Slovenská technická univerzita v Bratislave

FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Tímový projekt

REKONŠTRUKCIA 3D SCÉNY V2

Inžinierske dielo

Vedúci projektu:

Ing. Vanda Benešová, PhD.

Členovia tímu:

Bc. Jakub Ginter (SI)
Bc. Miroslav Haščič (SI)
Bc. Mário Hunka (SI)
Bc. Viktor Košťan (IS)
Bc. Richard Pintér (IS)

Názov tímu: 3DRecon (tím č. 11)
Web: http://team11-16.studenti.fiit.stuba.sk/
Kontakt: teamfiit11@gmail.com
Akademický rok: 2016/2017
Dátum odovzdania: 13. 12. 2016

OBSAH

1. ÚVOD

Tento dokument vznikol ako technická dokumentácia k projektu 3DRecon, ktorý sme dostali ako zadanie na tímový projekt na FIIT STU. Rekonštrukcia 3D scény je v oblasti výskumu veľmi aktuálnou témou, pretože ma pomerne veľké využitie. Aktuálne existuje veľké množstvo komerčných riešení, ktoré vedia spracovávať dáta zo senzorov skenujúcich 3d scénu. Ich hlavným problémom je, že dokážu spracovávať dáta iba z určitých senzorov (značky).

My chceme vytvoriť systém, ktorý bude spracovávať dáta v rôznom formáte bez závislosti na senzore čím umožníme rozšíriť využitie medzi širokú verejnosť. Snažíme sa umožniť naskenované dáta jednoducho spracovať a konvertovať ich do uhladeného formátu kde budú ďalej ručne spracovávane v programoch ako AutoCAD a jemu podobné. To výrazne zjednoduší prácu architektov a interiérových návrhárov ale odvetvia kde sa skenovanie objektov a priestorov tu zďaleka nekončia.

Tento dokument ponúka detailný pohľad na náš produkt a na proces jeho vývoja. Nachádzajú sa v ňom naše čiastočné i globálne ciele ale aj architektúra systému a opis fungovania.

2. GLOBÁLNE CIELE PRE ZS

Projekt sme dostali v rozpracovanej forme preto bolo prvotným cieľom celého tímu oboznámiť sa so systémom, s jeho architektúrou, triedami a celkovým spôsobom fungovania. Súčasne s týmto oboznamovaním sme museli študovať princípy segmentácie 3D scény a používané metódy a algoritmy.

Ďalšie definované ciele na semester boli:

- Refaktoring projektu
    - Zmena štruktúry kódu
    - Odstránenie nepotrebných tried a častí kódu
    - Návrh novej architektúry
    - Tvorba nového GUI
- Analýza možných spôsobov selekcie outliers
    - Analýza dostupných metód
    - Konzultácia v rámci tímu
    - Implementácia vybraných metód

- Získanie nových datasetov
    - Získanie senzoru od spolupracujúcej spoločnosti
    - Naskenovanie nových priestorov
    - Rekonštrukcia týchto dát
    - Testovanie na týchto dátach
- Získanie používaných datasetov
    - Nájsť na internete nové datasety, ktoré sme ešte nepoužili
- Ovládanie vizualizácie prostredníctvom 3D myši
    - Interakcia s vizualizérom pohybmi 3D myšou
- Presnejšie výsledky používaných algoritmov
- Ukladanie medzivýsledkov
    - Uloženie medzivýsledkov pre zrýchlenie výpočtov

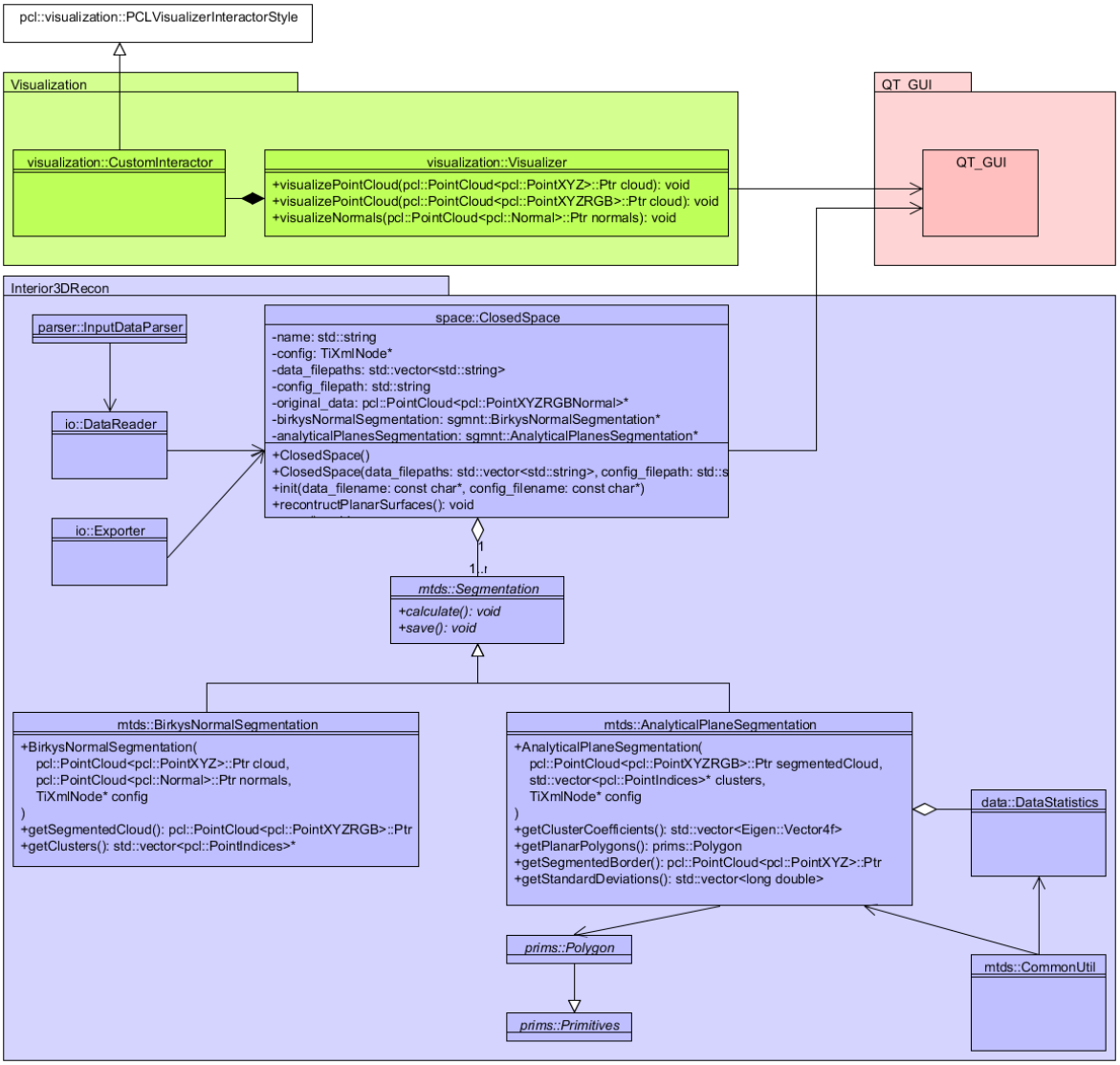
Ďalšie úlohy sa špecifikujú postupom času na základe výstupov a výsledok predošlých úloh. Výsledky sa pravidelne konzultujú s vedúcou projektu, ktorá nám radí akým ďalším smerom sa máme pohybovať. Nakoľko tento projekt má výskumný charakter nie je možné dopredu definovať veľké množstvo konkrétnych cieľov, pretože smerovanie sa vždy odvíja od predošlých poznatkov.

3. CELKOVÝ POHĽAD NA SYSTÉM

V tomto projekte pokračujeme po minuloročnej skupine, ktorá na projekte začínala. Preto sme museli ako prvú vec analyzovať aktuálnu formu projektu. Celý náš tím (5 členov) musel prejsť projekt a pochopiť presné fungovanie jednotlivých tried. Nakoľko bola forma pomerne rozhádzaná a projekt obsahoval množstvo nepotrebného kódu a nepoužívaných funkcií išlo o zdĺhavý proces.

Po tejto analýze sme sa rozhodli, že pristúpime k celkovému refaktoringu tohto projektu a zmeníme jeho štruktúru. Bol navrhnutý nový architektonický model systému založený na predošlom systéme avšak s väčším ohľadom na zameniteľnosť jednotlivých častí a s oddelením používateľského rozhrania od funkcionality. V starom projekte sa časť funkcionality nachádzala aj v projekte používateľského rozhrania čo nám nepripadalo správne v rámci zaužívaných pravidiel a nechceli sme takto postavený projekt ďalej nabaľovať o ďalšiu funkcionalitu.

Nová architektúra je postavená na rovnakých knižniciach ako tá predošlá. Jediný rozdiel je v organizácií projektu. Analýzu možných alternatívnych riešení sme nerobili, pretože sme to nepovažovali za rozumné a časovo efektívne. Aj keby sme našli lepšie riešenia na problematiku knižníc v obore 3D rekonštrukcie bolo pri by časovo nerentabilné prerábať celý projekt. Preto sme sa zamerali na použitie PCL teda PointCloudLibrary, ktorá ponúka rozsiahlu funkcionalitu v tejto oblasti.



Ako zo zadania vyplýva museli sme dbať najmä na dátovo-formátovú nezávislosť aby bola aplikácia skutočne použiteľná na všeobecnej úrovni. S tým je zviazaná aj dátovo-typová nezávislosť. Zlepšili sme zameniteľnosť a flexibilitu projektu. Z dôvodu náročných výpočtov sme výrazne dbali aj na zníženie pamäťovej náročnosti tejto aplikácie.

3.1 INTERIOR3DRECON

Interior3DRecon predstavuje hlavnú funkcionalitu nášho projektu. Obsahuje triedy na segmentáciu, import a export 3D dát.

Jej aplikačné rozhranie predstavuje trieda *CosedSpace*, ktorá v sebe zoskupuje načítané pôvodne 3D dáta, konfiguračné parametre a metódy na výpočet/prístup ku segmentovaným dátam. Je navrhnutá tak, aby jej inštancia obsahovala segmentované dáta pre práve jednu kombináciu pôvodných dát a konfiguračných parametrov. Je to preto, aby sa zabezpečila lepšia viditeľnosť a umožňuje to jednoduchšie a prehľadnejšie ukladanie dát inštancie.

Jednotlivú segmentáciu 3D dát vykonávajú podtriedy triedy *Segmentation*. Podobne ako trieda ClosedSpace sú navrhnuté tak, aby si po inicializácii a vykonaní *calculate* metódy uchovali vypočítané dáta a v prípade, že bude potrebné segmentovať iné dáta alebo použiť inú konfiguráciu vytvorí sa nová inštancia. Umožní to uchovať si výsledky tej istej segmentačnej metódy s rôznymi konfiguráciami, čo sa môže hodiť v prípade, že sú potrebné tieto rozličné výsledky na ďalšie spracovanie.

V súčasnom projekte sú implementované dve triedy, ktoré dedia od triedy Segmentation - *BirkysNormalSegmentation* a *AnalyticalPlanesSegmentation*. *BirkysNormalSegmentation* segmentuje point cloud na základe odhadnutých normál bodov. Tieto segmentované dáta berie na vstupe *AnalyticalPlanesSegmentation,* ktorá ďalej segmentuje 3D scénu na planárne polygóny pomocou fitovania rovín cez clustre bodov zo vstupu.

Triedy *CommonUtil* a *DataStatistics* obsahujú podporné metódy pre podtriedy triedy *Segmentation*.

Metódy a funkcie na import a export dát sú obsiahnuté v triedach *Exporter* a *DataReader* pričom *DataReader* používa triedu *InputDataParser*, ktorá parsuje ľubovoľné ASCII dáta do point cloudu v XYZ koordinačnom systéme.

3.2 QT_GUI

Pomocou tohto modulu je spúšťaný celý projekt. Zvyšné moduly sú kompilované do knižníc, ktoré *QT_GUI* používa. Jeho jediná funkcionalita je pritom používateľské okno, ktoré umožňuje volať funkcionalitu modulov *Interior3DRecon* a *Visualization*.

Jeho jediná rovnomenná trieda *QT_GUI* obsahuje premennú typu *ClosedSpace,* ktorá poskytuje aplikačné rozhranie modulu *Interior3DRecon* a premennú typu *Visualizer*, ktorá poskytuje aplikačné rozhranie modulu *Visualization*. Po vybratí súborov s 3D dátami a konfiguračného súboru sa zavolá inicializačná metóda triedy *ClosedSpace* a sprístupnia sa tak tlačidlá, ktoré volajú jej zvyšné metódy a metódy *Vizualizéru.*

Na vytvorenie používateľského rozhrania sme použili populárny softvér *QT*. Umožnil nám pomerne rýchlo vytvoriť grafické používateľské rozhranie, a taktiež jeho podpora cross-platformového vývoja sa môže do budúcna hodiť, v prípade že sa bude implementovať rekonštrukcia dát z MS Kinect.

