

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

Just18

Dokumentácia k inžinierskemu dielu

Vedúci tímu: Ing. Peter Kapec, PhD.

Členovia tímu: Bc. Martin Gašpar

Bc. Michal Knapík

Bc. Tomáš Krupa

Bc. Peter Marušin

Bc. Bence Ligárt

Bc. Miloslav Slížik

Bc. Marek Škriečka

Akademický rok: 2017/2018

Obsah

1	Úvod	4
2	Globálne ciele	6
2.1	Globálne ciele pre zimný semester	6
2.2	Globálne ciele pre letný semester	7
3	Celkový pohľad na systém	7
3.1	Architektúra	7
3.2	Dátový model	8
3.2.1	Stručný opis vybraných entít	9
3.3	Diagram tried	9
3.4	Moduly	9
4	Zimný semester	10
4.1	Aktualizácia závislostí projektu a build projektu	10
4.1.1	macOS	10
4.1.2	Linux	11
4.1.3	WSL	11
4.1.4	Windows	12
4.2	RefaktORIZÁCIA Cmake	13
4.3	OpenPose	14
4.3.1	Úvod	14
4.3.2	Analýza	14
4.4	Vagrant	16
4.5	Provisioning	17
4.6	Continuous integration - návrh	17
5	Letný semester	18
5.1	Include What You Use (IWYU)	18
5.2	Migrácia na GitLab a synchronizácia	19
5.3	RefaktORIZÁCIA logovania projektu	19
5.4	RefaktORIZÁCIA modulu Data	19
5.5	Analýza knižnice sol2	20
5.6	RefaktORIZÁCIA modulu LeapLib	21
5.7	RefaktORIZÁCIA modulu GitLib a ParserLib	21
5.8	RefaktORIZÁCIA modulov LuaGraph a LuaInterface	21

5.9	RefaktORIZÁCIA render balíka.....	22
5.10	Continuous Integration – implementácia	22
6	Prílohy	23
6.1	Inštalačný manuál.....	23
6.1.1	Linux	23
6.1.2	Windows Subsystem for Linux (WSL).....	24
6.1.3	macOS (vytvárané a testované pre verziu 10.13 High Sierra).....	26
6.1.4	Windows.....	28
6.2	Používateľský manuál pre 3DSoftviz	33
6.2.1	Ovládacie prvky	33
6.2.2	Záložka GRAPH	34
6.2.3	Záložka CONSTRAINTS	35
6.2.4	Záložka CLUSTERING.....	38
6.2.5	Záložka CONNECTIONS.....	39
6.2.6	Záložka EVOLUTION.....	40
6.2.7	Záložka MORE FEATURES	41
6.2.8	Hlavné okno.....	45
6.2.9	Git repozitár.....	46

1 Úvod

Predkladaný dokument slúži ako technická dokumentácia k predmetu Tímový projekt na Fakulte informatiky a informačných technológií Slovenskej technickej univerzity v Bratislave. Opisuje štruktúru už existujúceho univerzitného projektu 3DSoftviz na vizualizáciu informácií v obohatenej realite a prácu na danom projekte.

Prvá kapitola dokumentu opisuje globálne ciele pre zimný a letný semester, ktoré sa bude snažiť náš tím naplniť. Vzhľadom na rozsah projektu, stav dokumentácie a hlavne požiadavku na podporu viacerých platforiem sme svoju snahu v zimnom semestri venovali hlavne zmenám v infraštruktúre projektu a údržbe. Snažili sme sa vytvoriť vhodné podmienky na plynulé zavedenie continuous integration. Continuous integration pri budúcom vývoji zautomatizuje proces zostavenia a testovania, no taktiež pomôže monitorovať a urýchliť zavádzanie zmien do projektu. Tiež sme sa snažili o vytvorenie replikovateľného vývojového prostredia, ktoré budúcich vývojárov 3DSoftvizu odbremení od počiatočnej konfigurácie či sťahovania závislostí.

Druhá kapitola obsahuje technický popis stavu projektu v čase, keď sme na projekte začali pracovať. Jednotlivé podkapitoly sú venované architektúre, dátovému modelu, triedam a vzťahom medzi nimi a modulom systému.

Ďalšie kapitoly popisujú prácu na projekte z hľadiska analýzy, návrhu, implementácie a testovania nami modifikovaných častí systému.

Záverečná kapitola dokumentu obsahuje inštalčné manuály pre vývojárov na rôznych platformách a tiež používateľský manuál 3DSoftviz-u. Ten sme prebrali po tímových projektoch z minulých rokov a mierne aktualizovali. Nové Inštalčné manuály obsahujú tiež aktualizovaný zoznam potrebných závislostí a prípadné známe chyby vyskytujúce sa počas konfigurácie aj s popisom riešenia.

Podiel práce na dokumentácii k inžinierskemu dielu

Kapitola	Autor
Globálne ciele pre zimný semester, Aktualizácia závislostí projektu a build projektu - macOS	Peter Marušin
Aktualizácia závislostí projektu a build projektu - Linux	Miloslav Slížik
Aktualizácia závislostí projektu a build projektu - WSL, Formát	Marek Škriečka
Aktualizácia závislostí projektu a build projektu - Windows	Bence Ligárt

Celkový pohľad na systém	Bence Ligárt, Marek Škriečka
Analýza knižnice OpenPose	Martin Gašpar
RefaktORIZÁCIA CMake	Michal Knapík

2 Globálne ciele

2.1 Globálne ciele pre zimný semester

V zimnom semestri sa všetci členovia tímu budú venovať analýze aktuálneho stavu projektu. Na začiatku sa budeme snažiť nakonfigurovať si vývojové prostredia na našich počítačoch, aby sme projekt vedeli spustiť a mohli pracovať na jeho vývoji. Dokumentácia projektu obsahuje inštalčné manuály pre vývojárov vo forme osobitného dokumentu pre každú z troch podporovaných platforiem: Windows, macOS a Linux (Debian-based distribúcie, napr. Ubuntu). Budeme sa pridŕžať existujúcich príručiek, čím zároveň overíme ich relevantnosť a v prípade potreby ich aktualizujeme.

Projekt so sebou prináša nutnosť inštalácie mnohých závislostí a preto je tiež potrebná synchronizácia inštalovaných verzií na všetkých platformách použitých pri vývoji. Projekt stále závisí na knižnici Qt vo verzii 4, no radi by sme prešli na verziu 5 a taktiež sa po počiatočnej analýze pokúsili o aktualizáciu ostatných závislostí. Ak dokážeme aktualizovať závislosti na novšie verzie, budúcim vývojárom uľahčí počiatočnú konfiguráciu (novšie verzie sú ľahšie dostupné v package manageroch systémov macOS a Linux) a projekt si so sebou nemusí niesť už nevyvíjané a nepodporované verzie.

Ďalším cieľom vytvorenie vhodných podmienok na zavedenie continuous integration a continuous development (CI/CD). Pri vývoji multiplatformového softvéru ich dôležitosť narastá - proces zostavovania pre konkrétne platformy či testovania je automatizovaný a tiež je možné monitorovať zostaviteľnosť projektu po vykonaní zmien v repozitári. Projekt je momentálne uložený vo vzdialenom git repozitári na GitHub-e, no v pláne je jeho migrácia na GitLab, ktorý poskytuje natívne mechanizmy pre CI/CD. Podporované je zostavovanie pre všetky 3 platformy. Po odladení do spustiteľnej podoby na všetkých platformách môže byť teda projekt premigrovaný a zavedené CI/CD.

Ďalším cieľom vytvorenie replikovateľného vývojového prostredia s už nainštalovanými závislosťami 3DSoftvizu, nastavenými premennými prostredia a samotným 3DSoftvizom. V tom bude bez ďalšej väčšej konfigurácie možný vývoj projektu. Prostredie chceme vytvoriť pomocou softvéru Vagrant určeného na tvorbu virtuálnych vývojových prostredí prostredníctvom virtualizačných nástrojov ako napr. VirtualBox. Bude predstavovať ďalšiu ľahko použiteľnú alternatívu pre vývojárov, ktorým sa nepodarí projekt rozbehať na svojom počítači.

Dlhodobým cieľom bude aj údržba už existujúceho kódu, refaktorizácia, prečistenie dokumentácie a odstraňovanie nájdených bugov. Práci na projekte bolo venovaných mnoho bakalárskych a diplomových prác či tímových projektov.

2.2 Globálne ciele pre letný semester

Keďže väčšinu zimného semestra sme analyzovali existujúci stav projektu a snažili sa o vytvorenie vývojového prostredia, v letnom sa budeme venovať už iba refaktoringu a oprave nájdených chýb.

Členenie kódu v projekte 3DSoftviz nie je v súlade so žiadnym architektonickým štýlom. Časté sú cyklické závislosti, zle členenie modulov či tried. Jednotlivé časti projektu na sebe silno závisia. Naším hlavným cieľom preto je začať s deľbou kódu do modulov a knižníc a zlepšiť logické členenie jednotlivých častí kódu. Jedná sa o veľa práce, no aj ak nestihneme zrefaktorovať celý projekt, v našej práci môžu pokračovať ďalší študenti.

Chceme tiež zmigrovať projekt na GitLab a zaviesť CI, keďže v zimnom semestri sme to nestihli. Bude potrebné domyslieť aj synchronizáciu vykonaných zmien, keďže na projekte okrem nás pracuje ďalšia skupina študentov v rámci svojich diplomových a bakalárskych prác.

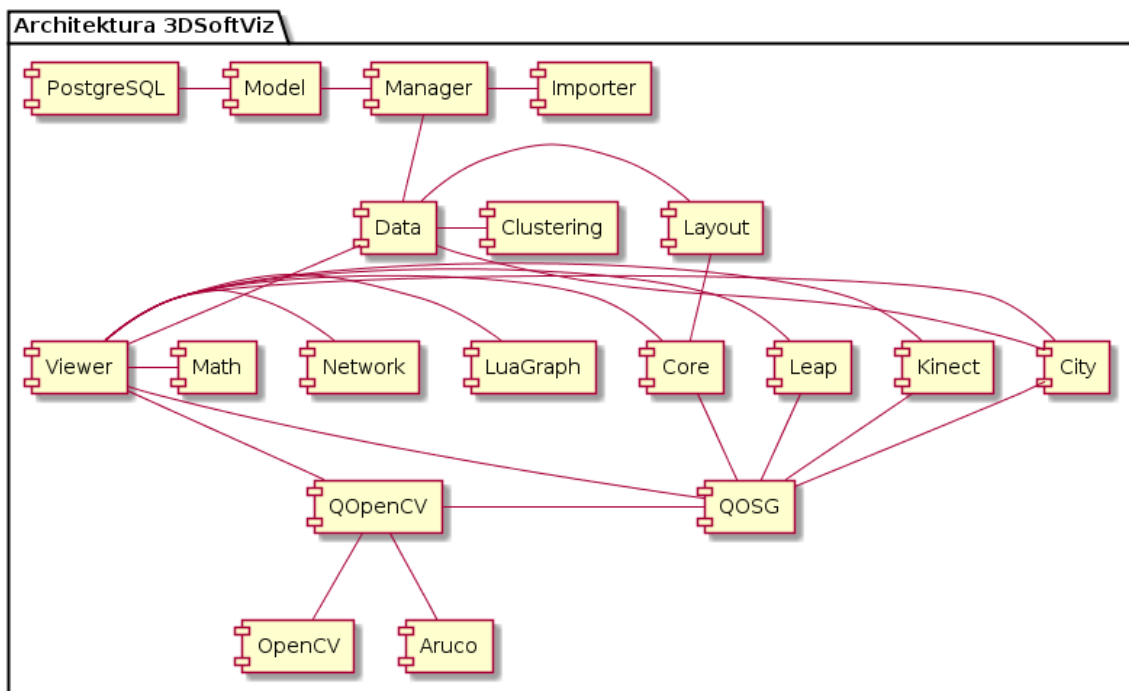
V neposlednom rade budeme pokračovať v refaktorizácii CMake-u, opravovať nájdené bugy či aktualizovať závislosti projektu (rovnako ako počas zimného semestra).

3 Celkový pohľad na systém

Táto kapitola obsahuje celkový pohľad na systém z hľadiska jeho architektúry, dátového modelu, prítomných tried a modulov projektu.

3.1 Architektúra

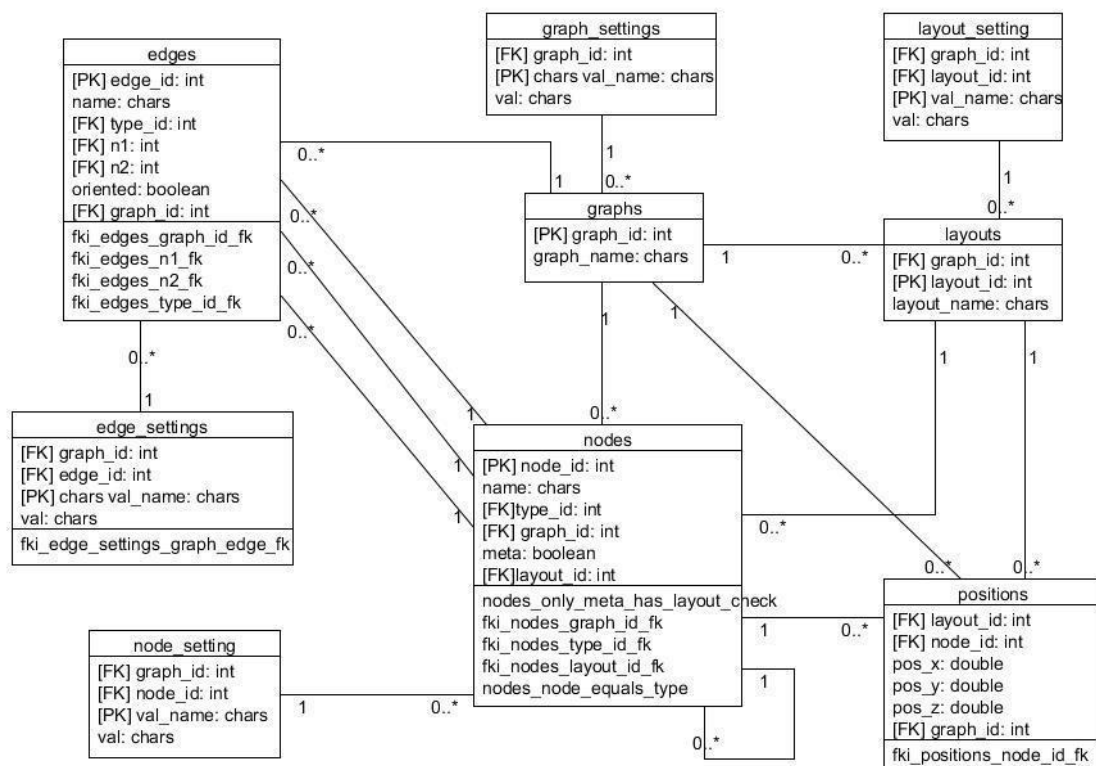
Na obrázku 1 sme znázornili štruktúru projektu 3DSoftViz na začiatku semestra. Projekt v predošlom ročníku bol rozšírený o nový modul, pomocou ktorého projekt už dokáže načítať a vizualizovať nový typ grafu - graf modulov. Do architektúry sa tým pridal modul City, ktorý slúži na vytváranie metafory mesta, kde uzly modulov sa znázorňujú ako hierarchické štruktúry poskladané z regiónov, budov a gúľ. Projekt bol rozšírený aj o ďalšie triedy a zmeny, ktoré ale celkovú architektúru nezmenili.



