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 I
Kontakt: teamtp05@gmail.com
Akademický rok: 20213/2014, zimný semester

Obsah

1 Úvod

Tento dokument predstavuje dokumentáciu 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. Príloha B zachytáva návod na inštaláciu softvéru 3DVisual, a v prílohe C je opísaný používateľský manuál k nemu. Obsah elektronického média je zdokumentovaný v prílohe D 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 CESCG. Prog-

ramuje v C/C++, Jave 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 priraďujeme 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 odpútali 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 sprehľadnenie 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 informácii 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 multimediálnym 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
  - Analýza OpenSceneGraph
  - Zmena pozadia v programe
- Bc. Dávid Durčák
  - Celkový pohľad na architektonický model prebratého softvérového riešenia
  - Oprava chýb v predchádzajúcom riešení
- Bc. Ján Handzuš
  - Analýza ARToolkit
  - Úprava programu pre novšie verzie knižníc
- Bc. Patrik Hlaváč
  - Analýza Kinectu a jeho import do softvéru
  - Spojazdnenie servera (vrátane Redmine) a tvorba webstránky
- Bc. Marek Jakab
  - Rozpoznanie tváre a jeho import do softvéru

  - Import OpenCV do softvéru

- Bc. Matej Marcoňák
  - Analýza Git
  - Oprava chýb v predchádzajúcom riešení
  - Rozbitie programu na moduly
- Bc. Daniel Soós
  - Analýza Kinectu a jeho import do softvéru
  - Tvorba dokumentácie
- Bc. Martina Trégerová
  - Analýza osgART
  - Rozpoznávanie markerov

3.3 Podiel práce členov na jednotlivých častiach dokumentácie

3.3.1 Dokumentácia k inžinierskemu dielu

- Bc. Duško Dogandžić
  - Kapitola 2.1.1, 6.6
- Bc. Dávid Durčák
  - Kapitoly 1, 4.3, 4.4, 5.2, 6.2
- Bc. Ján Handzuš
  - Kapitoly 3.3, 3.4, 4.5, 5.4, 6.4
- Bc. Patrik Hlaváč
  - Kapitola 2.3
- Bc. Marek Jakab
  - Kapitoly 2.1.2, 3.5, 4.1, 5.1, 6.1
- Bc. Matej Marcoňák

  - Kapitola 2.2, 3.6, 4.2, 5.3, 6.3

- Bc. Daniel Soós
  - Kapitola 3.2, 5.5

- Bc. Martina Trégerová
  - Kapitoly 3.1, 5.4, 6.5

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
Zápisnice zo stretnutí	všetci
Návod na inštaláciu	Bc. Dávid Durčák Bc. Ján Handzuš Bc. Marek Jakab
Prílohy C, D, E	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. Zimný semester je rozdelený na päť šprintov. V tabuľke 1 vidíme náš semestrový plán.

Tabuľka 1: *Harmonogram nášho tímu v zimnom semestri*

týždeň v semestri	plán
1.	odovzdanie zoznamu kompentencií tímu
2.	pridelenie témy, rozdelenie úloh
3.	naplánovanie prvého šprintu a jeho riešenie
4.	finalizácia šprintu č.1
5.	práca na šprinte č.2
6.	finalizácia šprintu č.2
7.	práca na šprinte č.3
8.	finalizácia šprintu č.3
9.	práca na šprinte č.4 a odovzdávanie priebežných dokumentov
10.	finalizácia šprintu č.4
11.	práca na šprinte č. 5
12.	finalizácia šprintu č. 5, finalizácia výstupov a odovzdávanie

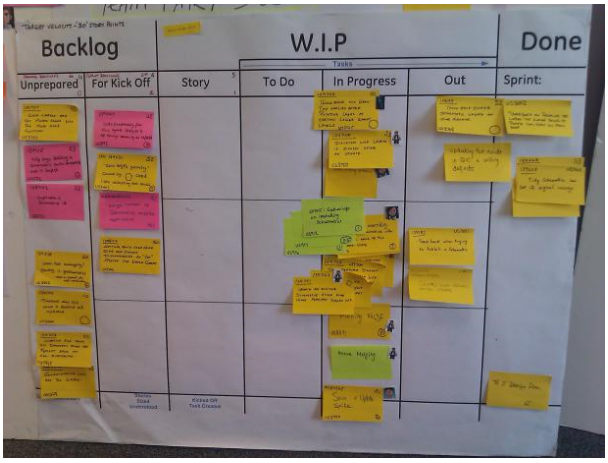
5 Metodiky použité pri vývoji

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 realizácií 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 neskoršie použité pri analyzovaní stavu úloh. A táto analýza poslúži 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 špecifikovane 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 další Sprint a vyhodnotenie predchádzajúcej práce.

