

---

# **3dSoftViz Documentation**

*Release v0.1*

**Alphareach**

**Jun 06, 2017**



## CONTENTS



Obsah:



## PRE POUŽÍVATEL'OV

### 1.1 Inštalačný návod (VC12)

v1.6 (13. 03. 2016)

#### 1.1.1 Changelog

##### **v1.6 - 13.03.2016**

- návod pre MS VisualStudio 2013
- aktualizovaný Qt (5.5.1)

##### **v1.5 – 17.11.2015**

- aktualizované Aruco využíva FreeGlut
- upravené nastavenie pre build v QtCreator

##### **v1.4 - 27.10.2015**

- aktualizovaná knižnica OSG (v3.4) – nahradené zdroje
- zmazaný import chýbajúcich knižníc

##### **v1.3 - 11.10.2015**

- pridaný krok vytvorenia build konfigurácie

##### **v1.2 - 06.10.2015**

- pridaný postup pre buildovanie OSG knižníc
- pripojený zdroj s uploadnutými OSG knižnicami
- zvýraznené dôležité poznámky

##### **v1.1 - 28.09.2015**

- doplnené obrázky
- aktualizované zdroje
- vyriešené chyby
- zmenené poradie krokov

## 1.1.2 Časté problémy

Inštalácia softvéru 3DSoftViz pozostáva z mnohých krokov. Pozorné postupovanie podľa inštrukcií je nevyhnutné, inak sa môžu vyskytnúť problémy. Nasledujúce problémy sme identifikovali.

### Ponuka generátorov je prázdna

Ak sa generátor nedá vybrať, je dôležité zmazať súbor CMakeLists.txt.user.2.5pre1 z *%3DSoftViz%* a zopakovať predošlý krok.

### CMake - chýbajúce moduly

1. cez git shell prejsť do *%3DSoftViz%*
2. uistiť sa, že je nastavený na master vetvu
  - na prepnutie: git checkout master
3. zadať príkaz: git submodule update –init

### CMake - cesta nebola nájdená

Ak nebola nájdená cesta k niektorému modulu, treba skontrolovať nastavenia premenných v RapidEE.

### Chyba spojená s glut.h, glut.lib, glut.dll

Pokiaľ sa vyskytnú problémy typu:

- nevie nájsť glut.h
- hlási chyby priamo v glut.h
- niečo iné s glut.h, glut.lib alebo glut.dll

Tak najjednoduchšie riešenie je premenovať:

```
- %OSG_DIR%/ThirdParty/VC10/x86/include/GL/glut.h > ../glut.h.bak
- %OSG_DIR%/ThirdParty/VC10/x86/lib/glut32.lib > ../glut32.lib.bak
- %OSG_DIR%/ThirdParty/VC10/x86/lib/glut32D.lib > ../glut32D.lib.bak
```

### Pri spustení crash programu s nullpoint exception

Pokiaľ sa projekt spustí iba cez argument: `-DCMAKE_BUILD_TYPE=Release`, ale cez `-DCMAKE_BUILD_TYPE=Debug` hlási nullpoint exception pri načítavaní obrázkov na pozadie a program crashne. Tak riešením je znovu stiahnuť/nakonfigurovať OSG podľa návodu.

### Zm### pri buildovaní

Ak došlo k erroru pri buildovaní: `-Zm###`, treba vykonať úpravu v CMakeLists.txt > riadok 128 zmeniť `/Zm216` na `/Zm240` a spustiť CMake odznovu.



### 1.1.3 Návod pre Windows

Tento návod bol úspešne otestovaný na operačných systémoch Windows 7, 8, 8.1, 10.

#### Potrebný softvér

**Uvedené programy neinštalovať do Program Files, PATH nesmie obsahovať medzeru!** Na inštaláciu potrebujeme klonovať projekt 3DSoftViz (Tím č.4 klonuje z /cimox/3dsoftviz) z Githubu a stiahnuť nasledovné:

- CMake (v3.6.2)
- OpenSceneGraph (*jeden z nasledujúcich*)
  - OpenSceneGraph ([source](#)) - iba zdrojové súbory -> treba buildnúť (cca 40-60min)
  - stable release: OpenSceneGraph (v3.4.0) - buildnuté (13.03.2016)
  - developer release: OpenSceneGraph (v3.5.1) - buildnuté (13.03.2016)
- Kinect for Windows SDK 1.8
- Microsoft VisualStudio 2013 (**okrem Express edition!**)
- Qt (v4.8.6)
- QtCreator (v3.6.1)
- OpenCV (v2.4.10)
- Boost (v1.57.0)
- RapidEE - program na prácu s premennými prostredia
- Inštalácia knižnice 3rd party dependencies VC12
- OpenNI2
- NiTE2
- Debugging Tools for Windows
  - WinDbg - Win 8.1
  - WinDbg - Win 10

#### Postup inštalácie

**Uvedené programy neinštalovať do Program Files, PATH nesmie obsahovať medzeru! Ideálne všetky programy, knižnice, atd.**

1. Nainštalovať CMake. (Cesta je v dokumente označená ako %CMAKE\_DIR%). Pri inštalácii zvolit' "Add CMake to the system for all users".
2. Nainštalovať Qt (%QT\_DIR%)
3. Nainštalovať QtCreator.
4. Vytvoriť zložku OpenSceneGraph (%OSG\_DIR%)
5. V prípade stiahnutia zbuildovaných súborov OSG (mega.nz)
  - (a) Rozbalit' zložky build a install do %OSG\_DIR%
  - (b) Vynechať nasledujúci krok.

6. V prípade stiahnutia iba zdrojových súborov OSG (oficiálna stránka)
  - (a) Rozbalit' OSG do `%OSG_DIR%/source`
  - (b) Vytvorit' zložku build a install v `%OSG_DIR%`
  - (c) Rozbalit' 3rd Party Knižnice (VC12) do `%OSG_DIR%/ThirdParty/`
  - (d) Spustiť CMake (`cmake-gui.exe`)
    - i. source code > `%OSG_DIR%/source`
    - ii. binaries > `%OSG_DIR%/build`
    - iii. stlačiť Configure (VS2013 kompilátor)
    - iv. nastaviť 3rdParty na `%OSG_DIR%/ThirdParty/VC12/X86`
    - v. stlačiť Generate
    - ak došlo k erroru: File > Delete Cache a skúsiť znovu