Obrázok 1: Architektúra projektu

3.2 Dátový model

Dátový model databázy sme prebrali od predchádzajúcich prác, zobrazený je na Obrázok 2.



Obrázok 2: Dátový model

3.2.1 Stručný opis vybraných entít

Graphs – predstavuje jednotlivé záznamy grafov,

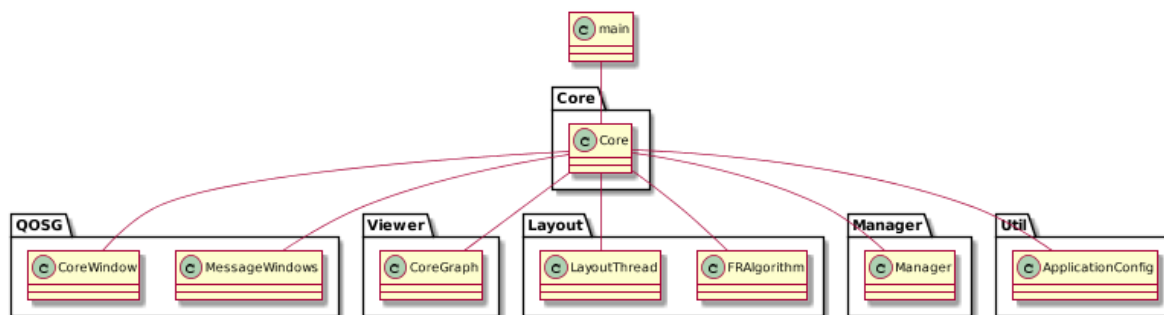
Layouts – obsahuje layout pre daný graf. Predstavuje konkrétne rozloženie množiny uzlov a hrán grafu v priestore,

Nodes – obsahuje uzly a typy v grafe (graf je vytvorený z množiny uzlov poprepájaných hranami, pričom každý uzol má svoj typ),

Positions – obsahuje súradnice uzlov v priestore. Pozostáva z troch súradníc, ktoré sú naviazané na konkrétny uzol a layout,

Edges – má uložené hrany spájajúce uzly v grafe. Hrana môže byť orientovaná a neorientovaná a má začiatkový a koncový uzol a typ.

3.3 Diagram tried



Obrázok 3: Diagram tried

Na diagrame tried sme znázornili najdôležitejšie triedy, ktoré používa trieda main a Core.

3.4 Moduly

Core – obsahuje jadro systému, inicializuje základné časti systému.

Data - dátový modul pre opis štruktúry grafu, obsahujúci triedy reprezentujúce jednotlivé prvky grafu (graph, node, edge, type, layout, ...).

Importer - modul pre parsovanie vstupných súborov vo formátoch GraphML, RSF a GXL.

Layout – modul, ktorý má na starosti rozmiestňovanie uzlov v 3D priestore, taktiež obsahuje implementácie layout algoritmu a triedy pre pridávanie ohraňení rozmiestnenia.

Manager - modul pre prácu s grafom.

Math - model pre rozšírenie práce s kamerou.

Model – modul pre komunikáciu systému s databázou. Funkcionalitou je mapovanie objektov do databázy, vytvorenie spojenia s databázou a základne funkcie výberu a uloženia grafu. Taktiež umožňuje uloženie uzlov aj s ich atribútmi a viacero rozmiestnení pre 1 graf.

Network - modul pre podporu kolaboratívnej práce nad grafom. Poskytuje klient/server funkcionality.

Noise - modul pre vytvorenie generovaného 3D priestoru pre pozadie.

OsgBrowser - modul zahŕňa viazanie udalostí pre jednotlivé klávesy a akcie myši medzi

rozhraniami Qt a OpenSceneGraph a vizualizáciu načítaných grafov.

QOSG – modul pre prácu s grafickými prvkami softvéru. Má na starosti vytvorenie hlavného okna a prácu s pomocnými oknami a widgetami.

Util - zabezpečuje konfiguráciu nastavení aplikácie a funkcie pre vyčistenie pamäte.

Viewer - modul zabezpečuje pohyb v 3D priestore a prácu s kamerou. V module sa tiež pripravuje graf a jeho pre zobrazenie a vytvorenie 3D kocky pre pozadie.

Kinect – modul pre komunikáciu a ovládanie zariadenia Kinect. Medzi jeho funkcionality patrí získavanie informácií a ich nasledovné spracovanie. Obsahuje detegovanie gest, ktoré nahrádzajú ovládanie myšou, otáčanie a pohyb grafu a gestá pre ďalšie ovládanie.

Speech - implementuje funkcionality rozpoznávania hlasu.

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

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

Aruco – obsahuje funkcionality, ktorá vie rozpoznávať značky z kamery použitím knižnice Aruco.

5DTGloves – zabezpečuje detegovanie gesta ruky a vykonávanie korešpondujúcich akcií.

Leap senzor – deteguje dve ruky používateľa v 3D priestore a sleduje pohyby rúk až na úroveň článkov prstov.

3D myš – poskytuje ovládanie kamery pomocou tohto zariadenia.

4 Zimný semester

4.1 Aktualizácia závislostí projektu a build projektu

4.1.1 macOS

Na platforme macOS projekt beží s verziami závislostí:

- OpenSceneGraph 3.5.6
- Qt 5.9.2
- OpenCV 2.4.13
- CMake 3.9.5

Kompilujeme všetci Clangom a projekt sa nám podarilo dostať do spustiteľnej podoby na verziách macOS 10.13 (High Sierra), . Aktualizovali sme Qt zo 4 na 5 a tiež sme projekt rozbehali s takmer najnovším OSG. Pri Qt bolo ale potrebné samostatne si stiahnuť a zostaviť modul osgQt, kt. už vo verzii 5 nie je distribuovaný spolu s OSG. Modul bol neskôr pridaný ako závislosť do repozitára projektu.

Pri počiatkoch testovaní sme sa aj na systéme macOS často stretávali s problémami ohľadom dostupnosti špecifických verzií závislostí.

Zaujímavé v tomto kontexte je hlavne porovnanie systémov macOS s Debian-based distribúciami Linuxu. Oba podporujú správu softvérových balíkov pomocou package management systémov. Repozitáre apt-u sú však centrálné regulované a často ich správcovia pri vydaní novej verzie balíka chvíľu počkajú, kým ju začlenia do ekosystému package manažera (či už z obozretnosti pred nespoľahlivosťou alebo nedostatočnej podpore).

V ekosystéme Homebrew na macOS balíky nie sú až tak striktne regulované, tvorcovia zvolili prístup "bleeding edge" - nové verzie sú často začlenené do package manažera hneď po ich vydaní aj napriek riziku nespoľahlivosti, nedostatočnej podpore či nepripravenosti vývojárov na aktualizáciu. Problém hlavne nastáva, nová verzia nahradí starú a tá už nie je priamo dostupná v package management systéme.

To presne často zaskočilo aj integrátorov na systémoch macOS. Homebrew napríklad počas semestra aktualizoval verziu OpenSceneGraph-u na 3.5.7. 3DSoftviz ale zatiaľ funguje s verziou 3.5.6, ktorá už nie je v Homebrew dostupná. Preto boli niektorí z nás nútení závislosť následne sami získať z oficiálneho git repozitára v správnej verzii a zbuildovať.

Z tohto hľadiska je macOS pri našom projekte ťažšie udržiavateľný ako linuxové distribúcie s apt-om. Komunita apt-u navyše disponuje veľkým množstvom Personal package archives (PPA) - súkromných repozitárov, kde sú často balíky aj vo verziách, ktoré už nie je možné získať z oficiálnych zdrojov.

4.1.2 Linux

Projekt už bol na linuxe funkčný, našou úlohou bolo regresne otestovať projekt s novými verziami knižníc. Konkrétne OpenSceneGraph 3.5.6 a Qt 5.9.1. Prechod z Qt4 na Qt5 si vyžiadaval úpravy v programe pretože knižnica nie je spätne kompatibilná. OpenSceneGraph vo verzii 3.5.6 oddelil osgQt do samostatného repozitára, čo si vyžadovalo úpravu v CmakeList.txt.

Po týchto úpravách je projekt bez problémov spustiteľný.

4.1.3 WSL

Keďže pri buildovaní projektu na platforme Windows vznikali viaceré problémy rozhodli sme sa otestovať novú funkcionálnosť systému - Windows Subsystem for Linux. Tento modul umožňuje za behu systému Windows spustiť systém Linux, konkrétne Ubuntu vo verzii 16.04. Tento subsystem je priamo súčasťou OS Windows a nefunguje ako virtuálny stroj. Po nainštalovaní je k dispozícii terminál, cez ktorý je celý systém ovládaný. Tento systém

podporuje ovládanie len z terminálu, avšak výstup sa dá exportovať na lokálny server, takže s grafickými aplikáciami nie je problém.

Pri inštalovaní a buildovaní projektu sa prišli na nedostatky v podobe podpory oficiálnych závislostí, ktoré projekt 3D softviz potrebuje. Ako tím sme sa dohodli, že projekt aktualizujeme na čo najnovšie knižnice, no WSL malo v repozitári len staršie verzie. Jednou z dôležitých knižníc bola knižnica Qt vo verzii 5, ktorá bola kvôli tomu inštalovaná z externých úložísk.

Projekt sa napokon cez tento systém podaril rozbehnúť a aj nainštalovať. Overením tejto funkcionality sme pridali ďalšiu možnosť inštalácie projektu na operačnom systéme Windows, čo výrazne ušetrí čas, pretože inštalovanie 3DSoftviz na tejto platforme je náchylnejšie na chyby.

4.1.4 Windows

Projekt na windowse sme skúšali rozbehať s rôznymi verziami závislostí a knižníc. Najprv sme postupovali podľa existujúceho inštalačného manuálu, s MSVC 12 (Microsoft Visual Studio 2013) a s aktualizovanými verziami knižníc, pričom všetky komponenty sme mali 32 bitové. Projekt sa nám podarilo úspešne buildnúť a nainštalovať, ale pri spustení projektu vyskytla chyba (chybová hláška: Error: OpenGL version test failed, requires valid graphics context., FATAL [default] CRASH HANDLED; Application has crashed due to [SIGSEGV] signal). Na fórach sme našli potenciálne riešenia na problém, medzi ktorými bola aj zlá verzia Qt, sme ich vyskúšali, ale stále sme sa dostali ku tej istej chybe.

Následne sme prešli na MSCV 14 (Microsoft Visual Studio 2015), pričom knižnice a závislosti sme nechali 32 bitové. Projekt sme úspešne buildli a nainštalovali ale taktiež sa vyskytla chyba pri štarte (The procedure entry point ?defaultConnection@QSqlDatabase@@@2PBDB could not be located in the dynamic link library C:\Timak\3dsoftviz_install\bin\3DSoftviz.exe).

Nakoniec s MSVC 14.11 (Microsoft Visual Studio 2017) sa nám podarilo rozbehať projekt. K úspešnému rozbehaniu projektu sme potrebovali aktualizovať závislosti na nasledujúce verzie:

- Microsoft Visual C++ (MSVC) 14.11 - Microsoft Visual Studio 2017
- CMake 3.9.5
- OpenSceneGraph 3.4.1
- Qt 5.9.2
- OpenCV 2.4.13.4
- Boost 1.65.1

Na Windowse na prácu s projektom v predošlých rokoch používali 32 bitové verzie závislostí. Keďže pri buildovaní projektu podľa starého inštalačného manuálu pre platformu Windows sme mali viac problémov, ktoré neboli spomenuté v inštalačnom manuáli a Microsoft Visual Studio 2013 už nie je voľne dostupný pre študentov fakulty, sme sa rozhodli aktualizovať všetky závislosti na 64 bitové verzie a Microsoft Visual Studio na najnovšiu verziu (2017).

OpenSceneGraph sme našli prebuildovaný s MSVC 14.11 na internete, ale k OpenCV sme našli len zdrojové súbory. Existujúce prebuildované verzie neboli kompatibilné s ostatnými závislosťami. OpenCV sme teda museli buildnúť. Na buildnutú verziu sme potom dali odkaz na stiahnutie do inštalačného manuálu, aby to nemusel každý buildovať.

Qt sme tiež aktualizovali z verzie 4 na 5. QtCreator už sme nemuseli zvlášť inštalovať, pretože na rozdiel od predošlých tímov Qt sme inštalovali cez online inštalátor, ktorý automaticky nainštaluje aj najnovšiu verziu QtCreatora. QtCreator sme teda tiež aktualizovali na aktuálnu najnovšiu verziu.

Problémy boli aj so submodulom libnoise, v ktorom sme tiež museli urobiť niekoľko zmien kvôli novej verzii MSVC. Tak isto sme museli aktualizovať aj moduly Leap a Leap-Orion na najnovšie 64 bitové verzie zo staršej 32 bitovej. K úspešnému buildu a spusteniu projektu sme museli urobiť ešte zmenu v súbore `\src\Clustering\Figures\Sphere.cpp`, tiež kvôli novej verzii MSVC. Poslednú zmenu sme museli urobiť v súbore `\resources\scripts\app\main.lua`, kde sme museli zakomentovať prvý riadok.