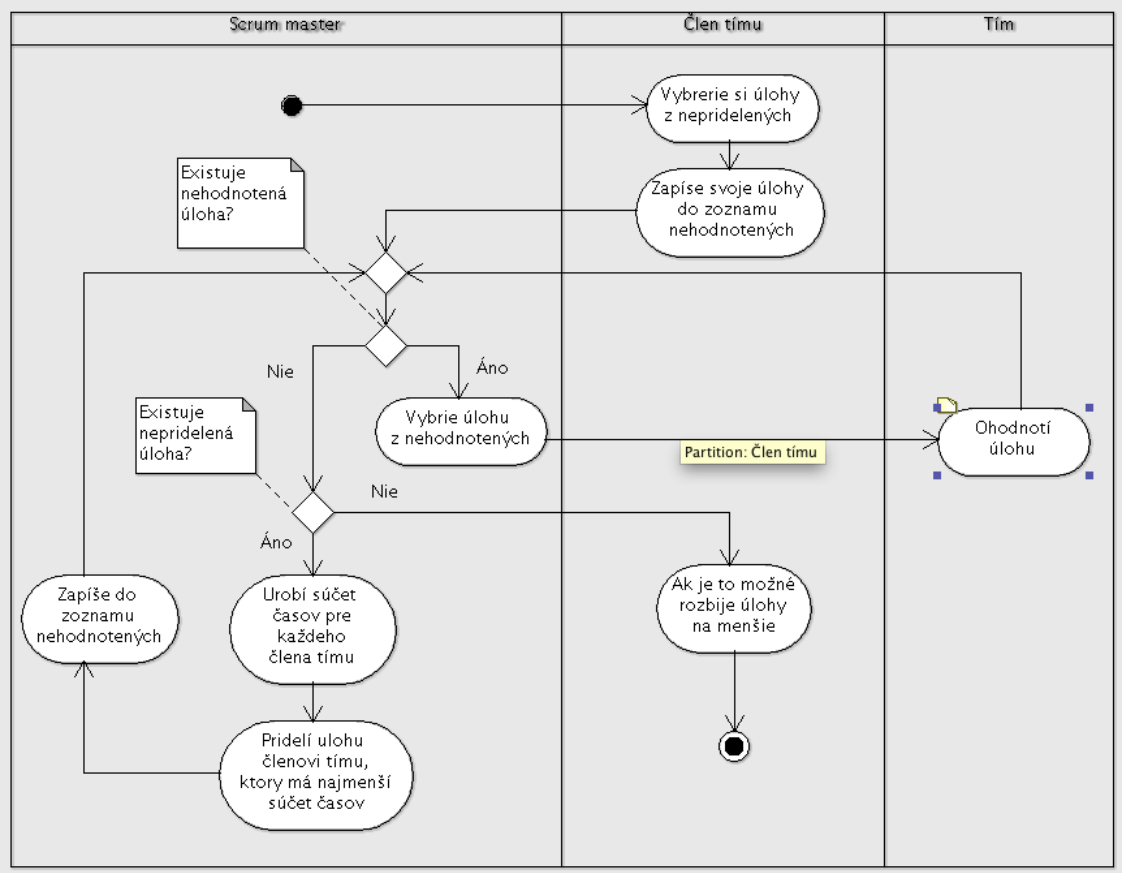


Obr. 1: *Scrum Board*

- **Scrum board** - tabuľa, ktorá obsahuje informácie o všetkých cieľoch a ich prioritách, ktoré má spĺňať 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 spĺňať. Zodpovedá hlavne za priority všetkých cieľov, ktoré sú potrebné pri plánovaní Sprintu. Tento člen tímu však nehodnotí jednotlivé úlohy, ale môže pomocť 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 další 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 Öhodnotenie 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 súčeť 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 další 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 spĺňať produkt. V tomto prípade bude požiadaný 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 úlohy 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 metodík

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, centrálne a lokálne.
- **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 funkcionalitu, 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 funkcionalitu.
- **Commit** - zoznam zmien, ktoré nastali v lokálnom repozitáre.

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

5.2.3 Role a ich zodpovednosti

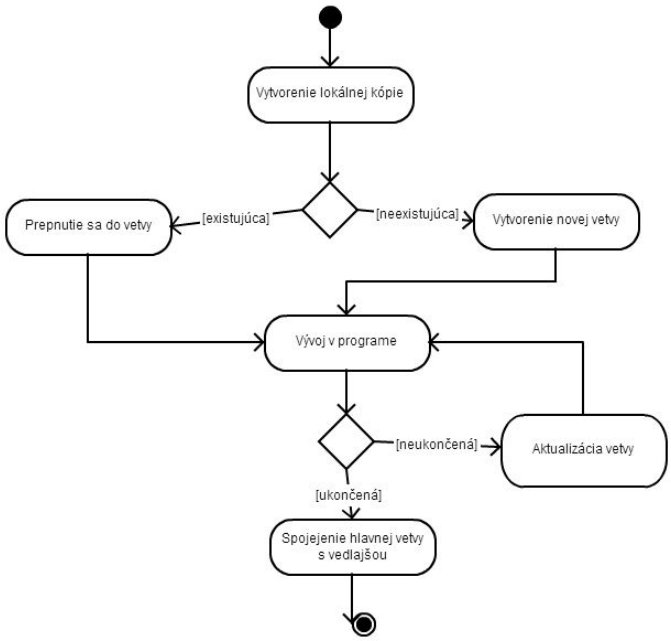
Rola	Zodpovednosť
Programátor/Vývojár	- Vytváranie novej funckionality - Oprava chýb - Komentovanie kódu
Správca repozitára	- Správa centrálneho repozitára

5.2.4 Procesy manažmentu verzií

Pri procese správy verzií 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 verzií*

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 centrálneho 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 funkcionalitu, 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 vykované na lokálnom úložisko 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 vhodne 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 pridájú 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 centrálneho 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 centrálneho 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 centrálneho 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 dodržujú 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 pisaniu 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 závisle 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 ïnclude".
- 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 namespacom. 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 analogický 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 neoddeľujeme.

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 funkcionalitu 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){
        <pozadovana funcionalita>
    }
    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, semafórmi 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 $QOject$ 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 templatoch 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. Vyhýbať 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 číselïným 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. Vyhýbať 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 metodík

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	- Vytvoriť funkcionalitu 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	- 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	- Vykonáva merge vetvy vypracovanej developerom s hlavnou develop vetvou - Možnosť konečného prijatia funkcionality pre následné testovanie alebo odmietnutia pre prepracovanie.
Člen tímu	- 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 funkcionalitu 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

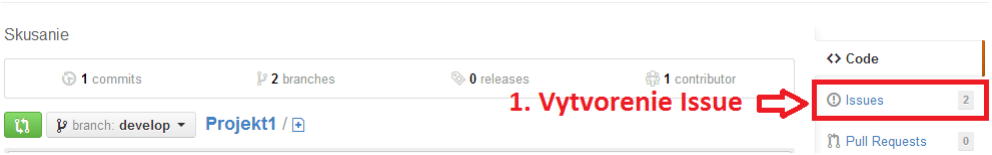
Pri vytváraní funkcionality pre systém sa riadime metodikou pre github(2.3), kde sa vytvárajú branche pre jednotlivé funkcionality na krátky čas. Toto je dôvod, prečo nemusí prehliadka kódu prebehnúť zvlášť pre každý github commit, ale je možné zahájiť proces prehliadky kódu až po dokončení funkcionality.

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 funkcionalitu otestovať na rôznych platformách. Vytvorenie issue prebieha nasledovne:

1. Vo vetve, ktorú chceme prehliadnuť spustíme issue.

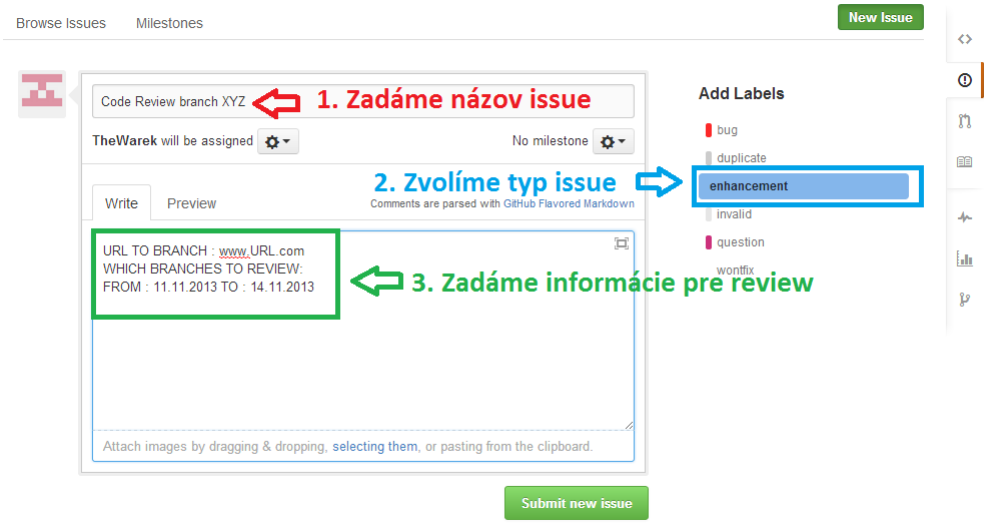


2. Nastavíme parametre pre issue:

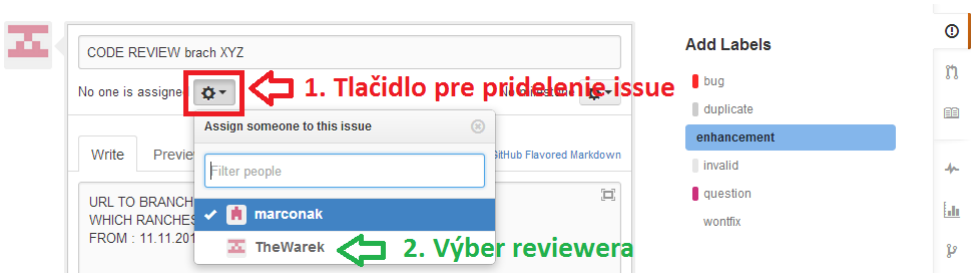
   (a) Názov: ”Code Review branchä pridáme názov samotnej vetvy.