3.3 VISUALIZATION

Vizualizáciu sme dokázali kompletne oddeliť od používateľského rozhrania a pomocné výpočtové hodnoty posielame z GUI ako parametre, takže v GUI je iba volanie funkcie. Vizualizér pracuje s inštanciou ClosedSpace, ktorá je hlavnou inštanciou celého programu a všetky operácie sa robia nad ňou, resp. všetky vizualizované dáta získavame z nej.
Vizualizácia prebieha s využitím metód knižnice PCL. Metódy vyberáme na základe pointcloudu, ktorý chceme vyberať, teda podľa toho aké ma vlastnosti. Jeho vlastnosti sa menia postupným spracovávaním prostredníctvom implementovaných metód. Spôsob vizualizácie si vyberá používateľ sám a na základe jeho voľby sa zavolá konkrétny konštruktor, pomocou ktorého sa vytvorí okno vizualizéru a naplní sa prostredníctvom zvolených metód knižnice.
Máme vytvorený vlastný spôsob interakcie ktorý sme prebrali od minuloročného tímu ale je potrebné prepracovať ho. Pri stlačení pohybových kláves w,s,a,d je obzvlášť pri pohybe hore a dole skok príliž

veľký a preto musíme túto funkcionalitu upraviť. Rotácia v priestore sa rotuje okolo zle vybraného bodu, čím sa môže stať, že sa popri rotovaní otáča obraz úplne nezmyselne. Interakcia je príliš neintuitívna a nepraktická. Na jej zmene sa aktuálne pracuje.

Ďalšou úpravou interakcie a celkovej vizualizácie je pridanie 3d Myši. Na tomto riešení sa aktuálne pracuje a analyzujú sa dostupné možnosti. Toto riešenia sa uberá dvoma smermi a vybraný bude ten, ktorý bude fungovať lepšie. Jednou alternatívou je použiť na celú vizualizáciu iba okno PCL vizualizéru a doplniť ho interakciou 3d Myši. Druhou možnosťou je použiť vizualizér, ktorý ponúka 3d myš, avšak zatiaľ sa nám nepodarilo úspešne doňho vložiť náš pointcloud.
Pri ďalšom vývoji vizualizéru bude pridaná možnosť pozrieť si pôvodný point cloud a súčasne aj výsledok jeho spracovania zvolenou metódou. Preskakovať medzi jednotlivými vizualizáciami bude možné stačením vybraných kláves. Táto funkcionalita sa však zatiaľ vyvíjať nezačala.

4. MODULY SYSTÉMU

Jednotlivé moduly systému sme previazali po predošlom tíme. Tie sú podrobne zdokumentované na ich prezentačnej stránke[1]. Medzi tie časti, ktoré sme si osvojili a zahrnuli do novej štruktúry systému, ktorý má vzísť z refaktoringu partia:

- **BirkysNormalSegmenation** - segmentácia point cloud do clusterov na základe vypočítaných normál bodov.
- **AnalyticatPlaneReconstruction** - po novom *AnalyticalPlaneSegmentation* keďže sa jedná o segmentáciu a nie rekonštrukciu je modul, ktorý má po refaktoringu segmentovať 3D scénu na planárne polygóny. Tieto polygóny sa neskôr použijú na segmentáciu stien miestností a menej členitých objektov ako napríklad stôl alebo dvere.
- **CustomInteractor** - trieda na ovládanie PCL vizualizačného okna. Bude upravená aby poskytovala intuitívnejšie ovládanie.
- **Načítavanie konfiguračných údajov z XML** - v novej verzii projektu je taktiež použitý opensource TinyXML na načítavanie konfiguračných údajov
- **Export segmentovaných dát** - trieda *Exporter*, ktorá bude doplnená o metódy na export dát, ktoré sú potrebné na to aby bolo možné uložiť inštanciu triedy *ClosedSpace*.

Novú funkcionalitu zatiaľ náš systém neponúka. Vzhľadom na globálne ciele (uvedené vyššie) sme identifikovali 4 problémy, ktoré budú predmetom riešenia nášho tímu:

1. Metódy pre rozpoznávanie a segmentáciu stien
2. Úplná segmentácia outlayerov
3. Rekonštrukcia dát z kinectu
4. Zlepšenie vizualizácie

[1] Prezentačná stránka tímu *RED*

link: http://labss2.fiit.stuba.sk/TeamProject/2015/team05is-si/#final-product

4.1 ANALÝZA METÓD PRE ROZPOZNÁVANIE A SEGMENTÁCIU STIEN

Hoci súčasný systém dokáže segmentovať polygóny, resp. steny z point-cloudu, kvalita výsledkov segmentácie a jej výpočtová náročnosť nie je postačujúca na ďalšie spracovanie. Preto bola súčasná implementácia analyzovaná vzhľadom na iné práce, ktoré sa venovali rozponávaniu a segmentácii stien z point-cloudu.

4.1.1 METÓDA ROZPOZNÁVANIA STIEN

Rozlišujeme 3 skupiny metód na rozpoznávanie rovín v point-coulde:

- Region growing techniky
- Metódy založené na RANSAC-u
- Metódy založené na transformácií priestoru príznakov (napr. Houghova transfornácia)

Súčasná implementácia používa PCL implementáciu RANSAC-u. Práca [1] navrhuje použitie metódu rozpoznávania stien pomocou Houghovej transformácie, ktorá dosahuje stabilnejšie výsledky pri rozpoznávaní stien zo sekvencie 3d dát v čase pričom si zachováva porovnateľnú výpočtovú náročnosť. Keďže ale náš projekt rieši rozpoznávanie zo statických 3d dát je táto výhoda nepodstatná.

RANSAC má nevýhodu, že bez optimalizácie je výpočtovo náročný. Nazvdory tomu bolo ukázané, že dokáže sa vysporiadať s datasetom obsahujúcim 50% outlayerov. Práca [3] rieši rozpoznávanie tvarov z point-cloudu použitím optimalizovaného RANSAC algoritmu pomocou hodnotiacej funkcie. V jednotlivých iteráciách tohto algoritmu sa vyberie tvar s maximálnym ohodnotením zo skupiny kandidátov, zistí sa pravdepodobnosť, že nebol prehliadnutý lepší kandidát a v prípade, že je táto pravdepodobnosť dostatočne vysoká vybraný kandidát je vymazaný zo skupiny kandidátov. Algoritmus končí v prípade, ak pravdepodobnosť, že v skupine kandidátov nie je lepší kandidát ako používateľom definovaný minimálny tvar je dostatočne veľká.

Súčasná implementácia používa na fitting rovín RANSAC nad segmentmi point-cloudu, ktoré boli segmentované pomocou dominantných normál point-cloudu a následného použitia Region growing algoritmu. Je teda primárne zameraná na rozpoznávanie planárnych objektov, pričom si vyžaduje pomerne výpočtovo náročný preprocessing. Preto stojí na zváženie použitie RANSAC algoritmu spolu s optimalizáciou, podobne ako to bolo v prípade práce [3].

4.1.2 VALIDÁCIA ROZPOZNANÝCH ROVÍN

Po fittingu stien pomocou RANSAC-u sa rozpoznané roviny evaulujú, či segment skutočne predstavuje planárny objekt, resp. povrch na základe pomeru 2:1 inlayers a outlayers. V prípade, že outlayer bodov je viac ako polovica inlayer bodov je táto stena (rovina) zamietnutá a ďalej sa s ňou neráta.
V tejto implementácii vidieť nedostatok, pretože sa takto môže zamietnuť segment, ktorý obsahuje stenu alebo planárny povrch s iným priľahlým objektom. Preto sa navrhuje implementovať odstránenie potencionálneho priľahlého objetku napríklad odstránením bodov segmentu, ktoré sa nachádzajú za určitou vzdialenosťou od rozpoznanej roviny.

4.1.3 REFITTING STIEN

Pre práce, ktoré rozpoznávajú steny pomocou RANSAC-u je obvyklé, že po rozpoznaní steny nasleduje vylepšenie (refining) steny pomocou algoritmu najmenších štvorcov. Optimalizuje sa tak chyba určenia roviny v segmente.
Toto vylepšenie určenia roviny steny chýba v súčasnej implementácii. Je preto na zváženie jeho použitie, nakoľko by pravdepodobne zlepšil výsledky segmentácie outlayer bodov, hoci by to znamenalo spomalenie segmentácie.

4.1.4 ZALAMOVANIE STIEN V SÚČASNEJ IMPLEMENTÁCIÍ

Na základe analýzy segmentačného algoritmu predošlého tímu sa zistila príčina zalamovania polygónov. Pri zjemňovaní okrajových bodov konkávnej obálky sa tieto body transformujú na najbližšie body segmentu, ktoré ale neležia na jednej rovine a preto nie je výsledný polygón planárny. Je potrebné ešte transformovať tieto body pomocou kolmých priemetov na analyticky určenú rovinu segmentu.

4.1.5 ODKAZY

[1] HULIK, Rostislav, et al. Continuous plane detection in point-cloud data based on 3D Hough Transform. Journal of visual communication and image representation, 2014, 25.1: 86-97.
[2] ROTHG., LEVINE M. D.: Extracting geometric primitives.CVGIP: Image Underst.58 , 1 (1993), 1–22.
[3] SCHNABEL, Ruwen; WAHL, Roland; KLEIN, Reinhard. Efficient RANSAC for point-cloud shape detection. In: Computer graphics forum. Blackwell Publishing Ltd, 2007. p. 214-226.

4.2 ÚPLNÁ SEGMENTÁCIA OUTLAYEROV

4.2.1 ANALÝZA

Analýza požiadaviek prebieha počas spoločných stretnutí s product ownerom. Na základe diskusií sme identifikovali problém s tzv. *outlayer-mi* - výčnelkami (kľučka, svetlo apod.). Problém úzko súvisí so segmentáciou polygónov, na ktorú je využitá metóda opísaná vyššie (***AnalyticalPlaneReconstruction***). Odstránením tohto problému je možné značne vylepšiť presnosť segmentácie polygónov, ktoré sú kľúčové pri identifikácií stien či objektov.

Konkrétne, segmentácia polygónov prebieha v týchto krokoch. Z troch bodov sa definuje rovina, táto rovina sa preloží cez vytvorené segmenty. Pri evaluácií sa zistí koľko percent zo segmentovaných clusterov sa pretína s definovanou rovinou. Samozrejme, je potrebná určitá odchýlka, ktorá určuje kedy bod patrí a kedy nepatrí do roviny. Odchýlka je určená vzhľadom na šum, ktorý vstupné dáta

obsahujú. Náš problém spočíva teda v tom, ako oddeliť šum od reálnych objektov. Napríklad, sme si vedomý toho aký nepresný je náš hardware. Avšak, objekty, ktoré sú relatívne malé vzhľadom na snímanú miestnosť - napr. kľučka - sa nám javia ako šum - čo ale nie je pravda. V prípade, že by sme tieto objekty vedeli identifikovať a nejakým spôsobom ich zo scény odstrániť (vysegmentovať) vieme zaručiť lepšiu identifikáciu polygónov.

Analyzovali sme viacero spôsobom, ktorými je možné lokalizovať a následne segmentovať outlayeri. Jedným znich je ABOD (Angle-Based outlayer detection). Pri tejto metóde zisťujeme kde sa bod nachádza vzhľadom na ostatné body. Ak sa "pozerá" na väčšinu ostatných bodov v jednom smere - je outlayer. Ak existuje viacero smerov, v ktorých vidí body - outlayer nie je. Pri tejto metóde je hlavná nevýhoda jej zložitosť a to O(n^3). Ďalším spôsob je využitie neurónovej siete. Avšak tu nastáva problém s označkovaným datasetom. ABOD by bolo možné optimalizovať tým, že nám používateľ označí miesta, kde sú potenciálne outlayeri. Tým pádom, by sme nemuseli prehľadávať celý point cloud, ale iba jeho časti. Avšak, naše riešenie má byť plne automatizované, takže táto možnosť nepripadá do úvahy. Nakoniec sme sa rozhodli pre otestovanie metódy, ktorá je opísaná nižšie v Návrhu.

4.2.2 NÁVRH A IMPLEMENTÁCIA

Navrhli sme metódu, ktorá je založená na projekcií z 3D do 2D. Prebieha v týchto krokoch:

1. StatisticalOutlierRemoval - túto funkcionalitu ponúka PCL. Dokáže zabezpečiť redukovanie šumu z 3D dát.
2. Projekcia do 2D - spravíme prierez segmentu, čím sa premietne do 2D.
3. Best line fit - cez body sa snažíme preložiť priamku, ktorá pretne čo najviac bodov
4. Lokalizácia outlayers - body, ktoré ležia v kolmej vzdialenosti +- treshold, sú outlayeri
5. Flag - nakoniec označíme tieto body nejakou značkou, ktorá bude signalizovať outlayer - tieto body nebudú potom brané do úvahy pri AnalyticalPlanesSegmentation

4.2.3 TESTOVANIE

Evaluácia výsledkov riešenia tohto problému bude prebiehať nasledovným spôsobom. Je potrebné vždy poznať aj výsledky starej metódy, preto je potrebné si ich niekam uložiť. Pri tej istej konfigurácií, identifikujeme polygóny bez toho aby sme odstránili outlayer-i a uložíme si výsledky o úspešnosti (koľko percent bodov zo segmentovaných častí patrilo do roviny). Následne pripojíme našu metódu segmentácie outlayerov a znovu spustíme proces identifikácie polygónov. Výsledky vieme teda ľahko pomocou tejto metriky porovnať. Ak dosiahneme značne lepšie výsledky metóda bude integrovaná do projektu.