Projekt na platforme Windows sa nám nakoniec podarilo rozbehať, ale bolo to časovo náročné. V našom tíme primárne sa bude pracovať na platformách Linux a MacOS a Windows bude slúžiť len ako testovacia platforma, pretože aj pri doterajších zmenách chyby sa vyskytli väčšinou na Windowse.

4.2 Refaktorizácia Cmake

Hlavný CMakeLists.txt súbor je veľký a pomerne neprehľadný, primárne sme sa snažili dostať buildovanie externých dependencií do samostatného CMakeLists súboru. Tento cieľ sa podarilo splniť, no počas vykonávania úlohy sa ukázalo že bude potrebné zjednodušiť aj iné časti hlavného cmake súboru a teda po splnení jednej úlohy nám pribudli ďalšie, a však tie by po dokončení mali ešte viac zjednodušiť prácu s cmake súbormi.

Prenesenie buildovania externých dependencií do samotného cmake súboru sme vykonali pomocou cmake príkazu: `add_subdirectory(dependencies)`, kde parameter predstavuje adresár vrámci hlavnej úrovne projektu. V adresáre sú všetky externe dependencie spolu so svojimi cmake súbormi. Do tohto adresára sa pridal nový CMakeLists.txt súbor kde sú cmake príkazy potrebné pre buildovanie všetkých externých závislostí. Výsledkom je jednoduchšia možnosť práce s buildovaním externých projektov.

Ďalej sme pracovali na presunutí priradovania zdrojových a hlavičkových súborov, tie sa do hlavného CMakeLists.txt súboru zapracovali pomocou príkazov `add_subdirectory(src)` a `include(include/Headers.cmake)`. Vytvorili sme teda nový CMakeLists.txt súbor v podadresári `src/` a nový Headers.cmake súbor v podadresári `include/` kde sa presunuli potrebné cmake skripty z hlavného súboru. Taktiež sa vytvoril CMakeLists.txt súbor v podadresári `tests/` ktorý sa zapracuje do hlavného súboru už spomínaným príkazom `add_subdirectory(tests)`. V rámci refaktorizácie sme sa rozhodli zapracovať do procesu buildovania aj niekoľko nových prvkov, ide o pridanie git informácií ako aktuálna vetva, posledný commit a pod. ktoré sa majú zobrazovať počas procesu buildovania. Ďalej sme pridali možnosť používania nástroja ccache ktorý má za úlohu zrýchliť proces buildovania.

4.3 OpenPose

4.3.1 Úvod

Detekcia prstov na ruke je jedna z najdôležitejších častí projektu a jej správnosť a presnosť predstavuje kľúč k úspechu celého projektu. Rôzne tvary a veľkosti prstov, množina uhlov, v ktorých sa prsty môžu nachádzať a ďalšie faktory znamenajú potrebu implementácie komplexného riešenia, vlastného alebo existujúceho.

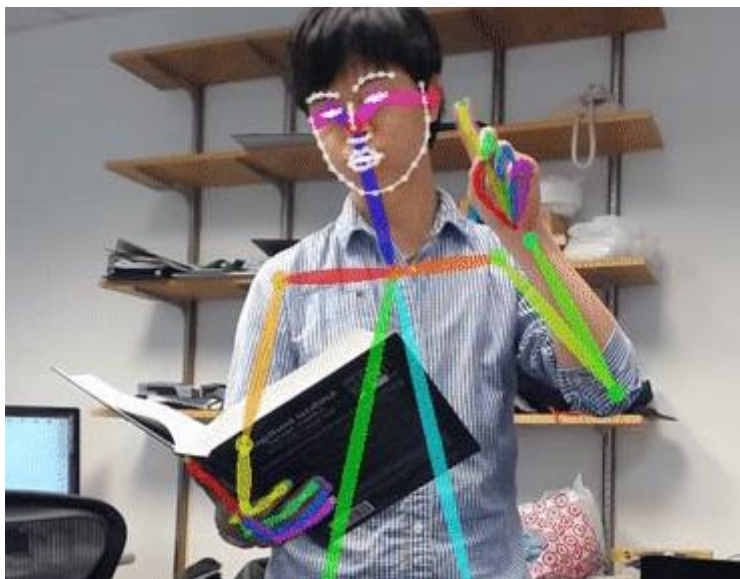
Existencia knižníc s požadovanou funkcionalitou nás môže viesť k predpokladu, že integrácia takejto knižnice/produktu, do nami vyvíjaného systému môže byť efektívnejšia, ako do času tak aj do výsledku, ako vytváranie vlastného riešenia. Rôznorodosť riešení ale so sebou nesie aj rozdiel v použitých technológiách a obmedzeniach s tým súvisiacimi, náročnosti integrácie a požiadavkách na výpočtový výkon, či rozdiel medzi deklarovanými a skutočnými vlastnosťami.

To je dôvod, prečo je každú knižnicu potrebné najskôr analyzovať, otestovať a až potom pristúpiť k jej integrácii do aplikácie resp. zvolenie si cesty vlastného riešenia. Čas, ktorý sa spotrebuje takouto analýzou je rozhodne menší ako ten, ktorý sa minie pri snahe použiť niečo, čo nebude naplňovať naše potreby

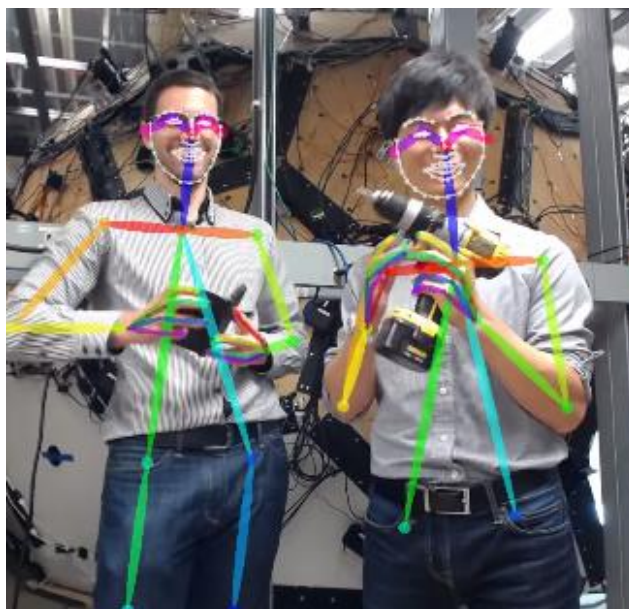
4.3.2 Analýza

OpenPose je nová knižnica pre detekciu kľúčových bodov tela, ruky a tváre v reálnom čase, napísaná v jazyku C++, využívajúca technológie CUDA, OpenCV a Caffé. Jej vznik sa datuje k aprílu tohto roka (2017), čo neprispieva k množstvu informácií, ktoré o nej je možné získať. Jej väčšina je sústredená priamo na GitHub¹, kde sa aj riešia problémy pri inštalácii, nápady na rozšírenie použitia resp. otázky k vývoju v budúcnosti.

¹ <https://github.com/CMU-Perceptual-Computing-Lab/openpose>



Obrázok 4: Detekcia kľúčových bodov tela pri jednej osobe²



Obrázok 5: Detekcia kľúčových bodov tela pri dvoch osobách³

Verzia 1.0.0 vyšla len 8. júla 2017. Súčasná verzia je 1.2.0 a hoci je písaní chystaní update na verziu 1.2.1 žiadne bližšie informácie o dátume alebo obsahu tohto updatu nie sú k dispozícii⁴. Je dostupná zadarmo(pri nekomerčnom použití). Podľa dostupných informácií môže byť na ako zdroj na vstup obrázok, video, webkamera, či IP kamera alebo Kinect⁵, pri

² <http://www.cs.cmu.edu/~yaser/>

³ <http://www.researchcareer.com.au/archived-news/code-released-for-better-body-tracking>

⁴ https://github.com/CMU-Perceptual-Computing-Lab/openpose/blob/master/doc/release_notes.md

⁵ https://github.com/CMU-Perceptual-Computing-Lab/openpose/blob/master/doc/demo_overview.md

ktorom niektorí používatelia riešili aj použitie ako obvyčajnej webkamery⁶. Výstupom je grafická obrazovka, kľúčové body identifikované na vstupnom obrázku/videu vo formáte JSON, XML a alebo renderované video/obrázok už s kľúčovými bodmi vo formáte JPG, PNG, AVI.⁷

Knižnica OpenPose využíva technológiu CUDA, kvôli uplatňovaniu hlbokého učenia. Táto technológia je vlastnená spoločnosťou Nvidia, a je dodávaná výhradne ku grafickým akceleratorom tejto spoločnosti. Nie je v pláne pridanie podpory pre konkurenčné AMD (FurStream)⁸, čo predstavuje určité obmedzenie v oblasti hardvéru potrebného k práci. Tiež podpora medzi operačnými systémami je limitovaná na Windows 8 a 10, Ubuntu 14 a 16 a Nvidia Jetson TX2. Hoci tvorcovia uvádzajú, že knižnica bola použitá aj na iných OS ako napr. macOS, CentOS, oficiálne podporované nie sú⁹. Výhodou ale môže byť, že pre spomínané operačné systémy existujú dobré návody obsahujúce všetky kroky, ktoré treba vykonať pre plnohodnotné využívanie funkcií knižnice. Pre platformu Windows tiež existuje aj demo. Je vhodné spomenúť, že dokumentácia je automaticky generovaná cez doxygen, čo je vhodné vzhľadom na rovnaký spôsob generovania aj nášho projektu

Na stránke projektu je dostupných pomerne veľa informácií¹⁰, ale vzhľadom na dĺžku existencie projektu, informácií z iných zdrojov je veľmi málo, čo prispieva k argumentu, že bez vyskúšania funkcionality nie je možné urobiť kvalifikované rozhodnutie.

Autori deklarujú, že v prípade nájdania videa, kde ich algoritmus nefunguje dobre sú ochotný ho upraviť/vylepšiť a zároveň prosia o nahlásenia chýb alebo pridanie vlastných vylepšení, ktoré by mali význam pre ostatných používateľov. Veľkým problémom, ktorí sa snažia niektorí používatelia zmierniť vlastnými úpravami je rýchlosť resp. výpočtová náročnosť, no napriek pokusom o jej zlepšenie, ak je strata presnosti príliš veľká, napr. rovnaká ako nárast rýchlosti, autori takéto riešenie neimplementujú, hoci niektorým by aj nižšia presnosť pri zvýšení rýchlosti mohla vyhovovať.

Autori deklarujú, že výpočtový výkon potrebný pre detekciu kľúčových bodov je rovnaký bez ohľadu na počet detekovaných ľudí z obrazu.

4.4 Vagrant

Z dôvodu zložitosti nainštalovania všetkých závislostí pre build 3dsoftvizu sme sa rozhodli použiť Vagrant na vytvorenie vývojového prostredia.

⁶ <https://github.com/CMU-Perceptual-Computing-Lab/openpose/issues/189>

⁷ <https://github.com/CMU-Perceptual-Computing-Lab/openpose/blob/master/doc/output.md>

⁸ <https://github.com/CMU-Perceptual-Computing-Lab/openpose/issues/45>

⁹ <https://github.com/CMU-Perceptual-Computing-Lab/openpose/blob/master/doc/installation.md#operating-systems>

¹⁰ <https://github.com/CMU-Perceptual-Computing-Lab/openpose/tree/master/doc>

Vagrant je softvér, ktorý umožňuje vytvorenie a údržbu prenosnej VM. Vstup pre vagrant je Vagrantfile, ktorý obsahuje informácie:

- Aký VM nástroj použiť (VMware, Virtualbox atď.)
- Koľko HW pridelí virtuálnemu stroju
- Aký provisioning má pri prvom spustení vykonať
- Aký OS má použiť

VM sa spustí príkazom: `$ vagrant up`

Prebehne provisioning a používateľ má k dispozícii funkčné vývojové prostredie.

4.5 Provisioning

Provisioning je príprava prostredia, zahŕňa inštaláciu správnych verzií knižníc potrebných pre build 3dsftvizu. Na provisioning využívame Ansible – program, ktorý vykoná provisioning.

Príklad ansible syntaxe pre nainštalovanie gitu pre systém, ktorý využíva APT package manager:

```
- name: install git
  apt: name=git state=present
```

Name - názov úlohy

Apt - zadefinuje, že sa má na inštaláciu použiť apt package manager

Name = git - názov balíku, s ktorým sa bude manipulovať

State = present – balík má byť nainštalovaný po dokončení príkazu

Znalosti získané pri vytváraní provisioningu cez ansible plánujeme využiť pri CI.

4.6 Continuous integration - návrh

S plánovanou migráciou z githubu na gitlab sme plánovali zaviesť aj CI. Idea bola taká, že sa využije .yaml súbor využitý pri provisioningu, ktorý sa rozšíri o príkazy na build. Týmto súborom vytvoríme docker image. Pokiaľ celý proces prebehne bez chyby, tak je program skompilovateľný a môže prebehnúť merge branche na gite.

Pri riešení CI sme narazili na problém, keď gitlab CI runner stále nepodporuje spustenie testu iba pri merge branche, ale iba pri každom commite. Čo nie je pre náš projekt vhodné, pretože provisioning + build trvá cca 30-40 minút.

Čiže CI je pripravené a otestované lokálne. Čaká sa migráciu projektu na gitlab a taktiež nato kým gitlab implementuje, potrebnú funkcionálnosť do ich CI runneru. Čo by malo byť už čoskoro. Pokiaľ sa tak nestane tak budeme uvažovať nad workaroundom.

5 Letný semester

5.1 Include What You Use (IWYU)

V začiatkovej fáze rozčleňovania projektu je potrebné upraviť includnuté knižnice v triedach. Snažili sme sa o voľnejšiu previazanosť – znížiť sa pravdepodobnosť, že trieda je závislá na knižniciach, ktoré nie sú includované priamo ňou. Môžeme odhaliť cyklické závislosti a bude ľahšie existujúce triedy rozdeľovať či presúvať.

