event pre select a dokoncit kod.	
 Issue: 155	[Marek.Jakab] Napisanie abstraktu na IIT SRC Vytvori abstrakt na IITSRC konferenciu. Navrhy su rozpracovane na https://docs.google.com/document/d/1d169bhi8j3L5vgNXUel3IGSj98016N8ur3b4g376Qls/edit Pripadne na mailing liste.	[17. 02. 2014] -> [19. 02. 2014]
	20. 02. 2014 Vytvori abstrakt na IITSRC konferenciu. Navrhy su rozpracovane na https://docs.google.com/document/d/1d169bhi8j3L5vgNXUel3IGSj98016N8ur3b4g376Qls/edit Pripadne na mailing liste.	
	22. 02. 2014 Abstrakt bol napisany. Obsah na: https://docs.google.com/document/d/1d169bhi8j3L5vgNXUel3IGSj98016N8ur3b4g376Qls/edit + Vykonana korekcia kolegami z timu Tlacena verzia odovzdana (Patrik).	
 Issue: 156	[Marek.Jakab] Oprava optimalizacie rozpoznania a uzavretie okna -> GitHub vetva https://github.com/marconak/Arvis/tree/features-optimalisationFaceDetection-Marek -Marek Odstranit problem vyskytu chyby pri najdeni objektu tvare na/za hranicou frameu. Pridat event pre uzavretie okna rozpoznania pri zavreti hlavneho okna programu. Nasledne vykonat pull request.	
	20. 02. 2014 GitHub vetva https://github.com/marconak/Arvis/tree/features-optimalisationFaceDetection-Marek Odstranit problem vyskytu chyby pri najdeni objektu tvare na/za hranicou frameu. Pridat event pre uzavretie okna rozpoznania pri zavreti hlavneho okna programu. Nasledne vykonat pull request.	
	23. 02. 2014 https://github.com/marconak/Arvis/commit/324c79aa8fddddd3dcb4967b8de26083acb750fac closing event	
	24. 02. 2014 github commit: https://github.com/marconak/Arvis/commit/cc1e8d7d1a1903706c16f2016212347ccdab215e Odstranil som predoslu verziu porovnavania ci je stvoruholnik okolo tvare len pripadnim "trasenim" Zmenil som to na sposob, ze v pripade ze sa tvar nachadza priblizne na rovnakom mieste (threshold) ostava stvoruholnik rovnaky. Snazim sa touto zmenou sposobit co najmenej nepotebnych zmien kde je pohyb tvarou sprevadzany celkovym trasenim vysledneho grafu. Ma to samozrejme aj negativa, ako su mensie skoky (ktore su vsak lepsie ako vela malych skokov aj ked sa clovek nehybe) Napadlo ma, ze neskor ak ostane cas mozeme to vyriesit pripadnou interpolaciou za urcity cas ci podobne.	
 Issue: 160	[Patrik.Hlavac] Prezentacia po zimnom semestri Priprava pptx	[23. 02. 2014] -> [24. 02. 2014]

23. 02. 2014

cca 8 slajdov s fotkami a osnovou pre prednasajuceho



[Patrik.Hlavac] Kinect modul

[23. 02. 2014] ->

Issue: 161



[Patrik.Hlavac] Export úloh z redmine

[02. 03. 2014] ->

Issue: 164

Evidencia úloh Po každom stretnutí tímu vyexportuje tím aktuálny stav úloh zo svojho systému manažmentu projektu, v čitateľnej a prehľadnej forme, do dokumentu vo formáte PDF a vystaví ho na webovej stránke tímu. Tím na stránke vystavuje a uchováva všetky takto exportované dokumenty. Tieto sú zároveň súčasťou projektovej dokumentácie, ktorá sa odovzdáva v kontrolných bodoch.



[Matej.Marconak] Oprava warningov v rámci 3DVisual programu

[18. 02. 2014] ->

Issue: 157

20. 02. 2014

V rámci programu počas jeho vývoja neboli nastavené príslušné parametre pre kompiláciu pre detegovanie warningov, čiže potenciálnych problémov. V rámci začiatku nového semestra je vhodné opraviť, čo najväčšie množstvo týchto warningov. Príslušný odkaz na Fórum <https://groups.google.com/forum/?hl=en#!topic/teamtp05/fUQfiHkKzD4>

20. 02. 2014

Vytvorená príslušná vetva na GitHube <https://github.com/marconak/Arvis/tree/hotfixes-repairWarnings-Matej> Oprava najčastejšie sa opakujúcich warningov - konverzia typov <https://github.com/marconak/Arvis/commit/bb76fb7274ac5ffba710bedfc70d0c0a561e4938> Oprava Switch a začiatok opravy nevyužívajúcich sa parametrov <https://github.com/marconak/Arvis/commit/19f428b44148ad40a238a3c60659beafe6df9e7b>

20. 02. 2014

Postupna oprava mensich warnings v najvacsich balikov v programe balik Data a Dao <https://github.com/marconak/Arvis/commit/39deca033c50a912c5ff3c10d47ef7fb560aa6a2> balik Network <https://github.com/marconak/Arvis/commit/beb8858d0da4d52a891e558a2975eaa8d2100564> balik Layout <https://github.com/marconak/Arvis/commit/37116f267022744be236bf1cf0bef6445699987a>

20. 02. 2014

Dokoncenie opravy nepouzivanych parametrov <https://github.com/marconak/Arvis/commit/b5bae9c7b309cc429b0dbf3f679a8120f36f2b80> Update konverzie medzi roznyimi typmi <https://github.com/marconak/Arvis/commit/413fa3098c094209182d43d786e0bf5d194dec6f> Testovanie v ramci platformy Linux a Windows Testovanie ostatnych parametrov na odlahovanie warningov - ostava 147 Podrobnejsie info na fore Konzultacia na fore so specifickymi problemami

23. 02. 2014

Dokoncena oprava warnings konverzii <https://github.com/marconak/Arvis/commit/086adcf50daf58e1c319908ddb0bf917b367f3a6> ostava - 26 chyb v porovnavani s float (double) 12 chyb z toho je v ramci tried databaz - oprava nie je nevyhnutna z dovodu ,ze nie su pouzivane, lebo funkcia databazy nie je dokoncena Z ostatnych asi 2 az 3 nie su v metodach, ktore nie su vobec volane, TODO doriesenie ostatnych (priblizne 10) chyb pri porovnavani

03. 03. 2014

oprava dalsich chyb na porovnavanie floatov <https://github.com/marconak/Arvis/commit/3ea7d78dd7d12248aaa39363142fa3bc1e64c6d0> oprava chyb pre windows <https://github.com/marconak/Arvis/commit/e6d9c07e0a2aae087069ae6dae8aeb79c4e5e955>

03. 03. 2014

Merge <https://github.com/marconak/Arvis/commit/dbb0e64e693399af8160510d5dea8d52d17be5dd> problemy s velkym poctom konfliktov problemy s automatickym mergom s doprednou deklaraciou



[Matej.Marconak] Oprava unity build na Windowse

->

Issue: 165

03. 03. 2014

Oprava unity build, aby mohol fungovať na systéme Windows
<https://github.com/marconak/Arvis/commit/1ebf96f23f2d0180ebf350863ea15d384be84875>



[Matej.Marconak] Technická podpora pri mergovaní a riešení chýb na systéme Linux

->

Issue: 166

03. 03. 2014

Riesenie mergov do develop vetvy <https://github.com/marconak/Arvis/commit/3d56e26d46b901377009b98f26ca944e699ba8a1>
 Riesenie konverzií v OpenCV triedach
<https://github.com/marconak/Arvis/commit/d72110b8756147d7c684b4f214833c3b8925237e> Riesenie warningov a chýb v Aruco vetve <https://github.com/marconak/Arvis/commit/cb030dfe2bdc8b21bbe9f2479888480e7ca4d800> Aktualizácia aruco vetvy <https://github.com/marconak/Arvis/commit/4d0ed8727ea1ad84cad0bb517c8bb6be009a6b8f>



[David.Durcak] Začlenenie Aruco do projektu

[17. 02. 2014] -> [02. 03. 2014]

Issue: 144

21. 02. 2014

Opis úlohy: Cieľom dokončiť už začaté začlenenie modulu Aruco do projektu a jeho úpravu, prípadne opravu. Najskôr je potrebná úprava CmakeListu, nakoľko momentálne je samotný modul už začlenený ale trieda ArucoCore ktorá ho využíva sa nekompiluje spolu s projektom. Následne bude potrebná jej úprava, aby spĺňala konvencie projektu. Ďalej pridanie metódy ktorá bude obaľovať potrebnú funkcionálnosť pre jej jednoduchšie použitie - napr.: metóda, ktorá dostane ako parameter obraz z kamery a vráti maticu detekovanej značky. A v neposlednom rade aj testovanie a oprava existujúcich metód.

21. 02. 2014

Úprava CmakeListu: pridanie arucocore k SRC pod podmienok ak sa naslo OpenCV podobne ako pri module zabezpečujúcom rozpoznanie tváre a oprava includovania hlavičkového súboru.
<https://github.com/marconak/Arvis/commit/b98751faa779a63635cac92fda02939161b2b340> Úprava triedy ArucoCore tak aby spĺňala stanovené konvencie. Nahradenie "cout" za qDebug() pri výpise chyby.
<https://github.com/marconak/Arvis/commit/64ebc444f2c1293bb6ba74fcd6d864a50a35d757> Pridanie namespace ArucoModul a oprava chyby pri výpise chyby pomocou qDebug().
<https://github.com/marconak/Arvis/commit/5795924ab30ee93617706e7d758612be79dde10d> Vytvorenie metódy obaľujúcej ostatné metódy, ktorá takto vykoná všetko potrebné. Dostane ako parameter snímku, deteguje značku vráti maticu opisujúcu polohu značky. Zmena ostatných funkcií na privátne.
<https://github.com/marconak/Arvis/commit/2744804170a4a29b2c122d0b1b06684cd396d048> Pridanie kalibračného súboru resources/config/camera.yml potrebného pri vytváraní inštancie Aruca. Pridanie metódy showDetImage pre debugovanie a kontrolu funkcionality Aruco - neskôr je plánované jej zmena nahradením zobrazenia snímky pomocou cv::imshow() signálom posielajúcim snímku pre riadne zobrazenie pomocou gui aplikácie (napr. v FaceRecognitionWindow)
<https://github.com/marconak/Arvis/commit/ef6582a4a87e8811b8d38bcab6e5432ff440b473>

03. 03. 2014

[CMakeList] oprava importovania aruco pri unity builde po zlúčení s developom.
<https://github.com/marconak/Arvis/commit/776e70fedae81805316362ac5b63ae1911398b34> [ArucoCore] pridanie getDetectedPosAndQuat() metódy pre detekciu značky, ktorá dostane ako parameter frame a vráti vektor jej posunutia a quaternion jej rotácie. Je to lepšia alternatíva podobnej existujúcej, ktorá poskytovala maticu. Je to nutné vylepšenie, nakoľko, ostatné časti projektu využívajú quaterniony, ktoré sú menej výpočtovo náročnejšie a aj ich posielanie je pamäťovo úspornejšie.
<https://github.com/marconak/Arvis/commit/3e799b24722d2a34f06888f999d7f4b6f7a28081>



[David.Durcak] Vytvorenie threadu pre Aruco

[17. 02. 2014] -> [02. 03. 2014]

Issue: 145

21. 02. 2014

Popis úlohy: Cieľom je vytvorenie threadu pre Aruco a jeho pridanie do CmakeListu. Thread by mal obnášať nasledovnú funkcionality: Mal by umožňovať pozastavenie a riadne ukončenie threadu. Samotná funkcia run by mala inicializovať kameru pomocou špeciálneho singletonu pre správu kamier a inštanciu triedy ArucoCore. Následne v cykle vykonávať získanie obrazu z kamery a jeho spracovanie pomocou ArucoCore, ktoré vráti maticu, s opisom pozície detekovanej značky. Táto matica bude následne emitovaná pre ďalšie spracovanie signálom threadu.

21. 02. 2014

Vytvorenie triedy ArucoThread (+ jej pridanie do CmakeListu). Vytvorenie základných metód threadu, pričom thread zatiaľ nič nerobí - run() obsahuje len prázdný cyklus.

<https://github.com/marconak/Arvis/commit/061a68a18ca685231a90bb138166e37367387fc2> Implementácia run() metódy.