Je potrebné pôvodný výstup kontrolovať aj ručne - pomocou vizualizéru, a teda porovnať výstup pôvodných metód v podobe point cloud s výstupom nových metód segmetnácie a identifikácie outlier-ov.

4.3 REKONŠTRUKCIA DÁT Z KINECTU

4.3.1 ANALÝZA

Získavanie testovacích dát - voľne dostupné datasety na internete, datasety od firmy *Photoneo s.r.o.* - obmedzuje vývojoví tím pri testovaní nových výskumných metódach. Ak by sme chceli dataset šitý na mieru, je časovo náročné (možno aj nemožné) takýto dataset zabezpečiť. Vzhľadom nato, sme na spoločných stretnutiach prišli k záveru, že je vhodné implementovať vlastnú rekonštrukciu dát.

4.3.2 NÁVRH

Preskúmali sme viacero možností ako pri skenovaní miestnosti minimalizovať odchýlky, ktoré môžu spôsobiť zlé mapovanie bodov v priestore. Všetky preskúmané možnosti využívali určitú variantu **SLAM algoritmu (simultaneous localization and mapping).** Algoritmus je rozdelený do dvoch častí: mapovanie a lokalizácia. mapovanie predstavuje proces nadväzovania a odstavovania chýb zo skenovaných dát. Lokalizačná časť je zase zameraná na výpočet polohy a správne rozpoznávanie natočenia senzoru, polohy v priestore atď. Problémom môže byť náročnosť algoritmu na výpočtový výkon a špeciálne prípady pri SLAM algoritme ako je opätovné navštívenie oblastí. Po presúmaní možností sme sa rozhodli pre otestovanie nasledujúceho algoritmu:

1. získanie dát zo senzoru
2. rozpoznanie pohybu
3. vytvorenie point cloud so natočením a posunom vzhľadom na pohyb senzoru
4. redukcia šumu
5. výber blendovacej funkcie
6. spojenie dát do globálneho Point cloud

.

4.3.3 IMPLEMENTÁCIA

Implementácia bude prebiehať v rámci 4. šprintu.

4.3.4 TESTOVANIE

Testovanie riešenia tohto problému bude prebiehať manuálne s použitím vizualizéru. V tomto prípade je to najlepšie riešenie, ak vieme čo sme nasnímali, vieme si to zobraziť a odhadnúť podobnosť s realitou.

Pri jednotlivých metódach rekonštrukcie je vhodné ich neskôr vyhodnocovať aj kvantitatívne a navzájom porovnávať presnosť s akou boli dáta namerané.

4.4 ZLEPŠENIE VIZUALIZÁCIE

4.4.1 ANALÝZA

V rámci nášho projektu je vizualizácia podstatná časť. Správnosť metód je v mnohých prípadoch overovaná vizuálne - dá sa pohľadom povedať či má zmysel sa metóde venovať alebo nie. Pri všetkých metódach je vizuálna kontrola prvým stupňom testovania a v niektorých prípadoch aj jediný. Práve preto je potrebné mať funkčný a intuitívne používateľný vizualizér.
Ďalším dôvodom pre zlepšenie vizualizéru je naša tímová prezentácia a celková prezentácia produktu. Pri prezentácií je práve vizualizér náš jediný výstup, ktorý vníma pozorovateľ. Ten nemusí mať s problematikou rekonštruovania 3d scény a v podstate ani s grafikou či informatikou ako takou žiadne skúsenosti. A práve vizualizér je jediná vec, prostredníctvom ktorej môže tento pozorovateľ a teoreticky budúci „zákazník“ s naším projektom prísť do kontaktu. A práve dobre vypracovaný vizualizér je preto dôležitou časťou nášho projektu a našej prezentácie.

4.4.2 NÁVRH

V našej práci sa zameriavame hlavne na implementáciu intuitívneho pohybu pomocou myši. Ďalšou navrhnutou metódou pohybu je ovládanie pomocou 3D myši. Pre správnu a hlavne intuitívnu implementáciu pohybu v priestore je nutné navrhnúť interakciu s vizualizérom tak aby bola priateľská pre používateľa a ľahko naučiteľná. Samotné dáta, ktoré vizualizujeme sú pripravené na prezentáciu implementovanými metódami segmentácie.
Nový vizualizér, ktorý je založený na vstavanom vizualizéri PCL knižnice, spočíva prevažne v implementácii vlastného ovládania (správne reakcie na pohyb myšou, správne interpretované klávesové vstupy do pohybov kamery vo vizualizéri) pre vizualizér a vyvinutie metód na spracovania vstupov z 3D myši. Okrem tohto hlavného problému vizualizéru sa tiež venujeme správnemu vkladaniu dát do priestoru v ktorom sa kamera pohybuje. Štartovacia pozícia kamery a jej orientácie v priestore je tiež veľmi dôležitá pre zorientovanie používateľa.
Pre používateľa je veľmi dôležité aby videl ako sa pomocou segmentačných metód mení používaný Point Cloud. Keď sa bude môcť prepínať medzi pôvodným a výsledným Point Cloudom dokáže identifikovať problémy metódy a uvidí čo zvolená metóda vlastne urobila.

4.4.3 IMPLEMENTÁCIA

Implementácia vizualizéru je riešená prostredníctvom funkcií z PCL knižnice. Jedna funkcia vytvorí okno s parametrami ako farba pozadia, koordináty a inicializačný bod pre kameru. Druhá funkcia inicializuje interakciu pre toto okno. Interakcia je sprostredkovaná našou triedou kde máme

definovaný spôsob interakcie. Celá táto funkcionalita je obsiahnutá v jednej metóde, ktorá sa zavolá pri kliknutí na tlačidlo vizualizovať v GUI. Metóda je volaná s parametrom typu PointCloud, ktorý je získaný z inštancie ClosedSpace. Ten sa v metóde vloží do vytvoreného okna kde je možné s nim pracovať kým nebude okno zatvorené alebo zastavená vizualizácia.
Podľa predtým aplikovaných metód nad Point Cloudom sa pridá vizualizačná možnosť pre upravený Point Cloud. Keď je táto metóda vybraná vytvorí sa konštruktor vizualizéru a naplní sa dátami potrebnými pre vizualizovanie týchto metód. Aktuálne vieme vizualizovať všetky výsledky našich metód, vysegmentované plochy a ich hrany alebo ofarbený Point Cloud, ktorý je výstupom Birkis metódy.
Aktuálne je rozpracovaná implementácia 3d myši prostredníctvom poskytovaného SDK od spoločnosti 3Dconexxion. Spôsob implementácie a jej podstata je opísaná v časti Celkový pohľad na systém. Rovnako sa pracuje na úpravách interakčných metód v už implementovanom vizualizéri.

4.4.4 TESTOVANIE

Overenie správnosti riešenia pre vizualizér sa bude testovať hlavne pomocou UX experimentu, do ktorého zapojíme členov tímu, ale aj neskúseného používateľa, ktorý nám môže priniesť užitočné návrhy na spríjemnenie používania vizualizéru.
Prvá fáza testovania prebehla tak, že vizualizér testovali všetci členovia tímu, pričom sme identifikovali niekoľko problémov, ktoré už boli odstránené. Ďalšie testovanie bude prebiehať až po implementácií nových súčastí vizualizéru.

Slovenská technická univerzita v Bratislave

FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Tímový projekt

REKONŠTRUKCIA 3D SCÉNY V2

Riadenie

Vedúci projektu:

Ing. Vanda Benešová, PhD.

Členovia tímu:

Bc. Jakub Ginter (SI)
Bc. Miroslav Haščič (SI)
Bc. Mário Hunka (SI)
Bc. Viktor Košťan (IS)
Bc. Richard Pintér (IS)

Názov tímu: 3DRecon (tím č. 11)
Web: http://team11-16.studenti.fiit.stuba.sk/
Kontakt: teamfiit11@gmail.com
Akademický rok: 2016/2017
Dátum odovzdania: 13. 12. 2016

Obsah

1 ÚVOD

Tento dokument obsahuje postupy a metódy, ktoré boli definované naším tímom za účelom riadenia tímového projektu. Projekt rieši segmentáciu 3D scény z rekonštruovaných 3D dát. Hlavnou myšlienkou je umožniť segmentáciu scény z dát v ľubovoľnom formáte pričom výstup by mal byť použiteľný softvérom *AutoCAD*[1].

Štafetu sme prebrali po minuloročnom tíme, ktorý viedol Bc. Lukáš Hudec. Vzhľadom nato, sme pokračovali v používaní niektorých metodík (metodika písania zdrojového kódu), avšak definovali sme si aj vlastné, ktoré sú špecifické pre náš tím.

Projekt je výskumného charakteru. Je teda potrebné umožniť testovanie rôznych segmentačných metód, ktoré po evaluácií môžu skončiť ako prototyp na zahodenie. Samotná práca s 3D dátami je výpočtovo, ale aj pamäťovo náročná. V súvislosti stým, sme sa spočiatku začali uberať smerom, ktorý by urýchlil prototypovanie (ukladanie medzivýsledkov, ktoré neovplyvňujú výsledky testovania novej metódy). Ako ďalší z našich cieľov, je úplná segmentácia tzv. *outlayerov* (kľučka, svetlo atď.), ktoré vytvárajú nepresnosti pri segmentácií objektov, pri ktorých sa nachádzajú, prípadne sú ich súčasťou (stena, stôl atď.). V neposlednom rade, rozširujeme aj možnosti formátov vstupných dát. Ako ďalšie máme v pláne rekonštruovať 3D dáta nasnímané pomocou zariadenia *Kinect* [2]z rôznych uhľov pohľadov.

Výsledkom našej práce bude knižnica, ktorá poskytuje rôzne metódy segmentácie 3D dát využiteľná hlavne vo výskumnej sfére. V rámci vývoja tejto knižnice je samozrejme dostupné aj používateľské rozhranie, ktoré umožňuje používateľovi ľahko a efektívne, testovať nové segmentačné metódy.

1.1 PREHĽAD DOKUMENTU

V časti 2 sú opísané manažérske činnosti a zodpovednosti členov nášho tímu. V podkapitolách môžeme nájsť aj podiel práce členov tímu na tomto dokumente. V 3 kapitole sme uviedli aplikácie manažmentov, ktoré

[1] Nástroj určený na dizajn http://www.autodesk.com/products/autocad/overview

[2] Hardware Senzor od Microsoftu http://www.xbox.com/en-US/xbox-one/accessories/kinect

používame pri riadení nášho tímu. V kapitole 4 sme sumarizovali výsledky našich 3 šprintov. V rámci vývoja nášho softvéru používame určité metodiky, ktoré sú uvedené v časti 5. V ďalších častiach tohto dokumentu sa nachádzajú exporty z nástroja používaného na manažovanie (TFS), globálna retrospektíva a zápisnice zo stretnutí.

2 ROLE ČLENOV TÍMU A PODIEL PRÁCE

2.1 MANAŽÉRSKE ČINNOSTI

TABUĽKA 1: ROZDELENIE ÚLOH V TÍME

Meno	Rola	Zodpovednosť
Jakub Ginter	vedúci tímu, biznis manažér	analýza a návrh požiadaviek, komunikácia s produkt ownerom
Miroslav Haščič	manažér kvality	vykonávanie testov, riešenie a správa chýb (bug-ov)
Viktor Košťan	manažér vývoja a architektúry	návrh a implementácia architektúry, správa hlavných segmentáčných metód
Mário Hunka	integrátor, manažér propagácie	prepájanie jednotlivých modulov projektu, tvorba a udržiavanie web stránky
Richard Pintér	manažér dokumentácie	udržiavanie dokumentácie a záznamov zo stretnutí

2.2 PODIEL PRÁCE

TABUĽKA 2: PODIEL PRÁCE NA DOKUMENTÁCIÍ RIADENIA

Časť	Člen tímu
Úvod	Mário Hunka
Role členov tímu a podiel práce	Mário Hunka
Manažment komunikácie	Jakub Ginter
Manažment vývoja a integrácie	Jakub Ginter
Manažment plánovania	Mário Hunka

Manažment riadenia	Miroslav Haščič
Manažment dokumentovania	Richard Pintér
Sumarizácia šprintov	Richard Pintér
Metodika zadávania úloh do TFS	Viktor Košťan
Metodika prehliadok kódu	Mário Hunka
Metodika verziovania kódu	Jakub Ginter
Metodika testovania	Miroslav Haščič
Metodika dokumentácie	Richard Pintér
Globálna retrospektíva	Mário Hunka
Export úloh z TFS	Jakub Ginter
Integrácia a finalizácia Kontrolný bod 1	Mário Hunka
Integrácia a finalizácia Kontrolný bod 2	Richard Pintér

TABUĽKA 3: PODIEL PRÁCE NA DOKUMENTÁCIÍ K INŽINIERSKEMU DIELU

Časť	**Člen tímu**
Úvod	Jakub Ginter
Globálne ciele pre ZS	Jakub Ginter
Celkový pohľad na systém	Viktor Košťan
Moduly systému – úvod	Viktor Košťan
Segmentácia Outlayerov	Mário Hunka
Rekonštrukcia dát z Kinectu	Mário Hunka
Zlepšenie vizualizéra	Miroslav Haščič
Analýza metód na rozpoznávanie a segmentáciu stien	Viktor Košťan
Integrácia a finalizácia	Richard Pintér

3 APLIKÁCIE MANAŽMENTOV

3.1 MANAŽMENT KOMUNIKÁCIE

Naša komunikácia v tíme by sa dala rozdeliť na dva hlavné celky. Prvým sú tímové stretnutia a druhým komunikácia v nástrojoch

3.1.1 TÍMOVÉ STRETNUTIA

Spoločne sme sa stretali podľa rozvrhu každý štvrtok o 13:00 v laboratóriu VGG 4.46. Na týchto stretnutiach sme spolu s vedúcou tímu zhodnocovali čo sa spravilo a čo sa bude robiť. Každý člen vysvetlil na čom pracoval a kde boli problémy, resp. aké by mohli v budúcnosti nastať. Tieto stretnutia prebiehali podľa agilnej metódy SCRUM, ktorá je opísaná v manažmente plánovania. Tam je možné nájsť ako presne tieto stretnutia prebiehali.