Projekt pri zostavovaní používa nástroj Cotire¹¹ (optional). Ten urýchľuje proces zostavenia pomocou techník *Single Compilation Unit* a *Precompiled Header* – počas zostavenia spája všetky triedy do jedného súboru, ktorý je následne skompilovaný rýchlejšie. To ale tiež znamená, že v triedach môžu chýbať potrebné includnuté knižnice a binárku bolo možné zostaviť len vďaka „zlepeniu“ všetkých tried dokopy Cotire-om (vždy sa našla aspoň jedna ďalšia trieda, ktorá obsahovala chýbajúci include).

Na analýzu a následnú úpravu include-ov sme použili nástroj Include What You Use (IWYU)¹². Ten analyzuje jednotlivé symboly (dátové typy, funkcie či makrá) v .cpp/.h súbore a kontroluje, či sú includnuté headre, ktoré symboly poskytujú. Vie odhaliť chýbajúce includey, no tiež zbytočne includnuté headre.

Nástroj je vo verzii alpha – úpravu include-ov nebolo možné automatizovať a prechádzali sme väčšinu tried projektu manuálne. Nástroj poskytuje aj možnosť zadefinovania pragiem¹³, ktorými je možné upraviť výsledky jeho analýzy – možno označiť includey, ktoré označil ako nepotrebné za nutné a naopak.

```
/Users/peto.marusin/VizReal/sources/LeapLib/include/LeapLib/LeapActions.h should add these lines:
namespace Leap { struct Vector; }
namespace LeapLib { class LeapManager; }

/Users/peto.marusin/VizReal/sources/LeapLib/include/LeapLib/LeapActions.h should remove these lines:
- #include "LeapLib/LeapManager.h" // lines 5-5

The full include-list for /Users/peto.marusin/VizReal/sources/LeapLib/include/LeapLib/LeapActions.h:
#include <Leap.h> // for Gesture, Hand
#include "LeapLib/DirectionDetector.h" // for DirectionDetector, DirectionD...
#include "LeapLib/LeapExport.h" // for LEAPLIB_EXPORT
namespace Leap { struct Vector; }
namespace LeapLib { class LeapManager; }
```

Obrázok 6 Ukážka výstupu IWYU pre LeapActions.h

IWYU vo verzii 0.8 bol testovaný na macOS a Linuxe. Boli problémy na macOS 10.13 (High Sierra) s Clangom (asi zapríčinené Apple verziou Clangu). Stále však bolo možné IWYU používať.

Inšpekciu includov sme okrem uvoľnenia previazanosti začali s úmyslom zmenšenia ich celkového počtu (urýchlenie zostavenia). To sa nám však asi nepodarilo. Zmeny v develop

¹¹ <https://github.com/sakra/cotire>

¹² <https://github.com/include-what-you-use/include-what-you-use>

¹³ <https://github.com/include-what-you-use/include-what-you-use/blob/master/docs/IWYUPragmas.md>

vetve činili 792 nových a 550 odstránených riadkov kódu. Aj s includmi navyše sú však triedy menej previazané. Najlepšie by bolo robiť inšpekciu include-ov 3DSoftizu aj naďalej počas celého vývoja. Nástroj je možné aktivovať prepínačom v hlavnom CMake-u projektu. Pri jeho použití musí byť vypnutý Cotire.

5.2 Migrácia na GitLab a synchronizácia

V prvej tretine semestra prebehla migrácia hlavného repozitára z GitHub-u na GitLab (príprava na nasadenie CI). Výhodou GitLab-u sú tiež privátne repozitáre aj pri free účte – je možné rozšíriť kód o časti, pri ktorých si vlastníci nepreje ich zverejnenie.

Tímový projekt, diplomanti aj bakalári majú separátne repozitáre. Synchronizáciu zmien sme riešili vytvorením synchronizačnej vetvy v každom repozitári. Synchronizácia prebiehala raz týždenne (piatok). Dovedy sme sa snažili vyriešiť všetky čakajúce pull-requesty. Následne sme zosynchronizovali synchronizačnú vetvu s aktuálnou verziou develop vetvy a vytvorili pull request *sync vetva TP -> sync vetva DP/BP*. Synchronizáciu je možné robiť obojsmerne. Sync vetva by mala po synchronizácii obsahovať rovnaké commity vo všetkých repozitároch, ktoré pracujú na 3DSoftize, čo aj platilo. Tento spôsob synchronizácie sa v praxi celkom osvedčil.

5.3 RefaktORIZÁCIA LOGOVANIA PROJEKTU

Pri analýze refaktORIZÁCIE sme zistili, že vývojári na projekte 3DSoftviz nedodržiavali konvencie a metodiku správneho logovania. Navyše na logovanie využívali rôzne metódy z rôznych knižníc a niekedy logy nedávali zmysel a boli tu zanechané len ako pomocné výpisy.

Preto sme pristúpili na prepísanie všetkých logovacích správ (ich počet bol na jednotný formát podľa metodiky a využili sme pri tom jednotne knižnicu *easylogging*, ktorá je dostupná pre C++. Tie, ktoré sa ukázali ako nepotrebné pre projekt a potreby odhaľovania chýb sme zmazali.

Pri logovaní teraz využívame všeobecnú štruktúru:

```
LOG( WARNING/ERROR/FATAL ) << "MENO_BALIKU/MENO_TRIEDY/MENO_FUNKCIE(PARAMETRE)"
```

5.4 RefaktORIZÁCIA MODULU Data

Hlavným cieľom zrefaktarovania modulu Data bude jeho oddelenie do samostatnej knižnice ktorá nemá žiadne závislosti na zvyšok 3Dsoftviz, ani na externé knižnice. Výsledkom bude tiež zlepšená hierarchia tried a odstránenie nepotrebných súčastí modulu Data.

RefaktORIZÁCIA bude prebiehať vo viacerých fázach. V prvej fáze sa vytvorí hierarchia tried pre entitu Graph. Táto hierarchia zníži komplexnosť zdrojového kódu a umožní väčšiu modularitu celkového projektu.

V ďalšej fáze sa budeme zaoberať odstraňovaním interných závislostí na iné triedy v rámci projektu 3Dsoftviz, ako aj externých knižníc. Pri analýze sme zistili, že niektoré metódy používali smart pointer z knižnice OSG, pričom nevykonávali žiadnu logiku spojenú s touto knižnicou. Tento pointer sa dá nahradiť jedným zo smart pointrov, ktoré sa nachádzajú v balíku std.

V ďalšej fáze chceme odstrániť závislosť modulu Data na DAO triedach. Tieto činnosti by sa mali vykonávať v inom moduli, keďže slúžia na načítanie a ukladanie dát do databázy. Naším cieľom bude ich presunutie do namespaceu GraphManager.

V poslednej fáze sa budeme zaoberať s odstránením zbytočnej previazanosti entity Type na triedu ApplicationConfig, cez ktorú číta nastavenia. Tento nedostatok napravíme tak, že všetky nastavenie budú posilať v konštruktoe pri vytváraní nových inštancií tejto entity.

Výsledkom refaktORIZÁCIE tohto modulu je:

- Bolo odstránené používanie DAO tried z modulu.
- Bola vytvorená nová hierarchie pre entitu Graph.
- Podarilo sa nám odstrániť previazanie na ApplicationConfig z väčšiny z tried z modulu. Previazanie ostalo len v entite Graph a Vizgraph. Toto priviazanie sa tiež dá odstrániť.
- Smart pointre neboli nahradené, keďže sme zistili, že nedajú na jednoducho nahradiť obyčajným smart pointrom.
- Boli odstránené ďalšie zbytočné previazania a nepoužívané funkcie z modulu.

5.5 Analýza knižnice sol2

Knižnica Sol2 slúži rovnako ako Diluculum na získavanie dát z Lua scriptov, tzv. binding pre C++. Ide go-to framework, s minimálne podľa dokumentácie, ľahko použiteľným API.

V podstate sa jedná o header-only knižnicu. Je zabezpečená podpora Lua 5.1, 5.2 a 5.3 a LuaJIT (Just-In-Time-Compiler for Lua) a tiež C++14. Z Githubu projektu je zrejmé, že na projekte sa stále pracuje, ale informácie o budúcich verziách tam nie sú. Poskytuje tutoriál, ktorý obsahuje predovšetkým príklady z krátkych kódov s minimom textu. Viac je možné zistiť až z dokumentácie, ktorá by mohla byť napísaná aj prehľadnejšie.

Sol2 podporuje základné konštrukcie ako je podpora stavov, tabuliek, nastavovanie resp. získavanie hodnôt rôznych typov. Tabuľka predstavuje hlavný bod interakcie medzi jazykom Lua a C++. K dispozícii sú dva druhy tabuliek, globálne a neglobálne, no obe poskytujú rovnaký interface a všetky globálne tabuľky sú konvertovateľné na neglobálne. Je k dispozícii aj operator[] proxy.

Sol2 dáva k dispozícii aj CMake script, avšak dokumentácia uvádza, že je primárne určený na testovacie účely. Pri problémoch so CMake súborom nie je poskytovaná plná podpora, nakoľko ide o výtvor dobrovoľníka.

Vytváranie objektov funguje cez deklaráciu `sol2::`. Volanie funkcií cez `protected_function` slúži nielen na odchytyvanie errorov a ale aj ich spracovanie. Podľa dokumentácie akýkoľvek C++ alebo Lua error by mal byť odchytený a spustený cez voliteľný `error_handler`.

Stránka knižnice obsahuje aj informácie o možných erroroch, ktoré sa pri jej používaní môžu vyskytnúť, uvádza problém s LuaJIT na Mac OS X, problém s Visual Studio množstvo ďalších.

5.6 Refaktorizácia modulu LeapLib

Hlavným cieľom pri refaktorizácii tohto modulu bolo odstrániť z neho všetky nepotrebné závislosti a tým umožniť jeho osamostatnenie od 3dSoftvize. Momentálne je tento modul používaný už ako samostatná knižnica s vlastným namespace bez závislosti na 3dSoftvize. Pri oddeľovaní tejto novo vzniknutej knižnice bolo potrebné vytvoriť v `Softviz::Leap` balíku novú triedu `Listener`, ktorá dedí z triedy `LeapLib::LeapListener`, pretože nebolo možné odstrániť všetky závislosti z pôvodnej `LeapLib` a tak bolo najvhodnejším riešením niektoré časti rozdeliť do viacerých vrstiev.

5.7 Refaktorizácia modulu GitLib a ParserLib

Modul `GitLib` bol oddelený od časti `Repository/Git` do samostatnej knižnice bez akýchkoľvek závislosti na 3dSoftvize, spolu s ním boli presunuté aj jeho testy.

Modul `Importer::Parsing` bol oddelený do samostatnej knižnice, čím sa opäť podarilo o čosi zmenšiť veľkosť samotného 3dSoftvize. Táto novo vzniknutá knižnica má na starosť parsovanie zdrojových súborov jazyka Java. Do tejto knižnice linkujeme externú knižnicu *parserlib*, výsledná knižnica je používaná samotným 3dSoftvizom.

5.8 Refaktorizácia modulov LuaGraph a LuaInterface

Cieľom bolo oddeliť zdrojové kódy týkajúce sa tried `LuaGraph` a `LuaInterface` do samostatných znovupoužiteľných celkov, bez spätných závislostí na hlavnú aplikáciu či iné nesúvisiace moduly. Z oboch modulov vznikli samostatné verziované repozitáre, ktoré sú zahrnuté do projektu ako git submodule. Sú plne samostatne zostaviteľné a inštalovateľné, no hlavne schopné zahrnutia do akéhokoľvek iného projektu. Všetky závislosti projektu sú vyriešené možnosťou vyhľadania v súborovom systéme alebo samostatným zostavením. Oba moduly boli pokryté testami, jednotkovými aj integračnými. V rámci modulu `LuaInterface` sa znovupoužil BDD testovací framework `Igloo`. V module `LuaGraph` sa experimentálne integroval testovací framework `Catch2`. Tento testovací framework je modernejší, poskytuje

väčšiu funkcionálnosť a je v aktívnom vývoji, a v rámci ďalšieho refaktoringu a aktívneho pokrývania projektu testami úplne nahradí framework Igloo.

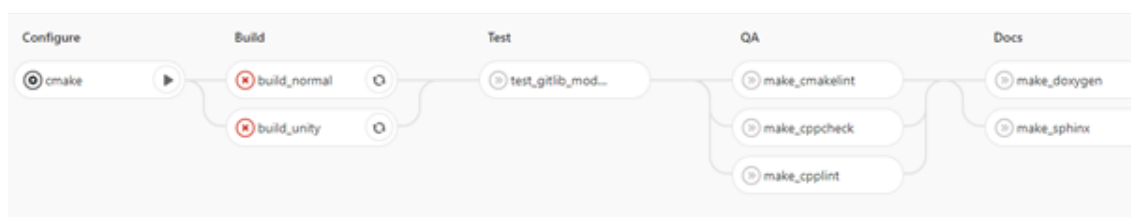
5.9 RefaktORIZÁCIA render balíka

Primárnou úlohou bolo vytvoriť samostatný nový balík a knižnicu a osamostatniť tak z 3dSoftviz-u ďalšie moduly. Namespace City a Shapes boli presunuté do tohto balíka, taktiež trieda ResourceManager. Bolo nutné odstrániť závislosti na 3dSoftviz, konkrétne na ApplicationConfig a Layout. Týmto prístupom do balíka pribudla trieda LayoutAlgorithms a naopak trieda Module bola presunutá naspäť do 3dSoftviz-u nakoľko jej previazanie nebolo možné rozumným spôsobom zredukovať. Nový balík (knižnica) má aj nový vlastný namespace.

5.10 Continuous Integration – implementácia

V letnom semestri sme implementovali CI pre 3dsoftviz. Po migrácii projektu na gitlab sme analyzovali možnosť využitia nástroja GitlabCI, ktoré Gitlab ponúka. CI pipeline, ktorá sa nám zdala najvhodnejšia pre náš projekt sa skladá z 5 častí a vyzerá nasledovne:

1. Configure – v tomto kroku sa vykonáva iba 1 úloha – cmake príprava na build, ktorá sa následne uloží a v ďalších fázach sa použije. Týmto sa získala časová úspora cca. 30 sekúnd pre každý nasledujúci job.
2. Build - v tomto kroku testujeme obidve možnosti buildu nášho projektu
3. Test – vykonávajú sa unit testy pre knižnice, ktoré ich majú napísané. V našom prípade ide iba o 1 knižnicu GitLib.
4. QA - krok v ktorom sa testuje kvalita a úprava kódu. Používame nástroje cppcheck, cpplint a cmakelint.
5. Docs – generujeme doxygen a sphinx dokumentáciu projektu.