Pridanie predbežnej funkcionality, ktorú bude thread zabezpečovať, aby bolo možné začať testovať ArucoCore a aj samotný thread, než bude implementovaný singleton pre kameru. Tieto metódy nakoniec nebudú volané priamo(camera nebude vytváraná priamo v threade) ale pomocou kamerového singletonu, čím sa nakoniec metóda run() podstatne zjednoduší.

<https://github.com/marconak/Arvis/commit/f8a6983ebeaffc30d43449de0aafc9a1e3342d9f> Drobné zmeny v threade(názov konfiguračného súboru, a dočasná náhrada while nekonečného cyklu za konečný).

<https://github.com/marconak/Arvis/commit/ef6582a4a87e8811b8d38bcab6e5432ff440b473>

03. 03. 2014

[ArucoThread] pridanie možnosti vypínať a zapínať korekciu a posielanie aktuálne spracovaného framu.

<https://github.com/marconak/Arvis/commit/f93b72b19bcc32be356791358da3edd5a3d98c3a>

03. 03. 2014

[ArucoThread] pridanie slotu updateCorectionPar();, ktorý spúšťa nastavenie parametrov z aktualneho snímku, až pokiaľ sa to nepodarí, a následne o tom pošle pridaný signál corParUpdated().

<https://github.com/marconak/Arvis/commit/ff9f2644bea57884279b0bfa7830ee207f8bb58>



[David.Durcak] Vytvorenie signalu, slotu a spojenia pre Aruco

[17. 02. 2014] -> [02. 03. 2014]

Issue: 146

21. 02. 2014

Opis úlohy: Cieľom je vytvorenie signálu v triede ArucoThread odosielajúceho maticu s informáciou o polohe značky. Ďalej vytvorenie slotu prijímajúceho rovnakú maticu pravdepodobne do triedy CameraManipulation v module Viewer, ktorý bude buď aktualizovať podobnú členskú premennú triedy, alebo spúšťať inú metódu ovplyvňujúcu polohu grafu so získanou maticou ako parametrom. Tiež aj vytvorenie spojenia vo vhodnej metóde, aby bolo toto spojenie možné priebežne rušiť alebo ukončovať, ak to bude potrebné.

23. 02. 2014

Do triedy CameraManipulation bol pridaný slot updateArucoGraphPosition(QMatrix4x4 mat), ktorý bude po napojení prijímať maticu z ArucoThreadu pre ďalšie spracovanie.

<https://github.com/marconak/Arvis/commit/483ee5ff322be8a2c21d787e9a7d4ec72410bb00> Pridanie

pushArucoMVMat(QMatrix4x4 mat) signálu do ArucoThread, a pridanie jeho volania do cyklu po získaní matice od aruca.

<https://github.com/marconak/Arvis/commit/15d8957e783fd2aec1fa24f830e07c4debec9cb0> Tieto signál a slot spoja

pravdepodobne (ako v prototypu) v OpenCVCore, kde sa najpravdepodobnejšie bude vytvárať inštancia ArucoThreadu, podobne ako pri detekcii tvaru.

28. 02. 2014

[CoreGraph] Bol pridaný nový graphRotTransf MatrixTransform uzol medzi root a graphGroup pre možnosť transformácie grafu.

Taktiež bolo pridané dedenie od QObject pre umožnenie pridania nového slotu updateArucoGraphRotation(QMatrix4x4 mat).

Tento slot transformuje maticu mat do požadovaného tvaru a aktualizuje uzol graphRotTransf, čím vlastne transformuje graf podľa matice mat a môže byť využitý pre rotáciu grafu podľa rotácie značky z aruca. Kvôli dedeniu od QObject bol aktualizovaný aj CMakeList. <https://github.com/marconak/Arvis/commit/86124bc72f3858d4028174a3aa7dd65ab02d241d>

<https://github.com/marconak/Arvis/commit/3ea27b87478e13e7b7c3b9b90a154cf72757acc0>

03. 03. 2014

[CameraManipulator] pridaný slot updateArucoGraphPosition(), ktorý už spracováva vektor namiesto matice. Bol vylepšený aj spôsob prepočtu pozície kamery oproti pôvodnému riešeniu.

<https://github.com/marconak/Arvis/commit/e671270d261509ac211c5e1b0d8c406e83cf8b96> [CoreGraph] pridanie slotu

updateArucoGraphRotation(), ktorý spracováva quaternion namiesto matice. Do triedy core/Core bol pridaný getCoreGraph(), aby bolo možné v budúcnosti pripojiť tento vytvorený slot s Aruco threadom.

<https://github.com/marconak/Arvis/commit/2d067be7fa06276b8839eb44a5a0272fc6c41757>



[David.Durcak] Implementacia algoritmu pre pohyb grafu znackou

[17. 02. 2014] -> [02. 03. 2014]

Issue: 148

21. 02. 2014

Opis úlohy: Cieľom je analyzovať možnosti zmeny grafu v priestore a nájsť spôsob ako túto polohu previazať s pohybom značky. Úlohou je teda implementácia metódy pre transformáciu matice s opisom polohy značky do vhodného tvaru a metódy meniacej premenné opisujúce polohu grafu podľa matice, podobne ako pri zmene polohy kamery pri zmene pozície hlavy. Ďalej aj prípadne pridanie parametrov pre budúce nastavenie.

23. 02. 2014

Zatiaľ jednoduchá implementácia pohybu značky podľa grafu. Je realizovaná pridaním dodatočnej translácie vo metódach `getMatrix` a `getInverseMatrix`. Pozícia grafu je ovplyvnená vo všetkých 3 smeroch, vrátane hĺbky. Je ešte potrebné upraviť niektoré parametre, hlavne aj po testovaní napr. pri priehľadnom plátne. Implementovaná ešte nieje rotácia, ktorá bude nasledovať. <https://github.com/marconak/Arvis/commit/483ee5ff322be8a2c21d787e9a7d4ec72410bb00>

26. 02. 2014

[ArucoThread] pridanie metódy pre výpočet korekčnej matice. Ta sa bude vypočítavať na začiatku ešte pre samotným cyklom threadu a neskôr bude v cykle slúžiť pre prepočet matice opisujúcej aktuálnu polohu značky, pred jej poslaním ďalej na spracovanie do pohybu grafu do `CameraManipulation`. Doteraz musela byť kamera vo vodorovnej rovine, čo znemožňovalo reálne detegovať značku položenú napr. na stole, pretože bola pod veľkým uhlom. Takto môže byť kamera umiestnená pod uhlom, (pozerať sa pod ľubovольným uhlom šikmo dole), a po jednoduchej kalibrácii = získaní matice pre výpočet korekčnej matice sa bude výsledná matica (= corekčná * získaná) dopočítavať, ako keby bola kamera vodorovne. * Boli pridané aj pomocné debug metódy a návrh testu, ktorý je potrebné implementovať. * Metóda je len implementovaná ale nepoužitá v run metóde, * Bude potrebné vytvorenie gui pre jej získanie vstupnej matice <https://github.com/marconak/Arvis/commit/4b53fd11eefec9592312d3e54192219e0532a8ec>

28. 02. 2014

[CoreGraph] Všetky skupiny uzlov (`GraphNode`) grafu, ktoré boli pôvodne priamo deťmi root uzla, sa stali deťmi novo pridaného `GraphNode` uzla, ktorý sa stal synom root uzla. Upravené boli aj ostatné metódy, ktoré doteraz spoliehali na poradie, že prvým dieťaťom root uzla je úráve pozadie (`skybox`). Takto bol celý graf oddelený do samostatnej vetvy aj pre jeho ľahšiu transformáciu v budúcnosti a taktiež aj ľahšiu možnosť zmeny pozadia. <https://github.com/marconak/Arvis/commit/31420f526417d1bd02b688f7f431f7cb929905e8> [CoreGraph] Bol pridaný nový `graphRotTransf` `MatrixTransform` uzol medzi root a `graphGroup` pre možnosť transformácie grafu. Taktiež bolo pridané dedenie od `QObject` pre umožnenie pridaného slotu `updateArucoGraphRotation(QMatrix4x4 mat)`. Tento slot transformuje maticu mat do požadovaného tvaru a aktualizuje uzol `graphRotTransf`, čím vlastne transformuje graf podľa matice mat a môže byť využitý pre rotáciu grafu podľa rotácie značky z aruca. Kvôli dedeniu od `QObject` bol aktualizovaný aj `CMakeList`. <https://github.com/marconak/Arvis/commit/86124bc72f3858d4028174a3aa7dd65ab02d241d> <https://github.com/marconak/Arvis/commit/3ea27b87478e13e7b7c3b9b90a154cf72757acc0> TODO: použiť namiesto matic quaterniony

03. 03. 2014

[ArucoThread] Úprava `run()` metódy v threade, aby získavala výsledky z aruca vo forme vektora posunutia a quaternionu rotácie namiesto matice. Pridanie 2 súvisiacich signálov, pre ich posielanie. Pridaná `computeCorQuatAndPos()` pre vytvorenie korekčného vektora a quaternionu a tiež metóda `correctQuatAndPos()`, ktorá ich pomocou nich koriguje vektor a quaternion pre aktuálny stav. Tieto sú vylepšením predchádzajúcej metódy `computeCorMat()`, ktorá využívala matice. Ešte však táto korekcia nefunguje korektné. Premenovanie a doplnenie ďalších drobností aby zodpovedali konvenciám. <https://github.com/marconak/Arvis/commit/d9a2e0f827c8d440922a30e06a8457775afe1ed4> [ArucoThread] Výmena typov vektoru a quaternionov na `osg::Vec3d` and `osg::Quat` namiesto `QVector` and `QQuaternion`, ktoré spôsobovali problémy a tiež úprava príslušných operácií. Pridaná korekcia y-osi vo vektory z aruca, nakoľko táto nie je pri rozdielnych pomeroch strán obrazu z kamery centrovaná ako os x. <https://github.com/marconak/Arvis/commit/aa5b5d6a0184d3ad700eadf2480f8571b5e70b4f> [ArucoThread] oprava matematicky nesprávnej inicializácie korekčného quaternionu v metóde `computeCorQuatAndPos()`. Pridanie tiež chýbajúceho nastavenia premennej `mCorSetted`, ktorá zabraňuje použitiu korekcie, ak táto nemá nastavené potrebné parametre. Ďalšie vylepšenie výpočtu pozície kamery podľa pozície značky v slotu `updateArucoGraphPosition()`. Podľa testov sa už korekcia pozície správa podľa očakávaní. <https://github.com/marconak/Arvis/commit/0b4bc9cb8ccc200cd937b9e83ad98fb759cdef9> [CoreGraph, CameraManipulator] vymazanie nepotrebných slotov pre úpravu pozície grafu a jeho rotácie využívajúce matice. <https://github.com/marconak/Arvis/commit/1ddccdc88c02d59526a8f90a7ea92bee02789b68> [ArucoThread] oprava korekcie aktuálnej rotácie. Graf sa už teraz pri otáčaní značky otáča v správnych osiach a správnym smerom. Pridaná možnosť nastavenia, či je značka pred obrazovkou, alebo zaňou, nakoľko v týchto 2 prípadoch sú rozdielne smery otáčania. <https://github.com/marconak/Arvis/commit/70ddd6895ad0f441149bb66860c7551af5918a8>



[Martina.Tregerova] Napisanie priebežnej spravy

[17. 02. 2014] -> [20. 02. 2014]

Issue: 154

20. 02. 2014

Vytvorila som základnu priebežnú spravu, hlavne z textového hladiska, poslala som ju dalej na kontrolu a Danovi na dokončenie. Druhu ulohu assignujem Danovi pretože dokončuje a nahadzuje všetky dokumenty. Martina



Issue: 158

[Martina.Tregerova] Pridanie vhodného GUI pre Aruco

Cieľom je začlenenie Aruca do používateľského rozhrania aplikácie. Úlohou je vhodne modifikovať triedu FaceRecognition window tak, aby umožňovala spúšťanie ArocoThreadu, prípadne aj pauzu a riadne jeho ukončenie napr. pomocou buttona, ale tiež aj riadne ukončenie v prípade zavretie okna. Ďalej pridanie možnosti zobrazenia aktuálne spracovaného snímku Arucum podobne ako pri detekcii tváre, čo zahŕňa pridanie potrebného signálu v ArucoThreade. Ďalej možnosť prepínať (napr. radio buttonami) medzi zobrazením snímok z rozpoznávania tváre, alebo detekcie značky z ArucaThreadu, čo môže byť realizované prepínaním medzi signálmi zasielajúcimi obraz snímky do okna - vytváranie a rušenie spojení z oboch zdrojových threadov (QObject::connect-unconnect).