   (b) Zvolíme typ issue. Pre prehlaidku kódu to je: ënhancement”.

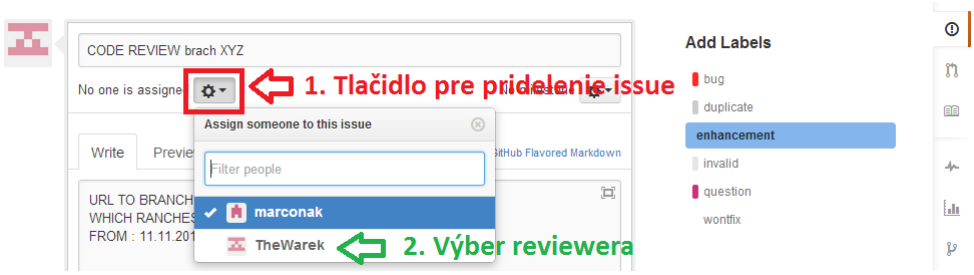
   (c) Zadáme parametre pre reviewera podľa obrázku. Nesmie chýbať odkaz na branch. Predpokladá sa prehliadka celého branchu. V prípade prehliadky jednotlivých commitov ich konkretizujeme.



3. Zvolíme komu chceme issue priradiť.



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

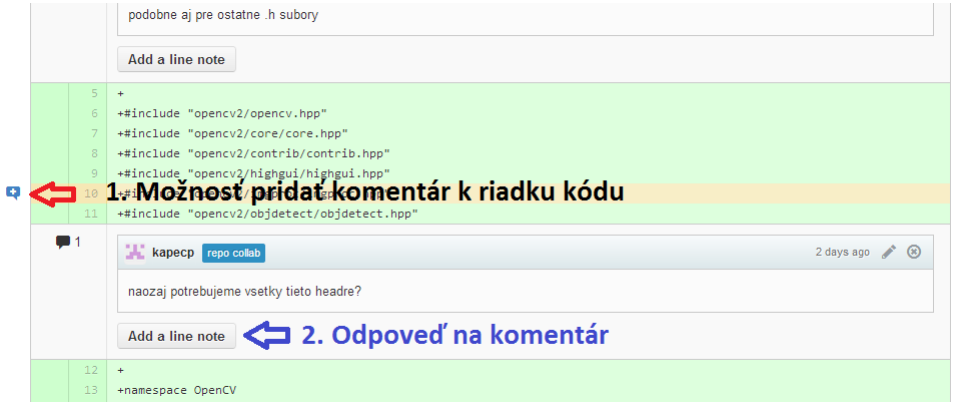


Účastníci budú upozornení e-mailom a môžu začať vykonávať prehliadku kódu.

5.4.4.3 Vykonanie prehliadky kódu

Vstup: Pridanie issue pre reviewera
Výstup: Vykonaná preliadka kódu
Zúčastnený: Reviewer

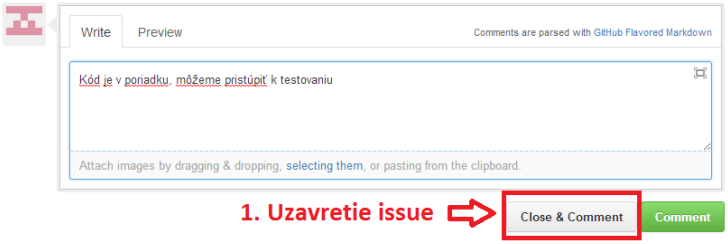
Reviewer podľa pokynov vykoná prehliadku kódu. Účelom je mať prehľadný zdrojový kód podľa metodiky písania programového kódu (2.1). Konvencie pre reviewera pri prehliadke kódu môžeme nájsť v prílohe. V prípade konfliktu reviewer opíše prostredníctvom github komentáru ku riadku kódu problém a zároveň 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 funkcionalita 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 funkcionalita 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 umožnuje 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 prezerať kód, doplniť doňho inú funkcionalitu 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 popis 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 destruktor.
        */
    ~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 funkcionalitu,
- 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 funkcionalitu 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 prezerať 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 reviever: 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 funkcionalitu, ktorá nadväzuje na inú, takú na ktorej pracoval alebo pracuje iný člen tímu. V prípade, že programátor potrebuje túto funkcionalitu 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 funkcionalitu za neúplnú, prípadne ju chce z akéhokoľvek dôvodu rozšíriť. V tomto prípade rovnako vytvorí blok kódu, kde definuje túto funkcionalitu.

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 zložitosťou - 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 zadefinovaný používatelia, je možné prejsť na kapitolu 5.6.3.2. Aby mohol systém fungovať interaktívne pre každého používateľa, je nutné zadefinovať jednotlivé práva používateľov a zadelenie 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 nadradeného 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 aplikácií 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 Ïn 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.

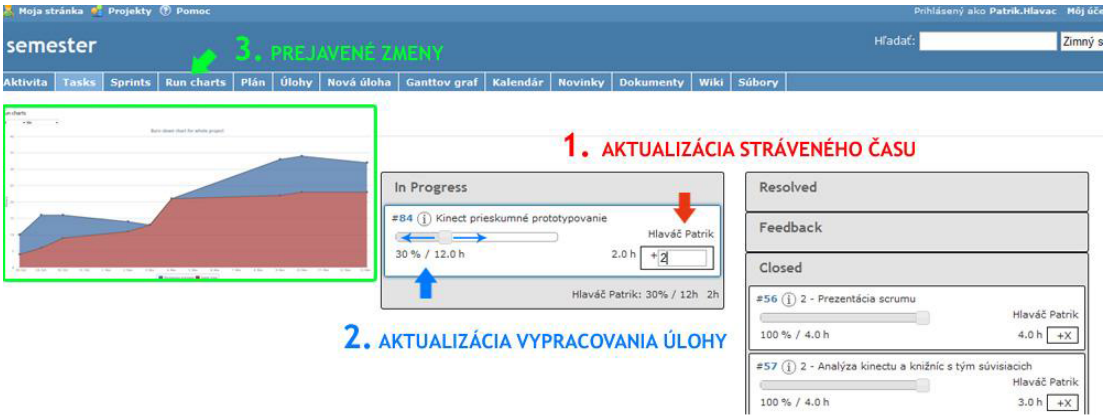
5.6.3.4 Aktualizácia

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.



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 ”feedbackälebo ”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. Testeri a vývojári spolupracujú vo vypracovaní testovacích úloh. Unit testy sú zadefinované a vykonané. Testy na funkcionalitu sú taktiež potrebné, aby nedošlo ku konfliktu v porovnaní s požiadavkami vedúceho. Po každej reintegrácii jedného komponentu je potrebné vykonať zadefinované testy, aby bol komponent kompatibilný s ostatnými časťami softvéru.