Mimo týchto stretnutí sme sa stretávali aj v iných dňoch a riešili problémy, na ktoré sme narazili a bolo potrebné ich vyriešiť čo najskôr alebo jednoducho pracovali. Tieto stretnutia prebiehali zväčša vo VGG alebo v knižnici.

3.1.2 KOMUNIKAČNÉ NÁSTROJE

Na našu komunikáciu sme najčastejšie používali dva nástroje (Slack a Google Drive). V prvotných začiatkoch prác na projekte sme komunikovali prostredníctvom skupinového Facebook chatu. To sa však ukázalo ako neefektívne bolo potrebné nájsť vhodný nástroj.

Slack je aplikácia určená na komunikáciu v tímoch a zvýšenie ich produktivity. Je možné vytvoriť komunikačné kanály čo umožňuje rozdeliť komunikáciu do tematických celkov. V rámci kanálov je možné nastavovať notifikácie pre jednotlivých členov alebo skupinu. To má veľké využitie obzvlášť pri mobilnej aplikácií, ktorá je tiež k dispozícií. Používateľ tak môže nastaviť upozornenia iba na správy kde bol označený a vie, že je potrebné reagovať čo najskôr.

Pomocou tejto aplikácie je možné nahrávať súbory, zdieľať ukážky zdrojového kódu alebo upozorňovať pri commite vetvy.

Google Drive sme používali ako tímové úložisko, kde sme uchovávali dokumentáciu, záznamy zo stretnutí a rôzne iné dokumenty potrebné pre tím. Rovnako sme tu mali uložené datasety pre náš projekt.

Menej často sme používali Skype alebo Gmail, na ktorom sme mali založený tímový alias odkiaľ sa pošta preposielala každému členovi.

3.2 MANAŽMENT VÝVOJA A INTEGRÁCIE

V našom projekte je úloha manažéra vývoja a manažéra integrácie rozdelená medzi dvoch členov. Ich úlohy pozostávajú z manažmentu verzií, integrácie súčastí do jedného veľkého celku ale aj konfigurácia systému a iné. Manažér

integrácie je priamo zodpovedný za repozitáre v TFS a udržuje vetvy projektu. Rieši konflikty a pridáva novú funkcionalitu do hlavnej vetvy. Viac o manažovaní projektových vetiev a postup ako ich v projekte 3Drecon udržujeme sa nachádza v dokumente Metodika verziovania.

Aby sme dokázali udržiavať prehľadný kód a projekt používame TFS (Team Foundation Server) kde medzi sebou zdieľame vetvy projektu, manažujeme úlohy aj zapisujeme bugy. Tento nástroj je priamo integrovaný do Visual Studia, ktoré používame na vývoj a preto je pre nás TFS najlepšou možnou alternatívou. Nástroj ponúka jednoduchú synchronizáciu a aktualizovanie zmien, ktoré priamo prenáša medzi našim úložiskom aj s prípadnými správami.

Každý z členov tímu je zodpovedný za časť systému, ktorú vyvíja teda prípadné nedostatky a chyby v týchto častiach musí primárne vyriešiť jej tvorca. Tvorca nie je zodpovedný iba za funkcionálnu stránku ale aj za výstup a úpravu, teda za celkové spracovanie (správne okomentovaný a zadokumentovaný kód). Následná integrácia do projektu je vykonaná integrátorom. Tvorca s ním túto skutočnosť odkomunikuje podľa pravidiel manažmentu komunikácie a poskytne integrátorovi potrebnú súčinnosť pri práci.

3.3 MANAŽMENT DOKUMENTÁCIE

Písanie dokumentácie v našom tíme prebieha priebežne. Dokumentácia je vytváraná spôsobom spoločného dokumentu, typu GOOGLE DOC, ktorý máme uložený na spoločnom uložisku GOOGLE DRIVE. Nakoľko dokumentácia sa skladá z viacerých častí, na GOOGLE DRIVE sú postupne pridávané dokumenty, ktoré si následne prejdú viacerí členovia tímu a po spoločnej konzultácií sa dohodnú, či je dokument vyhovujúci alebo je nutné ho prerobiť. V prípade, že je dokument vyhovujúci je následne zaintegrovaný do finálnej dokumentácie. Aby sme predišli rôznym kolíziám typu odlišné riadkovanie alebo font písma, dohodli sme sa dodržiavať pravidlá písania dokumentácie, ktoré sú spísané v dokumente menom Metodika písania dokumentácie.

3.4 MANAŽMENT PLÁNOVANIA

Predpokladom úspešného plánovania je naplnený tzv. product backlog aspoň na 2 šprinty vopred. Product backlog sa vytváral na základe požiadaviek

nášho zadávateľa, ktorý je v tomto prípade zároveň náš vedúci tímu. Vzhľadom na požiadavky, boli hierarchicky vytvorené časti softvéru (Epics, Features), pre ktoré sa vytvárali backlog itemy. Tie sa radia vždy podľa priority – dôležitosť s akou treba daný backlog item spraviť, aby sme naplnili požiadavky čo najefektívnejším spôsobom.

Proces plánovania prebieha na začiatku každého šprintu. Podľa priority sú ku šprintu priradené backlog itemy, na ktorých sa pracuje počas najbližšieho šprintu. Backlog item má určité kritéria akceptácie, ktoré musia byť dodržané, aby bola úloha splnená. Na naplnenie týchto kritérií je väčšinou potrebné viacero úloh (taskov).

Následne sa ohodnotia jednotlivé úlohy vzhľadom na prvú ohodnotenú úlohu, ktorá bola vopred určená (base). Na každom stretnutí sa hrá tzv. *Scrum Poker*, pri ktorom sa odhaduje náročnosť úloh pomocou hlasovania, čo sprevádza diskusia o úlohách o ktorých náročnosti sa hlasuje. Touto aktivitou je zabezpečený čo najlepší odhad k jednotlivým úlohám. Vzhľadom na minulé šprinty prispôsobujeme náročnosť ďalšieho šprintu, preto je veľmi dôležité zabezpečiť čo najlepší odhad.

Následne sú zadané úlohy do manažovacieho nástroja TFS, ktoré sú buď priradené jednotlivým členom tímu, alebo si ich členovia tímu vyberajú počas šprintu. Progres v jednotlivých úlohách sa taktiež sleduje v nástroji TFS.

3.5 MANAŽMENT RIADENIA

Riadenie v našom tíme bolo založené na agilnej metóde vývoja SCRUM. V časti manažment plánovania sa môžete dozvedieť ako sme pomocou tejto metódy plánovali postup v našom projekte a ako sme určovali úlohy. Tieto úlohy sme potom evidovali v nástroji TFS. V každom šprinte sme vytvorili definované backlog items a k nim patriace úlohy. Každý člen mal definované svoje úlohy a ich hodnotu (ako sa táto hodnota získava nájdete v časti manažment plánovania). Niektoré úlohy nemali pridelenú zodpovednú osobu nakoľko nebolo jasné kto to bude robiť a nebolo ani potrebné to definovať. Dohodnuté však bolo že sa to musí urobiť a ten kto mal najviac času takúto úlohu pripísal v TFS sebe a urobil ju.

Pre úlohy existujú tri základné stavy: To do, In progress, Done. Logicky je cieľom, každú z definovaných úloh dopracovať do štádia done. Ak by sa však niektoré úlohy v šprinte nestihli urobiť, presúvajú sa do ďalšieho šprintu kde majú najvyššiu prioritu.

Dohľad nad vykonávaním úloh a nad ich rozdeľovaním má vedúci tímu. Ten je zodpovedný za správne rozdelenie úloh, prípadne riešenie problémov v tejto oblasti.

4 SUMARIZÁCIA ŠPRINTOV

4.1 ŠPRINT 1

V prvom šprinte sme sa primárne snažili urobiť všetko podľa odporúčaní odborníkov. Narazili sme však na niekoľko problémov, ktoré sa odzrkadlili aj na jeho kvalite. No práve vďaka týmto chybám teraz vieme omnoho lepšie ako na šprintoch pracovať. Veľkým problémom bolo, že sme sa museli naučiť fungovať ako tím a dokázať tak zvýšiť efektivitu našej práce. Druhým problémom bolo, že pokračujeme v projekte po minuloročnom tíme a na začiatku sme mali minimálne skúsenosti a znalosti v oblasti, ktorej sa projekt venuje. Preto bolo potrebné dlhé analyzovanie, pretože bez neho nebolo možné ďalej implementovať novú funkcionalitu. Fakt, že celý projekt je výskumného charakteru túto situáciu sťažoval ešte o trochu viac. So SCRUM-om sme v tíme skúsenosti nemali a s jeho použitím na výskumnom projekte už vôbec nie.

Začali sme používať nové nástroje ako SLACK na komunikáciu a TFS na evidenciu úloh. Nejaký čas nám trvalo aj naučiť sa pracovať s týmito nástrojmi, hlavne s TFS.

V prvom šprinte sme definovali 4 backlog items.

1. Zrýchlenie výpočtov ukladaním medzivýsledkov
2. (bug) Slabá viditeľnosť procesov
3. Analýza celkového kódu
4. (bug) Zlyhanie programu pri načítaní konfiguračného súboru (.xml)

4.2 ŠPRINT 2

Pri retrospektíve prvého šprintu sme sa všetci členovia tímu jednohlasne zhodli, že našou najväčšou slabinou bolo TFS a jeho správne používanie, čo sme sa rozhodli do ďalšieho šprintu zmeniť. V druhom šprinte sme sa po dohode s vedúcou projektu a s členom minuloročného tímu Lukášom Hudecom rozhodli pustiť do reštruktulizácie celého programu. Vedeli sme, že tento krok nás spomalí v napredovaní rozširovania funkcionalít ale na druhej strane sme vedeli, že ak si to navrhneme a prerobíme podľa seba, bude to mat pozitívny účinok do budúcnosti. Najprv sme vytvorili návrh class diagramu nového projektu, ktorý sme následne diskutovali s vedúcou cez Skype hovor. Na konci hovoru, po odsúhlasení štruktúry projektu sme si rozdelili úlohy a postupne sme na nich začali pracovať. Počas šprintu sme narazili na zopár problémov pri rozbehávaní projektu na notebookoch, ktoré sa nám podarilo našťastie vyriešiť. Na konci šprintu sme zistili, že sme v stanovenom čase

nedokázali spraviť všetku prácu, ktorú sme si naplánovali a ,že v tom budeme musiet pokračovať aj v treťom šprinte.

4.3 ŠPRINT 3

Pri retrospektíve druhého šprintu sme zistili, že naša práca s TFS je už na vyššej úrovni ako to bolo pri prvom šprinte no stále sme nerobili niektoré veci tak ako by sme mali respektíve ako by sme si predstavovali. Zistili sme, že pri taskoch nezadávame správne hodnoty v story pointoch a, že to má výrazný vplyv na nežiadúci výzor nášho Burndown chart diagramu. Taktiež sme zhodnotili že by sme mali v TFS riešiť aj neprogramátorske úlohy a mohli by sme vylepšiť aj spôsob robenia nášho code review. Tieto problémy sme si na stretnutí vydiskutovali a navrhli sme vhodné riešenie na ich odstránenie. Pri vytváraní Backlog Items pre tretí šprint sme takmer všetok čas nášho šprintu investovali do dokončenia úloh z predchádzajúceho šprintu, no taktiež sme už načrtli víziu do budúcnosti a identifikovali sme si ciele ako vyriešiť outliers a taktiež možnosť pracovať na našom projekte pomocou Kinectu. Pri hraní Scrum Pokeru sme si určili vzorový task na základe, ktorého sme pridelovali hodnoty ostatným úlohám. V rámci tretieho šprintu bolo nutné dokončiť dokumentáciu.