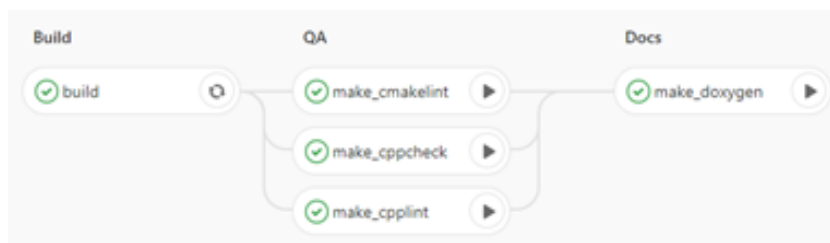
Obrázok 7: CI pipeline 3dsoftviz

Pretože náš projekt je komplexný a jeho build trvá pomerne dlho sme boli nútení čo najviac zrýchliť CI pipeline, pretože gitlab ponúka iba 2000 minút mesačne a zo začiatku trvalo priemerné zbehnutie celej pipeline cca 90 minút.

Pre zrýchlenie sme použili ccache, ktorý slúži ako cache pre kompilátor a takisto sme vytvorili vlastný docker image, v ktorom sú predinštalované všetky závislosti. Ako základ pre docker image sme použili Ubuntu rovnako ako pri vagrante, aby sa v budúcnosti čo najviac

zjednodušila ich údržba. Po implementácii spomenutých zlepšení sa nám podarilo skrátiť beh CI pipeline na menej ako 60 minút.

Počas semestra sa nám podarilo pri refaktoringu vybrať z 3dsoftvizu 2 submoduly LuaInterface a LuaGraph. Následne sme pre tieto moduly vytvorili CI pipeline, ktorá vyzerá veľmi podobne ako pre 3dsoftviz a pokrýva build, generovanie dokumentácie a QA kódu. Pipeline pre obidva moduly vyzerá zhodne:



Obrázok 8: CI pipeline LuaInterface a LuaGraph

6 Prílohy

6.1 Inštalačný manuál

6.1.1 Linux

Názvy knižníc sú ukázané pre Ubuntu 17.10

1. Cez Apt nainštalovať nasledujúce programy
 - Git - git
 - OpenSceneGraph - libopenscenegraph-dev //update na verziu 3.4.0
 - Qt5 - qt5* // update na qt 5.9.1
 - Qt5WebEngine - qtwebengine5
 - QtCreator - qtcreator
 - Boost - libboost-all-dev
 - Lua - liblua5.1-0-dev
 - OpenCV - libopencv-dev
 - FreeGlut3 - freeglut3-dev
 - CMake - cmake // verzia 3.9

Program je otestovaný aj s OpenSceneGraph verziou 3.5.6 , ale tá nie je momentálne v Ubuntu repozitári a ani v PPA, čiže treba build zo source kódu

Príkaz na inštaláciu závislostí : `$ sudo apt install git,libopenscenegraph-dev,qt5*,qtwebengine5,qtcreator,libboost-all-dev,liblua5.1-0-dev,libopencv-dev,freeglut3-dev,cmake`

1. Klonovať cez Git projekt 3dsoftviz
 - git clone ** url repozitára **

- `git submodule update --init --recursive`
- 2. Zbuildovanie projektu
 - Otvoriť QtCreator
 - File >> Open file or project >> Otvoriť CmakeList.txt v zložke 3dsoftviz
 - Vytvoriť projekt
 - Ak úspešne (dá sa kliknúť na FINISH) >> Kliknúť FINISH
 - Ak neúspešne – Nájsť chybu vo výpise
- 3. Nastaviť build v QtCreator-i
- 4. Projects >> Build and Run >> Build Build Settings >> Add >> Clone Selected >> pomenovať "unity" - automaticky prepne na unity build mode Build Steps >> Details >> zaškrtnúť `install_unity`
- 5. Klonovať cez Git projekt 3dsoftviz (AlphaReach klonuje z repozitára Cimox)
 - `git clone ** url repozitára **`
 - `git submodule update --init --recursive`
- 6. Nastavenie počtu jadier na buildovanie projektu... Details >> Additional arguments "-jN", kde N reprezentuje počet VIRTUÁLNYCH jadier

6.1.2 Windows Subsystem for Linux (WSL)

Inštalácia WSL

Note:

Inštalácia WSL je možná len pre 64-bitovú verziu Windows 10 a build 1607+

1. Spustiť **PowerShell** ako **administrátor** a spustiť nasledujúci príkaz:
`Enable-WindowsOptionalFeature -Online -FeatureName Microsoft-Windows-Subsystem-Linux`
2. Spustiť **CMD** a spustiť:
`bash`
3. Potvrdiť inštaláciu a nastaviť meno a heslo
4. WSL nepodporuje defaultne GUI aplikácie, no ich obraz sa dá exportovať na XServer:
 - nainštalovať Xlaunch (alebo iný)
 - v nastaveniach nastaviť port výstupného displeja
 - v terminály zadať `export DISPLAY=:port` (ten ktorý sme nastavili v Xlaunch)

Inštalácia projektu

1. Nainštalovať prostredníctvom Synaptic nasledujúce programy
 - **Git** - git
 - **OpenSceneGraph** - libopenscenegraph-dev

- **Boost** - libboost-all-dev
- **Lua** - liblua5.1-0-dev
- **OpenCV** - libopencv-dev
- **FreeGlut3**
 - freeglut3
 - freeglut3-dev
- **CMake** [3.5.0](#)
 - inštalovať zo source files, nie Synaptic

2. Nainštalovať Qt5 cez ppa (cez Synaptic je dostupná len verzia 4):

```
sudo add-apt-repository ppa:beineri/opt-qt591-xenial
sudo apt-get update
sudo apt-get install qt59-meta-full
```

3. Pre správne fungovanie je potrebné nastaviť flag pre spustiteľnosť knižníc Qt príkazom:

- `sudo execstack -c /opt/qt59/lib/*`
- v súčasnej verzii execstack nie je dostupný, treba ho najskôr nainštalovať

4. Klonovať cez Git projekt 3dsoftvizQt

- `git clone ** url repozitára **`
- `git submodule update --init --recursive`

5. Prepnúť sa do aktuálnej vetvy:

- `git checkout vetva`
- `git submodule update --init --recursive`

6. Zbuildovanie projektu

- Otvoriť QtCreator príkazom:

```
sudo ./opt/qt59/bin/qtcreator
```

- V QtCreator File >> Open file or project >> Otvoriť CmakeList.txt v zložke 3dsoftviz
- Vytvoriť projekt
 - Ak úspešne (dá sa kliknúť na FINISH) >> Kliknúť FINISH
 - Ak neúspešne – Nájsť chybu vo výpise

7. Nastaviť build v QtCreator-i

Projects >> Build and Run >> Build Build Settings >> Add >> Clone Selected >> pomenovať "unity" - automaticky prepne na unity build mode Build Steps >> Details >> zaškrtnúť install_unity

Nastavenie počtu jadier na buildovanie projektu... Details >> Additional arguments "-jN", kde N reprezentuje počet VIRTUÁLNYCH jadier

8. V Tools >> Options >> Build & Run >> QtVersions je potrebné cez *Add* pridať /opt/qt59/bin/qmake

9. V Tools >> Options >> Build & Run >> Kits je potrebné nastaviť verziu Qt5.9..

10. Spustiť buildovanie

11. Po buildovaní v Projects >> Run >> Add >> Custom Executable nastaviť

Executable: {project_dir}/_install/bin/3DSoftviz

Working directory: {project_dir}/_install/bin

12. Spustiť projekt

6.1.3 macOS (vytvárané a testované pre verziu 10.13 High Sierra)

1. Inštalácia Homebrew príkazom:

```
/usr/bin/ruby -e "$(curl -fsSL  
https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

2. Nainštalovať git:

```
brew install git
```

3. Nainštalovať OpenCV - MUSÍ byť verzia 2.x, nie 3 a vyššia! (po inštalácii skontrolovať príkazom `brew info opencv@2`, vyhovujúca verzia je napr. 2.4.13.4):

```
brew install opencv@2
```

4. Vytvoriť symbolické linky pre opencv@2:

```
brew link opencv@2 --force
```

5. Nainštalovať OpenNI2:

- možné pomocou Homebrew nasledovne:

- `brew tap homebrew/sciencebrew tap totakke/openni2brew install openni2`

- dostupné [na stiahnutie v Google Drive](#)

6. Nainštalovať Boost:

```
brew install boost
```

7. Nainštalovať CMake:

```
brew install cmake
```

8. Nainštalovať Qt (preferovaná verzia 5.9.2):

```
brew install qt
```

9. Vytvoriť symbolické linky pre Qt:

```
brew link qt --force
```

8. skontrolovať verziu a symbolické linky pre Qt príkazom: `qmake --version`. Ak je výstup `command not found`, skontrolovať v Homebrew: `brew info qt`. Ak je Qt nainštalované, no nevieme nájsť `qmake`, Homebrew nevytvoril správne symbolické linky. Skúsiť `brew unlink qt` a znova `brew link qt --force`, prípadne `reinstall`

9. Nainštalovať sphinx:

```
pip install sphinx
```

alebo

```
brew install sphinx
```

10. Nainštalovať astyle:

brew install astyle

11. Nainštalovať pcl:

brew install pcl

12. Nainštalovať doxygen:

brew install doxygen

13. Nainštalovať cppcheck:

brew install cppcheck

14. Nainštalovať ovládače pre 3D myš dostupné na odkaze alebo v Google Drive projektu.

15. Nainštalovať Qt Creator:

brew cask install qt-creator

16. Naklonovať projekt:

git clone <https://github.com/BergiSK/3dsoftviz> --recursive

17. Otvoriť Terminál, cd do priečinka ~/3dsoftviz a aktualizovať submoduly:

git submodule update --recursive

18. Otvoriť QtCreator a otvoriť CmakeLists.txt z priečinka, kde bol naklonovaný projekt.

Nastavenie Qt Creator:

- cmd + , pre otvorenie nastavení
- skontrolovať záložku Qt Versions, ak je dostupné Qt 5.x, nechať tak. Ak nie, treba nájsť binárku qmake (defaultne po inštalácii Qt cez Homebrew v /usr/local/bin/).
- skontrolovať záložku CMake. Ak nie je detekovaný žiadny, treba nájsť binárku cmake (defaultne po inštalácii CMake cez Homebrew v /usr/local/bin/).
- ísť do záložky Kits.
- Zvoliť konfiguráciu a vpravo kliknúť na Clone
- Kliknúť na novú konfiguráciu (dole sa otvorí editačné okno)
- Name zvoliť napr. unity
- C aj C++ Compiler zvoliť Clang
- Qt version zvoliť Qt 5.x
- CMake Tool zvoliť na CMake zo záložky CMake
- Vpravo hore kliknúť na Make Default
- Potvrdiť tlačidlom OK

Build & Run konfigurácia:

- V ľavom menu Qt Creator kliknúť na Projects
- Kliknúť na Build pri unity
- Edit build configuration zvoliť na Debug
- Build directory zvoliť na ~/3dsoftviz/build
- Pri položke Build steps kliknúť na Details a zaškrtnúť install_unity
- Kliknutím na Build project (kladivo) projekt zbuildovať
- Naľavo kliknúť na Run pri unity
- Pri Run configuration rozbaľiť Add menu a zvoliť Custom Executable

- Zvoliť binárku 3dsoftviz/_install/bin/3DSoftviz
- Working directory nastaviť na 3dsoftviz/_install/bin

Známe problémy, riešenia a rady

Q: Build error `"/usr/local/./mkspecs/macx-clang"`

CMake Error at /usr/local/lib/cmake/Qt5Core/Qt5CoreConfig.cmake:15 (message): The imported target "Qt5::Core" references the file `"/usr/local/./mkspecs/macx-clang"`

A: Chyba pravdepodobne je na strane Homebrew, pri inštalácii Qt5 nevytvorí dve symlinky (ani po zavolaní brew link). Musíte ich vytvoriť ručne (ak treba, upravte verziu v ceste, návod bol robený pri nainštalovanej 5.9.2):