Review Meeting
Kým sa spraví prehliadka 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 produktivia pre ďalšie iterácie.

5.7.2 Role a ich povinnosti

- Testeri 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 poskytli spätnú väzbu, akceptačné kritéria sú nanovo definované a vytvoria sa testovacie prípady.
- Ak sa používajú automatizované testovacie frameworky, testeri vyprodukujú 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.
- Testeri 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

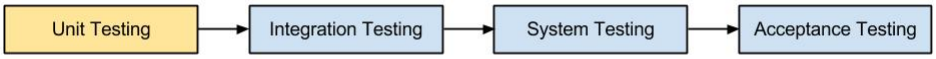
Unit testy sa konajú v každom šprinte s cieľom testovania funkcionality každého komponentu. Pričom pre každý jeden komponent sa napíše testovací scenár a ten je následne nasadený na ten komponent, aby sa zhodnotila jeho funkčnosť. Motivácia unit testov spočíva v tom, aby sa zistilo, nakoľko daný komponent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

Unit testy píšu vývojári a testeri. 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 píšu 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.

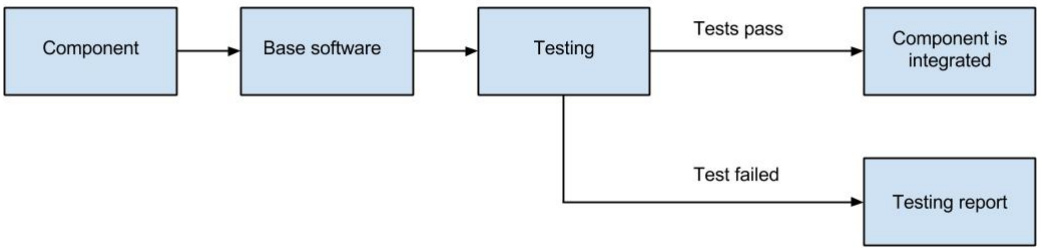
Unit testy by sa mali rozvíjať spolu so softvérom - refaktoring unit testov sa koná v každom šprinte, ak to je žiadúce.



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 vytvorenýmp rototypom 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 prehľadáva 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 prehľadne zobrazujú vo výstupnom dokumente.

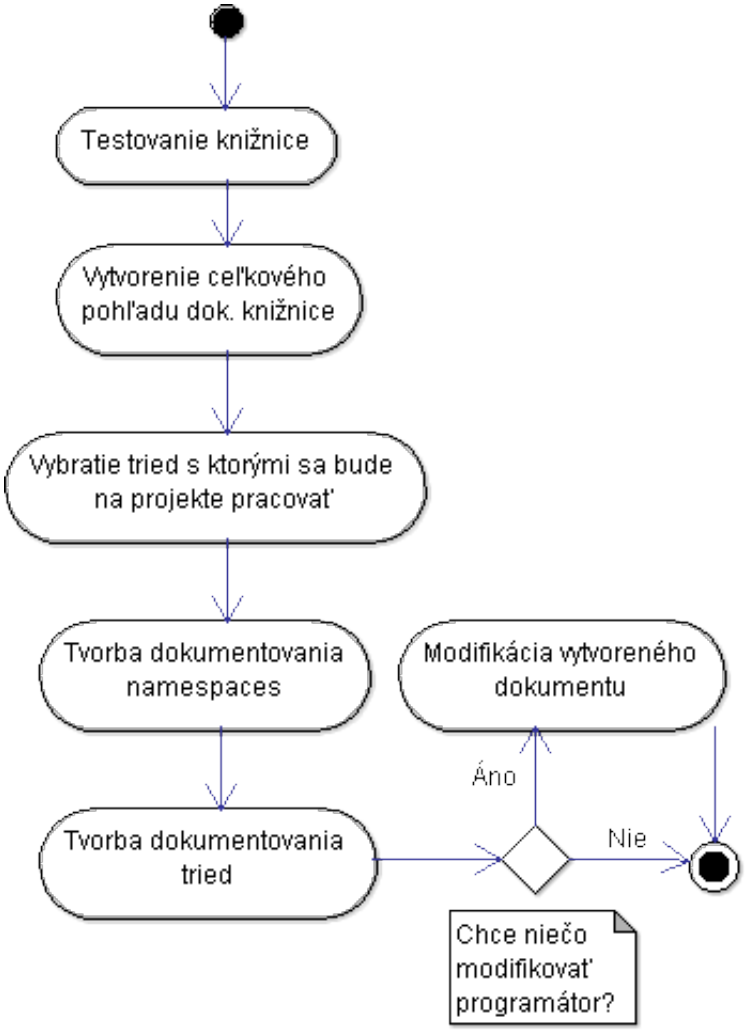
Táto metodika do malej miery nadväzuje na metodiku dokumentácie zdrojového kódu, keďže v tých procesoch vystupuje ako nástroj Doxygen, a všeobecné konvencie dokumentovania pomocou tohto nástroja sú dodržané aj v tejto metodike.

5.8.1 Roly

Rola	Procesy
Manažér popdory vývoja	Vytvorenie celkového pohľadu dokumentácie knižnice
Analytik	Tvorba dokumentovania *namespaces* 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*

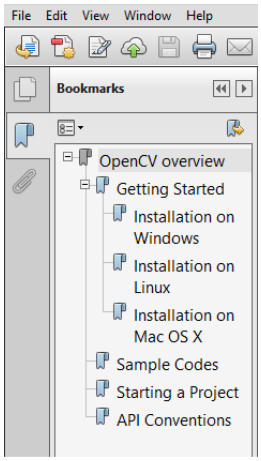
5.8.2.1 Vytvorenie celkového pohľadu dokumentácie knižnice

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 dokumentáciach.



Obr. 6: *Ukážka návestí v Readeri*

5.8.2.2 Tvorba dokumentovania namespaces

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

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 ukážať 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 klúčové 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 7, 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 štrukturovania 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 podporným 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 spätne 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 týmu 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.

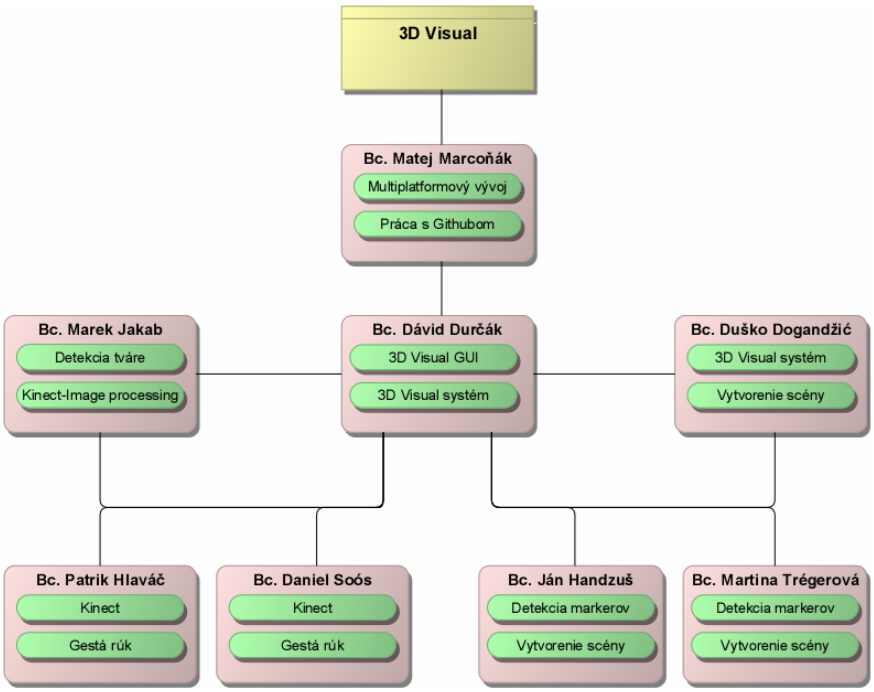
8 Manažment vývoja

Vzhľadom na fakt, že cieľom tímového projektu je doplniť funkcionalitu 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 používateľa 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 ošetrenie 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 vlastnostní 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/ý* a *vysoká/ý*.