[17. 02. 2014] -> [02. 03. 2014]

03. 03. 2014

Rozbehanie samotného arvisu trvalo dlhšie ako som predpokladala, unity build mi stále nefunguje preto až takto neskoro som sa k tomu dostala. Vytvorila som gui základne, bol pridany commit do vetvy features-OpencvArucoGui-Martina: <https://github.com/marconak/Arvis/commit/94ddabe8eae8a7aa7d17e717db78227d2e78d76d>

03. 03. 2014

Vytvorena funkcia pre radio buttons, zatiaľ prázdna, treba doplniť connect a disconnect, prídubne v rámci ďalších commitov. aktuálny commit: <https://github.com/marconak/Arvis/commit/70970ea3403eb3d16fa07768f8af49f1f28306b6>

03. 03. 2014

Pridana funkcionalita pre spodné checkboxy pre zapínanie a vypínanie threadov podľa toho ktorý checkbox je zaškrtnutý. Commit: <https://github.com/marconak/Arvis/commit/733e9aff3397b142fc2d58ebbbc248994aaff2b8>

03. 03. 2014

Upravené pomenovania premenných. Na stretnutí bolo dohodnuté, že keďže je táto uloha závislá na vytvorení singletonu pre tento sprint ju vyhlasujeme za ukončenú. Gui bolo upravené tak aby podporovalo rozoznávanie markera pomocou triedy aruco. Ďalšia implementácia zastavovania a spustania threadov bude doplnená po vytvorení singletonu. Posledný commit: <https://github.com/marconak/Arvis/commit/f5dd33c76f289bf3c1f682cd05ec50e0ad79f235>



Issue: 152

[Daniel.Soos] Napísanie abstraktu na IIT SRC

[19. 02. 2014] -> [20. 02. 2014]



Issue: 153

[Daniel.Soos] Napísanie priebežnej spravy

No description

[17. 02. 2014] -> [20. 02. 2014]



Issue: 159

[Daniel.Soos] Import OpenNI2 a NiTE

2 novovytvorené cmake suborov pre nájdenie kniznic pre prácu s Kinectom. Uprava projektového Cmakelists suboru tak, aby vedel program pokračovať v behu aj v prípade chýbajúcich cmake suborov [takisto ako OPENCV_FOUND nie je required]. Nalinkovanie libraries a header suborov do projektu, aby ďalej stacilo napísať do source kodu už len 'include openni.h' atd.

[23. 02. 2014] ->

24. 02. 2014

- commit na vlastnej vetve features-addOpenNIandNite-Daniel - pridane cmake subory a podľa toho upraveny Cmakelists.txt (find_package...) - ďalší krok bude merge s develop vetvou a otestovanie funkčnosti

24. 02. 2014

vytvorený pull request na vetvu develop; čaka sa na code review a následne mergnutie vetvy s develop vetvou - zmeny sa týkajú len 2 nových suborov a editovaní Cmakelists.txt

03. 03. 2014

korekcia CmakeLists - OpenNI2 a NiTE2 sa hlada len pre Win; pridanie potrebných dll do /_install/bin directory v CmakeLists. K tomu bolo potrebné modifikovať aj FindNiTE2.cmake subor. Updatnutý pull request.

03. 03. 2014

link na commit: <https://github.com/marconak/Arvis/commit/00e910f1e987ec42a5c3af3687530c95155b88ab>



[Jan.Handzus] Pridanie OSG objektu.

[24. 02. 2014] ->

Issue: 162

23. 02. 2014

Pridanie jednoducheho objektu, ktorý bude nasledne sluzit na zobrazenie realnej sceny.

03. 03. 2014

Ulohu nebolo možné dokončiť kvôli problémom po aktualizácii na platforme OSX. Vyvoj som musel presunúť na platformu Windows. Co si vyžiadalo nemalo hodín.

03. 03. 2014

Ulohu nebolo možné dokončiť kvôli problémom po aktualizácii na platforme OSX. Vyvoj som musel presunúť na platformu Windows. Co si vyžiadalo nemalo hodín.



[Jan.Handzus] Potiahnutie OSG objektu texturov sceny

[24. 02. 2014] ->

Issue: 163

23. 02. 2014

Potiahnutie objektu textúrou realnej sceny.

03. 03. 2014

Ulohu nebolo možné dokončiť kvôli problémom po aktualizácii na platforme OSX. Vyvoj som musel presunúť na platformu Windows. Co si vyžiadalo nemalo hodín.

B.2 Report č. 2 v letnom semestri

Sprint 2

ID	autor/názov/popis	od - do
 Issue: 170	[Marek.Jakab] Camera singleton Informacie o probleme: https://groups.google.com/forum/#!topic/teamtp05/ouoPuSTIH1k Vytvorenie singletonu pre pouzivanie kamery, vzhľadom na fakt, ze moze byt jej vyuzivanie vyzadovane viacero modulmi	[20. 02. 2014] ->
	05. 03. 2014 https://github.com/marconak/Arvis/commit/ca9c84069a8be902f5bc377a2897488434481979 Vytvorena hlavna funkcionalita. Stale este potreba vyriesit male problemy ako je zatvorenie okna (ak si nevyberieme ziadnu kameru), alebo vratenie objektu kamery (pouzita zatiaľ globalna premenna)	
	08. 03. 2014 https://github.com/marconak/Arvis/commit/3af0447f5b9d40aadadb064fecfa6a4e19012f56 Pridany closing event Pouzitie Singletonu: Funkcionalita sa nachadza vo vytvorenom objekte CapVideo Trieda CamSelectCore udrzuje vektor tychto objektov kamier Ziskanie kamery: CapVideo mCapVideo = OpenCV::CamSelectCore::getInstance()->selectCamera(); Kontrola na NULL ci kameru vratilo: if (mCapVideo==NULL) error Ziskanie matice: Mat mat = mCapVideo->queryFrame(); Release kamery: (ak nie je NULL) mCapVideo->release();	
 Issue: 174	[Marek.Jakab] Speech recognition	[03. 03. 2014] ->
 Issue: 172	[Patrik.Hlavac] Kinect modul	[23. 02. 2014] ->
	06. 03. 2014 Spravený CMakeLists a dve triedy v adresari Kinect. Nasleduje vyriesenie chyb pri kompilacii kodu z VisualStudia (pokracovanie ziskavanim a spracovanim obrazu)	
 Issue: 171	[Matej.Marconak] Prototyp giest s dvoma rukami Pomocou Kinect, sledovanie pohybu pre obe ruky.	[03. 03. 2014] ->
	09. 03. 2014 Riesenie problemu s Kinect kniznicami - problem na moje strane - pouzivanie 64 bit. kniznic Oprava problemu s importom dll - dopisane na WIKI Testovanie nitte:HandTracker Riesenie architektury	
	10. 03. 2014 Zapis navrhu do triedy MouseControl. Cakanie na Kinect core pre dalsiu implementaciu. Todo Testovanie s kinectom Vytvorenie docasnej implementacie na testy	
	10. 03. 2014 Testovanie s kinectom Bug Drivers kinect in instal/bin treba mat v osobitnom openni/driver pricinku Todo Uprava posunu mysky a ohranicenia, gest A kedy sa to aktivuje	



Issue: 175

[David.Durcak] Oprava prístupu do databazy**[03. 03. 2014] ->**

Zefektívnenie zobrazovania a nacistania.



Issue: 177

[David.Durcak] Dokončenie spoločného gui pre Aruco a Face Detection**[03. 03. 2014] -> [16. 03. 2014]**

Cieľom je dokončenie začlenená Aruco do používateľského rozhrania aplikácie. Cieľom je modifikovať triedu OpenCVCORE a FaceRecognition window tak, aby umožňovali spustenie a ukončenie Aruco a Face Detection threadov a ich riadne jeho ukončenie po ukončení aplikácie.

07. 03. 2014

[FaceRecognitionWindow] zmena includov na dopredne deklarácie, a vykonanie pár drobných zmien najmä v štýle a formátovaní, ktoré neovplyvňujú funkcionálnosť

<https://github.com/marconak/Arvis/commit/33f06302cd1645ebbd835c237e75bb7140a74853>

[FaceRecognitionWindow] začatie väčších zmien v organizácii triedy a zmien v spojení na thready, ktoré budú presunuté neskôr do OpenCVCORE. Zmeny v layoutovaní okna a pridanie niektorých nových widgetov a signálov a slotov, najmä pre ovládanie Aruco. V tomto štádiu nie sú pripojené žiadne signály a sloty, takže okno neovláda žiadnu funkcionálnosť.

<https://github.com/marconak/Arvis/commit/1e0d9333f2c1e81426216029634df3bba610344> [OpenCVCORE]

Vytvorenie niektorých hlavných spojení pre Aruco a čiastočne pre Face Detection a oprava pár chýb.

Ovládanie threadu pre aruco by už malo fungovať, ale chýbajú spojenia s ovládaním camery a grafu.

<https://github.com/marconak/Arvis/commit/f6d2d6af345ab7b69212cff97e77f54229e7aa29> TODO: - dokončiť zapojenie aruco, - dokončiť zapojenie pre face detection (bude treba pridať pár slotov a signálov do face detection thread) - pridať chýbajúce komentáre pre doxygen - vyriešiť vypznanie threadov po skončení hlavnej aplikácie, aby mohli bežať na poyadi aj pri yavretom okne FaceDetectionWindow, ktorý by mal len funkciu ovládania, ale zároveň aby nebolo potrebné mať ho otvorené stále.

07. 03. 2014

[OpenCVCORE, FaceDetectionWindow] Pridanie chýbajúcich spojení pre posielanie výsledkov z Aruco. Pridanie spojení pre Face Detection. Pridanie vypínania a zapínania posielania aktuálneho obrázku do Face Detection. Oprava problému s ukončením camery vo Face Det. threade po jeho ukončení. Oba moduly sú už funkčné a je ich už možné plne ovládať z okna, pod podmienkou, že nebudú spustené súčasne, kvôli chýbajúcemu camera singletonu. <https://github.com/marconak/Arvis/commit/cf7da8b5f58ff15dc7f91cb164a9fa8345fb0edc> TODO: - pridať chýbajúce komentáre pre doxygen - vyriešiť vypznanie threadov po skončení hlavnej aplikácie, aby mohli bežať na poyadi aj pri yavretom okne FaceDetectionWindow, ktorý by mal len funkciu ovládania, ale zároveň aby nebolo potrebné mať ho otvorené stále. - môže byť chýbne otvorených viac Face Detection okien súčasne will be solved with camera singleton. TODO: fix proble with exiting thread, when windows is closing, and fix problem that more than one Face detection window can be open.

08. 03. 2014

[OpenCVCORE] Úprava faceRecognition() tak aby bolo možné vytvoriť len 1 okno a tiež súvisiaca úprava closeEvent() vo FaceDetectionWindow, tak aby sa sice zavrelo okno, ale nezrušilo úplne a len sa zrušili všetky spojenia. Aby ho bolo možné opätovne zobraziť neskôr a pamätalo si stav (čo je efektívnejšie riešenie, než jeho úplne zrušenie, keď by si informácie o stave threadov muselo najsôr opätovne získať pri opätovnej inicializácii) a umožňuje to mať pustené thready na pozadí a okno mať zobrazené len v prípade potreby. <https://github.com/marconak/Arvis/commit/2da01f765980c1be9e46eadc6240f8eb4841ec78> TODO: - pridať chýbajúce komentáre pre doxygen - vyriešiť vypznanie threadov po skončení hlavnej aplikácie

09. 03. 2014

[OpenCVCORE, CoreWindow] Pridanie deštruktora do OpenCVCORE, ktorý ukončí oba thready, ukončí okno a uvoľní pamäť. Pridanie closeEvent() metódy do hlavného okna CoreWindow, ktorá sa volá tesne pred ukončením hlavného okna a v tejto sa zavolá pridaný deštruktork. Vďaka tomu sa môže zavrieť hlavné okno aj s bežiacimi threadami a otvoreným Face Detection oknom, ktoré sa korektné ukončia. Do CoreWindow bol pridaný aj vlastný AugmentedReality toolbar, do ktorého sa presunula funkcionálnosť z Collaboration toolbaru. Boli doplnené aj chýbajúce komentáre. <https://github.com/marconak/Arvis/commit/8fec911dba8d2918617b4722828fc82fba06c558>

 Issue: 173	[Martina.Tregerova] Rozoznanie prostredia za grafom [03. 03. 2014] -> Kinect senzor.
 Issue: 176	[Daniel.Soos] Tvorba dokumentu k retrospektive [03. 03. 2014] ->
07. 03. 2014 Napisana retrospektiva na zaklade google docu z februara a posledneho stretnutia k 1. sprintu v LS [3. tyzden]. Dokument sa nachadza na Dropbox v priecku Dokumenty/Reporty_retros_LS/ a onedlho aj na webstranke timu.	
 Issue: 168	[Jan.Handzus] Pridanie OSG objektu. [24. 02. 2014] ->
 Issue: 169	[Jan.Handzus] Potiahnutie OSG objektu texturov scény [24. 02. 2014] ->
 Issue: 167	[Dusko.Dogandzic] Oprava logických chýb [03. 03. 2014] -> Cez program cpp check.
10. 03. 2014 current changelog will be posted in group.	