```
ln -s /usr/local/Cellar/qt/5.9.2/mkspecs /usr/local/mkspecs
ln -s /usr/local/Cellar/qt/5.9.2/plugins /usr/local/plugins
```

Ak príkaz zlyhá, pretože symlinky už existujú, zmažte ich a skúste znova.

6.1.4 Windows

Tento návod bol úspešne otestovaný na operačnom systéme Windows 10 Pro.

Na inštaláciu potrebujeme klonovať projekt 3DSoftViz (Tím č.18 klonuje z /pmarusin/3dsoftviz) z Githubu a stiahnuť nasledovné: Nasledujúce programy neinštalovať do Program Files, PATH nesmie obsahovať medzeru!:

- Microsoft Visual Studio 2017 (môže byť nainštalovaný štandardne do Program Files)
- CMake (v3.9.5)
- OpenSceneGraph (v3.4.1) (jeden z nasledujúcich)
 - Full pack
 - Release + Debug + Knížnice 3rd party dependencies VC14.1
- Qt (v5.9.2) a Qt Creator (v4.4.1) - cez Qt online installer
- OpenCV (v2.4.13.4) (jeden z nasledujúcich)
 - Built - buildnuté
 - Source - iba zdrojové súbory, treba buildnúť
- Boost (v1.65.1)
- RapidEE - program na prácu s premennými prostredia
- Kinect for Windows SDK 1.8
- OpenNI2 (v2.2)
- NiTE (v2.21)
- Debugging Tools for Windows 10 - WinDbg - Win 10

Postup inštalácie:

Uvedené programy neinštalovať do Program Files, PATH nesmie obsahovať medzeru!

Ideálne všetky programy, knížnice, atď. dať do jedného adresára. Ideálne je mať všetko na spoločnom mieste kvôli prehľadnosti, napr. C:/Timak/

1. Nainštalovať Microsoft Visual Studio 2017 (môže byť nainštalovaný štandardne do Program Files) spolu aj s rozšíreniami:

1. VC++ 2017 v141 toolset
2. Windows 10 SDK for Desktop C++
3. Visual C++ tools for CMake
4. Visual C++ ATL support
5. MFC and ATL support
6. C++/CLI support
2. Nainštalovať CMake. (Cesta je v dokumente označená ako %CMAKE_DIR%). Pri inštalácii zvoliť "Add CMake to the system for all users".
3. Nainštalovať Qt a Qt Creator (*mal by sa nainštalovať automaticky s Qt*):
 1. Nainštalovať Qt online installer (%QT_INSTALLER_DIR%)
 2. Spustiť MaintenanceTool.exe > Skip > Add or remove components
 3. Zvoliť z Qt 5.9.2 nasledujúce veci:
 1. msvc2017 64-bit
 2. všetko od Qt Charts po Qt Script
 4. Zvoliť z Tools Qt Creator 4.4.1 CDB Debugger Support
 5. Nainštalovať zvolené položky (%QT_INSTALLER_DIR%/5.9.2/msvc2017_64)
4. Vytvoriť zložku OpenSceneGraph (%OSG_DIR%)
5. V prípade stiahnutia Full pack:
 1. Rozbaliť zložky do %OSG_DIR%
 2. Vynechať nasledujúci krok.
6. V prípade stiahnutia Release + Debug + Knižnice 3rd party dependencies VC14.1:
 1. Vytvoriť zložku Release v zložke %OSG_DIR%
 2. Rozbaliť do nej zložky zo stiahnutého Release
 3. Vytvoriť zložku Debug v zložke %OSG_DIR%
 4. Rozbaliť do nej zložky zo stiahnutého Debug
 5. Vytvoriť zložku 3rdParty v zložke %OSG_DIR%
 6. Rozbaliť do nej zložky zo stiahnutého 3rd party dependencies VC14.1
7. V prípade stiahnutia zbuildovaných súborov OpenCV
 1. Rozbaliť zložky do %OPENCV_DIR%
 2. Vynechať nasledujúci krok.
8. V prípade stiahnutia iba zdrojových súborov OpenCV
 1. Rozbaliť OpenCV do %OPENCV_DIR%/source
 2. Vytvoriť zložku install a build v %OPENCV_DIR%
 3. Spustiť CMake (cmake-gui.exe)
 1. source code > %OPENCV_DIR%/source
 2. binaries > %OPENCV_DIR%/build
 3. nastaviť 3rdparty:
 - stlačiť Add Entity
 - zadať Name: 3rdparty
 - zvoliť Type: PATH
 - zadať Value: %OPENCV_DIR%/source/3rdparty

- stlačiť OK
- 4. stlačiť Configure
- 5. nastaviť generator na Visual Studio 15 2017 Win64
- 6. stlačiť Finish
- 7. čakať na výpis
- 8. nastaviť CMAKE_INSTALL_PREFIX na %OPENCV_DIR%/install
- 9. stlačiť Generate
 - ak došlo k erroru: File > Delete Cache a skúsiť znovu
- 4. Nájst súbor OpenCV.sln v %OPENCV_DIR%/build
- 5. Otvoriť súbor vo VS2017 **(ako správca!)**
- 6. Nastaviť Solution Configuration na Debug, Solution Platform na x64
- 7. Nájst projekt ALL_BUILD > pravý klik > build
- 8. Po skončení nájsť projekt INSTALL > pravý klik > build
- 9. Nastaviť Solution Configuration na Release
- 10. Nájst projekt ALL_BUILD > pravý klik > build
- 11. Po skončení nájsť projekt INSTALL > pravý klik > build
- 12. Presunúť nainštalované súbory zo zložky install do %OPENCV_DIR%
- 13. Zložky install, build a sources môžu byť vymazané z %OPENCV_DIR%
- 9. Nainštalovať Boost (%BOOST_DIR%)
- 10. Otvoriť súbor environment.txt a upraviť v ňom cesty k programom a knižniciam, ktoré sa nachádzajú na začiatku súboru.
 1. Spustiť powershell ako správca
 2. Skopírovať do powershellu obsah celého súboru
 3. Vynechať pridávanie systémových premenných cez RapidEE (nasledujúci krok) a vykonať len kontrolu, či sa cesty správne nastavili.
- 11. Nainštalovať a otvoriť RapidEE, v ktorom sa vykonajú tieto zmeny **(ako správca!)**:
 1. do PATH pridať premenné:
 1. %CMAKE_DIR%/bin
 2. %QT_INSTALLER_DIR%/5.9.2/msvc2017_64/bin
 3. %QT_INSTALLER_DIR%/Tools/QtCreator/bin
 4. %OSG_DIR%/Debug/bin
 5. %OSG_DIR%/Release/bin
 6. %OSG_DIR%/3rdParty/x64/bin
 7. %OPENCV_DIR%/x64/vc15/bin
 2. Vytvoriť premennú CMAKE_INCLUDE_PATH a pridať:
 1. %OSG_DIR%/Debug/include
 2. %OSG_DIR%/Release/include
 3. %OSG_DIR%/3rdParty/x64/include
 4. %OPENCV_DIR%/include
 3. Vytvoriť premennú CMAKE_LIBRARY_PATH a pridať:
 1. %OSG_DIR%/Debug/lib

2. %OSG_DIR%/Release/lib
3. %OSG_DIR%/3rdParty/x64/lib
4. %OPENCV_DIR%/x64/vc15/lib
4. Vytvoriť premennú BOOST_INCLUDEDIR a pridať: %BOOST_DIR%/boost
5. Vytvoriť premennú BOOST_LIBRARYDIR a pridať: %BOOST_DIR%/libs
6. Vytvoriť premennú BOOST_ROOT a pridať: %BOOST_DIR%
7. Vytvoriť premennú OPENCV_DIR a pridať: %OPENCV_DIR%
12. Nainštalovať WinDbg.

Inštalácia projektu:

1. Naklonovať projekt 3DSofViz cez git shell (%3DSofViz%)
2. Cez command line prejsť do naklonovaného projektu a zavolať *git submodule update --init --recursive*
3. Vytvoriť v priečinku %3DSofViz% priečinky _build a _install
4. Spustiť QtCreator. Tools > Options... > Build and Run:
 1. záložka CMake – ak je nainštalovaný CMake, tak auto-detected, inak pridať manuálne %CMAKE_DIR%/bin/cmake.exe
 2. záložka Compilers – ak existuje VS2017, tak sú auto-detected
 3. záložka Qt Versions – ak je nainštalovaný Qt, tak auto-detected, inak zadať cestu %QT_INSTALLER_DIR%/5.9.2/msvc2017_64/bin/qmake.exe
 4. záložka Debuggers – ak je nainštalovaný WinDbg, tak sú auto-detected, inak pridať manuálne C:/Program Files (x86)/Windows Kits/10/Debuggers/x64/cdb.exe
 5. záložka Kits – vytvoriť nový a nastaviť hodnoty nasledovne:
 1. Name: Local PC
 2. Device type: Desktop
 3. Compiler: C: Microsoft Visual C++ Compiler 15.0 (amd64)
 4. Compiler: C++: Microsoft Visual C++ Compiler 15.0 (amd64)
 5. Debugger: Auto-detected CDB at C:/Program Files (x86)/Windows Kits/10/Debuggers/x64/cdb.exe
 6. Qt version: Qt 5.9.2 MSVC2017 64bit
 7. CMake Tool: System CMake at %CMAKE_DIR%/bin/cmake.exe
 6. záložka General – nastaviť Default build directory: %3DSofViz%/_build
 7. Potvrdiť – OK
5. File > Open File or Project... > vybrať CMakeLists.txt z %3DSofViz%
6. Skontrolovať výpis v časti 6 General Messages, na konci výpisu musí byť
7. Generating done
8. CMake Project was parsed successfully.
9. Vybrať Projects > Build & Run > Local PC > Build, v časti Edit build configuration kliknúť na Add > Clone selected, nazvať „unity“

10. Prejsť na vytvorený build config. „unity“, v časti Build Steps otvoriť Details a vypnúť pri build step Build: cmake.exe --build . --target možnosť *all* a označiť *install_unity*
11. Stlačiť Build (kladivo vľavo dole - potrebné spraviť znova po každej následnej úprave systémových premenných)
12. Po úspešnom zbuildovaní vybrať Projects > Build & Run > Local PC > Run, v časti Run pridať Add > Custom Executable a nastaviť:
 1. Executable: %3DSoftViz%/_install/bin/3DSoftviz.exe
 2. working directory: %3DSoftViz%/_install/bin/
13. Spustiť program pomocou zeleného tlačidla Run (vľavo dole)

V prípade, že aplikácia ihneď po spustení crashne, napriek úspešnému buildu, jedná sa pravdepodobne o problém s grafickou kartou. Na notebookoch, ktoré majú externú grafickú kartu NVidia, je v tomto prípade treba cez NVidia Control Panel nastaviť jej použitie pre 3DSoftViz.exe

Rozšírenie 3DSoftviz o Kinect

1. Nainštalovať Kinect for Windows
2. Skontrolovať v RapidEE či sa vytvorila premenná %KINECTSDK10_DIR%, keď nie, vytvoriť a pridať: C:/Program Files/Microsoft SDKs/Kinect/v1.8
3. Nainštalovať OpenNI2 (%OPENNI2_DIR%)
4. Skontrolovať v RapidEE či sa vytvorili premenné (ak na konci majú 64, vymazať 64):
 1. %OPENNI2_INCLUDE%, keď nie, vytvoriť a pridať: %OPENNI2_DIR%/Include/
 2. %OPENNI2_LIB%, keď nie, vytvoriť a pridať: %OPENNI2_DIR%/Lib/
 3. %OPENNI2_REDIST%, keď nie, vytvoriť a pridať: %OPENNI2_DIR%/Redist/
 4. %OPENNI2_ROOT%, keď nie, vytvoriť a pridať: %OPENNI2_DIR%
5. Nainštalovať NiTE2 (%NITE2_DIR%)
6. Skontrolovať v RapidEE či sa vytvorili premenné (ak na konci majú 64, vymazať 64):
 1. %NITE2_INCLUDE%, keď nie, vytvoriť a pridať: %NITE2_DIR%/Include/
 2. %NITE2_LIB%, keď nie, vytvoriť a pridať: %NITE2_DIR%/Lib/
 3. %NITE2_REDIST%, keď nie, vytvoriť a pridať: %NITE2_DIR%/Redist/
 4. %NITE2_ROOT%, keď nie, vytvoriť a pridať: %NITE2_DIR%
7. Pridať do premennej CMAKE_INCLUDE_PATH:
 1. %OPENNI2_INCLUDE%
 2. %NITE2_INCLUDE%
8. Pridať do premennej CMAKE_LIBRARY_PATH:
 1. %OPENNI2_ROOT%/Driver
 2. %OPENNI2_REDIST%
 3. %OPENNI2_REDIST%/OpenNI2/Drivers
 4. %OPENNI2_LIB%
 5. %NITE2_ROOT%/Samples/Bin/OpenNI2/Drivers
 6. %NITE2_LIB%

9. Pridať do premennej PATH:
 1. %OPENNI2_REDIST%/OpenNI2/Drivers
 2. %OPENNI2_REDIST%
 3. %NITE2_REDIST%
 4. %NITE2_ROOT%/Samples/Bin
10. Spustiť CMake a skontrolovať vo výpise:
 1. OpenNI2 FOUND
 2. NITE2 FOUND
 3. KINECTSDK FOUND

6.2 Používateľský manuál pre 3DSoftviz

Okno s aplikáciou je rozdelené na tri základné časti:

- menu
 - File – načítanie grafu zo súboru, z databázy, uloženie grafu, uloženie layoutu, ukončenie aplikácie
 - Settings – nastavenia aplikácie; konfiguračný súbor používa bodkovú notáciu, ktorá umožňuje identifikovať význam konfiguračnej premennej
 - Help
 - Test - pušťa základné grafy pre rýchle testovanie (100-uzlový, 500-uzlový, Veolia, Lua Graph, Module Graph)
- hlavné okno - zobrazuje graf a umožňuje s ním používateľovi interagovať
- ovládací panel – nástroje pre prácu s grafom

6.2.1 Ovládacie prvky

Ovládanie kamery:

- Vyber prvkov grafu - ľavé tlačidlo myši + pohyb myšou
- Otáčanie kamery okolo grafu - pravé tlačidlo myši + pohyb myšou
- Ovládanie priblíženia obrazovky - koliesko na myši
- Ovládanie pomocou klávesnici:
 - Hore - PgUp
 - Dole - PgDn
 - Vľavo - šípka vľavo