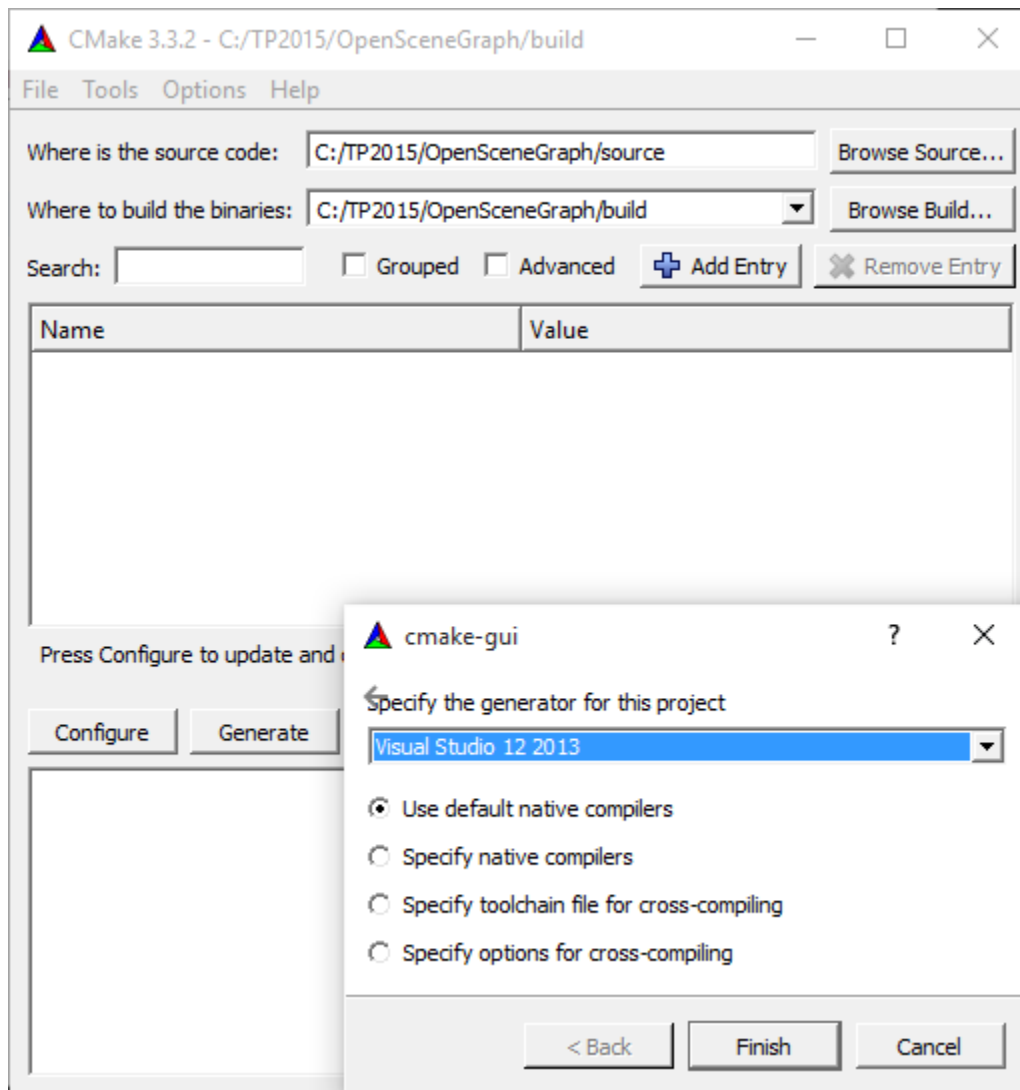


Fig. 1.1: CMake pre OSG

- (e) Nájst' súbor OpenSceneGraph.sln v *%OSG\_DIR%/build*
  - (f) Otvoriť súbor vo VS2013 (**ako správca!**)
  - (g) Nastaviť Solution Configuration na Debug
  - (h) Nájst' projekt ALL\_BUILD > pravý klik > build
  - (i) Po skončení nájsť projekt INSTALL > pravý klik > build
  - (j) Nastaviť Solution Configuration na Release
  - (k) Nájst' projekt ALL\_BUILD > pravý klik > build
  - (l) Po skončení nájsť projekt INSTALL > pravý klik > build
  - (m) Presunúť nainštalované súbory (default *c:\Program Files (x86)\OpenSceneGraph*) do *%OSG\_DIR%/install*
7. Nainštalovať OpenCV (*%OPENCV\_DIR%*)
  8. Rozbalit' Boost (*%BOOST\_DIR%*)

Ideálne je mať všetko na spoločnom mieste kvôli prehľadnosti, napr.

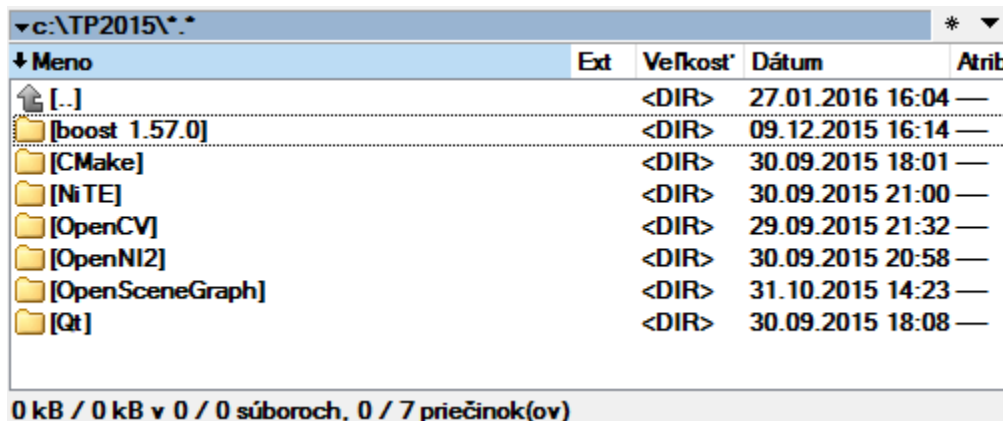


Fig. 1.2: Nainštalovaný SW

9. Otvoriť súbor environment.txt a upraviť v ňom cesty k programom a knižniciam, ktoré sa nachádzajú na začiatku súboru.

```

2 # cesta k projektu
3 $timak = "C:\Timak"
4 $osg = "${timak}\OpenSceneGraph"
5 $opencv = "${timak}\opencv-2.4.10"
6 $boost = "${timak}\boost_1_57_0"
7 $qtCreator = "${timak}\qt\qtcreator-4.2.1"
8 $openNi2 = "${timak}\OpenNI2"
9 $nite2 = "${timak}\NiTE2"