B.3 Report č. 3 v letnom semestri

Sprint 2

ID	autor/názov/popis	od - do
 Issue: 170	[Marek.Jakab] Camera singleton Informacie o probleme: Vytvorenie singletonu pre pouzivanie kamery, vzhľadom na fakt, ze moze byt jej vyuzivanie vyzadovane viacero modulmi	[20. 02. 2014] ->
	05. 03. 2014 https://github.com/marconak/Arvis/commit/ca9c84069a8be902f5bc377a2897488434481979 Vytvorena hlavna funkcionalita. Stale este potreba vyriesit male problemy ako je zatvorenie okna (ak si nevyberieme ziadnu kameru), alebo vratenie objektu kamery (pouzita zatiaľ globalna premenná)	
	08. 03. 2014 https://github.com/marconak/Arvis/commit/3af0447f5b9d40aadadb064fecfa6a4e19012f56 Pridany closing event Pouzitie Singletonu: Funkcionalita sa nachadza vo vytvorenom objekte CapVideo Trieda CamSelectCore udrzuje vektor tychto objektov kamier Ziskanie kamery: CapVideo mCapVideo = OpenCV::CamSelectCore::getInstance()->selectCamera(); Kontrola na NULL ci kameru vratilo: if (mCapVideo==NULL) error Ziskanie matice: Mat mat = mCapVideo->queryFrame(); Release kamery: (ak nie je NULL) mCapVideo->release();	
 Issue: 174	[Marek.Jakab] Speech recognition	[03. 03. 2014] ->
	15. 03. 2014 https://github.com/marconak/Arvis/commit/008a9fe5b8d5e093d109cf5cacbfaf5540ac65c0 Kinect SDK podpora, pre rozpoznavanie hlasu. Potreba systemovej premennej KINECTSDK10_DIR, ktoru ale nastavi samotna instalacia Okrem ineho treba instalovat aj Kinect speech SDK	
	16. 03. 2014 Funkcionalitu budem momentalne vytvarat u seba na PC bez commitovania vzhľadom na to, ze potrebujem KinectCore z vetvy features-createKinectModule-Daniel. Pri vytvarani novej vetvy z developu a merge Danielovej vetvy nastali konflikty ktore som momentalne nebol schopny vyriesit. Navyse to vyzerá tak, ze niektorí maju problem s headerom glut.h v Danielovej vetve. vetvu features-SpeechRecognition-Marek teda zatiaľ ponechavam prazdnu a funkcionalitu vytvaram na vlastnom PC. Po mergnutí vetiev danu vetvu updatnem a dodam pozadovane zdrojove kody a funkcionalitu na rozpoznavanie hlasu.	
	17. 03. 2014 Nutne vyriesit problem, vid. https://groups.google.com/forum/#!topic/teamtp05/a-dwFIVH3b0	
 Issue: 172	[Patrik.Hlavac] Kinect modul	[23. 02. 2014] ->
	06. 03. 2014 Spravený CMakeLists a dve triedy v adresari Kinect. Nasleduje vyriesenie chyb pri kompilacii kodu z VisualStudia (pokracovanie ziskavanim a spracovanim obrazu)	
	21. 03. 2014 https://github.com/marconak/Arvis/commit/e83998f91475467b960e34e950a5ba5b69aca686 window https://github.com/marconak/Arvis/commit/a45b4ef02f97622ada1a586cdef6692981c9ef kinect core	



Issue: 171

[Matej.Marconak] Prototyp giest s dvoma rukami [03. 03. 2014] ->

Pomocou Kinect, sledovanie pohybu pre obe ruky.

09. 03. 2014

Riesenie problemu s Kinect kniznicami - problem na mojej strane - pouzivanie 64 bit. kniznic Oprava problemu s importom dll - dopisane na WIKI Testovanie nitte:HandTracker Riesenie architektury

10. 03. 2014

Zapis navrhov do triedy MouseControl. Cakanie na Kinect core pre dalsiu implementaciju. Todo Testovanie s kinectom Vytvorenie docasnej implementacie na testy

10. 03. 2014

Testovanie s kinectom Bug Drivers kinect in instal/bin treba mat v osobitnom openni/driver priecinku Todo Oprava posunu mysy a ohranicenia, gest A kedy sa to aktivuje

19. 03. 2014

Prvy prezentovany prototyp bol s windows API na mys, funkcie ako POINT CurrentMousePos; GetCursorPos(&CurrentMousePos); SetCursorPos() po prezentacii sa tento sposob zamietol z hladiska ze by funkcionálne bol len na systéme Windows. druhy zacaty sposob bol zalozeny na externej kniznici, no vsak bol zamietnuty. Nasledne sme diskutovali o sposoboch umiestnenia objektu a jeho naslednej manipulacia, ale aj tento sposob bol zamietnuty a nasledne po konzultacii sme sa dohodli na inom princípe. Z týchto dovodov je tento task zamietnuty.

19. 03. 2014

Uzatvorena. Zamiestnuta.



Issue: 175

[David.Durcak] Oprava pristupu do databazy [03. 03. 2014] -> [17. 03. 2014]

Zefektivnenie zobrazovania a nactania.

17. 03. 2014

Opis úlohy: Cieľom je dokončiť už pred letným semestrom začatú úlohu na už existujúcej vetve, ktorej cieľom bolo odstrániť nepotrebné vytváranie konecie na databázu pri štarte programu, tiež nepotrebnú snahu uložiť graf do databázy hneď pri jeho načítaní zo súboru a ďalšie nepotrebné prístupy k databáze, ktoré v prípade, že táto nie je k dispozícii značne predlžujú niektoré úkony. Je potrebné dokončiť tieto podúlohy: - presunúť ukladanie layoutu z CoreWindow do GraphManager - pri ukladani layoutu ak ešte nie je uložený graf tak ho predtým uložiť namiesto skončenia - vyhľadať a odstrániť ďalšie zbytočné prístupy k databáze a vytvoríť spojenie čo najneskôr ak nie vôbec

17. 03. 2014

[DB] pridanie metódy createConnection(), v ktorá zabezpečuje vytvorenie spojenia s databázou. Táto sa volá v tmpGetConn(), čiže sa spojenie vytvorí, až keď je potrebné, nie ako doteraz v konštruktoze. Vymazanie zbytočného volania getGraphs z GraphManager konštruktoza a getLayouts() z Graph::addLayout(), ktoré zbytočne pristupujú k databáze.

<https://github.com/marconak/Arvis/commit/81a371d6cc5aa42fd3ed2bfab96fa8c379efc9ca> [GraphManager] aktualizácia metódy saveActiveGraphToDB(). Predtým bola funkcionálna pre ukladanie grafu rozdelená medzi CoreWindow (ktorý by ju nemal sam vykonávať, ale iba zavolať) a touto metódou, pričom sa duplicitne vykonávali niektoré testy. Teraz sú všetky testy potrebné pred uložením grafu vykonané práve tu a spojenie sa netestuje zbytočne 2-krát.

<https://github.com/marconak/Arvis/commit/555954c898d7b65e6f20873bf3fb498931149778> [GraphManager] pridanie saveActiveLayoutToDB() metódy, do ktorej bola vrámcami možností presunutá funkcionálna z CoreWindow podobne ako pri ukladaní grafu. Core window iba yíska meno pre layout a zavola túto metódu, ktorá už zabezpečí potrebné testy a uloženie layoutu do databázy.

<https://github.com/marconak/Arvis/commit/e3ca095e6f45994d3e3c15163bd2ca84a8e6248a> [GraphManager] úprava saveActiveLayoutToDB() tak, aby v prípade, že pred uložením layoutu ešte nie je uložený graf, uloží sa najskôr tento graf a k nemu okrem originalného layoutu aj nový layout, ktorý chceme uložiť.

<https://github.com/marconak/Arvis/commit/59b44a5ba737d15ca4ff3e83d15b940a07179557>



Issue: 177

[David.Durcak] Dokončenie spoločného gui [03. 03. 2014] -> [16. 03. 2014] pre Aruco a Face Detection

Cieľom je dokončenie začlenenia Aruca do používateľského rozhrania aplikácie. Cieľom je modifikovať triedu OpenCVCore a FaceRecognition window tak, aby umožňovali spustenie a ukončenie Aruco a Face Detection threadov a ich riadne jeho ukončenie po ukončení aplikácie.

07. 03. 2014

[FaceRecognitionWindow] zmena includov na dopredne deklarácie, a vykonanie pár drobných zmien najmä v štýle a formátovaní, ktoré neovplyvňujú funkcionálnu

<https://github.com/marconak/Arvis/commit/33f06302cd1645ebbd835c237e75bb7140a74853>

[FaceRecognitionWindow] začatie väčších zmien v organizácii triedy a zmien v spojení na thready, ktoré budú presunuté neskôr do OpenCVCore. Zmeny v layoutovaní okna a pridanie niektorých nových widgetov a signálov a slotov, najmä pre ovládanie Aruca. V tomto štádiu nie sú pripojené žiadne signály a sloty, takže okno neovláda žiadnu funkcionálnu.

<https://github.com/marconak/Arvis/commit/1e0d9333f2c1e81426216029634df3bba610344> [OpenCVCore]

Vytvorenie niektorých hlavných spojení pre Aruco a čiastočne pre Face Detection a oprava pár chýb.

Ovládanie threadu pre aruco by už malo fungovať, ale chýbajú spojenia s ovládaním camery a grafu.

<https://github.com/marconak/Arvis/commit/f6d2d6af345ab7b69212cff97e77f54229e7aa29> TODO: - dokončiť zapojenie aruca, - dokončiť zapojenie pre face detection (bude treba pridať pár slotov a signálov do face detection thread) - pridať chybajúce komentáre pre doxygen - vyriešiť vypznanie threadov po skončení hlavnej aplikácie, aby mohli bežať na pozadí aj pri zavretí okna FaceDetectionWindow, ktorý by mal len funkciu ovládania, ale zároveň aby nebolo potrebné mať ho otvorené stále.

07. 03. 2014

[OpenCVCore, FaceDetectionWindow] Pridanie chýbajúcich spojení pre posielanie výsledkov z Aruca. Pridanie spojení pre Face Detection. Pridanie vypínania a zapínania posielania aktuálneho obrázku do Face Detection. Oprava problému s ukončením camery vo Face Det. threade po jeho ukončení. Oba moduly sú už funkčné a je ich už možné plne ovládať z okna, pod podmienkou, že nebudú spustené súčasne, kvôli chýbajúcemu camera singletonu. <https://github.com/marconak/Arvis/commit/cf7da8b5f58ff15dc7f91cb164a9fa8345fb0edc> TODO: - pridať chybajúce komentáre pre doxygen - vyriešiť vypznanie threadov po skončení hlavnej aplikácie, aby mohli bežať na pozadí aj pri zavretí okna FaceDetectionWindow, ktorý by mal len funkciu ovládania, ale zároveň aby nebolo potrebné mať ho otvorené stále. - môže byť chybne otvorených viac Face Detection okien súčasne will be solved with camera singleton. TODO: fix problem with exiting thread, when windows is closing, and fix problem that more than one Face detection window can be open.