 - Vpravo - šípka vpravo
 - Dopredu - šípka hore
 - Dozadu - šípka dole

Inicializácia automatického pohybu začne po stlačení kláves Alt + Shift a kliknutím myši na zvolenú hranu, či uzol. V závislosti od nastavenia aplikácie sú pred inicializovaním pohybu ešte automaticky vybrané body záujmu. Pokiaľ je automatický výber uzlov vypnutý, body záujmu je možné zvoliť manuálne myšou alebo stlačením klávesy Q (pre náhodný výber uzlov alebo pre výber uzlov pomocou metrík). Automatické použitie metrík je možné vypnúť v

nastavení aplikácie pomocou parametra „Viewer.PickHandler.SelectInterestPoints“ nastaveného na hodnotu 1.

Iné ovládacie prvky:

- Kláves "T" – skrytie všetkých ovládacích prvkov
- Kláves "S" - štatistiky vykresľovania
- Kláves "Shift" - pridávanie ďalších objektov do výberu
- Kláves "Ctrl" - odstránenie objektov z výberu

6.2.2 Záložka GRAPH



- manipulácia s prvkami grafu (no-select mód), pohyb vybraných uzlov v priestore



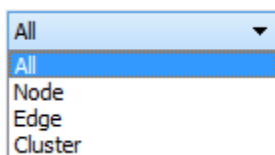
- výber jedného prvku grafu (single-select mód)

Umožňuje sústredenie sa na práve jeden objekt – môže to byť hrana aj uzol.



- výber viacerých prvkov grafu (multi-select mód)

Umožňuje vybrať v trojrozmernom zobrazení viacero objektov naraz.



- typ výberu: všetko, iba uzly, iba hrany, klastre



- centrovanie pohľadu vzhľadom na vybraný prvok grafu

V prípade, že nie je označený žiadny element, kamera bude vycentrovaná na ťažisko grafu



- pridanie meta uzla do grafu



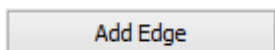
- odstránenie vybraných meta uzlov z grafu



- ukotvenie vybraných uzlov na aktuálnej pozícii, t. j. nebudú sa pohybovať v závislosti od pôsobenia síl ostatných uzlov



- uvoľnenie ukotvených uzlov



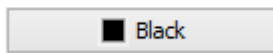
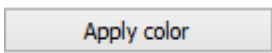
- pridanie hrany medzi dvoma vybranými uzlami

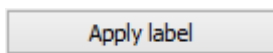
Umožňuje pridať hranu medzi dvoma vybranými uzlami, kde ešte nie je hrana, inak končí akcia chybovou hláškou. Ideálne je čiernou šípkou vybrať jeden uzol a bielou šípkou ho nastaviť na také miesto, kde sa ho dá spojiť s druhým uzlom - je potrebné mať nastavenie Node v takomto prípade spolu s Multi-select mode.

 - pridanie uzla do stredu pohľadu



 - odstránenie vybraných elementov (uzly alebo hrany)


Ak sa rozhodneme pre zmazanie hrany, uzly prepojené s touto hranou v grafe zostávajú.


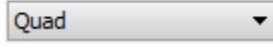
  - zafarbenie zvolených uzlov a hrán farbou vybranou z palety nad tlačidlom

 - aplikovanie textového označenia na vybrané uzly podľa textu z poľa nad tlačidlom

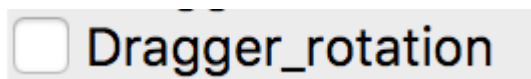
 - zapnutie/vypnutie zobrazovania popisov uzlov a hrán

  - spustenie/zastavenie rozmiestňovania (animovania) uzlov grafu

 - zmena odpudivých síl pôsobiacich medzi uzlami

  - výber vizuálnej reprezentácie uzla (square, sphere) a výber vizuálnej reprezentácie hrany (quad, cylinder, line)

- pridanie nového manipulátora (geometria kocky) do scény, ktorý umožňuje pohyb kamery ľavým tlačidlom myši

 - pridanie nového manipulátora (geometria gule) do scény, ktorý umožňuje rotáciu grafu okolo jeho stredu ľavým tlačidlom myši


 - reprezentácia grafu ako vizualizačnej metaforu mesta (namiesto klasického grafu).

6.2.3 Záložka CONSTRAINTS

 - aplikovanie priestorového ohraničenia: povrch gule

 - aplikovanie priestorového ohraničenia: obsah gule

 - aplikovanie priestorového ohraničenia: rovina

 - aplikovanie priestorového ohraničenia: zjednotenie gule a roviny

Zjednotenie gule a roviny je vhodné pre zobrazenie grafov s hustým stredom, alebo na veľké grafy.

 - aplikovanie priestorového ohraničenia: kružnica

Aplikovanie obmedzenia na kružnicu na uzly v celom grafe je vhodné pre veľmi riedke grafy alebo na grafy s pravidelnou štruktúrou. Pri hustých grafoch sa hrany medzi uzlami prekrývajú

 - aplikovanie priestorového ohraničenia: kužeľ

Obmedzenie na kužeľ je vhodným riešením v prípadoch, kedy má jeden uzol výrazne vyšší počet hrán ako ostatné uzly.

 - aplikovanie priestorového ohraničenia: kužeľový strom

Po aplikácii sa uzly rozdelia do skupín podľa spoločného rodiča. Na tieto skupiny sa aplikujú obmedzenia na kužeľ, ktoré sú následne obmedzené na roviny v závislosti od hĺbky uzlov v strome. Kužeľový strom sa aplikuje automaticky na celý graf na základe používateľom vybraného koreňového uzla. Jedine v prípade, že graf nie je spojitý, tak sa aplikuje iba na komponent, ktorý obsahuje koreňový uzol.


 - odstránenie vybraných priestorových ohraničení

 25  - aplikovanie priestorového ohraničenia: povrch valca

Vloží do scény tzv. bod záujmu, od ktorého sú uzly zobrazovaného grafu odtláčané do tvaru valca. Polomer valca sa dá nastaviť pomocou číselníka.

 25  - aplikovanie priestorového ohraničenia: povrch kužeľa

Vloží do scény tzv. bod záujmu, od ktorého sú uzly zobrazovaného grafu odtláčané do tvaru kužeľa. Polomer kužeľa sa dá nastaviť pomocou číselníka. Veľkosť kužeľa sa nastavuje automaticky podľa toho, kam sa používateľ prostredníctvom kamery pozerá.

 - aplikovanie radiálneho rozmiestnenia na označené uzly

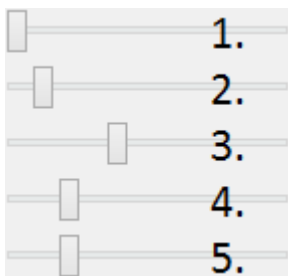
Odporúča sa používať pri stromovom type grafu. Použitie rozmiestnenia na označené uzly dáva používateľovi nové možnosti ako zväčšenie priestoru označením uzlom, alebo manuálne zhlukovanie uzlov.



- výber módu vykreslenia radiálneho rozmiestnenia (drôtený, plný)

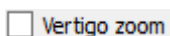


- nastavenie módu 2D/3D radiálneho rozmiestnenia



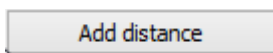
1. nastavenie veľkosti rozmiestnenia
2. nastavenie priehľadnosti rozmiestnenia
3. nastavenie počtu zobrazených gúľ
4. nastavenie faktora zosilnenia odpudivých síl v radiálnom rozmiestnení pre uzly, ktoré nie sú na rovnakej vrstve
5. nastavenie faktora zosilnenia odpudivých síl v radiálnom rozmiestnení pre uzly, ktoré sú na rovnakej vrstve

Veľkosť radiálneho zobrazenia sa dá nastaviť v rozmedzí 0 – 300, parameter priehľadnosti 0 - 100 %, veľkosť faktora zosilnenia odpudivých síl sa nastavuje medzi hodnotami 1 - 5000.

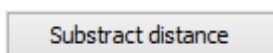


- prepínač medzi normálnou a vertigo kamerou

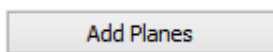
Tento mód kamery je vhodné použiť vtedy, keď chce používateľ meniť dva rôzne pohľady na graf: lokálny pohľad, pri ktorom môže používateľ s väčšou presnosťou skúmať jednotlivé uzly a vzťahy medzi nimi a globálny pohľad, pri ktorom môže používateľ skúmať vzťahy medzi uzlami a rozloženie uzlov v daných hĺbkach kostry grafu v globálnom kontexte.



- zvýšenie vzájomnej vzdialenosti medzi rovinami




- zníženie vzájomnej vzdialenosti medzi rovinami



- pridanie dvoch paralelných rovín

Obmedzenie na roviny sa aplikuje pri grafoch s minimálnou maximálnou hĺbkou kostry grafu hodnoty 2. Koreňový uzol v kostre grafu určí program - vyberie uzol s najväčším počtom hrán. Pri zrušení obmedzenia sa uzly „odpoja“ od roviny.

 - odobranie dvoch paralelných rovín

 - zmena násobiča odpudivých síl medzi uzlami

Násobič odpudivých síl medzi uzlami je na začiatku nastavený na 1 kvôli prvému pridaniu dvoch rovín do priestoru - nechceme, aby sa hneď zväčšili odpudivé sily.

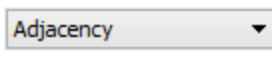
 - vypnutie všetkých predchádzajúcich obmedzení


6.2.4 Záložka CLUSTERING

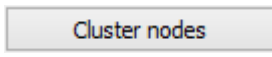
 - zlúčenie vybraných uzlov

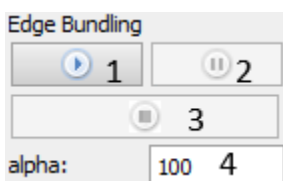
Umožňuje zlúčiť vybrané uzly do jedného spoločného uzla. Takýto uzol sa bude v pokračovaní zobrazovať modrou farbou.

 - zrušenie zlúčenia vybraných uzlov

 - definovanie algoritmu, ktorým sa bude zhľukovať graf (adjacency, leafs, neighbours)

Depth:  - nastavenie počtu rekurzií pre vybraný algoritmus

 - spustenie zhľukovania nad aktívnym grafom
Ak zhľukovanie trvá viac ako 1 sekundu, objaví sa indikátor postupu.



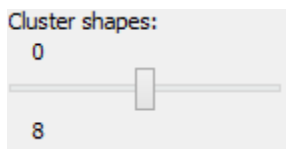
1. spustenie algoritmu na zväzovanie hrán
2. pozastavenie algoritmu na zväzovanie hrán
3. úplne zastavenie algoritmu na zväzovanie hrán a zobrazenie pôvodného grafu
4. vstupné pole na zadanie konštanty, určujúcej silu akou sú hrany k sebe počas zväzovacieho algoritmu priťahované

Po použití funkcie zhľukovania, sa odkrývajú nasledujúce možnosti:

Opacity:
☐ auto
☐ selected
auto - automatická priehľadnosť - mení sa na základe vzdialenosti zhľukov od kamery
selected - priehľadnosť označeného zhľuku – pomocou posuvníka(nižšie) sa mení priehľadnosť len označených zhľukov



- posúvaním upravíme priehľadnosť označených zhlukov

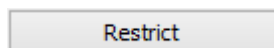


- posúvaním sa mení prahová hodnota, pri ktorej sa menia tvary

zhlukov

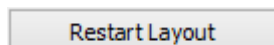
Spodné číslo udáva, koľko uzlov obsahuje daný zhluk (v tomto prípade 8).

Pri označení konkrétneho zhuku sa odkrývajú nasledujúce možnosti:

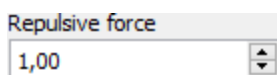


- kliknutím zmeníme označený zhluk na obmedzovač

Obmedzuje pozície uzlov tak, aby z neho nevyšli von. Keď obmedzovač posunieme dostatočne ďaleko, t.j. mimo pôvodnej pozície uzlov, uzly sa začnú lepiť na jeho stenu a posúvať spolu s ním. Ignoruje príťažlivé a odpudivé sily medzi ním a ostatnými uzlami grafu (posunutie zhuku bez obmedzovača spôsobí posun celého grafu za týmto zhukom). Obmedzovač začína svoje pôsobenie ako kocka, je možné zmeniť jeho tvar natáhovaním a stlačením.



- znovurozmiestnenie uzlov v priestore po tom, ako sa nalepia na hranu obmedzovača



- upravenie odpudivej sily medzi uzlami v označenom zhuku

Čím je hodnota väčšia, tým budú uzly ďalej od seba.

Ďalšie funkcie obmedzovača:

Ak na zhluk zaregistrujeme obmedzovač, môžeme s ním jednoducho pohybovať a meniť jeho tvar pomocou klávesových skratiek a myši:

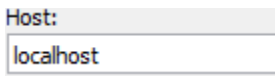
- Pohyb – metóda ťahaj a pušť (drag & drop)
- Zmena veľkosti – držíme **Ctrl** a točíme kolieskom myši
- Zmena tvaru
 - na osy x – držíme **X** a **Ctrl** a točíme kolieskom myši
 - na osy y – držíme **Y** a **Ctrl** a točíme kolieskom myši
 - na osy z – držíme **Z** a **Ctrl** a točíme kolieskom myši

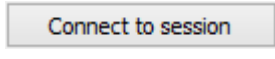
6.2.5 Záložka CONNECTIONS

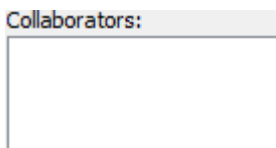


- napísanie mena, pod ktorým bude používateľ vystupovať v kolaborácii

 - spustenie/zastavenie servera

Host:
 - napísanie IP adresy servera

 - pripojenie(odpojenie) ku(od) kolaborácii