```

Fig. 1.3: Pozadovane premenne, ktoré treba nastaviť

- (a) Spustiť powershell ako spravca .
  - (b) Skopirovať do powershellu obsah celeho suboru.
  - (c) Vynechať pridávanie systémových premenných cez RapidEE (nasledujúci krok) a vykonať len kontrolu, či sa cesty správne nastavili.
10. Nainštalovať a otvoriť RapidEE, v ktorom sa vykonajú tieto zmeny (**ako správca!**):

- (a) do PATH pridať premenné:
  - i. `%CMAKE_DIR%/bin`
  - ii. `%QT_DIR%/bin`
  - iii. `%QT_DIR%/Qtcreator/bin`
  - iv. `%OSG_DIR%/build/bin`
  - v. `%OSG_DIR%/ThirdParty/VC12/x86/bin`
  - vi. `%OPENCV_DIR%/build/x86/vc12/bin`

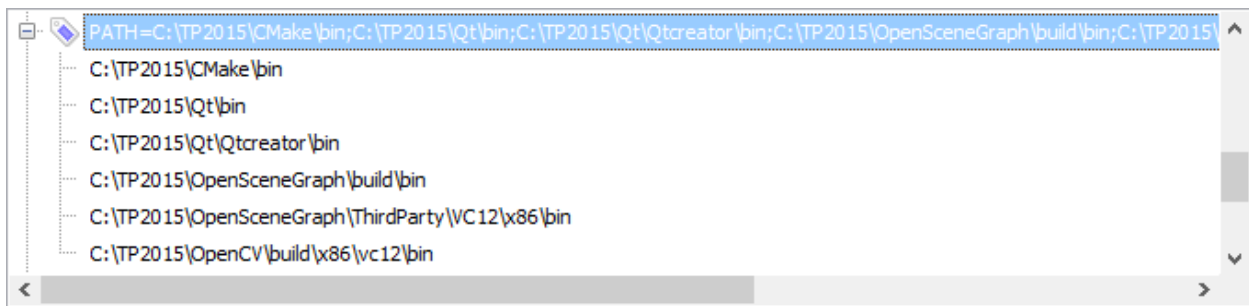


Fig. 1.4: PATH premenná

- (b) Vytvoriť premennú CMAKE\_INCLUDE\_PATH a pridať:
  - i. `%OSG_DIR%/install/include`
  - ii. `%OSG_DIR%/ThirdParty/VC12/x86/include`
  - iii. `%OPENCV_DIR%/build/include`

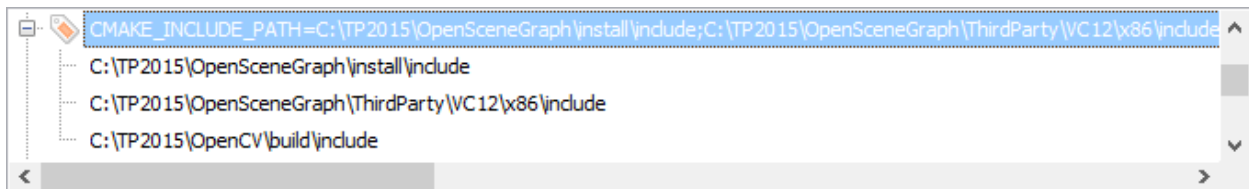


Fig. 1.5: CMAKE\_INCLUDE\_PATH premenná

- (c) Vytvoriť premennú CMAKE\_LIBRARY\_PATH a pridať:
  - i. `%OSG_DIR%/build/lib`
  - ii. `%OSG_DIR%/install/lib`
  - iii. `%OSG_DIR%/ThirdParty/VC12/x86/lib`
  - iv. `%OPENCV_DIR%/build/x86/vc12/lib`
- (d) Vytvoriť premennú BOOST\_INCLUDEDIR a pridať: `%BOOST_DIR%/boost`