08. 03. 2014

[OpenCVCore] Úprava faceRecognition() tak aby bolo možné vytvoriť len 1 okno a tiež súvisiaca úprava closeEvent() vo FaceDetectionWindow, tak aby sa sice zavrelo okno, ale nezrušilo úplne a len sa zrušili všetkz spojenia. Aby ho bolo možné opätovne zobrazit neskôr a pamätalo si stav(čo je efektívnejšie riešenie, než jeho úplne zrušenie, keď by si informácie o stave threadov muselo najskôr opätovne získať pri opätovnej inicializácii) a umožňuje to mať pustené thredy na pozadí a okno mať zobrazené len v prípade potreby.
<https://github.com/marconak/Arvis/commit/2da01f765980c1be9e46eadc6240f8eb4841ec78> TODO: - pridať chybajúce komentare pre doxygen - vyriesit vypznanie thredov po skončení hlavnej aplikácie

09. 03. 2014

[OpenCVCore, CoreWindow] Pridanie deštruktora do OpenCVCore, ktorý ukončí oba thredy, ukončí okno a uvoľní pamäť. Pridanie closeEvent() metódy do hlavného okna CoreWindow, ktorá sa volá tesne pred ukončením hlavného okna a v tejto sa zavolá pridaný deštruktork. Vďaka tomu sa môže zavrieť hlavné okna aj s bežiacimi threadami a otvoreným Face Detection oknom, ktoré sa korektné ukončia. Do CoreWindow bol pridaný aj vlastný AugmentedReality toolbar, do ktorého sa presunula funkcionlita z Collaboration toolbaru. Boli doplnené aj chýbajúce komentáre.
<https://github.com/marconak/Arvis/commit/8fec911dba8d2918617b4722828fc82fba06c558>



[Martina.Tregerova] Rozoznanie prostredia za grafom

[03. 03. 2014] ->

Kinect senzor.

Issue: 173

10. 03. 2014

Presla som viacero clankov ale v konecnom dosledku som dospela k uvahe, ze najjednoduchšie bude vytvorit reprezentáciu plochy pomocou bodov získaných z hlbkovej mapy, ktoré sa nachádzajú v urcitom intervale. Takto sme to uzavreli aj na cviceni, s tym ze dalsi postup, ktorý mam v plane je nainstalovat kinect senzor, zistit akym sposobom poskytuje hlbkovu mapu a z tejto vyskusat vytvorit jednotlivé plochy. Ak tato teoria nevyjde, v zalohe stale ostava rozpoznanie ploch pomocou jednotlivých aruco znaciak a pomocou ich definicie vytvorenie vyslednych ploch. Tuto ulohu uzatvaram a vytvaram si nove pre realnu implementáciu a skusanie kinectu.



[Martina.Tregerova] Rozpoznavanie ceny pomocou modulu Kinect

[10. 03. 2014] -> [17. 03. 2014]

Uloha zahrna: - rozbehnutie senzoru Kinect (pomocou OpeNNI, chcem to vyskusat nalinux, c bude fungovat) - zistenie v akom formate je vystupna hlbkova mapa, ktoru vracia Kinect - pokus o vytvorenie plochy steny a ak tento bude uspesny nasledne - pokus o vytvorenie plochy stola (realne reprezentovany pomocou tej istej plochy avsak jeho rozpoznanie bude na inej úrovni)

Issue: 178

15. 03. 2014

po vlastnej uvahe a naslednej konzultácii s veducim na stretnuti sme dospeli k zaveru, ze najvhodnesjei riesenie pre hladanie ploch, ktoré budu vytvarat obmedzenie pre grafy bude pomocou hlbkovej mapy najst oblasti ktorá maju priblizne rovaku hlbku. Priblizne rovnaka hlbka je v súčasnosti pracovny pojem a znamena taku hlbku ktorá sa nachadza v intervale hlbok, ktorý je rozumne dany napríklad od +0.2 do -0.2 od nejakej strednej hodnoty danej oblasti. To by znamenalo ze ak v oblasti su hlbky v rozmedzi od 1.5 do 1.7 hlbka 4 uz bude definovana ako ina plocha. V ramci dalsieho pokracovania prace na ulohu sa budeme sustredit na RANSAC algoritmus, a otestujeme moznost jeho vyuzitia pre nas projekt.



[Daniel.Soos] Tvorba dokumentu k retrospektive

[03. 03. 2014] ->

Issue: 176

07. 03. 2014

Napísana retrospektiva na základe google docu z februára a posledného stretnutia k 1. sprintu v LS [3. týždeň]. Dokument sa nachádza na Dropbox v priečinku Dokumenty/Reporty_retros_LS/ a onedlho aj na webstránke tímu.



[Jan.Handzus] Pridanie OSG objektu. [24. 02. 2014] ->

Issue: 168



[Jan.Handzus] Potiahnutie OSG objektu [24. 02. 2014] -> texturov scény

Issue: 169



[Jan.Handzus] Uprava cmake listu [03. 03. 2014] -> [17. 03. 2014]

Bolo potrebné upraviť cmake list a to v časti pre aruco a libnoise tak aby bol prehľadnejší a neobsahoval duplicitné nastavenia premenných.

Issue: 180

16. 03. 2014

Bolo potrebné upraviť cmake list a to v časti pre aruco a libnoise tak aby bol prehľadnejší a neobsahoval duplicitné nastavenia premenných.



[Dusko.Dogandzic] Oprava logických chýb [03. 03. 2014] ->

Cez program cpp check.

Issue: 167

10. 03. 2014

current changelog will be posted in group.

B.4 Report č. 4 v letnom semestri

Sprint 3

ID	autor/názov/popis	od - do
 Issue: 183	[Marek.Jakab] Speech recognition Kniznice: Microsoft Kinect SDK Microsoft Speech SDK Využitie hlasových povelov v programe. Aktivacia rozneho druhu (buttony, selecty a pod.) TASK: Vyriesit cim skor. Nutny kinect core (merge)	[03. 03. 2014] ->
	23. 03. 2014 https://github.com/marconak/Arvis/commit/a3a34c8373bbeb4bf6d26171f491edbf91d53b96 Pracuje sa na DLL kniznici, ktora sa prilinkuje. (Momentalne zabera viac casu ako bolo planovane, ale uz je to na dobrej ceste) Zakladne triedy a Cmakelist dokonceny.	
	23. 03. 2014 https://github.com/marconak/Arvis/commit/0c347cadaccb6fda743e8a1ded41818d5d5fc387 Bolo vytvorene DLLko pre speech recognition SpeechSDK header a SpeechSDKClass sa budu pouzivat pre zakladnu funkcionalitu rozpoznavania hlasu.	
	24. 03. 2014 https://github.com/marconak/Arvis/commit/25c860360b80e0e5acfd9651abc5d287ac0fb87 Pridany CMake support pre kniznice a DLL	
 Issue: 181	[Matej.Marconak] Uprava Kinect modulu Na zaklade uz vytvoreného kinect modulu nebola celkova komunikacia vyriesena. Na tomto podnete sa tento modul upravuje na zaklade tychto parametrov - cv::Mat konverzia, dve mozne obrazovky - farebna alebo hlbkova mapa - tradicny sposob slotov a signalov - ziaden GLUT obrazovka	[19. 03. 2014] ->
	19. 03. 2014 Hotovy zaklad tlacidla spolu s novou komunikaciou https://github.com/marconak/Arvis/commit/546bef06b8ba287b68ddb78cce58060ac6438967	
	19. 03. 2014 Vytvaranie zakladov na Kinect Windows https://github.com/marconak/Arvis/commit/d5e47e7dbcca1b646b5cdfa2b29363d113114f16	
	19. 03. 2014 Testovanie jednotlivych konverzií medzi cv::Mat a roznyimi druhmi openni - Handtracker, user tracker https://github.com/seiketkm/nite2sample/blob/master/HelloNiTE2/HelloNiTE2.cpp Vytvorenie zakladnych 2 sposobov zobrazovania - hlbkova mapa a jednoduche farebne zobrazenie	
	20. 03. 2014 https://github.com/marconak/Arvis/commit/4ca89bcb4386eb8daf331373765dcca6028d0ac7 https://github.com/marconak/Arvis/commit/ab05ca6b56d8e3822903f65282d1b4b8c493584c pridanie treadu a nasledne spojenie s oknom	
	20. 03. 2014 https://github.com/marconak/Arvis/commit/fe1791a1917aac7774e85a2c48e4e4f839004699 prva testovacia verzia, ktora posielala len do okna kameru, ale uz su vytvorene vsetky potrebne ostatne komponenty	
	20. 03. 2014 Pridana Kinect Recognition class https://github.com/marconak/Arvis/commit/168e1b22115c9fd780a427a2f08d3a4a38984304 Problemy - konverzia s vytvorenej cv::Mat do QImage riesim to druhym sposobom konverzie, ale v tej nie je funkcný Depth robrazovanie	

20. 03. 2014

<https://github.com/hiroyky/OniPlayer/blob/master/imageconverter.cpp> -- zdroj konverzie Na zaklade problemov boli pridane konverzie do QImage, ale bohuzial este neboli otestovane na spravnost funkcionality comit konverzii - <https://github.com/marconak/Arvis/commit/ffd45e50629dc2f5adc27feeee6b5ab03cbcf86> comit docasna uprava thread na kameru - <https://github.com/marconak/Arvis/commit/63b00cef0f72bf9d08d0b6708e1d169d8f688e4d> TODO doplnenie vsetkych slotov a signalov na start - stop, a praca s Threadom - je mozne aj bez Kinectu nasledne otestovanie QImage konverzie a doplnenie okna o dva prepinače - zatiaľ farebný výstup alebo hľbková mapa - potrebný Kinect nasledne je možné už s tým pracovať - treba vedieť čo treba ešte presne chceme dostávať a návrh gest

23. 03. 2014

<https://github.com/marconak/Arvis/commit/ed5ccdc27cf0e82194a7046bad0712fc86a8299> - core <https://github.com/marconak/Arvis/commit/3deb21a647ca3db9fabbe83805fe648fea8e6372> - thread <https://github.com/marconak/Arvis/commit/3ee3432db465c2c7b521a840993bb03f0965d5d4> - window zaklade upravy na kontrolu okna a threadov Koniec Tasku - dalsie upravy a pripadne hotfixy sa budu riesit nasledne TODO zatiaľ okno zobrazuje len obyčajnu kameru existujú funkcie na ovládanie Kinectu - staci len ich prepojenie



[Matej.Marconak] Vytvaranie vlastnych eventov pre mysku [19. 03. 2014] ->

Zakladom tejto ulohy je vytorit sposob ako kontrolovat mysku, a simulovat jednotlivé eventy.

Issue: 182

19. 03. 2014

Zakladne testovanie QT mysi na zaklade tohto navodu <http://stackoverflow.com/questions/18551387/tracking-mouse-coordinates-in-qt>

24. 03. 2014

ViewerQT spolu s adapter - geter a povolenie protected funkcií pre aktiváciu QMouseEventv Docasne demo pod klavesou I



[David.Durcak] Otáčanie grafu pomocou myši [21. 03. 2014] -> [30. 03. 2014]

Opis úlohy: V súčasnom stave sa dá otáčať graf pravým tlačidlom myši, avšak v skutočnosti sa graf neotáča, ale sa otáča kamera, čo môže byť v prípade nahradenia jedného obrázku v skyboxe obrázkom z kamery nežiadúce. Preto je nutné pridať možnosť otáčať samotným grafom myšou a neotáčať kamerou, a tiež možnosť prepínať medzi nimi.

Issue: 185

22. 03. 2014

[CoreGraph] pridanie slotu updateGraphRotByMouse() pre rotáciu grafu quaternionom z CameraManipulator úprava existujúceho slotu pre rotáciu podľa Aruca

<https://github.com/marconak/Arvis/commit/553a095e39c836b5e5c896c025eddeaca12246c7> [CoreGraph] Oprava bugov v slotoch pre rotáciu, quaterniony sa teraz už aj ukladajú.

<https://github.com/marconak/Arvis/commit/aca69d046316753cd70de62d752bb0145bcd655c>

[CameraManipulator] Pridanie slotu a signálu pre posielanie rotácie myšou a úprava calcMovement() metódy, aby ju emitovala. <https://github.com/marconak/Arvis/commit/110dc56cc6f7be68a673910de5a3219cc8dcaff6>

[CameraManipulator] Pridanie bool premennej _cameraCanRot a slotu, ktorý ju nastavuje pre možnosť vypínať rotáciu kamery myšou. Vytvorenie spojenia medzi CoreGraph a CameraManipulator vo ViewerQt konštruktore pre posielanie quaternionu. Oprava bugu s inverznou rotáciou.

<https://github.com/marconak/Arvis/commit/5bfac0f21af98751aa81572c1a45b89c84c3e160> [CoreGraph]

Pridanie checkboxu pre vypnutie rotácie kamery do Augment reality toolbaru a vytvorenie spojenia s príslušným slotom v CameraManipulator. Podľa prvých testov sa zdá byť otáčanie grafu myšou funkčné, avšak bude potrebné ešte riadne otestovanie.