Collaborators:
 - zoznam používateľov (zoradený abecedne), v ktorom je možné jedného vybrať a použiť nasledujúce funkcie:

☐ Spy
☐ Center
☐ Shout - po výbere si môžeme zvoliť jednu funkciu z dvojice: *Spy* (špehovať) a *Center* (centrovať).

Po aktivovaní funkcie *Spy* získa používateľ pohľad iného používateľa, ktorý je priebežne aktualizovaný – znamená to, že pohybom sledovaného používateľa sa aktualizuje aj pohľad sledujúceho. Po aktivácii *Center* nasmeruje pohľad používateľa tak, aby v jeho strede bol iný používateľ. Pri centrovaní platí to isté, čo pri špehovaní – teda pri aktualizácii polohy centrovaného používateľa sa natáča aj pohľad centrujúceho používateľa. Po označení políčka *Shout* sa ostatným používateľom v scéne zobrazí pri vašom mene ikona znázorňujúca, že sa pokúšate upútať pozornosť.

Avatar scale
 - nastavenie veľkosti avatarov v scéne

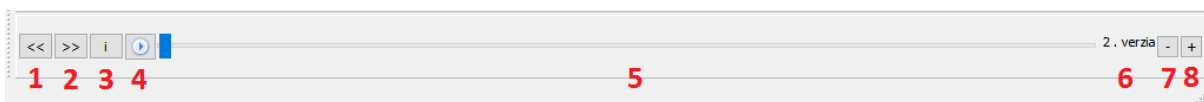
Avatar je kužeľ, ktorého kruhová podstava znázorňuje smer, ktorým sa používateľ pozerá.

6.2.6 Záložka EVOLUTION

- Po rozkliknutí tabu *Evolution* (1) sa zobrazia možnosti evolúcie

1. *Lifespan* - možnosť ponechania vymazaných uzlov vo vizualizácii. Prednastavená hodnota 0 znamená, že vymazané uzly sa automaticky vymažú z grafu. V prípade hodnoty väčšej ako 0 vymazané uzly v grafe zotrvávajú o verzie dlhšie podľa nastavenej hodnoty
2. *Change commits* - prepínač spracovania Git repozitáru. Ak je zaškrtnutý, inicializuje sa spracovanie na úroveň grafu volaní. V opačnom prípade - na úroveň histórie Git repozitáru
3. Kombo box s výberom vizualizácie - prepínanie sa medzi jednotlivými možnosťami vizualizácie grafu volaní

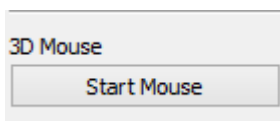
- *LuaStat* - vizualizácia softvérových metrík pomocou analýzy Lua zdrojového kódu
 - *Difference* - pohľad na zmeny, ktorými softvér prešiel pri prechode na novú verziu
 - *Changes* - aktivovanie filtrovania nad práve aktívnou vizualizáciou
4. Kombo box s výberom filtra - výber vhodnej skupiny filtra
 - Prednastavená možnosť *All* - všetky prvky grafu sú zobrazené
 - *Authors* - filtrovanie podľa autorov zmien v softvéri
 - *Structure* - filtrovanie podľa štruktúry
 5. Kombo box zoznamu možností - možnosti závisia od vybraného filtra
 - zoznam autorov s možnosťou zobrazenia zmien všetkých autorov - *All*
 - štruktúra - *Files* (zobrazí v grafe volaní len uzly reprezentujúce adresáre a súbory), *Local Functions* (zobrazí rozšírenú možnosť Files spolu s uzlami lokálnych funkcií), *Global Functions* (zobrazia sa uzly možnosti Local Functions spolu s uzlami globálnych funkcií) a *Modules* (zobrazí všetky štruktúry, ktoré sa v grafe nachádzajú)



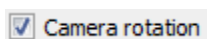
- panel ovládania evolúcie

1. Prechod na predchádzajúcu verziu - možnosť, kedy sa stav grafu vráti o jednu verziu dozadu
2. Prechod na nasledujúcu verziu - možnosť, kedy sa stav grafu posunie o jednu verziu dopredu
3. Tlačidlo informácií o verzii - zobrazí informácie o aktuálne zobrazenej verzii. Medzi zobrazené informácie patrí identifikátor, autor a dátum commitu spolu so zoznamom súborov, ktoré boli zmenené
4. Spustenie/zastavenie animácie - aktivovanie/zastavenie automatického prechodu na novú verziu
5. Posuvník - presun na konkrétnu verziu pomocou skokového prechodu medzi verziami
6. Indikátor verzie - poskytuje informáciu o aktuálne zobrazenej verzii
7. Spomalenie animácie - regulovanie rýchlosti animácie
8. Zrýchlenie animácie - regulovanie rýchlosti animácie

6.2.7 Záložka MORE FEATURES



- zapnutie 3D myšky (musí byť aktivovaný driver)



- ak je zaškrtnuté, kamera nasmerovaná na graf sa pohybuje na základe pohybu tváre, značky alebo rúk, inak sa na základe týchto akcií rotuje samotný graf

☒ Camera enabled - povoľuje použitie kamery

Start camera - otvorenie okna pre prácu s kamerou

Start Speech - otvorenie okna pre prácu s hlasovým ovládaním

Note

Speech je momentálne vylúčený z projektu

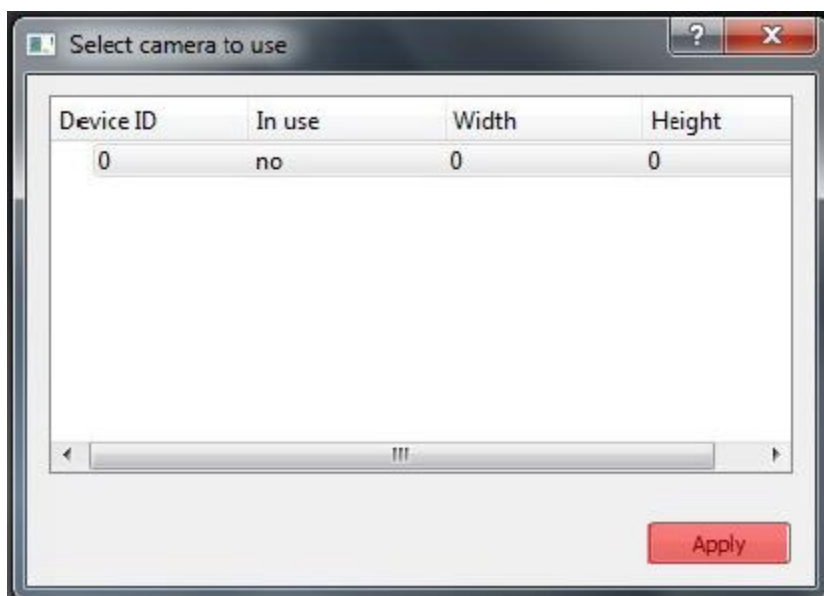
Start Leap - zapnutie ovládania pomocou Leap Senzor-u

Okno pre prácu s kamerou

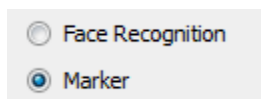
☒ Face Recognition
☐ Marker - prispôsobenie ľavej strany okna pre ovládanie funkcionality rozpoznávania tváre (pri zapínaní treba zaškrtnúť Camera rotation a Camera enabled).

Start Face Recognition - zvolenie kamerového zariadenia a následným potvrdením objavenie záberu z kamery

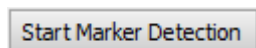
Ukončiť túto akciu je možné tlačidlom „StopFaceRec“ (ak používateľ zatvoril okno, môže ho vrátiť na grafický interface opätovným kliknutím na „StartCamera“ a potom pozastaviť detekciu). V prípade detegovanej tváre (detekcia je reprezentovaná zeleným obdĺžnikom) sa kamera alebo graf pohybuje vďaka pohybu tváre.



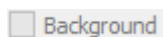
- okno pre výber snímacieho zariadenia



- prispôsobenie ľavej strany okna pre ovládanie funkcionality rozpoznávania značky

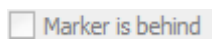


- zvolenie kamerového zariadenia a následným potvrdením objavenie záberu z kamery určenej pre rozpoznávanie značky a graf sa začne otáčať a pohybovať so značkou



- nastavenie aktuálne snímanie ako pozadie pre graf

Je potrebné zmeniť parameter „Viewer.SkyBox.Noise“ v konfiguračnom súbore na hodnotu 2 alebo 3 (odporúčané je 3).



- prepínanie medzi pohybom podľa značky ako keby sa kamera pozerala na používateľa a naopak

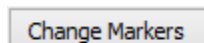


- zapnutie korekcie

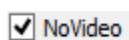


- nastavenie korekčných parametrov

Podľa predvolených nastavení sa značka pohybuje tak, ako keby sa kamera pozerala vo vodorovnom smere. Ak by sa pozerala napr. na stôl pod miernym sklonom dole, graf by sa pri posúvaní značky po stole neposúval korektne. Preto je možné nastaviť korekčné parametre. Najskôr je potrebné nastaviť značku do polohy, kedy je detegovaná na spodnom okraji a následne stáčiť toto tlačidlo. Po nastavení sa aktivuje opcia „Correction“ (uvedené vyššie), ktorou je možné zapnúť korekciu.



- zmena spôsobu použitia značky v prípade, že používateľ má k dispozícii len jednu značku



- vypnutie/zapnutie zobrazenia videa

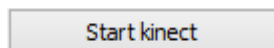
Toto prepínanie a vypnutie zobrazenia video má vplyv len na zobrazenie v rámci tohto ovládacieho okna „Face Recognition“ and „Marker Detection“ a neovplyvňuje to ani voľbu kamery pre video pozadie.



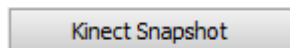
- Interakcia s vizualizáciou v obohatenej realite

- Možnosť *Custom light*, vyznačená modrou, ktorá slúži na prepínanie vlastného a základného zdroja svetla
- Možnosť *Shadow*, vyznačená žltou, ktorá slúži na zapínanie a vypínanie generovania tieňov
- Možnosť *Base*, vyznačená červenou, ktorá slúži na zobrazenie a skrytie základne
- Možnosť *Axes*, vyznačená ružovou, ktorá slúži na zobrazenie a skrytie pomocných osí
- Tlačidlo *Center graph*, vyznačené svetlo modrou, ktoré slúži na umietnetie grafu nad stred základne

Okno pre prácu s kinectom a arucom



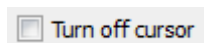
- zapnutie detekcie



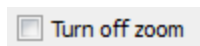
- zachytenie kádra s následnou možnosťou dať ho na pozadie



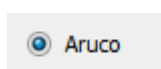
- zapnutie rozpoznávania značiek



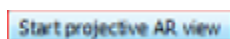
- prepínanie medzi detekovaním ruky pre manipuláciu grafu alebo kamery v podobe rotovania a medzi detekovaním ruky pre funkciu "klik" (pohyb ruky do hĺbkky, nie vertikálne alebo horizontálne)



- vypne možnosť približovania



- nastavenie práce s arucom



- zobrazenie okna projekčného zobrazenia

Projector	
FOV:	30,00 Å°
Pos X:	-0,665 m
Pos Y:	-1,345 m
Pos Z:	0,825 m
Dir X:	-0,085 m
Dir Y:	1,345 m
Dir Z:	-0,587 m

- v projekcnom zobrazení - Projector - spinboxy na zmenu parametrov projektora (odhora) - zorné pole, pozícia (súradnice x, y, z), smer projekcie(súradnice x, y, z)

Viewer	
FOV:	90,00 Å°
Pos X:	-1,880 m
Pos Y:	-0,950 m
Pos Z:	1,720 m
Dir X:	1,130 m

- v projekcnom zobrazení - Viewer - spinboxy na zmenu parametrov pozorovateľa (odhora) - zorné pole, pozícia (súradnice x, y, z), smer projekcie (súradnice x, y, z)

Graph	
Pos X:	-0,750 m
Pos Y:	-0,250 m
Pos Z:	0,250 m
Radius:	0,500 m
Place graph:	<input checked="" type="checkbox"/>

- v projekcnom zobrazení - Graph - spinboxy na zmenu parametrov grafu (odhora) - pozícia (súradnice x, y, z), polomer, checkbox Place graph na potvrdenie použitia parametrov grafu (štandardne označený)

Apply scene - potvrdenie zadaných parametrov scény

Hlasové príkazy pre Speech

- select all nodes - vybratie všetkých uzlov
- select left side - vybratie uzlov na ľavej strane
- select right side - vybratie uzlov na pravej strane
- clear screen - zrušenie vybratia uzlov
- sphere - sformovanie gule pre vybrané uzly
- unset restrictions - návrat k pôvodnému stavu - zrušenie akcie "sphere"

6.2.8 Hlavné okno

edges filter - filtrovanie hrán

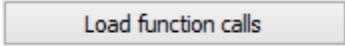
nodes filter - filtrovanie uzlov

Príklady príkazov:


- "params.type like 'file' or params.type like 'directory'"

- “params.name like ‘init%.lua’ and params.type like ‘function’”
- “params.type like ‘function’”

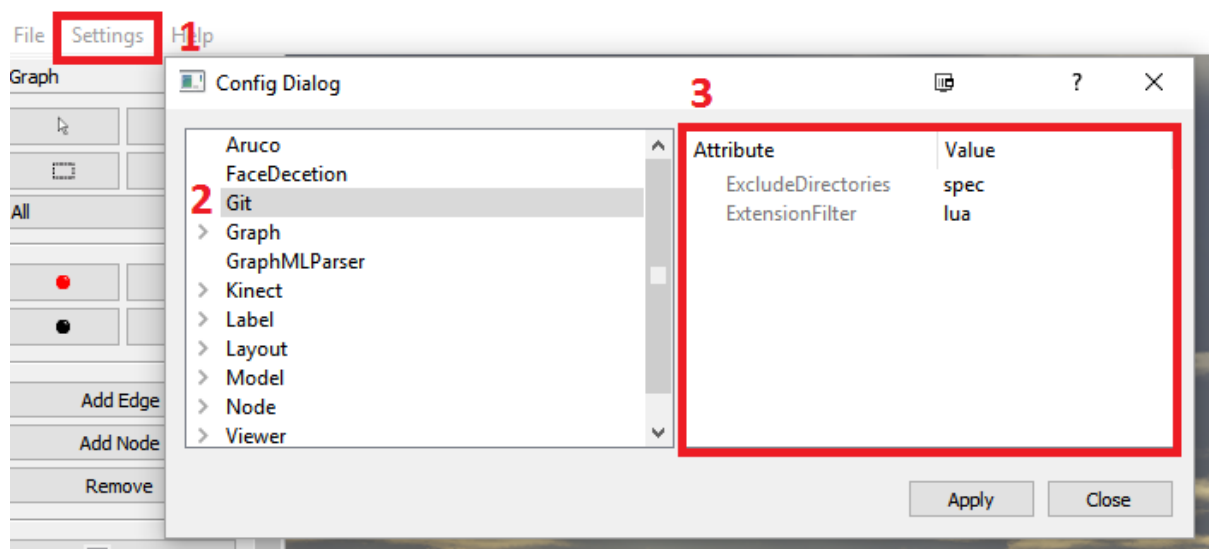
Filter je navrhnutý pre grafovú vizualizáciu softvéru s využitím softvérových metrick jazyka Lua a je vyhodnotený po stlačení klávesu „Enter“.

 - zobrazí dialóg pre výber súborov a po vybratí vykreslí do poľa pod tlačidlom graf volaní funkcií týchto súborov

Pri označení práve jedného vrcholu v poli sa zobrazí stromová štruktúra informácií o tomto vrchole.

 - prepínanie medzi zobrazovaním jedného prehliadača pre každý uzol a zobrazovaním jedného prehliadača pre všetky vyznačené uzly

6.2.9 Git repozitár



1. Settings / Options - zobrazenie dialógového okna s konfiguráciou
2. Možnosť Git - zobrazia sa možnosti konfigurácie spracovania Git repozitáru
3. Možnosti konfigurácie spracovania Git repozitáru
 - vyčlenenie adresárov (ExcludeDirectories) - ľubovoľný počet názvov adresárov oddelených znakom ",". Pre zadanú hodnotu sa pri spracovaní Git repozitáru odignorujú všetky súbory, ktoré vo svojej relatívnej ceste obsahujú adresár spec.
 - ExtensionFilter - funguje obrátene, pričom ponecháva len tie súbory, ktorých koncovka súboru sa zhoduje s jednou zo zadaných hodnôt. Hodnota taktiež môže obsahovať viacero koncoviek súborov, pričom musia byť oddelené znakom ","