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 prepojiteľnosť 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: Zodpovedný:	15.10.2013 Dávid Durčák
Pravdepodobnosť: nízka	Dopad:	stredný	Frekvencia:	nízka
Následky	Tím sa musí zaoberať novými riešeniami situácie, čo spomalí celkové napredovanie a zmení výsledné riešenie.			
Prevencia	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: Zodpovedný:	21.10.2013 Martina Trégerová
Pravdepodobnosť: nízka	Dopad:	stredný	Frekvencia:	nízka
Následky	Možná strata kľúčových odborníkov, nemožnosť dokončiť projekt v danom rozsahu.			
Prevencia	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.			
Prevencia	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:	vysoký	Frekvencia:	stredná
Následky	Nesplnenie požadovanej funkcionality			
Prevencia	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 používateľa 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.			
Prevencia	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 perifémymi 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.			
Prevencia	Dbanie na bezpečnosť a pozornosť pri práci. Zamedzenie manuálnemu poškodeniu.			
Náprava	Získanie nového zariadenia.			

9.7 Zlá prepojiteľnosť funkcionality s pridaným ovládaním

Názov rizika	Zlá prepojiteľnosť funkcionality s pridaným ovládaním			
Opis	Zlý prístup ku sledovaniu používateľa a neskoré (prípadne žiadne) reagovanie na sledované správanie.		Identifikované: Zodpovedný:	11.11.2013 Daniel Soós
Pravdepodobnosť: nízka	Dopad:	stredný	Frekvencia:	nízka
Následky	Nefunkčnosť ovládania aplikácie, nutné hľadanie nových riešení.			
Prevencia	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 prejaviť v spomalenom reagovaní alebo nefunkčnosti.		Identifikované: Zodpovedný:	11.11.2013 Ján Handzuš
Pravdepodobnosť: stredná	Dopad:	nízky	Frekvencia:	stredná
Následky	Pomalé testovanie, nižšia použiteľnosť.			
Prevencia	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 cely semester. Preto táto časť bude priebežne dopĺňaná počas semestra.

Každé dva tyždne 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 súčastne. Za dokončenie takejto úlohy v šprinte zodpovedá každý, komu bola takáto úloha pridelená. Pokiaľ je to možne, 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 zostavý zoznam cieľov na další šprint.

4. Ak je to možne, 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 medzisemestrálnom 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 sprehľadnenie 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 dlhodobejšie 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. Pritom 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 repozitári 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 dokumentáciam,
- 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 Zápisnice 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. Zápisnice zároveň obsahujú stručný obsah týchto stretnutí.

A.1 Zápis 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, nadstavba 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 vyzerať výstup projektu a bližšie sme sa oboznámili so základnou problematikou.

A.2 Zápis 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 vyzerať 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 Zápis 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 funkcionalitu 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 funkcionalitu 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 vyzerať 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 záležať 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 Zápis 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 inštalaácii 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 Zápis 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 tvobra 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 verzie
    - 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 pospájal 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 analýzujúci 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 prehĺbiť 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 Zápis 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 zložitosť 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 kartičky a ohodnocovanie používateľských príbehov (prípadne z toho vyvodených 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 ostávajúcich 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 retrospetívu a naplánovali sme si tretí šprint už aj s rozdelenými úlohami.

A.7 Zápis 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 další týždeň v rámci šprintu ideálny na vyriešenie úlohy
    - Martina a Ján o prieskumnom prototypovaní osgART a ARToolkit: druhý menovaný je už zastaralý a má menšiu podporu, s rozbehaním 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 funkcionalita 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 funkcionalitu 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 rozpoznaním tváre: je to vhodný postup; taktiež zdôraznil, že v zásade pre každý ďalší používateľský príbeh pôjdu vlastné namespacy – Marek dodal, že on sám vo svojom riešení využil dva namespacy
  - 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 vyskúsať 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 treťom 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 šprintu 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 najsľubnejšie 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
  - striktnejšia verzia kompilátora so zmenenými parametrami vo Visual Studio nefungovalo, takže takéto takáto podoba testovania kódu bude možné skôr len v Linux prostredií
  - dokumentácia ešte finišuje v deň stretnutia, kedže 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 opciu 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	Impoert 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 zložitosť úloh a priradili ich. Niektoré úlohy boli ešte v teň 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 Zápis 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 častejší 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 opciu 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	Impoert 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 Zápis 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 Marcoňák
	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 opciu 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 funkcionalita 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	Impoert 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 Zápis 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 dopĺňať,
    - tím s vedúcim sa dohodol, že prvoradé 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 Návod na inštaláciu

V tejto časti sú opísané návody na inštaláciu a spustenie programu 3DVisual, ktorý bol pôvodne vytvorený v rámci predchádzajúcich projektov na tejto fakulte, a náš tím postupne pridával na funkcionalite tohto softvéru.

Návod na Linux

Návod bol otestovaný na Ubuntu 12.04. V niektorých verziách Linuxu je nainštalovaná verzia Qt 5, ktorá môže spôsobovať problémy, keďže sa v projekte používali staršie verzie Qt.

Najprv je potrebné stiahnuť aplikáciu z https://github.com/marconak/Arvis/ a rozbaliť to napr. do priečinka QtWorkspace. Ako prvý krok je treba mať nainštalované Qt, Cmake, OpenSceneGraph a ďalšie knižnice.

Ukážka 3: *Inštalácia potrebných softvérov a knižníc*

```
sudo apt-get install synaptic
sudo apt-get update

sudo apt-get install git
sudo apt-get install make
sudo apt-get install cmake cmake-gui # cmake-qt-gui

sudo apt-get install qt4-dev-tools libqt4-dev libqt4-core libqt4»
  » -gui
sudo apt-get install qt-sdk qtcreator

sudo apt-get install openscenegraph libopenscenegraph-dev
sudo apt-get install libpng12-dev  # libpng12-0
```

V prípade, ak máme v domovskom adresári QtWorkspace a je tam aplikácia rozbalená, presunieme sa tam, a pokračujeme spôsobom ako to je v ukážke 4.