4.4 ŠPRINT 4

Plánovanie. Plánovanie pri šprinte 4 prinieslo rozdelenie nášho tímu do viacerých smerov. Narozdiel od predošlých šprintov, kde sme riešili refaktoring a novú architektúru softvéru, tieto tasky boli konkrétnejšie a ich úlohou bolo priniesť novú funkcionalitu. Venovali sme sa segmentácií outlayerov, ktorá je aplikovateľná pri segmentácií stien a môže priniesť lepšie výsledky celkovej segmentácie. Ďalej sme sa venovali problému zalomenia stien, ktorý vzniká pri segmentácií. Jeden člen z nášho tímu hľadal možnosti využitia MS Kinect v našom projekte - rekošntrukcia nasnímaných dát. V neposlednom rade sme sa venovali vylepšovaniu nášho vizualizéra, ktorý je potrebný pri testovaní a samozrejme aj pri prezentácií nášho projektu. Ako posledný task sme si zadali priebežnú dokumentáciu kódu. Určili sme člena, ktorý sa bude venovať DoxyGen-u a bude kontrolovať aké komentáre sa píšu a či spĺňajú všetky dohodnuté pravidlá.
Priebeh. V šprinte išlo hlavne o analytické úlohy, ktoré mali priniesť vhodné návrhy riešení, ktoré budú otestované. Analýzy sme predstavili na stretnutí v strede šprintu, kde sme ich bližšie diskutovali s našou vedúcou.
Retrospektíva. Retrospektíva priniesla výraznú zmenu pre náš tím. Zistili sme 2 problémy, ktoré sú kritické. Je potrebné riadiť verziovanie a zmeniť

dĺžky šprintov. V prípade analytických šprintov (ako bol z veľkej časti tento) sme sa rozhodli pre 1 týždňový šprint, ktorá nám dopomôže k tomu, ako naplánovať ďalší 2 týždňoví šprint čo najpodrobnejšie. Taktiež je potrebný manažér verziovania, keďže sa stáva, že sme často závislí jeden od druhého a je potrebné, aby niekto kontroloval priebeh a dokončenie novej funkcionality (schválené pull requesty, nové branche a pod.).

5 GLOBALNA RETROSPEKTIVA

V rámci celkovej retrospektívy sme do tohto bodu semestra narazili na viacero problémov, ktoré sme sa snažili riešiť v každom šprinte. Celkovo môžeme vytýčiť niekoľko skutočností, ktoré ovplyvnili vývoj nášho tímu najviac.

1. **Práca s TFS** – jednoznačne jeden z najväčších problémov. Keďže sme doteraz nemali možnosť pracovať v tíme pozostávajúcom z viac ako dvoch ľudí, je to pre nás nová skúsenosť. Nemali sme žiadne skúsenosti s používaním systémov na manažment úloh. Taktiež spadajú pod tento problém aj iné náležitosti, ako napr. tvorba backlogu či zadávanie estimate-u pri taskoch a vlastne celkové poznatky v oblasti manažmentu úloh.

   V rámci 3. šprintu vidieť značný progres v tejto časti. Každý pracuje s tfs, interaguje s tabuľou, zadáva bugy a pod. Výsledok je vidieť na burn-down charte, ktorý naberá správny spád a taktiež aj na velocity, ktorá sa postupne zvyšuje.

2. **Plánovanie** – v rámci plánovania šprintu sme učiňovali viacero chýb. Nemali sme vytvorený backlog, z ktorého by sme len podsúvali podľa priority item-y do šprintov. Bolo to ovplyvnené hlavne tým, že pokračujeme po minuloročnom tíme a nevedeli sme presne kam sa chceme pohnúť a navyše je projekt výskumného charakteru, čo znamená, že backlog sa nám neustále mení vzhľadom na výsledky jednotlivých prototypov. Planning poker sme nehrali korektne a nezapisovali sme celý jeho opis rovno do description pri jednotlivých taskoch.

   V rámci 3. šprintu sme v tejto časti spravili jednoznačne najväčší krok vpred. Aj keď je pravdou, že výskumný projekt nie je možné naplánovať na pol roka dopredu, lebo nikdy nie je isté čo-ako dopadne. Napriek tomu sme sa posnažili naplniť backlog minimálne na najbližšie 3 šprinty. Z týchto item-ov sme podľa priority vybrali

tie s najvyššou a presunuli sme ich do šprintu. Je cítiť oveľa väčší prehľad v tom čo robíme.

Planning poker v 3. šprinte bol jednoznačne prínosnejší ako tie predtým. Hlasovalo sa o všetkých taskoch a diskusia sa písala rovno do ich opisu. Brainstoriming zabezpečil lepší odhad a taktiež aj presný opis toho čo je treba v budúcnosti urobiť.

V neposlednom rade môžeme vyzdvihnúť našu komunikáciu. Od začiatku používame *Slack*, štruktúrujeme komunikáciu na jednotlivé témy a všetci sa aktívne zapájame. Taktiež využívame *Google Drive* na zdieľanie jednotlivých dokumentov. Okrem iného, sa stretávame osobne, mimo nášho oficiálneho stretnutia – to hodnotíme kladne a určite v tom chceme pokračovať aj naďalej.

6 POUŽÍVANÉ METODIKY

TABUĽKA 4: POUŽÍVANÉ METODIKY

METODIKA ZADÁVANIA ÚLOH DO TFS	Metodika opisuje spôsob plánovania jednotlivých šprintov a napĺnania backlogu.
METODIKA PREHLIADOK KÓDU	Opisuje spôsob ako vykonávať prehliadky kódu a na čo sa pri nich zamerať.
METODIKA VERZIOVANIA KÓDU	Opisuje spôsob členenia projektu na jednotlivé verzie a ako správne používať Git.
METODIKA TESTOVANIA	Ako správne testovať náš systém.
METODIKA PÍSANIA DOKUMENTÁCIE	Tu sú spísané pravidlá, ktoré je potrebné pri písani dokumentácie dodržať.

7 EXPORT ÚLOH

Project: 3Drecon **Server:** 3drecon.visualstudio.com\3drecon **Query:** 3Drecon Team - Sprint 1 - Backlog **List type:** Tree

ID	Work Item Type	Title 1	Title 2	State	Assigned To	Description	Remaining Work	Activity
19	Product Backlog Item	Zrýchlenie výpočtov ukladaním medzivýsledkov		Done	Mario Hunka	Prioritou tohto backlogu je ukladať medzivýsledky a obmedziť tak zbytočné čakanie na to isté pri každom spustení. Nakoľko program používa zložité výpočty bolo na bežnom počítači veľmi zdĺhavé testovať novú funkcionalitu.		
10	Task		Pridanie checkboxu pre export medzivýsledkov	Done	Mario Hunka	Do gui je potrebné pridať checkbox aby bolo možné exportovať medzivýsledky.		
7	Task		Ukladanie medzivýsledkov do formátu .pcd	Done	Mario Hunka	Vytvorenie funkcionality ukladania výsledkov výpočtovo náročných funkcií programu s cieľom urýchlenia ladenia a vyvíjania programu.		Development
31	Task		Načítavanie .pcd uložených medzivýsledkov.	Done	Mario Hunka	Uložené medzivýsledky je potrebné správne načítať do objektu ClosedSpace, kvôli ďalšej práce s nimi.		Development
21	Bug	Slabá viditeľnosť		Done				
13	Task		Vytvorenie loading screenov	Done	Jakub Ginter	Používateľa je potrebné informovať o tom, že program práve pracuje na výpočtoch.		
8	Product Backlog Item	Analýza celkového kódu		Done		Po prevzatí projektu po predošlom tíme je potrebné zanalyzovať jeho štruktúru, jednotlivé moduly a triedy. Výsledky analýzy budú referované členmi tímu na ďalšom stretnutí.		
11	Task		Vyhľadanie nástroja na zapisovanie chýb	Done	Mario Hunka	Nakoľko v každom projekte vznikajú nejaké chyby, je potrebné ich nejakým spôsobom evidovať. Treba analyzovať dostupné možnosti a vybrať tú najvhodnejšiu pre náš projekt.		
14	Task		Analýza modulu 3DReconstruction/Methods	Done	Viktor Košťan			
22	Task		Analýza modulu 3DReconstruction/Objects	Done	Jakub Ginter	Analýza objektov používaných pri rekonštrukcii.		
23	Task		Analýza modulu 3DReconstruction/Primitives	Done	Jakub Ginter	Analýza tried v Primitives, ktoré sú využívané pri rekonštrukcii.		
26	Task		Analýza zvyšných tried v module DataHandling	Done	Mario Hunka	Analýza tried z DataHandlingu mimo konfigurácie.		
28	Task		Analýza modulu gui/GUI	Done	Mario Hunka	Analýza používateľského okna.		
29	Task		Analýza modulu gui/Widgets	Done	Mario Hunka	Analýza Widget tried, ktoré zabezpečujú výpočty v projekte GUI.		
39	Task		Prefarbenie bodov	Done	Jakub Ginter	Prefarbit body na ine farby pre lepsie orientovanie v kode		
40	Task		odstánenie zbytočných kopírovaní PointCloud	Done	hascicm.mh	po preskumani som zistil ze kopirovania su opodstatnene, ale otvorila sa moznost navrhnut vlastny data parser do pcl::PointXYZ		
25	Task		Analýza modulu	Done	hascicm.mh			
9	Bug	Zlyhanie programu pri načítaní		Done	Viktor Košťan			
32	Task		Implementovanie bezpečného načítania konfiguračných	Done	Viktor Košťan			

Project: 3Drecon **Server:** 3drecon.visualstudio.com\3drecon **Query:** 3Drecon Team - Sprint 2 - Backlog **List type:** Tree

ID	Work Item Type	Title 1	Title 2	Assigned To	Description	Remaining Work	Activity
75	Product Backlog Item	Refaktoring tried segmentácie		Richard Pinter			
76	Task		Identifikácia častí kódu týkajúcich sa segmentácie	Richard Pinter			
77	Task		Osamostatnenie kódu týkajúceho sa segmentácie	Richard Pinter			
74	Product Backlog Item	Vytvorenie abstraktnej triedy pre Segmentáciu		Richard Pinter			
81	Task		Vytvorenie triedy pre segmentáciu	Richard Pinter			
72	Product Backlog Item	rekonštrukcie pre nový model		hascicm.mh			
83	Task		prepracovanie triedy do nového projektu	hascicm.mh			
97	Task		objektov do nového projektu	hascicm.mh			
71	Product Backlog Item	refaktorovanie tried rekonštrukcie		hascicm.mh			
84	Task		refaktorovanie tired	hascicm.mh			
65	Product Backlog Item	Vytvorenie nového projektu Interior3DRecon		Viktor Košťan			
66	Task		Vytvorenie a nastavenie projektu Interior3DRecon	Viktor Košťan	projektov 3DReconstruction, DataHandling a Exporter.		
51	Product Backlog Item	Refactoring náčítania dát zo súborov		Viktor Košťan			
82	Task		Premiestnenie DataReader do Interior3DRecon	Viktor Košťan			
50	Product Backlog Item	Refactoring triedy ClosedSpace		Viktor Košťan			
67	Task		Vytvorenie základnej verzie triedy ClosedSpace	Viktor Košťan			

45	Product Backlog Item	Vytvorenie projektu pre GUI		Mario Hunka	rozhranie. Preto je potrebné vytvoriť nový projekt pre GUI a nastaviť všetko tak, aby aplikácia bežala bez problémov.		
57	Task		Integrovanie QT do projektu	Mario Hunka	iný framework -QT. Z čoho vyplýva, že je nutné nastaviť cesty ku zdrojovým súborom QT. Taktiež nainštalovať addon do MCVS 2013, pomocou, ktorého je možné prepojiť QT Creator a MCVS.		
58	Task		projektu s ostatnými projektami	Mario Hunka	Nastavenie dependencies pre GUI (iné moduly, ktoré GUI používa)		
59	Task		Vytvorenie zakladneho pouzivatelskeho rozhrania	Mario Hunka	Vytvorenie triedy pre gui a základného template.		
60	Task		Vytvorenie noveho GUI pomocou QT	Mario Hunka	Design nového GUI.		
61	Product Backlog Item	Prepojenie GUI s projektom		Mario Hunka	Je potrebné zabezpečiť prepojenie GUI s logickou časťou projektu.		
62	Task		EventHandler pre rekonštrukciu	Mario Hunka	Handler pre elementy, ktoré súvisia s rekonštrukciou scény.		
63	Task		EventHandler pre vizualizaciu	Mario Hunka	Handler pre elementy, ktoré súvisia s vizualizáciou.		
64	Task		EventHandler pre načítanie súboru	Mario Hunka	Handler pre elementy, ktoré súvisia s načítavaním/exportovaním súboru.		
80	Task	Vytvorenie novej triedy pre vizualizer		Jakub Ginter	Vytvorenie projektu pre refaktoring starého kódu Vizualizácie priestoru		
89	Task	Odstránenie nepotrebného kódu		Jakub Ginter	Zmazanie starého kódu, ktorý sa nepoužíva		