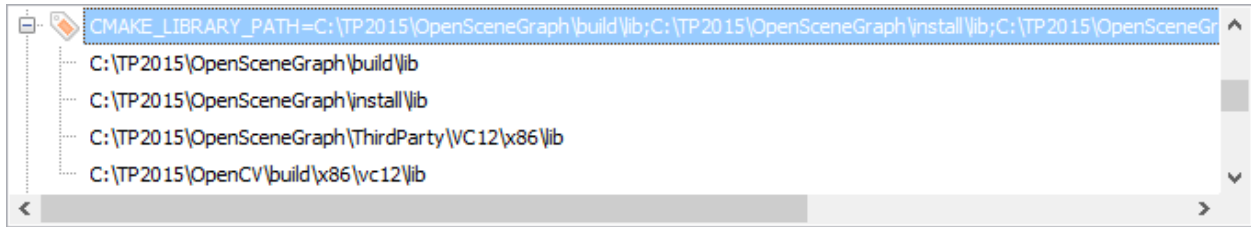


Fig. 1.6: CMAKE\_LIBRARY\_PATH premenná

- (e) Vytvorit' premennú BOOST\_LIBRARYDIR a pridať: `%BOOST_DIR%/libs`
- (f) Vytvorit' premennú BOOST\_ROOT a pridať: `%BOOST_DIR%`

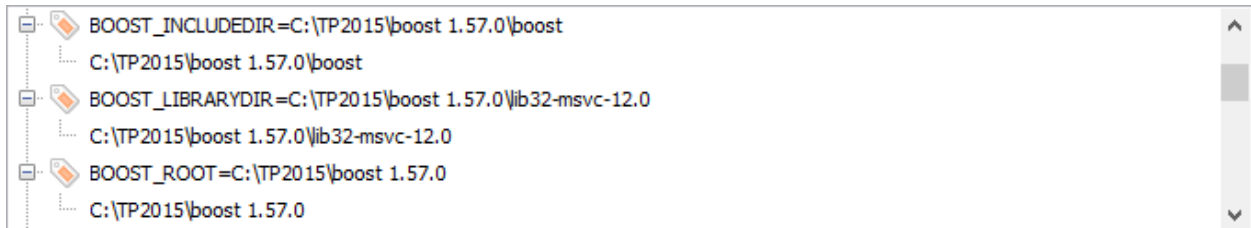


Fig. 1.7: BOOST premenné

- (g) Vytvorit' premennú OPENCV\_DIR a pridať: `%OPENCV_DIR%/build`

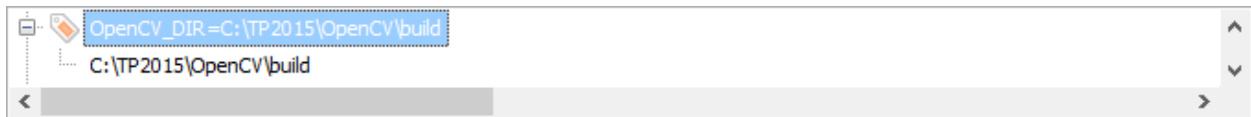


Fig. 1.8: OPENCV\_DIR premenná

11. Nainštalovať Debugging Tools for Windows.
12. Naklónovať projekt 3DSoftViz cez git shell (`%3DSoftViz%`)

1. Naklónovať projekt 3DSoftViz cez git shell (`%3DSoftViz%`)
2. Cez command line prejsť do naklonovaného projektu a zavolať `git submodule --init --recursive`
3. Vytvorit' v priečinku `%3DSoftViz%` priečinky `_build` a `_install`
4. Spustiť QtCreator. Tools > Options... > Build and Run:
  - (a) záložka CMake – zadať cestu `%CMAKE_DIR%/bin/cmake.exe`
  - (b) záložka Compilers – ak existuje VS2013 tak sú autodetected
  - (c) záložka Qt Versions – zadať cestu `%QT_DIR%/bin/qmake.exe`
  - (d) záložka Kits – vytvorit' nový a vybrať hodnoty nasledovne:
  - (e) záložka General – nastaviť default build directory: `%3DSoftViz%/_build`
  - (f) Potvrdiť – OK
5. File > Open File or Project... > vybrať CMakeLists.txt z `%3DSoftViz%`