Ukážka 4: *Spustenie Cmake*

```
mkdir \_build
cd \_build
cmake ..
make -j4
```

```
make install
cd ../_install
```

Ostáva spustiť QtCreator, tam otvoriť súbor cmakelist.txt. V argumentoch pre cmake zadať *-DCMAKE_BUILD_TYPE=Debug* alebo *-DCMAKE_BUILD_TYPE-=Release*. V ľavom menu, kde sa nachádza *Projects*, treba rozbaliť *build steps*, pridať na základe jadier nášho procesoru do *additional details* napr. *-j4*. Potom *Add Build Step, Custom Steps*. Pridať ako príkaz */usr/bin/make*. Do argumentov *install* a zapnúť *Custom Processes*.

Ďalej sa treba prepnúť v ľavom menu na *Projects* → *Run Settings* → *Run*, tam zmeniť working directory o adresár vyššie a nastaviť na _install. Následne treba kladivom dať build a potom *Do Project*.

Návod na Mac OS X

Tento návod bol testovaný na OSX 10.8.5. Aj po správnej inštalácii sa môžu vyskytnúť problémi s načítavaním PNG formátov. Nasledujúce softvéry a knižnice sú potrebné pre inštaláciu:

- **CMake 2.8.11 (a vyššia) dmg súbor,**
- **QT libraries 4.8.5 pre Mac a debug libraries,**
- **OpenSceneGraph 3.0.1 binárne súbory.**

Následne ich inštalácia - Qt treba cez *.mpkg* súbory, ktoré sa nachádzajú v dmg súboroch. CMake cez *.pkg* súbor, ktorý sa taktiež nachádza v dmg.

V ukážke 5. je opísaný priebeh inštalácie libpng.

Ukážka 5: *Inštalácia libpng na OS X*

```
$ cd /usr/local/src #Ak src neexistuje vytvor ho cez sudo mkdir »
  » src na uvedenej ceste.
$ sudo curl --location --output libpng-1.6.6.tar.gz http://»
  » download.sourceforge.net/libpng/libpng-1.6.6.tar.gz
$ sudo tar -xzvf libpng-1.6.6.tar.gz
$ cd libpng-1.6.6
$ sudo ./configure --prefix=/usr/local/libpng-1.6.6
$ sudo make -j4 # Za parametrom -j das pocet vlakien. Pocet »
  » zalezi od poctu jadier na procesore.
$ sudo make install
$ sudo ln -s libpng-1.6.6 /usr/local/libpng
```

Ak nebude stačit nainštalovať libpng, treba si upraviť vrchný návod a nainštalovať *libpng-devel*.

V ukážke 6. je opísaná inštalácia OpenSceneGraphu.

Ukážka 6: *Inštalácia OpenSceneGraph na OS X*

```
$ cd ~ # Chod do domovskeho priecinku.
$ nano .bash_profile # Otvor/vytvor a edituj subor ".»
  » bash_profile".
# Pripis nasledujuci obsah (OSG = OpenSceneGraph -3.0.1 »
  » priecinok v root). Cesty zmen podla potreby:
export DYLD\_LIBRARY\_PATH="/OSG/lib"
export PATH="$PATH:/OSG/bin:/usr/local/libpng/bin:/Applications/»
  » CMake 2.8-11.app/Contents/bin"
export CMAKE\_INCLUDE\_PATH="/OSG/include:/usr/local/libpng/»
  » include"
export CMAKE\_LIBRARY\_PATH="/OSG/lib:/usr/local/libpng/lib"
#Uloz obsah a zatvor ("ctrl + x" potom "y").
$ chmod +x .bash_profile #Zmen mu prava, aby bol spustitelny.
$ nano .bashrc # Otvor/vytvor a edituj subor.
# Pripis nasledujuci obsah:
source ~/.bash_profile
#Uloz obsah a zatvor ("ctrl + x" potom  "y").
$ chmod +x .bashrc #Zmen mu prava, aby bol spustitelny.
# Otvor nove okno terminalu ("cmd + n"). Tym sa spustia »
  » vytvorene subory.
```

Ukážka 7. opsiuje inštaláciu a spustenie 3DVisual.

Ukážka 7: *Inštalácia OpenSceneGraph na OS X*

```
$ cd /3Dsoftviz-master
$ mkdir _build
$ mkdir _install
$ cd _build
$ cmake ..
$ make -j4
$ make install
$ cd ..
$ cd install
$ ./3DVisual
```

Návod na Windows

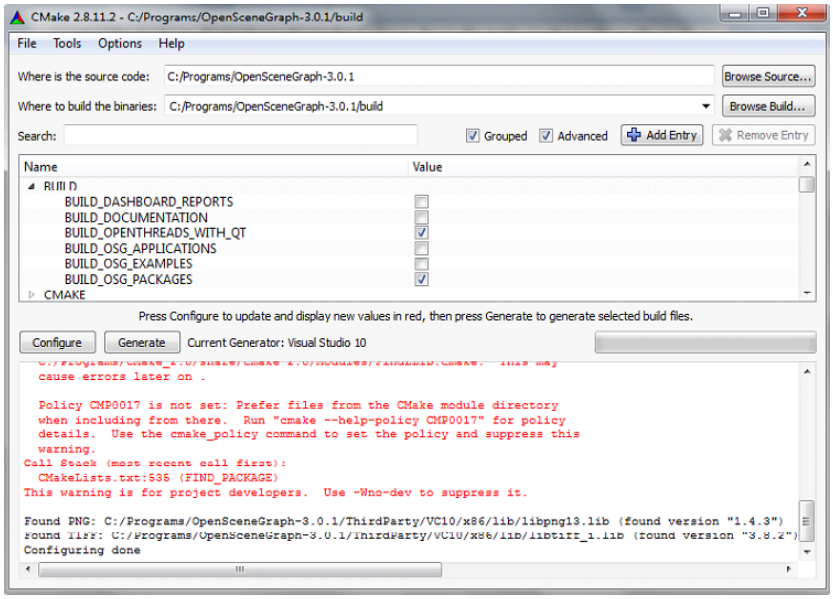
Tento návod bol úspešne otestovaný na Windows 7. Na inštaláciu potrebujeme 3DSoftviz, CMake 2.8.11, OpenSceneGraph (source, ktoré budú buildované vo Visual Studio 2010). Je dôležité mať nainštalované Microsoft Visual Studio 2010 SP1. Ďalej treba mať Qt 4.8.5 a keďže sa jedná o staršiu verziu, treba k tomu doinštalovať aj QtCreator. Pre pohodlnejšiu prácu s premennými prostrediami je tiež vhodné mať nainštalovaný program RapidEE. Taktiež je nutné mať pre inštaláciu knižnice. 3rd party dependencies.[1] Postup na inštaláciu by mal vyzerať nasledovne:

1. Nainštalovať CMake. Cesta ku CMake ďalej v texte ako %CMAKE_DIR%.
2. Nainštalovať Qt a QtCreator. Cesta do Qt dalej ako %QT_DIR%.
3. Stiahnuť OSG - cestu do OSG ďalej v texte ako %OSG_DIR%.
4. Stiahnuť 3rd party knižnice, ideálne vytvoriť priečinok %OSG_DIR%\ThirdParty\.
5. Otvoríme si RapidEE pre zmenu enviroment variables.
   (a) do PATH pridáme nasledovné premenné:
      - %CMAKE_DIR%\bin
      - %QT_DIR%\4.8.5\bin
      - %OSG_DIR%\ThirdParty\VC10\x86\bin

   (b) vytvoríme variable CMAKE_INCLUDE_PATH a pridáme:
      - %OSG_DIR%\include
      - %OSG_DIR%\ThirdParty\VC10\x86\include

   (c) vytvoríme variable CMAKE_LIBRARY_PATH a pridáme:
      - %OSG_DIR%\ThirdParty\VC10\x86\lib.
6. Ak ste stiahli OSG binary files preskočte na krok 10. Spustíme Cmake-gui. Ako source priečinok si nastavíme %OSG_DIR% a build cestu %OSG_DIR%\build. Spustíme configure a vyberieme Visual Studio 2010.