<https://github.com/marconak/Arvis/commit/fd967a42c0d70875cae8a0a7ae3e75320f8c070c> TODO: Pri posúvaní podľa Aruca sa posúva kamera, čo tiež môže narušiť plán nahradiť bozadie v skyboxe obrazom s kamery, preto je potrebné ešte zabrániť aj tomuto posunutiu a upraviť ho tak, aby sa sa posúval graf namiesto kamery.

23. 03. 2014

[OpenCVCORE] Oprava problému s chýbajúci includom po mergi. Úprava na posielanie konštantných quaternionov z Aruco, kvôli efektívnosti. Úprava výpočtu posunutia kamery Arucou.

<https://github.com/marconak/Arvis/commit/6b74ac22cff02db2bd31af57442e809cf0f8b84c> Zistilo, že hoci Aruco a taktiež aj pri klávesové šípky posúvajú kameru, toto narušenie od rotácie Skybox kompenzuje, a výsledný efekt je ako keby sa posúval graf a pozadie sa nemenilo, preto posúvanie nespôsobuje problém.



Issue: 186

[David.Durcak] Úprava Aruca, aby používal Camera singleton**[17. 03. 2014] -> [30. 03. 2014]**

Opis úlohy: Nakoľko sa Aruco a Singleton pre kamery vyvíjali osobitne, Aruco tento singleton doposiaľ nepoužíva. Preto je ho potrebné upraviť, aby ho oužíval rovnakým spôsobom ako Face Detection. Taktiež počas mergovania mohli vyniknúť rôzne logické chyby, ktoré je potrebné opraviť a je potrebné otestovať, či môžu Face Detection a Aruco používať súčasne rovnakú kameru, a ak nie, je potrebné upraviť Camera Singleton, tak aby neposkytoval kameru, ale len obraz z kamery.

22. 03. 2014

[FaceRecognitionThread] oprava získania a ukončenia inštancie CapVideo, ktorá manažuje kameru z konštruktora do run() metódy. Vykonanie pár formátovacích zmien.

<https://github.com/marconak/Arvis/commit/eb6ec540db6f9acd0ce16f5772de1fcc1f8e755b> [FaceRecognition] Úprava selekcie v CamSelectWindow, tak aby nepoužívala nepotrebný slot a jeho odstránenie. Pridanie signálu a slotu pre nastavenie CapVideo inštancie do FaceRecognitionWindow a ich spojenie. Pridanie volania pre nastavenie kamery pred spustením threadu. Je to potrebné jednak, pretože nevieme, či to je kamera pre Aruco alebo pre FaceDetection a taktiež to dáva možnosť zvoliť inú kameru alebo iné nastavenia pri reštarte threadu. <https://github.com/marconak/Arvis/commit/e3a727989a1f7743f8afd3fd72045b8e2e168c31>

22. 03. 2014

[CapVideo] Úprava getGrayframe() metódy a vzmazanie nepotrebných creatGreatImage() metódy a vykonanie drobných zmien vo formátovaní.

<https://github.com/marconak/Arvis/commit/a797274cc36eaaaf72e4e76bc0638f499f5bb540> [ArucoThread]

Úprava Aruca, aby používal Camera Singleton. Pridanie inštancie mCapVideo, ktorá bude poskytovať frame a pridanie slotu setCapVideo() pre jej nastavenie. Vymazanie nepotrebného CaptureVideo.

<https://github.com/marconak/Arvis/commit/e52ba26e0e8808d781e75b4a30105debb7856dd>


[CamSelectWindow, CapVideo] Pridanie dopredných forward deklarácií do tried súvisiacich s FaceDetection.

<https://github.com/marconak/Arvis/commit/9ddc9bfb16d7ffd03c400ce0684708b59df96db6> [CamSelectCore, OpenCVCORE] Úprava selectCamera(), aby nevytvárala CamSelectWindow dynamicky, pretože 0je modálne a pri každom spustení sa vytvorí nové. Pridanie volania deštruktora pre CamSelectCore do OpenCVCORE deštruktora. <https://github.com/marconak/Arvis/commit/ef0be10e2c8ebf0ea66d1e08b33e75bd03ee96a5>

 Issue: 188	[Martina.Tregerova] Vytvorenie sceny za grafom [17. 03. 2014] -> [31. 03. 2014] Snaha o vytvorenie sceny za grafom. - Vyskusat moznost vyuzitia RANSAC algoritmu pre nas projekt. - Vytvorit plochu reprezentujucu zadnu stenu. Tymto sa pokusime vytvorit plochu ktora bude predstavovat obmedzenie pre graf tak aby nebolo mozne ist za alebo do tejto plochy grafom.
	17. 03. 2014 Prejdene moznosti RANSAC algoritmu, najdene vedecke a informacne clanky o algoritme, jeho vyuziti a podpore v kniznici openCV.
 Issue: 179	[Daniel.Soos] Kinect modul [13. 03. 2014] -> Issue #172 pokr.
	13. 03. 2014 Potrebne dll pre spustenie example v nasom projekte netreba dalej natvrdo davat do INSTALL_BIN, commit vytvara adresar OpenNI2 a do toho nainstaluje adresar Drivers, kde su dll. Predosly (Patrikov) commit zahrnul do projektu Kinect modul spolu aj so spustenim okna (GL), kde sa otvori example SimpleViewer. link na commit: https://github.com/marconak/Arvis/commit/556874171733b7fb47f17fa8543ab96b225879b2
 Issue: 187	[Daniel.Soos] Odovzдание camera ready abstraktu [22. 03. 2014] -> IIT SRC 2014
	23. 03. 2014 Rozsireny abstrakt odovzdany do konf. systemu; pribudli paragrafy reagujuce na paper review. Dokument sa nachadza na Dropboxe a suvisiaci mail bol odoslaný tiež do googlegroups.
 Issue: 184	[Jan.Handzus] Pridat vystup z kamery. [17. 03. 2014] ->

B.5 Report č. 5 v letnom semestri

Sprint 4

ID	autor/názov/popis	od - do
 Issue: 194	[Marek.Jakab] [FaceRecognition] uprava threadu a posuvania grafu Znížit zataž procesora pridaním sleep Preformulovať funkciu pre posun na základe interpolácie (prípadne ine napady), zabrániť "skakaniu" kamery	[02. 04. 2014] ->
	07. 04. 2014 https://github.com/marconak/Arvis/commit/d4160e31701a237141cead44ef08ffaa54802f1d Bola vykonaná uprava rozpoznávania. Pridany region vacšej veľkosti okolo hlavy. Výsledok: Stabilnejšie rozpoznávanie a predpokladaná menšia zataž CPU	
	07. 04. 2014 https://github.com/marconak/Arvis/commit/25b7aa0453084f7343b74e985b53c78299876f2d https://github.com/marconak/Arvis/commit/6775ce993e1f7dd86da4640d0098c8e9bec61693 https://github.com/marconak/Arvis/commit/2393280c7e74f6c7b81891bae5d13169148b78d8 Pridaná funkcionálnosť pre výpočet priemerných koordinátov očí. Pomocou zásobníka lifo. Odburávame tak inak potrebný for cyklus. Zrušený emit len v prípade pohybu o daný threshold, pre vytvorenie zdania jemnejšieho prechodu. Emit je vykonávaný v každom kroku, kedy je detegovaná tvár.	
	08. 04. 2014 Vykonaný pull request a merge. Bude sa ešte upravovať dequeue aby bol použiteľný aj mimo triedu.	
 Issue: 196	[Marek.Jakab] OBJ - Testovanie použitelnosti (vykreslenie do pozadia) s použitím kinect sdk vieme vygenerovať obj subor. Experimentálne prototypovanie vykreslenia na pozadie	[02. 04. 2014] ->
	15. 04. 2014 Prebehne hľadanie možného a podobného riešenia na internete	
 Issue: 203	[Marek.Jakab] [Queue] Vytvorenie queue o určenej veľkosti Pridat výpočet priemeru pre posuny grafu na scene Pridat source vo vlastnej triede	[08. 04. 2014] ->
	08. 04. 2014 https://github.com/marconak/Arvis/commit/c312e75342ed6d50b476393dd74de7f45afb252d Pridané základné triedy	
	08. 04. 2014 https://github.com/marconak/Arvis/commit/4c711be31397f470ae5a814ac6c6ffc35bed6c1b Pridaná funkcionálnosť pre SizedQueue konštruktor berie 2 argumenty. size = veľkosť stacku threshold = prah pod ktorý nezmení priemernú hodnotu.	
	08. 04. 2014 https://github.com/marconak/Arvis/commit/6fd2ec8e9d5b4d068d9de2944bf0a379e8f6df8e Použitá predosla implementácia stacku vo výpočte očí cez face detection.	
	08. 04. 2014 Vytvorený pull request David spravil code review mergnute	



[Patrik.Hlavac] Handtracker slider na rotovanie [01. 04. 2014] ->

Priprava triedy pre snimanie slajdovania grafu rukou.

Issue: 193



[Matej.Marconak] Refaktoring Kinect tried [05. 04. 2014] ->

V priebehu vyvoja Kinect modulu sa zanedbala dokumentacia a konzistencia kodu.

Issue: 198

05. 04. 2014

Upravil sa sposob deklaracie includov

<https://github.com/marconak/Arvis/commit/4b08c4e9b642231142947ff9c9bd145fb2c8e40bf> zdokumentovala sa cast kodu okolo kontroly mysy a pridala sa korekcia pohybu

<https://github.com/marconak/Arvis/commit/8e21ab18f9822c6c507e85668e18097d473f0d8e> uprava connectov a refaktoring kodu, cistenie

<https://github.com/marconak/Arvis/commit/24d445f6990a2facc2133daabd41bc2040cff7a2>



[Matej.Marconak] Pridanie parametrov do Kinectu na prisposobenie citlivosti ovladania [06. 04. 2014] ->

Citlivost ovladania na Kinecte sa zatiaľ neda nijako upravovat, preto je potrebne pridanie možnosti nastavovania parametrov

Issue: 200

06. 04. 2014

Najprv sa diala uprava start tlacidla

<https://github.com/marconak/Arvis/commit/c030f737556c31d569932e2896a9b8908c211569> Pridanie checkboxu na vypnutie kontroly mysy

<https://github.com/marconak/Arvis/commit/f125575b5306eb98061b4919dadd2f7ce29de077> a nasledne pridany slider na zaklade ktoreho sa da prisposobovat reakcie - zatiaľ pouzite priamo pre pohyb mysy

<https://github.com/marconak/Arvis/commit/90ff5b3b7cd3417b8a4f330f2fb894aec1ee4f09> Velka vdaka patri Marekovi za testovanie na Kinecte



[Matej.Marconak] Uprava CmakeList z pohladu Kinectu [09. 04. 2014] ->

V kinect moduloch sa zacalo pouzivat pri vykreslovani opencv a taktiez sa uz pouzivaju obe kniznice pre Kinect - Nite a openni - v tejto verzii to nie je nijako osetrene

Issue: 204

09. 04. 2014

- uprava CMakeListu - uprava pre tlacidlo Boli pridane podmienky pre Kinect kniznice a integraciu h a cpp pre kinect funkcionalitu Uprava CMakeListu z hladiska citatelnosti



Issue: 205

[Matej.Marconak] Pridanie funkcionality z externeho programu na vypocet rovin

[09. 04. 2014] ->

Martina vytvorila externy program, ktory dokaze vypocitavat roviny z dat, ktore nam poskytuje Kinect. Tento program je potrebny exportovat do nasho projektu 3DVisual

09. 04. 2014

Program bol vypracovany v systeme Microsoft Visual 2013, pricom vyuzival dost vela funkcionality, ktora v nasom projekte nie je podporovana. Z hladiska ze vyuzivame Microsoft Visual 2010 bol problem spustit na vlastnom stroji, nasledne po konzultacii s Marekom sme opravili chyby, ktore nam branili v spusteni a projekt sme mohli vyskusat.