Project: 3Drecon **Server:** 3drecon.visualstudio.com\3drecon **Query:** 3Drecon Team - Sprint 3 - Backlog **List type:** Tree

ID	Work Item Type	Title 1	Title 2	Assigned To	Description	Remaining Work	Activity
146	Product Backlog Item	Dokumentácia k inžinierskemu dielu					
156	Task		Štruktúra dokumentu	Jakub Ginter	Vytvorenie nového dokumentu na google drive. Návrh štruktúry - kapitoly, podkapitoly.		
157	Task		Kapitola úvod	Jakub Ginter			
158	Task		Kapitola ciele	Jakub Ginter			
145	Product Backlog Item	Dokumentácia riadenia					
147	Task		Štruktúra dokumentu	Mario Hunka	Vytvorenie nového dokumentu na google drive. Návrh štruktúry - kapitoly, podkapitoly.		Documentation
148	Task		Finalizácia dokumentu	Mario Hunka	Po dokončení dokumentácie je potrebné naformátovať celú dokumentáciu do jedného štýlu.	1	Documentation
149	Task		Kapitola úvod	Mario Hunka	Napísanie kapitoly úvod.		
150	Task		Integrácia dokumentácie	Mario Hunka	Vzhľadom na to, že dokumentáciu robia všetci členovia tímu. Je potrebné dať všetko do jedného súhrného dokumentu.		Documentation
151	Task		Sumarizácia šprintu 1	Jakub Ginter			
152	Task		Export evidencie úloh	Jakub Ginter			
153	Task		Manažment komunikácie	Jakub Ginter			
154	Task		Manažment vývoja a integrácie	Jakub Ginter			
155	Task		Manažment riadenia	hascicm.mh			
159	Task		Globalna retrospektiva	Mario Hunka			
160	Task		Manazment planovania	Mario Hunka			
142	Bug	Zlá konverzia stringu pri načítavaní nového		Viktor Košťan		3	Testing

143	Task		Oprava konverzie stringu pri načítavaní	Viktor Košťan			Development
144	Task		(TEST) Oprava konverzie stringu pri načítavaní			1	Development
140	Product Backlog Item	Webová prezentácia tímu		Mario Hunka			
141	Task		Update priebeznej dokumentacie na webe	Mario Hunka			Documentation
137	Product Backlog Item	Tvorba dokumentu na logovanie odrobených hodín		Viktor Košťan	na taskoch, aby mal produkt owner prehľad o odrobenej práci, keďže tfs nedovoľuje zdarma prístup do projektu pre viac ako 5 ľudí.		
138	Task		Vytvorenie excel sheetu na logovanie práce	Viktor Košťan	Je potrebné vytvoriť excel sheet na logovanie práce a vložiť do zdieľaného priečinku na drive.		Documentation
75	Product Backlog Item	Refaktoring tried segmentácie		Richard Pinter			
78	Task		Odstránenie nepotrebného kódu	Richard Pinter		5	
55	Product Backlog	Vizualizácia spracovaných dát		Jakub Ginter	Všetky dáta, ktoré načítavame a spracovávame je potrebné vizualizovať.		
91	Task		Vizualizácia Point cloud vo formáte XYZ	Jakub Ginter	Vizualizácia Point cloud vo formáte XYZ pre naskenovaný priestor	5	
93	Task		Vizualizácia Point cloud s farebným odlíšením	Jakub Ginter	Vizualizácia Point cloud s farebným odlíšením pre naskenovaný priestor	5	
52	Product Backlog	Refactoring ukladania (exportu) dát do		Viktor Košťan			
94	Task		Premiestnenie triedy Exporter do	Viktor Košťan			
95	Task		Refactoring triedy Exporter v	Viktor Košťan		3	
50	Product Backlog	Refactoring triedy ClosedSpace		Viktor Košťan			
88	Task		Vytvorenie segmentačného	Viktor Košťan		4	
106	Product Backlog Item	Finalizácia refaktoringu			Posledná časť refaktoringu, v ktorej sa vykoná merge všetkých branches refaktoringu do jednej výslednej.		

107	Task		Vytvorenie metódy calculate pre triedu	Richard Pinter		5	Development
111	Task		(MERGE) Vytvorenie metódy calculate pre			1	Development
110	Task		(TEST) Vytvorenie metódy calculate pre			2	Testing
112	Task		(REVIEW) Vytvorenie metódy calculate pre			2	Development
108	Task		Vytvorenie metódy calculate pre triedu	hascicm.mh		8	Development
113	Task		(MERGE) Vytvorenie metódy calculate pre			1	
114	Task		(TEST) Vytvorenie metódy calculate pre			2	
115	Task		(REVIEW) Vytvorenie metódy calculate pre			2	Development
109	Task		Integrácia segmentačných funkcií	Viktor Košťan		3	Development
116	Task		(MERGE) Integrácia segmentačných funkcií			1	Development
117	Task		(TEST) Integrácia segmentačných funkcií			2	Testing
118	Task		(REVIEW) Integrácia segmentačných funkcií			1	Development
119	Task		Handler pre GUI	Mario Hunka	Prepojenie logickej časti v GUI. Pridanie handlerov na výpočet segmentácie konkrétnou metódou a vizualizácie pomocou nového vizualizéru.	3	Development
120	Task		(MERGE) Handler pre GUI			1	Development
121	Task		(TEST) Handler pre GUI			2	Testing
122	Task		(REVIEW) Handler pre GUI			1	Development
123	Task		Vizualizácia segmentovaných dát	Jakub Ginter	Na vizualizér je potrebné odoslať dáta aj s informáciou, čo ideme vizualizovať. Je potrebné dorobiť možnosť vizualizácie už rekonštrovaných dát.	8	Development
124	Task		Interaktor vizualizéru	Jakub Ginter	Interakcia so starým vizualizérom nebola intiutívna. Treba zmeniť spôsob vizualizácie a urobiť interakciu v okne lahšie použiteľnou.	8	Development
125	Task		(MERGE) Vizualizér			1	Development

126	Task		(TEST) Vizualizér			5	Testing
127	Task		(REVIEW) Vizualizér			2	Development
131	Product Backlog Item	Tvorba metodík					
132	Task		Manažment code review	Mario Hunka	Metodika na to ako robiť prehliadky kódu.		Documentation
133	Task		Manažment verzií	Jakub Ginter	Metodika verziovania projektu		Documentation
134	Task		Manažment testovania	hascicm.mh			Documentation
135	Task		Manažment úloh	Viktor Košťan			Documentation
136	Task		Manažment dokumentácie	Richard Pinter		8	Documentation
100	Product Backlog Item	Migrácia existujúcej implementácie segmentácie outlayerov					
128	Task		Migrácia existujúcej implementácie	Mario Hunka	Presun logiky segmentácie outlayerov do nového projektu.	13	Development
129	Task		(TEST) Migrácia existujúcej			5	Testing
130	Task		(REVIEW) Migrácia existujúcej			2	Development

Project: 3Drecon **Server:** 3drecon.visualstudio.com\3drecon **Query:** 3Drecon Team - Sprint 4 - Backlog **List type:** Tree

ID	Work Item Type	Title 1	Title 2	Assigned To	Description	Remaining Work	Activity
209	Bug	Viacnásobný výpočet rovnakých výsledkov segmentačných metód		Viktor Košťan			Development
210	Task		Vytvorenie triedy zabezpečujúcej lazy evaluation	Viktor Košťan			Development
101	Product Backlog Item	Analýza rôznych metód lokalizácie outlayerov		Mario Hunka			
194	Task		Verifikácia analýzy	Mario Hunka			
195	Task		Analýza metódy ABOD	Mario Hunka			
196	Task		Analýza existujúcich metód	Mario Hunka			
197	Task		Analýza metódy založenej na hustote datasetu	Mario Hunka			
198	Task		Analýza určovania outliers pomocou neuronových sietí	Mario Hunka			
12	Bug	Zalamovanie stien.		Viktor Košťan			
20	Task		Identifikovať príčinu zalamovania stien	Viktor Košťan	Hypotéza je, že pridávanie hraničných bodov stien (polygónov) pomocou konkávnej obálky spôsobuje to, že sa medzi ne pridajú body, ktoré neležia na rovine steny.		
206	Task		Opraviť identifikovanú príčinu zalamovania stien	Viktor Košťan	Identifikovaná príčina stien - nie je implementované premietnutie bodov konkávnej obálky segmentu na rovinu segmentu pri upresnení ohraničenia segmentu.		Development
168	Product Backlog Item	Návrh prototypu pre lokalizáciu outlayerov		Mario Hunka			
169	Task		Vyhodnotenie analýzy	Mario Hunka			
214	Task		Návrh riešenie pomocou zobrazenia do 2D	Mario Hunka			
170	Product Backlog Item	Implementácia prototypu lokalizácie outlayerov		Mario Hunka			
171	Task		Vizuálne overenie	Mario Hunka		13	
172	Product Backlog Item	Zosnímanie testovacej vzorky Kinectom		hascicm.mh			
208	Task		snimanie dát pomocu Kinect SDK	hascicm.mh			
175	Task		Verifikácia nasnímaných dát	hascicm.mh			

173	Product Backlog Item	DataHandling pre Kinect dáta		hascicm.mh			
213	Task		načitanie a rekonštruovanie vytvoreného datasetu	hascicm.mh			
176	Task		Verifikácia načítania súborou z Kinect	hascicm.mh			
174	Product Backlog Item	Analýza metód rekonštrukcie 3D dát nasnímaných z viacerých uhľov pohľadu		hascicm.mh			
177	Task		Vyhodnotenie analýzy	hascicm.mh		13	
199	Task		analyzovanie existujúcich metód	hascicm.mh			
202	Task		analýza odborného článku	hascicm.mh	http://www.tuit.ut.ee/sites/default/files/tuit/atprog-courses-bakalaureuset55-loti.05.029-lembit-valgma-text-20160520.pdf		
178	Product Backlog Item	Analýza metód pre rozpoznávanie a segmentáciu stien		Viktor Košťan	Analýza metód rozpoznávania a segmentácie stien z point cloud dát (včetne tej, ktorá sa používa v tomto projekte).		
179	Task		Vyhodnotenie analýzy	Viktor Košťan		21	
205	Task		Analýza aktuálnej implementácie rozpoznávania a segmentácie stien.	Viktor Košťan			
207	Task		Rešeršovanie metód rozpoznávania a segmentácie stien	Viktor Košťan			
182	Product Backlog Item	Generovanie dokumentácie pomocou Doxygen		Richard Pinter			
183	Task		Verifikácia výstupu	Richard Pinter			
184	Product Backlog Item	Kontrola a oprava kódu podľa code convetions		Richard Pinter			
185	Task		Finalizacia opráv	Richard Pinter		13	
187	Product Backlog Item	Zmena interaktoru vizualizéra		Jakub Ginter			
191	Task		Overenie funkčnosti vizualizéra	Jakub Ginter		8	
188	Product Backlog Item	Vizualizácia segmentovaných polygónov		Jakub Ginter			
192	Task		Overenie vizualizácie segmentovaných dát	Jakub Ginter		3	
200	Task		Oprava vizualizácie originálnych dát	Jakub Ginter	Oprava chyby pri vizializovaní originálnych dát. Okno sa nedá zatvoriť ani nijak ovládať.		

Read-only

201	Task		Implementácia vizualizácie segmentovaných dát	Jakub Ginter			
189	Product Backlog Item	Integrácia 3d myši		Jakub Ginter			
193	Task		Verifkácia interakcie s 3D myšou	Jakub Ginter			
203	Task		Konfigurácia projektu s novým SDK pre 3d myš	Jakub Ginter			
204	Task		Analýza 3D myši	Jakub Ginter	Analýza dostupných funkcií a knižníc obsiahnutých v SDK pre 3d myš. Hľadanie dostupných funkcií pre náš projekt.		

8 PREBERACICE PROTOKOLY

Tímový projekt 2016/2017

Tím č. 11 – 3DRecon

Predmet odovzdávania:

Dokumentácia riadenia – prvých štryoch šprintoch

Projektová dokumentácia – po prvých štyroch šprintoch

Vedúci tímového projektu: Ing. Vanda Benešová, PhD.