[1] Zdroj: http://ulozto.sk/xaZJArtg/vc10-zip, dostupné z webu v novembri 2013.



Obr. 9: *Ukážka CMake*

7. Configure by vám mal prebehnúť v poriadku a nájsť aj PNG a iné knižnice, čo potrebuje. Odporúčam zaškrtnúť grouped a advanced a v BUILD záložke zaškrtnúť podľa obrázku, nech nebuildujete zbytočne example aplikácie a podobne.

8. Stlačíme znova configure, ak nie sú žiadne problémy následne stlačíme generate.

9. V priečinku %OSG_DIR%\build spustíme .sln pre visual a dáme build project (vytvoria sa knižnice a .dll). Môžeme vytvoriť pre Debug aj pre Release (cca. 10 minút).

10. Znova pridáme enviroment variables.

    (a) do PATH pridáme:

    - %OSG_DIR%\build\bin,

    (b) do CMAKE_LIBRARY_PATH pridáme:

    - %OSG_DIR%\build\lib.

11. V priečinku %3Dsoftviz% vytvoríme _build a _install.

12. Spustíme QtCreator. File → Open file or project. Vyberieme CMakeList z %3Dsoftviz%.

13. Ako build directory zvolíme %3Dsoftviz%_build

14. Arguments: -DCMAKE_BUILD_TYPE-
    =Debug alebo -DCMAKE_BUILD_TYPE=Release a Generator NMake pre verziu 4.8.5. V prípade, že nenašlo generator postupujeme nasledovne:

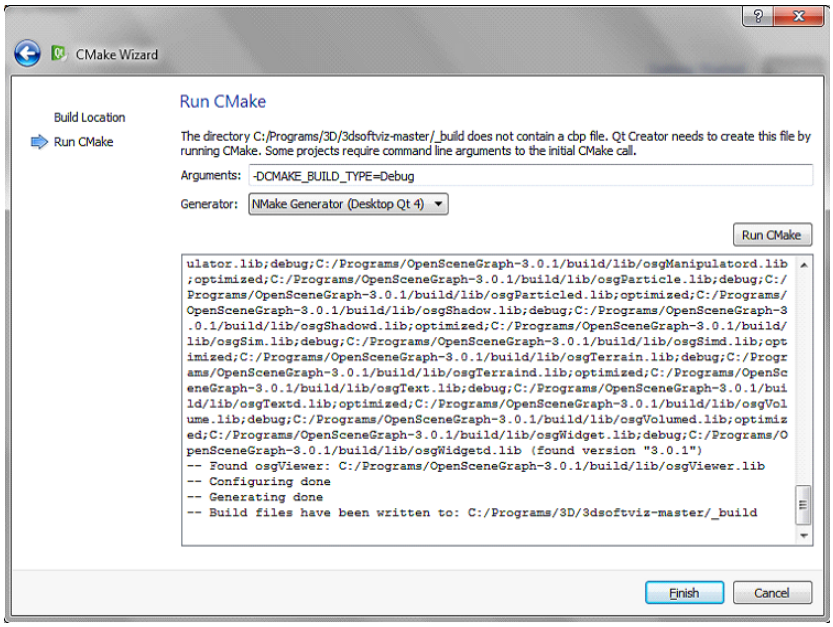
    (a) Tools → Options → CMake a ako cestu nastavíme %CMAKE_DIR%-\bin\cmake.exe,

    (b) V Kits by sme mali mať nastavené Qt verziu 4.8.5 a compiler VS 2010,

    (c) Vo Qt versions ak nenašlo cestu do %QT_DIR%\4.8.5\bin\qmake.exe , treba ju nastaviť,

    (d) teraz by vám už malo nájsť generator.

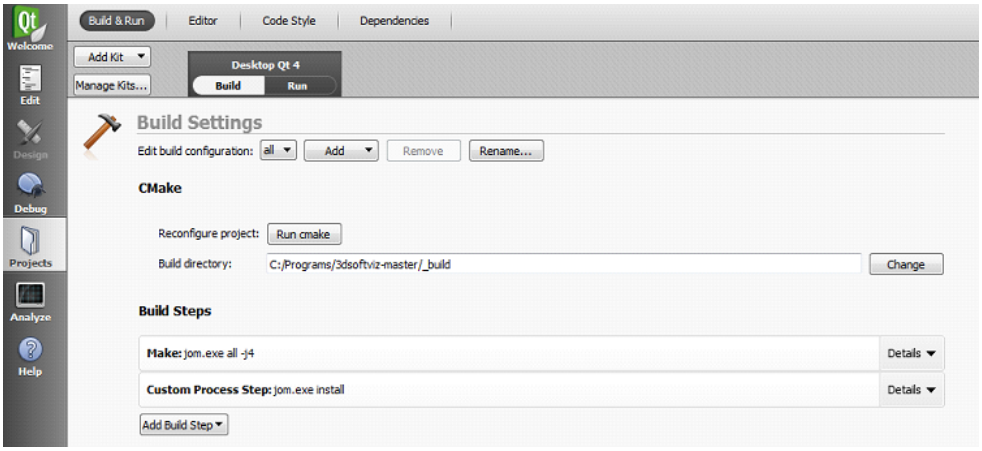
15. Spustime Run Cmake.



Obr. 10: *Spustenie CMake*

16. Po dokončení v ľavej lište vyberieme Projects → build & run → build a v Build steps nastavíme ku Make: *jom.exe* additional arguments *-j4*.

17. Pridáme k tomu ešte Add build step → Custom process step. command: *jom.exe* a argument *install*.



Obr. 11: *Spustenie CMake*

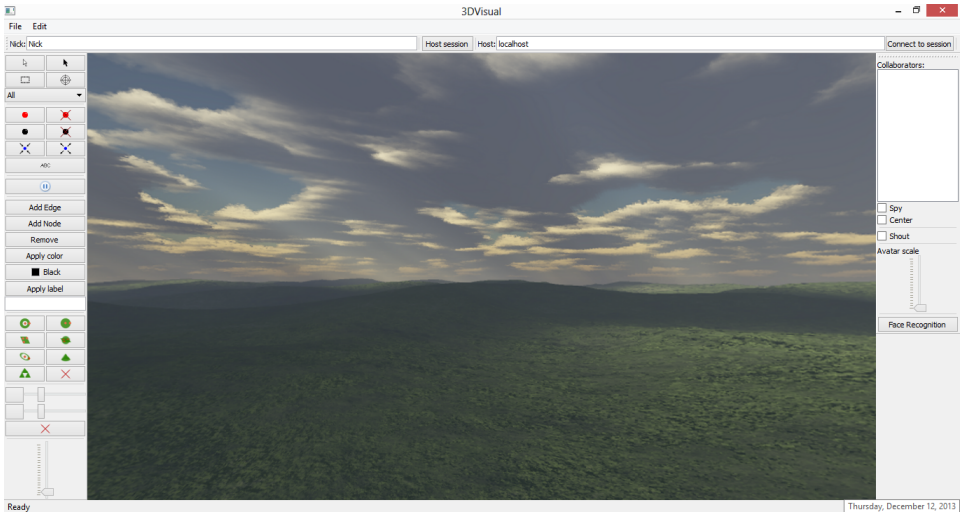
18. Následne celý program zbuildujeme (kladivko vľavo dole).