09. 04. 2014

<https://github.com/marconak/Arvis/commit/c02b930cc40874095392b2135680f03323d1d8c8> - pridanie potrebných suborov
<https://github.com/marconak/Arvis/commit/805a622df77b493d3424461bf7d909147521b86c> - pridanie prvých dvoch tried - do projektu som pridal prve dve triedy, ktore jedna sluzi ako cisty vektor, a druha uz pokryva nejaku funkcionalitu algoritmu - parametre suborov som sa zatiaľ rozhodol len cisto riesit za pomoci konfigu, kde moznost neskor sa prida nejake ta funkcionalita na konkretny subor TODO pridanie dalsej casti su vyzaduje preskumat jadro algoritmu, ktory nam poskytla Martina z hladiska pouzitych funkcií a ich napojenia pre dalsie pouzitie

13. 04. 2014

Pridana posledna trieda - maly refaktoring potreby pri prepise
<https://github.com/marconak/Arvis/commit/03d0795494e4dd952f914f5a85e4819ca0bd9bb2>
<https://github.com/marconak/Arvis/commit/1c56c992b8322915d76e9d8354aa8e799de247c2> Pridany button + dalsi refaktoring potrebný
<https://github.com/marconak/Arvis/commit/4c0885376b321151548b5ffb3c6aaa8b7c80b6e6>
<https://github.com/marconak/Arvis/commit/d8b7c6f39e1810c4a6141d949a5b9b4ab61f9d5c>



Issue: 199

[David.Durcak] Obraz z kamery ako pozadie

[31. 03. 2014] -> [13. 04. 2014]

05. 04. 2014

Opis: Úlohou je pridať možnosť zobrazíť obraz z kamery ako pozadie pre graf. Úloha má viacero podúloh: - pridanie vhodného objektu do scény - zabrániť rotácii a posúvaniu tohto objektu - upravíť ho pre vhodné správanie pri rezizovaní okna - zakomponovať možnosť posielat obraz z aruca alebo z face detection - zistiť ako je možné možné konvertovať cv::Mat pre pouzitie v osg. - zistiť ako je možné dynamicky aktualizovat textúru

05. 04. 2014

[CoreGraph] pridanie metódy createTextureBacground() v ktorej sa vytvára uzol scény vo forme(QUAD) štvorca potiahnutého textúrou načítanou zo súboru. Podarilo sa nájsť vhodnú orientáciu a tiež aj pridať transformáciu, ktorá zabezpečuje stabilnú polohu pri scrollovaní (zväčšovaní a zmenšování grafu). Tiež sa podarilo dosiahnuť jeho ignoráciu pri hlbkovom teste, čiže graf sa nachádza ako keby pred ním.
<https://github.com/marconak/Arvis/commit/bbbcd1efe7b092735ee5c5bda66ff7bd5ccc27a2>

06. 04. 2014

[CoreGraph] prídanie konverzie z cv::Mat do osg::Image. Oprava orientácie textúry.
<https://github.com/marconak/Arvis/commit/98114cfc419239508233f9633fd02f13fa19f404> [CameraStream] prídanie triedy CameraStream, ktorá dedí od osg::ImageStream. Objekt tejto triedy môže byť použitý namiesto osg::Image pri vytváraní textúry, pričom však umožňuje dynamicky meniť zdrojový obraz vo vnútri triedy. Tie táto trieda dedí z QObjectu, či bolo možné pridať aj slot updateBackgroundImage(cv::Mat cvImg), ktorý konvertuje cv::Mat v parameri a aktualizuje vnútorný osg::Image a potom zavola dirty(), čím vyvolá automatickú aktualizáciu všetkých textúr, ktoré ho používajú.
<https://github.com/marconak/Arvis/commit/e8a73b9ab868c930de2bc9a19a5dc0b9fa4637eb> [CoreGraph] aktualizácia metódy createTextureBackground(), tak aby používala nový CameraStream. Opravený bol aj bug, a teraz sa vztvára najskôr geometry object, a až potom testúra a stateset vlastnosti tohto objektu, čo predtým spôsobovalo pád programu.
<https://github.com/marconak/Arvis/commit/4b0019ab1a694751da0135317acbc6d1f425754a>

06. 04. 2014

[ArucoThread, FaceRecognitionThread] prídanie signálov pushBackgrImage() pre posielanie obrazov z kamery z oboch threadov pre pozadie ak je nastavená pridaná premenná, a tiež prídanie slotu pre jej nastavenie.
<https://github.com/marconak/Arvis/commit/1cecc5cdb6beed87383d179f480bdf092b372a65> [CoreGraph] prídanie get metódy pre objekt CameraStream, aby mohol byť neskôr pripojený z threadmi, ktoré mu budú posilať obraz z kamery na jeho slot updateBackgroundImage().
<https://github.com/marconak/Arvis/commit/ab9fadd8be255c60cc20d102738ae53f779fca8c> [FaceRecognitionWindow] prídanie checkboxov do okna a príslušných slotov, pomocou ktorých ja možné povolenie/zakázanie posielanie aktuálneho obrazu pre z Aruca alebo faceDet threadov, cez pridané signály. Bolo to urobené tak, aby mohol iba 1 thread aktualizovať pozadie. pričom je možné ľahko prepínať, ktorý to bude. Zároveň je to nezávislé od zobrazovaného obrazu vo FaceRecognition okne.
<https://github.com/marconak/Arvis/commit/0875c62139fd27bafa796bd49148837af9ec622e> [OpenCVCore] pridané 4 prepojenia. 2 pre posielanie obrazu z Aruco/FaceDet do CameraStream objektu v CoreGraphe pre aktualizáciu pozadia. Tiež 2 pripojenia medzi threadmi a FaceRecognition oknom aby bolo možné ovládať toto video pozadie (vypínať/zapínať a prepínať medzi threadmi).
<https://github.com/marconak/Arvis/commit/5cc45d103c318050b84813c9085c4ba409d7c9ae> Toto je prvý funkčný prototyp video pozadia. TODO: - problém z rezizovaním okna - iný pomer strán v geometry objekte oproti obrazu z kamery - textúra je deformovaná - automaticke zakázanie rotácie kamery chýba.

07. 04. 2014

[CameraManipulator] prídanie automatického zakázania rotácie kamery pri štarte ak sa ma použiť video pozadie. <https://github.com/marconak/Arvis/commit/4502efaf561654ad56be188e1f61626a4b679c60> [CameraStream] prídanie metódy updateGeometryCoords() ktorá aktualizuje osg štvorec ktorý tvorí pozadie (vertexArray v geometry), a upravuje ho na obdĺžnik s pomerom strán rovnakým ako zdrojový obrázok pre pozadie, čím už nedochádza k deformácii textúry a jej aplikovanie v CoreGraphe.
<https://github.com/marconak/Arvis/commit/1881b726f507db55a5b02c2c4e1ffd4b3931b7>

17. 04. 2014

[CamerStream] Oprava chyby s prevodom farieb (RGB->BGR), oprava chyby horizontálnou veľkosťou a zmena zoomu geometry objektu pre pozadie, aby lepšie sedelo do obrazovky.
<https://github.com/marconak/Arvis/commit/07854ffa088ed3f66755ae8fa0e4949cb0aac3fa> [CoreGraph] Kvôli problémom pri rezizovaní (čierne okraje) u predchádzajúceho riešenia (3d object), bola pridaná metóda createOrtho2dBackground(), v ktorej je video pozadie riešené prepnutím sa do ortho 2d módu. Obraz sa tak prispôsobuje celej šírke obrazovky, avšak môže dochádzať k jeho roztiahnutiu, čo však pri podobnom pomere strán okna a videa, je zanedbateľné.
<https://github.com/marconak/Arvis/commit/6ed846d3b68382d9fb1596956e360c9e9b25ae95> [CoreGraph] Prídanie metódy createBackground(), aby sa oddelili a sprehľadnili všetky spôsoby vytvárania pozadia od ich implementácie. <https://github.com/marconak/Arvis/commit/0f11557f767c762013f3cf1bdcdbd1871efef89da>

17. 04. 2014

[CoreGraph] Zmena pridaných pointerov na ref_ptr objekty, oprava orientácie textúry, prídanie dopredných deklarácií, aktualizácia config súboru.
<https://github.com/marconak/Arvis/commit/2a3676b4a39da14fab8a0f1aa0770991933d3349> [CoreGraph, CameraStream] Oprava slabého výkonu (sekanie), keď je video pozadie aktívne.
<https://github.com/marconak/Arvis/commit/1ac1473be61305234929fbc08f90b8c1d4ca495>

17. 04. 2014

[CameraStream] Aby bolo možné program používať aj bez OpenCV, bola treída CameraStream presunutá do modulu OpenCV. <https://github.com/marconak/Arvis/commit/1073ae7daf80c76937da92273acb50cfebbc7cc6>
[CameraStream] Oprava chýb nastavením štartovacieho pozadia pre video pozadí, a tiež pri prevode zo cv::Mat do osg::Image pri jeho aktualizácií, spôsobenými rozdielnym správaním a nekompatibilitou používaných metód medzi Win32 a Linux platformami a oprava pár drobných chýb.
<https://github.com/marconak/Arvis/commit/dd31d04cd449521da7f4a71f88b16da4397c2a40>
<https://github.com/marconak/Arvis/commit/71e6a6e8bae9d8511ecd84bb390baceb8fb68e>
<https://github.com/marconak/Arvis/commit/5f461c97fe2bdbcf72659265363c112431004a68>
<https://github.com/marconak/Arvis/commit/2855be012b9f7f3e3e9e173adaaf79c0560ac118>



[Martina.Tregerova] Vytvorenie rovin [06. 04. 2014] -> z hlbkovej mapy

Issue: 202

16. 04. 2014

Pre dany sprint ukoncena, na stretnuti boli prezentovane vysledky.



[Jan.Handzus] Pridat vystup z kamery. [17. 03. 2014] ->

Issue: 197



[Dusko.Dogandzic] CMake Tools for [06. 04. 2014] -> Visual Studio

Issue: 201

07. 04. 2014

Description: Enables editing and writing cmake code in visual studio syntax auto-fill intellisense colored code. Integrates cmake GUI. Works with vs 2010,2012(2013 not supported but might work) Download: <http://cmaketools.codeplex.com/> Install: -Download and install -NO configuration in IDE -GUI is under tools

B.6 Report č. 6 v letnom semestri

Sprint 5

ID	autor/názov/popis	od - do
 Issue: 214	[Marek.Jakab] Vytvorenie prezentačných videí Task zahrnuje aj prípravu prostredia -plexisklo, projektor - otestovanie funkcionality.	[23. 04. 2014] ->
	27. 04. 2014 Videá vytvorené. Este chyba pár funkcionality natocit. Zároveň pripraviť poriadne na testovanie. (koncovky, predzovacky). Videá sa práve spracovávajú u Pata.	
	30. 04. 2014 Dokončené videá (aruco-markery)	
 Issue: 207	[Patrik.Hlavac] Vytvorenie postera Popis z EMAILU Tlačit môžeme postery formátu *** A1 vytlačeny na Landscape *** (na to nezabudnite, že to musí byť landscape, aby to cele nejakú slušne vyžeralo a tiež sa lepšie prezentovalo). Ak okrem panelu nepredpokladáme nič iné, pojde v prvom rade o prezentáciu výsledkov formou diskusie, takže všetko čo chcete diskutovať dajte na poster, dajte si tam pomôcky vo forme obrázkov, schém, vzorcu, ale aj hlavná myšlienka, cieľ...	[19. 04. 2014] ->
 Issue: 213	[Patrik.Hlavac] Vytvorenie grafu s obrázkami - preskúmanie stavu jednotlivých grafov a ich schému - vytvorenie programu, ktorý doteraz potrebné informácie/ obrázky ku grafu - konzultácia s http://team05-13.ucebne.fiit.stuba.sk:443/issues/212 , v akom formáte spracovať nový graf	[19. 04. 2014] ->
	22. 04. 2014 Skúsil som viacero API na robenie screenshotov, nemilo ma prekvapili limity a platene verzie. Nakoniec som sparsoval všetkých 650 obrázkov, pričom 230 stránok už neexistovalo	
 Issue: 219	[Patrik.Hlavac] Video prezentácia Spraviť dlhšiu súvislú prezentáciu videí z produktu. Planovaná prezentácia na dopytovanom LCD-ku	[27. 04. 2014] ->