Podpisom potvrdzuje prevzatie vyššie uvedených častí dokumentácie

V Bratislave

Dátum Podpis

9 MOTIVAČNÝ DOKUMENT

Dobrý deň,

dovoľte mi predstaviť náš tím pre tímový projekt. Skladá sa zo 6 absolventov bakalárskeho štúdia na FIIT, pričom sme všetci študovali v odbore informatika. Menovite sa v tíme nachádza Jakub Ginter, Miroslav Haščič, Mário Hunka, Viktor Košťan, Richard Pintér a Matej Kollár. Nakoľko neštudujeme všetci rovnaký odbor sú v našom rozvrhu zastúpené takmer všetky odborné predmety inžinierskeho štúdia, ktoré nám poskytnú znalosti pre prácu na širokej škále projektov. Tím sme však neskladali náhodne. Pracovali sme spolu na viacerých menších projektoch v rámci štúdia, nie sme len kolegovia, ale aj kamaráti. Využívali sme rôzne technológie, primárne však prevládajú v našom tíme skúsenosti s vývojom v jazyku Java. S týmto jazykom majú široké skúsenosti všetci členovia nášho tímu. V rámci databázových technológií sú v našom tíme zastúpené PostgreSQL a MySQL ale tak isto aj MariaDB či Hibernate. Náš tím dokáže efektívne pracovať na vývoji ako webových aplikácií tak aj mobilných. Pokročilé skúsenosti máme s vývojom v Pythone(Django), C/C++ a už spomínaná Java. Pri tvorbe webovej aplikácie vieme využiť naše skúsenosti v HTML, CSS, JavaScript(AngularJS). V rámci školských povinností sme si vyskúšali prácu s použitím technológií v oblasti počítačových sietí, databáz, paralelného programovania alebo grafiky a videnia. V rámci bakalárskych prác sme pracovali na témach - vývoj aplikácie pre mobilné zariadenia s platformou Android, analýza správania používateľa, vizualizáciu dát v obohatenej realite a iné.

Počas mimoškolských aktivít sme nabrali skúsenosti v oblasti vývoja i testovania softvéru, tvorby webových stránok či v analýze dát.

Doposiaľ dosiahnuté vedomosti dokážeme efektívne využívať a rozdeliť si prácu, aby každý člen pracoval na tom čo ho zaujíma. Našou najsilnejšou stránkou je tímový duch a chuť pracovať na projekte spoločne a tým vytvoriť čo najzaujímavejší produkt s potenciálom pre reálne využitie praxi.

Motivácia EduSim

Náš tím sa prioritne zaujíma o tému Tvorba vzdelávacích simulácií, nakoľko pokladáme segment vzdelávania za veľmi neefektívny, obzvlášť čo sa týka využívania informačných technológií. Ako aj z vlastnej skúsenosti

vieme, predstava fyzikálnych alebo chemických javov, ako napríklad gravitačná sila, elektromagnetizmus alebo fotosyntéza, môže byť pre žiaka veľmi náročná. Práve preto by sme radi pracovali na aplikácií, ktorá by žiakom zjednodušila učenie sa. Sami si vieme predstaviť ako by takáto aplikácia zefektívnila výučbu na školách, ktoré sme navštevovali.

Máme skúsenosti s tvorbou webových aplikácií, avšak prácu vo väčších tímoch sme si zatiaľ nemali možnosť vyskúšať, resp. sme pracovali iba v menších tímoch v rámci školských projektov. Práve preto chceme primárne pracovať na tejto téme, pretože zaujala všetkých členov nášho tímu. Skúsenosťami v tíme sa dokážeme navzájom dopĺňať, dokážeme vytvoriť funkčný frontend aj backend.
V rámci štúdia vieme podporiť vývoj znalosťami z premetov ako Pokročilé databázové technológie, Architektúry informačných systémov aj Spracovanie obrazu, grafika a multimédia či predmetom bakalárskeho štúdia PPGSO. Myslíme si že naše znalosti pre vypracovanie tohto projektu sú dostatočné a rovnako aj náš záujem o školstvo a proces edukácie samotnej. Budeme radi ak nám bude umožnené podieľať sa na tomto projekte.

Motivácia eMotion

Ako sekundárnu v rámci priority sme vybrali tému Manažment zdravotného stavu pacienta prostredníctvom monitoringu emócií. Táto téma nás zaujala z dôvodu, že vytvorením takéhoto systému je možné reálne pomáhať ľudom v oblasti zdravia prostredníctvom informačných technológií. Tento segment pokladáme za veľmi zaujímavý a pri spolupráci so spoločnosťou, ktorá sa v ňom aj pohybuje by bolo možné vytvoriť veľmi zaujímavý systém.

Znalosti, ktoré by sme pri práci na tomto projekte vedeli využiť bude možné prepojiť v rámci štúdia s predmetmi ako Architektúra softvérových systémov, Objektovo-orientovaná analýza a návrh softvéru, Vizualizácia dát, Pokročilé databázové technológie, Vyhľadávanie informácií aj Objavovanie znalostí. Všetci členovia tímu majú pokročilé skúsenosti s programovaním v jazyku Java a tiež máme záujem získať vedomosti v oblasti programovania v jazyku Swift, ktorý ma veľký potenciál využitia. Naše skúsenosti a vedomosti pokladáme za dobrý predpoklad pre prácu na projekte eMotion a budeme radi ak nám to bude umožnené.

10 METODIKY

10.1METODIKA ZADÁVANIA ÚLOH DO TFS

V rámci tohto projektu (3dRecon) sa úlohy manažujú v 3 fázach:

1. *Tvorba Backlogu*
2. *Plánovanie šprintu*
3. *Práca na taskoch počas šprintu*

Manažment úloh sa v tomto prípade týka **Backlog Items**, **Bugs** a **Tasks**. Epics a Features nie sú opísané, keďže sa s nimi manipuluje iba zriedka.

Najdôležitejšia časť metodiky pre členov tímu je *Práca na taskoch počas šprintu*.

SLOVNÍK

BLI - Backlog Item
PO - Product Owner

10.1.1 TVORBA BACKLOGU

Tvorba Backlogu je činnosť, ktorá prebieha stále. Vytvoriť nový Backlog Item alebo Bug je možné hocikedy a nemusí sa čakať s vytvorením na konzultáciu s ostatnými členmi tímu alebo s Product Ownerom, aj keď práve PO dáva najviac podnetov na BLI vzhľadom na tému.

Životný cyklus BLI

1. *Vytvorenie BLI*
2. *Schválenie BLI Product Ownerom*
3. *Odhadnutie Effortu BLI*
4. *Práca na BLI*
   - *4.1. Priradenie do sprintu*
   - *4.2. Odloženie (pozastavenie) BLI*
5. *Dokončenie BLI*

(body 4 a 5 sú opísané v kapitolách *Plánovanie šprintu* a *Práca na taskoch počas šprintu*)

10.1.1.1 VYTVORENIE BACKLOG ITEMU

Aj keď je možné vytvoriť BLI hocikedy je dôležité aby sa zadali všetky nižšie popísané atribúty BLI-u aby sa neskôr nebrzdili procesy zadávaním týchto atribútov. Tiež v čase ich tvorby ich vieme opísať rýchlo, lebo sa nemusíme rozpamätávať.

Kroky tvorby Backlog Itemu:

1. **Vhodne nazvať BLI** - *Title* - Názov Backlog Itemu by mal byť dostatočne popisný aby vystihoval prácu ktorú predstavuje, pričom by nemal obsahovať názvy tried, funkcií atď. pokiaľ nie sú nevyhnutné pre popis aby sme neobmedzovali tasky vo voľbe implementácie. *(miesto názvu funkcie alebo triedy skôr použiť opis funkcionality)*. Zlý názov "Vytvorenie triedy Exporter" - Dobrý názov "Vytvorenie exportu segmentačných dát"

2. **Pridať "*Description*"** - *Description* - Popis by mal byť natoľko podrobný, aby umožnil členom tímu odhadnúť *effort* a určiť tasky na jeho implementáciu.

3. **Určiť "*Acceptance Criteria*"** - *Acceptance Criteria* - Kritérium akceptácie reprezentuje "*Definition of done*". Slúži na verifikáciu dokončenosti BLI. Mala by byť definovaná minimálne 1 akceptačná podmienka.

4. **Určiť "*Value area*"** - Ak sa BLI týka zmeny štruktúry programu a nepridáva funkcionalitu tak je *Architecture*, inak *Business*.

5. **Určiť "*Priority*"** - Pri určení priority je dobré vychádzať z priorít ostatných BLIs a určiť ju vzhľadom na ne.

6. **Priradiť ku "*feature*" (pridať vzťah *patent*)**

7. **Priradiť predchodcov (pridať vzťah *successor*)** - Je dôležité priradiť predchodcu BLI, aby sa pri plánovaní šprintu vedelo, či bude možné BLI implementovať (či nebude musieť BLI čakať na predchodcu).

8. **Usporiadať BLI podľa priority**

V rámci Backlog treba premiestniť nový BLI tak, aby boli Backlog Items v zozname usporiadané podľa priority.

10.1.1.2 SCHVÁLENIE BACKLOG ITEMU PRODUCT OWNEROM

Po schválení BLI Product Ownerom (väčšinou na stretnutiach) je potrebné zmeniť jeho stav z *New* na *Approved*. Ak sa BLI vytvára na podnet Product Ownera, tak po jeho vytvorení tiež treba zmeniť stav na *Approved*.

10.1.1.3 ODHADNUTIE EFFORTU BACKLOG ITEMU

Odhadnutie effortu BLI by sa malo vykonať na tímových stretnutiach pomocou *planning pokeru*, pričom sa môže upraviť aj jeho popis ak si ho počas planning pokeru členovia tímu lepšie upresnia.

10.1.2 PLÁNOVANIE ŠPRINTU

Pred začatím plánovania šprintu by mali mať všetky BLIs **určený effort**, aby sa vedelo podľa *velocity* z predošlého šprintu odhadnúť aké BLIs vybrať aby sa stihli spraviť do konca šprintu.

Kroky plánovania šprintu:

1. **Určiť Backlog Itemy pre nasledujúci šprint. -** Backlog itemy pre nasledujúci šprint sa určujú na základe *velocity* z predošlého šprintu a závislosti Backlog Itemov (niektoré môžu byť vykonané až po druhých)

2. **Určiť a vytvoriť tasky pre Backlog Itemy**

   2.1. *Title* - Názov tasku by mal konkrétne popisovať úlohu, ktorá sa ma vykonať. V prípade *Development* tasku by sa mali používať aj názvy funkcií a tried.

   2.2. *Description* - Popis tasku **netreba** pokiaľ je názov dostatočne opisný.

   2.3. *Activity* treba určiť podľa typu tasku

3. **Odhadnúť Remaining Work tasku**

4. **Určiť vzťahy medzi taskmi**

   4.1. *Predecessor/Successor* - potrebné pre všetky typy taskov, ak sa dá

   4.2. *Related* - vhodné pre tasky ako REVIEW a pod. ktoré môžu bežať súbežne a ovplyvňujú sa (súvisia) istým spôsobom

   4.3. *Tests/Tested By* - tasky typu *Testing* (Activity) majú vzťah *Tests* s taskami, ktoré testujú.

10.1.3 PRÁCA NA TASKOCH POČAS ŠPRINTU

Počas šprintu sa vykonávajú nasledujúce aktivity:

1. **Práca na tasku -** Pri začatí práci na tasku sa člen tímu priradí do daného tasku pokiaľ tak už neurobil a presunie task (zmení mu stav) do *In progress*.

2. **Dokončenie práce na tasku -** Ak sa dokončí práca na tasku je nutné ho presunúť do *Done*. Tiež sa skontroluje Backlog Item tasku, či má dokončené všetky tasky. Ak áno a **sú splnené *Acceptance Criteria***, tak sa nastaví stav Backlog Itemu na *Done*.

3. **Obnovenie práce na tasku -** V prípade ak sa ukáže, že task nie je dokončený, napr. ak neprejde cez testy, zmení sa jeho stav späť na *In progress*.

4. **Objavenie chyby -**Ak sa objaví chyba vytvorí sa nový *Bug* do Backlogu a podľa závažnosti a potreby sa môže pridať do aktuálneho šprintu. Pri jeho vytváraní sa nastavia atribúty *Severity*, *Effort*, *Remaining Work* a *Activity*. (atribúty *Effort* a *Remaining Work* sa nastavia na rovnakú hodnotu)

5. **Pridávanie potrebných taskov -** Počas šprintu sa často stáva, že sa zistí, že treba vytvoriť nový task, lebo pri plánovaní sa s ním nerátalo, alebo sa objavil bug a treba vykonať tasky na jeho opravu. V takých prípadoch sa môže vytvoriť task a pridať do BLI alebo Bugu pre daný šprint. Nevadí, že Burdown Chart stúpne, pretože podľa toho ako často a o koľko stúpne sa vie vyhodnotiť, ako dobre sa plánuje, prípadne ako často nastávajú chyby.

6.

10.2METODIKA PREHLIADOK KÓDU

Tento dokument pojednáva o spôsobe vykonávania prehliadok naším tímom. Každý *pull-request* musí prejsť schválením, ktorému predchádza prehliadka kódu.

10.2.1 ZÁKLADNÉ USTANOVENIA

1. Schváleniu predchádza prehliadka kódu, ktorú vykonáva vopred určený člen tímu. V nevyhnutných prípadoch je možné, aby prehliadku spravil niekto iný, avšak je potrebné dostatočne vopred oznámiť, že prehliadku daný človek nebude môcť vykonať, pričom treba uviesť relevantný dôvod.
2. Nikto by nemal robiť prehliadku pre kód dlhší ako **400** riadkov kódu. V prípade, že je kódu viac, je potrebné, aby bola prehliadka vykonaná viac ako jedným členom.
3. Prehliadka by mala trvať maximálne **1 hodinu**.