19. Mali by ste mať uz program v _install a mal by sa dať spustiť. Môžeme proces buildovania zopakovať aj pre Release/Debug

POZNÁMKA: okrem spomenutých knižníc v návode je potrebné cez git konzolu po klonovaní repozitára spustiť príkaz *submodule*, aby pribudol *libnoise*. Taktiež do projektu pribudla knižnica ArUco (potrebné inštalovať), ten má problémy na Windowse s OpenGL. Riešením je zakázať build v ArUco GL príkladoch, čo je možné nastaviť v $dependencies/aruco/utils$ v *CmakeLists* projektu. OpenNI2 a NiTe2 knižnice tiež treba mať pre prácu s Kinectom (funkcionalita zatiaľ riešená len na Windowse), a príslušné cesty treba pridať do $CMAKE\_INCLUDE\_PATH$, resp. $CMAKE\_LIBRARY\_PATH$. OpenCV má vlastný *.cmake* súbor vo svojom *build* adresári, stačí si túto cestu nastaviť ako systémovú premennú menom $OpenCV\_DIR$.

C Používateľská príručka

Aplikácia 3DVisual sa spúšťa cez súbor *3DVisual.exe*. K spusteniu aplikácie je potrebné mať nainštalované prostredia OpenSceneGraph a Qt. Na obrázku 12 vidíme používateľské rozhranie aplikácie 3DVisual.



Obr. 12: *Používateľské rozhranie 3DVisual*

V ľavom hornom rohu sa nachádza hlavné menu aplikácie, v ktorom sa dá načítať súbor (preddefinovaný je súbor s príponou *graphml*) z adresára alebo z databázy. Takisto sa dá uložiť graf a v sekcie *Edit* je možné zmeniť nastavenia programu.

Na ľavej strane je taktiež aj nástrojová lišta. Opciami *All*, *Node* a *Edge* je možné vybrať si, či chceme selektovať hrany spolu s uzlami, alebo len uzly, resp. hrany. Predvolene je program nastavený na *No-select mode* (biela šípka), čiernou šípkou sa môžeme prepnúť na stav *Single-select mode*, čo nám umožňuje sústredenie sa na práve jeden objekt - môže to byť hrana aj uzol. Tretia možnosť v menu je *Multi-select mode*, čím sa dá vybrať v trojrozmernom zobrazení viacero objektov naraz. Štvrtá opcia v tejto rubrike zobrazí centrálny pohľad na práve vybranú časť trojdimenzionálneho grafu. Pokiaľ sme nastavení na *No-select mode*, vieme hýbať vybraným uzlom.

Pod týmto menu sú tlačidlá na pridávanie meta-uzlov a na zmazanie meta-uzlov, pričom, pochopiteľne, zmazať sa dajú len tie popridávané meta-uzly. Existuje aj možnosť zafixovania a odfixovania vybraných uzlov. Možnosťou *Merge nodes together* sa naskytuje možnosť zlúčiť vybrané uzly do jedného spoločného uzla. Takýto uzol sa bude v pokračovaní zobrazovať s modrou farbou. Týmto spôsobom je ľahké zrušiť zlúčenie týchto uzlov možnosťou *Separate merged nodes*.

Ak hýbeme uzlami, často sa formuje aj celý graf - túto akciu môžeme vypnúť tlačidlom *Play*.

Opcia *Add Edge* pridáva hranu medzi dvoma vybranými uzlami, kde ešte nie je hrana, inak končí akcia chybovou hláškou. Ideálne je čiernou šípkou vybrať jeden uzol a bielou šípkou ho nastaviť na také miesto, kde sa ho dá spojiť s druhým uzlom - je potrebné mať nastavenie *Node* v takomto prípade spolu s *Multi-select mode*. *Remove* slúži na zmazanie uzlov alebo hrán. Ak sa rozhodneme pre zmazanie hrany, uzly prepojené s touto hranou na grafe ostávajú.

Aplikovanie rôznych farieb sa uskutočňuje spôsobom, že najprv sa spraví operácia, ktorú chceme znázorniť farebne (napr. výber jednej podčasti grafu) a až potom sa na to aplikuje farba. Textové pole sa používa tým istým spôsobom ako sfarbenie grafu. Zapnúť tieto texty treba tlačidlom, ktoré sa nachádza nad možnosťou *Play*. Ďalšie štyri možnosti tvoria aplikácie priestorových ohraničení (povrch gule, obsah gule a rovina), použité a vybrané ohraničenia je možné aj odstrániť.

Pod hlavným menu sa nachádzajú nastavenia na pripojenie. Zadať je potrebné používateľské meno a IP adresu, medzitým sa nachádza tlačidlo na spustenie, resp. zastavenie servera a posledným tlačidlom je pripojenie na kolaboráciu, resp. odpojenie od nej. Kolaboračný panel je umiestnený na pravej strane programu, tam sa nachádza okno zoznamu používateľov. *Spy* a *Center* sú funkcie, ktoré môže nadobudnúť používatelia. Ak sa označí funkcia *Shout*, tým sa znázorní pokus o upútanie pozornosti ostatných členov. Velkosť avatarov, ktoré majú používatelia zobrazené na scéne sa dajú tiež zväčšiť, resp. zmenšiť.

Pod touto opciou sa nachádza funkcionalita pridaná vďaka knižnici OpenCV. Po spustení tlačidla *Face Recognition* sa otvorí nové okno, v ktorom sa vykoná detekcia tváre z kamery.

Ak je potreba zväčšiť aplikačnú časť programu, je možné kliknúť na ľubovolné miesto hlavného panelu pravým klávesom, a vypnúť niečo z trojice *Tools* (nástroje), *Collaboration* (kolaborácie) alebo *Network* (pripojenie).

D Elektronické médium

/Arvis - vytváraný program s ďalšími podadresármi

/doc

- dok_riadenia - dokumentácia bakalárskej práce
- dok_ing_diela - dokumentácie inžinierskeho diela

- readme.txt

E Preberací protokol

Preberací protokol

Tímový projekt 2013/2014

Tím č. 5 - ARVis

Predmet odovzdávania:
Dokumentácia riadenia - verzia po prvých piatich šprintoch
Dokumentácia k inžinierskemu dielu - verzia po prývch piatich šprintoch

---

Vedúci tímového projektu: Ing. Peter Kapec, PhD.
Podpisom potvrdzuje prevzatie vyššie uvedených častí dokumentácie.

V Bratislave

....
Dátum