	<p>[Patrik.Hlavac] Natocenie videa Natocit implementovanu funkcionalitu.</p>	<p>[27. 04. 2014] -></p>
<p>Issue: 220</p>		
	<p>[Matej.Marconak] Priprava programu na odovzdanie tretej strane Program je potrebne otestovat a nasledne poopracovat bug, ktore obsahuje - zameranie je hlavne na nasu implementaciju a bugy, ktore sposobuju pad aplikacie</p>	<p>[24. 04. 2014] -></p>
<p>Issue: 215</p>		
<p>24. 04. 2014 Testovanie hlavných tlačidiel na načítanie grafu, a veci okolo databázy, ktoré spomalovali chod. https://github.com/marconak/Arvis/commit/e75e0036e366aa03a4f56932e6987743ce31a317 nasledne uprava veci ohladom povoleni buttonov , atd https://github.com/marconak/Arvis/commit/f2e97fbb2b0449b4a3bb0edde33989b7d8d16c9a https://github.com/marconak/Arvis/commit/a5479a46ecb298f32275d3ab5a6f2d9ff349ba38 https://github.com/marconak/Arvis/commit/1c10a1a86182f6b63532088127b91fdaf81c6054</p> <p>26. 04. 2014 Pridane dalsie bugy https://github.com/marconak/Arvis/pull/34 - zaciatok opravy Kinect spojenia https://github.com/marconak/Arvis/issues/31 https://github.com/marconak/Arvis/issues/30 riesene zatiaľ Kinect - refactor codu a zmenena logika zapinania - testovanie s Marekom - este prebieha konzultacia zaciatok riesenia mys - konzultacia s Davidom - objaveno dva problemy - problem s resize okna, problem s komunikaciou medzi threadmi</p> <p>26. 04. 2014 zaciatok prace na mys a Linuxe https://github.com/marconak/Arvis/tree/hotfixes-mouseRefactor-Matej - vetva nakoniec je potrebne presunut funkcionalitu do hlavneho threadu ostava ze vyriesit ostatne funkcie pre Kinect</p> <p>29. 04. 2014 Ukoncene - testovanie s Marekom, pridanie ukoncovanie, priprava scenarom pre prezentovanie na IIT.SRC</p>		
	<p>[Matej.Marconak] Pridanie readme - GITHUB Pridanie readme do repozitara zo zakladnymi informaciami</p>	<p>[26. 04. 2014] -></p>
<p>Issue: 218</p>		
<p>26. 04. 2014 https://github.com/marconak/Arvis</p> <p>29. 04. 2014 Male upravavy - ukoncene.</p>		
	<p>[David.Durcak] Ovladanie kurzora pomocou aruco značky Úlohou je rozšírenie funkcionality Aruca, kde sa momentálne manipuluje s grafom ako celkom, tak aby bolo možné pomocou ďalšej značky ovládať kurzor myši, podobne ako pomocou kinect. Cieľom je dokázať pohybovať kurzorom, vedieť simulovať kliknutie, napr. otočením značky, aby bolo možné označiť uzly grafu.</p>	<p>[14. 04. 2014] -> [27. 04. 2014]</p>
<p>Issue: 206</p>		

25. 04. 2014

[MouseControl] Presun triedy MouseControl do modulu Viewer a aktualizácia iných ovplyvnených súborov
<https://github.com/marconak/Arvis/commit/438880d095df7cce4da93bea03c6815df92135b7>
<https://github.com/marconak/Arvis/commit/4b33098f0d1e84dd0fcffd675320c676c6e93b72> [ArucoCore]
 pridanie metódy detect() ktorá bude len detekovať markery z obrázka a uloží si ich, aby bolo možné neskôr získať informácie o ktoromkoľvek z nich.
<https://github.com/marconak/Arvis/commit/ab3cd3103122880ae0a1ebc6b526f2aba42efd9f> [ArucoCore]
 Pridanie metódy getPosAndQuat(), ktorá vráti pozíciu a rotáciu zvoleného markera.
<https://github.com/marconak/Arvis/commit/e84d3bddd7810be67bdd17e1f1139141514224e0> [ArucoThread]
 aktualizácia run() metódy, aby využívala novšie metódy detect() a getPosAndQuat(). Takto bude možné využívať naraz viacero markerov pre súčasné ovládanie grafu, ale aj ovládanie myši.
<https://github.com/marconak/Arvis/commit/596adc02c24dd3c6578e0e87179c332f4bda5e8f>

25. 04. 2014

[ArucoThread] pridanie slotu interchangeMarkers(), aby bolo možné prepínať prioritu markerov. V prípade, že má užívateľ len 1 marker, bude možné prepínať medzi ovládaním myši a ovládaním grafu.
<https://github.com/marconak/Arvis/commit/67d2761a53ad7330a91c9ee6d4f04f0d9682163d> [ArucoThread]
 pridanie metódy imagesSending() pre oddelenie tejto funkcionality z metódy run(). Oprava niektorých konverzií. <https://github.com/marconak/Arvis/commit/418aef1762d6287d6fdbfa9130f3594987ef7120>
 [ArucoThread] pridanie metódy graphControlling() pre oddelenie funkcionality ovládania grafu Arucom z metódy run(). <https://github.com/marconak/Arvis/commit/70be65951abff6370a42d37c43c3c3a8896ef72c>
 [ArucoThread] pridanie metódy mouseControlling() pre ovládanie myši pomocou Aruca.
<https://github.com/marconak/Arvis/commit/f42a069aad0b5cc17eca37c9d76c4bf8ec39e0ff>

25. 04. 2014

[MouseControl] pridanie metódy moveMouseAruco(), ktorá prijíma normalizované koordináty a prepočítava ich na reálne koordináty na obrazovke. Ovládanie myši nie je implementované zatiaľ.
<https://github.com/marconak/Arvis/commit/ad56e7e956c2debea55055aac8205be7c43842e8> [ArucoThread]
 úprava metódy mouseControlling() tak, aby prepočítavala koordináty z aruca na normalizované koordináty. Pridanie MouseControll inštancie, ktorej sú potom tieto koordináty potom predané.
<https://github.com/marconak/Arvis/commit/b1787c3748abd4ff1bcc48a36942a5c28588edd9> [MouseControl]
 úprava metódy moveMouseAruco() tak aby hýbala kurzorom. Oprava chýb.
<https://github.com/marconak/Arvis/commit/613ca93c779c7477e2c6856c90f9b44c569d1919>
<https://github.com/marconak/Arvis/commit/85ed9b52571680d45c845a1f6671906e6a83c4fa>
<https://github.com/marconak/Arvis/commit/88d73b6733f5fbd6f94dc81c3369619a693a267> [ArucoThread]
 oprava chyby s posielaním inverzného koordinátu x.
<https://github.com/marconak/Arvis/commit/a6b1464e5c172de53ab1bfb4e3a9c3a3ed8e132>
 [FaceRecognitionWindow] pridanie tlačidla pre prepínanie priority markerov a jeho prepojenie so skôr pridaným slotom interchangeMarkers().
<https://github.com/marconak/Arvis/commit/65e5637758bb9ae5ff00d65bfb2333b35ddd3769>



[David.Durcak] Funkcia fullscreen

Pridanie možnosti prepínať do fullscreen módu, kvôli lepšiemu zobrazeniu a pohodlnejšej práci.

[24. 04. 2014] -> [27. 04. 2014]

Issue: 216

25. 04. 2014

[PickHandler] Pridanie možnosti prepínať medzi normálnym oknom a fullscreen zobrazením pri stlačení tlačidla [I/L]. <https://github.com/marconak/Arvis/commit/b60c4af6294445641a61332c41215647628c429d>



[David.Durcak] Pridanie možnosti súčasného použitia rotácie grafu pre Kinect a FaceDetecion.

Momentálne nie je možné súčasne rotovať grafom pomocou Kinectu a Face Detection, pretože používajú spoločný slot a dochádza tak k nesprávne zobrazeniu, kedy graf skáče z jednej rotácie do druhej. Cieľom je odstrániť tent problém.

[24. 04. 2014] -> [27. 04. 2014]

Issue: 217

25. 04. 2014

[CameraManipulator] pridanie slotov setRotationHeadKinect() a setRotationHeadDet(), ktoré zabezpečia oddelenie ovládania rotácie grafu pre Face Detection a Kinect, aby bolo možné používať oba súčasne. Aktualizácia metódy setRotationHead(), ktorá už bude zabezpečovať len výpočet rotácie s parametrov, ktoré jej pridané sloty aktualizujú a nebude ju možné použiť ako slot.
<https://github.com/marconak/Arvis/commit/679c9fb31b3035d499b37b6865fc20526921bb7> Aktualizácia spojení(v connect() metódach) v KinectCore a OpenCVcore abz používali nové sloty. Oprava chýb a aktualizácia komentárov.
<https://github.com/marconak/Arvis/commit/ae7cdce3d9eb325621851b0fcb6c2620f9d172d2>
<https://github.com/marconak/Arvis/commit/df67f690df8f4d48ce50bcadffb55aa6e0d1cb9f>



Issue: 208

[Martina.Tregerova] prezentacia timu na portali robime.it

[19. 04. 2014] ->

Info z EMAILU V tejto etape to znamená, že treba, aby ste ***** v termíne najneskor do IIT.SRC ***** spracovali krátku textovú obrázovú prezentáciu na portál robime.it. Prezentácia (text, obrázky) by mala predstaviť váš tím (kto ste, v čom ste dobrí), produkt, ktorý vyvíjate (čo je na nom inovatívne, prínosy pre používateľov) a aj ďalšie veci, ktoré môžu byť zaujímavé pre verejnosť (váš príbeh, ako sa váš projekt vyvíjal, čo ste sa naučili, čo nabuduce spravíte ináč). Prezentácia by nemala byť veľmi dlhá, aby si ju čo najviac ľudí prečítalo. Môžete kludne použiť čokoľvek aj z toho čo budete pripravovať ako odpovede na otázky do dotazníka, ktorý som pred chvíľou poslala. Preto to posielam teraz, že to je možné dobre spraviť aj naraz a neodkladať. Keď prezentáciu vášho tímu a projektu zverejníte na portáli, pošlite mail s odkazom p. Mrskovej (mrskova@fiit.stuba.sk). Nezapodnajte v prezentácii sa jasne identifikovať, že ste z FIIT STU a toto je váš projekt v rámci súťaže TP CUP.

26. 04. 2014

Vytvorený článok, čaká sa na schválenie ostatnými členmi alebo aspoň niektorými. Odkaz na článok: https://docs.google.com/document/d/1onAZ9_krJOYkK4nTAtmt3ZdxDMIUwGBTowr2YH2Gj0/edit V priebehu záťažovej bude doplnené, dopravné a poslané.

30. 04. 2014

Prezentácia tu: <http://robime.it/tp-cup-2014-arvis-zobrazenie-ovladanie-grafu-pomocou-gest-v-realnej-scene/>



Issue: 209

[Daniel.Soos] Používateľská príručka

[19. 04. 2014] ->

Na základe odovzdávania projektu na testovanie tretej strany je potrebné vypracovať používateľskú príručku. Obsahovať - starú príručku *popis Aruco a Face okna * popis Kinect okna * popis Speech - v prípade nájdenia chyby, problému s programom - reportovať - Matejovi alebo Marekovi



Issue: 210

[Daniel.Soos] Aktualizácia instalačnej príručky

[19. 04. 2014] ->

Od poslednej aktualizácie sme pridali do programu viacero nových technológií - preto je potrebné aktualizovať instalačnú príručku pre ďalšie generácie.

24. 04. 2014

Navod na instaláciu (pre Windows) je v priečinku Dropbox/Tim/Dokumenty/Navody_Prirucky (subor arvis_instalacia_win).



Issue: 212

[Jan.Handzus] Preskumanie možnosti nactavanie obrzkov v grafe [19. 04. 2014] ->

Treba preskumat ci v grafe sa da nactavat obrzku a v akej forme - kde sa nactavaju jednotlivé uzly, v akom formate - potrebne zmeny, keby sme chceli zobrazovat v tychto uzlov obrzky z internetu - preskumat staru verziu, ktora takuto funkcionalitu obsahuje



Issue: 221

[Jan.Handzus] Pridanie funkcionality na vykreslovanie obrzkov vo vntri grafov [29. 04. 2014] ->

Po preskumani funkcionality nactavanie grafu, je potrebne pridanie funkcie na nactavanie obrzkov a ich nasledne vykreslenie.

C Obsah elektronického média

- /Arvis/doc - adresár obsahujúci dokumentáciu k projektu,
- /Arvis/doc-other - adresár obsahujúci dopĺňajúce dokumenty projektu,
- /Arvis/install - adresár so spustiteľnou verziou aplikácie,
- /Arvis/sources - adresár obsahujúci zdrojové súbory aplikácie,
- /libraries - adresár obsahujúci inštalacné súbory knižníc OpenNI a Nite,
- /webpage - adresár obsahujúci statickú verziu webovej stránky projektu,
- readme.txt - popis obsahu elektronického média.

D Preberací protokol

Preberací protokol

Tímový projekt 2013/2014

Tím č. 5 - ARVis

Predmet odovzdávania:

Dokumentácia riadenia - finálna verzia po letnom semestri

Dokumentácia k inžinierskemu dielu - finálna verzia po letnom semestri

Vedúci tímového projektu: Ing. Peter Kapec, PhD.

Podpisom potvrdzuje prevzatie vyššie uvedených častí dokumentácie.

V Bratislave

.....

Dátum

.....