10.2.2 PRIEBEH PREHLIADKY

Pri prehliadke je potrebné pozrieť sa na kód z viacerých pohľadov, za účelom odhaliť čo najviac chýb a tak zabezpečiť vyššiu kvalitu kódu.

ARCHITEKTÚRA

- ***Single responsibility principle*** - každá trieda je zodpovedná za **práve** jednu konkrétnu vec. Tento princíp je možné uplatniť aj pri metódach. Ak metóda robí 2 veci, nie je to správne.
- ***Open/Closed principle*** - každá trieda musí byť orvorená pre rozširovanie (dá sa rozširovať, je možné pridávať novú funkcionalitu…), ale uzavretá pre modifikáciu (existujúcu implementáciu, je možné použiť inými triedami/modulmi)
- ***Duplikácia kódu*** - ak si všimneme duplikovaný kód, treba zvážiť či nie je možné vytvoriť metódu, ktorú je potom možné využiť na viacerých miestach.
- ***Error handling*** - používané metódy je potrebné ošetriť proti zlyhaniu (napr. načítanie *.pcd* súboru - *FileNotExistException* - je ošetrená ?).

Inými slovami, vždy musí byť alternatívny scenár, ktorý sa vykoná v prípade neúspechu primárneho priebehu.

- ***Efektívnosť*** - treba sa zamyslieť či daný algoritmus nie je možné urýchliť. Napr. iterovanie celého listu v prípade, že to nie je potrebné (hash tabulka, dictionary apod.)
- ***Potenciálne chyby*** - táto časť je jednoznačne najťažšia, ale je potrebné sa zamyslieť aj nad ňou. Každý kód by mal prejsť samozrejme testovaním, avšak, je možné, že sa nachádza v kóde nejaká logická chyba ? Ak si ju všimneme je výborné nato poukázať.

ŠTÝL

Vzhľadom na skutočnosť, že pokračujeme v minuloročnom projekte, je vhodné aby sme dodržiavali code convections, ktoré boli zavedené minulý rok. Táto metodika sa nachádza v minuloročnej dokumentácií a je možné si ju prezrieť. Pri prehliadke postupujme preto podľa tejto metodiky.

V minuloročnej metodike sú konkrétne návody ako písať kód. Uvediem preto zopár konvencií, ktorých je všeobecne správne sa držať a je vhodné nedostatky týchto konvencií hľadať pri prehliadkach kódu.

- ***Mená metód a premenných*** - mená musia byť jednoznačné. Ak napadne pri prehliadke niekomu lepšie pomenovanie, nebojte sa napísať svoj návrh.
- ***Dĺžka metódy*** – 50 riadkov
- ***Dĺžka triedy*** – 200-300 riadkov
- ***Komentovaný kód*** - komentovaný kód nemá čo hľadať pri žiadosti o mergnutie novej funkcionality. Ak chcete dať návrh, že niečo treba spraviť, niečo inak atď. Napíšte to do komentára a diskutujte to s ostatnými.
- ***Komentáre*** - sú dostatočne okomentované všetky metódy, premenné atď? Ak nie, treba nato upozorniť. Je veľmi dobré komentovať aj parametre, ktoré do metódy vstupujú, nie len samotnú funkcionalitu.

10.2.3 TESTING

- ***Testy*** - pre každú funckonalitu, musí byť dokončnené testovanie. Ak nie je, treba počkať dokedy bude.
- ***DFD*** - spĺňa kód *Definition of Done* (zabezpečuje funkcionalitu, ktorú má?)

10.2.4 AKO ROBIŤ PREHLIADKU

1. Pýtajte sa
2. Diskutujte (označte tagom v *tfs* daného človeka)
3. Argumentujte
4. Dajte návrhy nato, ako danú vec zlepšiť
5. Nezabudnite pochváliť, ak je to zaslúžené :)

Na záver ešte jedno zlaté pravidlo.

Skúste vždy pred pull requestom spraviť code review sám sebe.

10.3METODIKA VERZIOVANIA KÓDU

Na manažment verzií využívame TFS od spoločnosti Microsoft, ktorý je priamo integrovaný do Visual Studia.

SLOVNÍK

- **Branch** – vetva s projektovým kódom, prípadne jeho časťou.
- **Commit** – odoslanie zmien do úložiska.
- **Merge** – Zlúčenie vetiev kódu, vyriešenie konfliktov.
- **Konflikt** – vznikne v prípade že rovnaká vetva bola na rovnakom mieste upravená dvoma spôsobmi, resp. v dvoch vetvách.
- **Pull-Request** - teda získanie zmien zo vzdialeného repozitára a ich zamergovanie do hlavnej vetvy.

10.3.1 BRANCHES

Projekt je vetvený do 2 hlavných vetiev (*branches*).

- **master** – vetva obsahujúca iba skontrolovanú a overenú funkcionalitu. Je to hlavná projektová vetva. O túto vetvu sa stará integrátor projektu a iba on môže pripojiť do tejto vetvy novú funkcionalitu. Z tejto vetvy sa aplikácia nasadzuje.
- **dev** – je to developerská vetva určená pre verziovanie kódu. Do takejto vetvy je možné poslať iba funkčný kód po teste programátorom. Do tejto vetvy môže pridať funkcionalitu každý developer, musí však dodržať pravidlo, že všetka funkcionalita bude bez problémov fungovať aj s ostatnou v tejto vetve. Ostatný developeri teda nebudú mať problém použiť ju pri práci na svojej úlohe.
- **Pomocné vetvy** – každý developer môže vytvárať svoje pomocné vetvy a pracovať v nich. Každá vetva sa však vytvára za účelom práce na konkrétnom tasku alebo backlog iteme. Avšak ak chce túto vetvu pridať do nadradenej, musí dodržať určité pravidlá spísané nižšie v dokumente. V prípade, že celý tým bude pracovať na konkrétnom okruhu funkcionality, môže sa vytvoriť dočasná developerská vetva, z ktorej si všetci urobia svoju vetvu a pri dokončení sa tieto vetvy spoja. V jednej vetve pracuje jeden

človek. Výnimkou však môžu byť určité kedy bude v jednej vetve pracovať viac ľudí ale je potrebný daľši manažment.

V prípade, že sa developer rozhodne pridať novú funkcionalitu do master vetvy musí mať kód patrične zdokumentovaný podľa pravidiel dokumentovania. Kód musí byť okomentovaný a otestovaný. Ak je všetko splnené informuje o tom integrátora. Ten projekt skontroluje a ak bude všetko v poriadku spojí vetvy, ak nie, ohlási nájdené chyby developerovi a ten ich musí opraviť.

Medzi vetvami je možné prepínať sa iba v prípade, že všetky zmeny boli odoslané alebo ak majú vetvy rovnakú verziu. Ak nemajú rovnakú verziu nebude možné prepnúť sa medzi vetvami.

10.3.2 PULL-REQUEST

Pull-Request vetvy podlieha nasledujúcim pravidlám:

1. Funkcionalita vetvy je otestovaná a nenašli sa žiadne chyby.
2. Vetva je bez problémov skompilovateľná.
3. Nadradené sú vždy pravidlá vetvy, do ktorej sa pod-vetva pridáva.
4. Ďalšie pravidlá sú definované v dokumente manažment prehliadok, ktorým musí prejsť každá nová funkcionalita.

Každý Pull-Request podlieha procesu schvaľovania tzv. *code review*.

Nikto si nesmie svoj Pull-Request schváliť sám!

Požiadavka o kontrolu, teda nový Pull-Request sa objaví v komunikačnom nástroji. Jeden z developerov (nie autor) si vezme úlohu z boardu ohľadne schvaľovania tejto vetvy a skontroluje či je všetko splnené. Ak áno povolí Pull-Request, ak nie oznámi autorovi pridaním komentáru, prípadne aj v komunikačnom nástroji, aké našiel chyby a nedostatky a ten je povinný ich napraviť.
Ak pri Pull-Requeste nastanú kolízie je potrebné informovať o tom integrátora, ktorý ich vyrieši a urobí merge vetiev.

10.3.3 COMMIT

Commit je vlastne nahranie zmien do repozitára, čím vzniká nová revízia v histórií. Všetky zmeny urobené v počítači konkrétneho developera sa zapíšu do úložiska. Každý commit obsahuje správu ktorá pozostáva zo zoznamu zmien vo funkcionalite spolu s ich krátkym opisom, novú

funkcionalitu s opisom a prípadné nezrovnalosti alebo možné budúce chyby a problémy. Zo správy musí byť jasné čo je nové a prečo to bolo vytvorené.

10.3.4 SÚHRN PRAVIDIEL ODOSIELANÉHO KÓDU

Kód, ktorý chce developer pridať do nadradenej vetvy podlieha nesledujúcim pravidlám.

Odosiela sa výlučne:

- fungujúci kód
- otestovaný kód
- okomentovaný kód
- stabilný kód

10.4METODIKA TESTOVANIA

Tento dokument určuje spôsob akým je nutné testovať funkcionalitu pred tým ako sa zmena v projekte schváli a integruje sa do nadradenej branch. Keďže pracujeme na výskumnom projekte, pre ktorý nie je možné efektívne vyvinúť automatické testovanie, je nutné aby testovanie prebiehalo na čo najľahšie testovateľných častiach.

POJMY

- **Unit testovanie** – unit testovanie, alebo testovanie komponentov je testovanie, ktoré sa zameriava na špecifickú časť kódu, resp. konkrétnu funkcionalitu
- **Integračné testovanie** – cieľom integračného testovanie je overenie správnej interakcie jednotlivých modulov systému.
- **White box testovanie** – Testovania softvéru zamerané na vnútornú logiku a štruktúru systému.

10.4.1 ZÁKLADNÉ PRAVIDLÁ

- Testovanie je nutné vykonať pre každý pull request
- Každý programátor je zodpovedný za testovanie svojej pridanej funkcionality
- Je nutné vykonať unit test na pridanú funkcionalitu
- Po unit testoch programátor vykoná integračné testovanie
- Po úspešnom otestovaní môže pokračovať vo vytváraní pull request

Programátor nesmie vytvoriť pull request bez otestovania svojej práce.

10.4.2 PRIEBEH TESTOVANIA

Testovanie prebieha v podobe unit testov, ktoré vykonávajú samotný programátori. Pre každý task z backlog si zodpovedný programátor vytvorí branch. V rámci vytvorenej branch pracuje na tasku a po dokončení programovania a okomentovaní kódu otestuje svoju prácu. Testovanie prevedie hlavne na svoju časť kódu, ale taktiež otestuje celú integritu kódu po vytvorenej zmene.

10.5METODIKA PÍSANIA DOKUMENTÁCIE

Tento dokument slúži ako návod na písanie dokumentácie.

10.5.1 ZÁKLADNÉ POŽIADAVKY

Dokumentácia je vytváraná v nástroji MS WORD(Microsoft Word) od spoločnosti Microsoft, tento nástroj je zahrnutý v balíku MS Office, ktorý musí byť licencovaný. Dokumentácia musí byť písaná v slovenskom jazyku a v prípade anglických výrazov je nutné aby boli v dokumente preložené. Dokumentácia musí byť písaná pomocou diakritiky. V prípade, že dokonale neovládate slovenský pravopis alebo ho ovládať nechcete využívajte pomocný nástroj na kontrolu pravopisu, ktorý je integrovaný v MS WORD.

10.5.2 ZÁKLADNÉ PRAVIDLÁ

- **Font písma** – Times New Roman
- **Veľkosť písma** – 11
- **Farba písma -** čierna
- **Riadkovanie** – 1,5
- **Hlavný nadpis** – na titulný nadpis dokumentu používajte nadpis typu Názov
- **Podnadpis** – na podnadpis dokumentu používajte nadpis typu Nadpis 1
- **Zoznam** – v prípade zvolenia zoznamu využívajte odrážky
- **Obrázky** – Obrázok musí mať vždy priradený popis a musí byť aj správne očíslovaný
- **Diagramy** – Diagram musí mať vždy priradený popis a musí byť správne očíslovaný
- **Tabuľka** – Tabuľka musí mať vždy priradený popis a musí byť správne očíslovaná

- Popis – Popis k obrázkom, diagramom a tabuľkám musí byť v tvare objekt. Číslo – opis
- Umiestnenie popisu – popis musí byť umiestnený vždy pod objektom, v prípade, že sa nezmestí je nutné objekt s obrázkom vložiť na ďalšiu stranu
- Kapitola – každá nová kapitola musí začínať na novej strane, je nutné aby pred ňou bol zlom strany
- V prípade odkazovania sa na nejaký dokument je potrebné aby odkaz obsahoval hyperlinku

10.5.3 SPRÁVA DOKUMENTÁCIE

Každý dokument musí byť nahraný na spoločné úložisko Google Drive, aby bola prístupná aj ostatným členom tímu, Na tomto úložisku sa nachádza finálny dokument, do ktorého bude váš dokument v prípade, že je vyhovujúci zaintegrovaný. Dokument na Drive nazývajte spôsobom čosomspravil_ktotospravil. V prípade nejasností kontaktuje manažéra dokumentácie, ktorý s vami bude problémy riešiť.