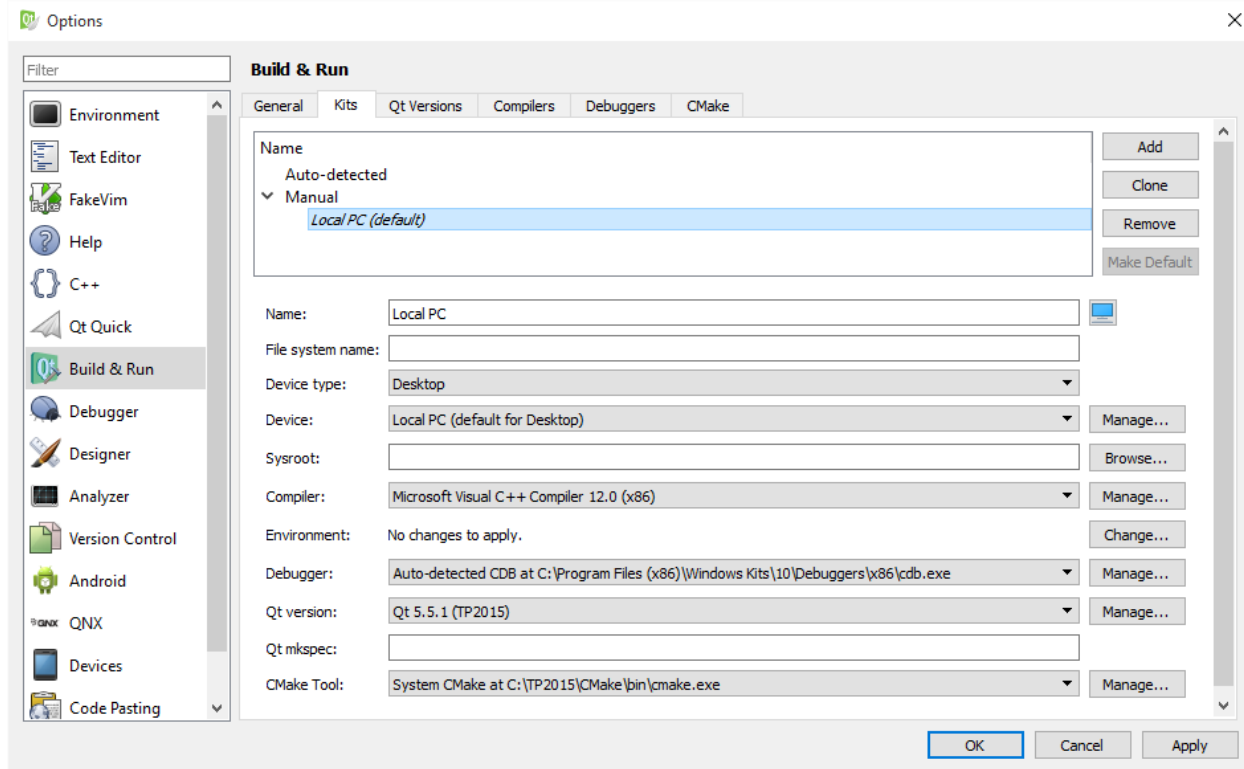


Fig. 1.9: QtC Kits nastavenia

6. Zadať do poľa Arguments jeden z nasledujúcich prepínačov:
  - -DCMAKE\_BUILD\_TYPE=Debug
  - -DCMAKE\_BUILD\_TYPE=Release
7. Vybrať z nastavený generátor – NMake Generator (názov kitu)
 

(Chyba: Generátor nebol nájdený <riešenie>)
8. Stlačiť Run CMake
 

(Chyba: Chýbajúce moduly <riešenie>)

(Chyba: Cesta nebola nájdená <riešenie>)
9. Ukončiť – Finish
10. Vybrať Projects > Build & Run > Build, v časti Edit build configuration kliknúť na Add > Clone selected, nazvať napr. „unity“
11. Prejsť na vytvorený build config. „unity“, v časti Build Steps otvoriť Details a zaškrtnúť pri build step *jom.exe* možnosť *install\_unity*
12. Skontrolovať nastavenie build config – unity
13. Stačiť Build (kladivko vľavo dole - potreba spraviť znova po každej následnej úprave systémových premenných)

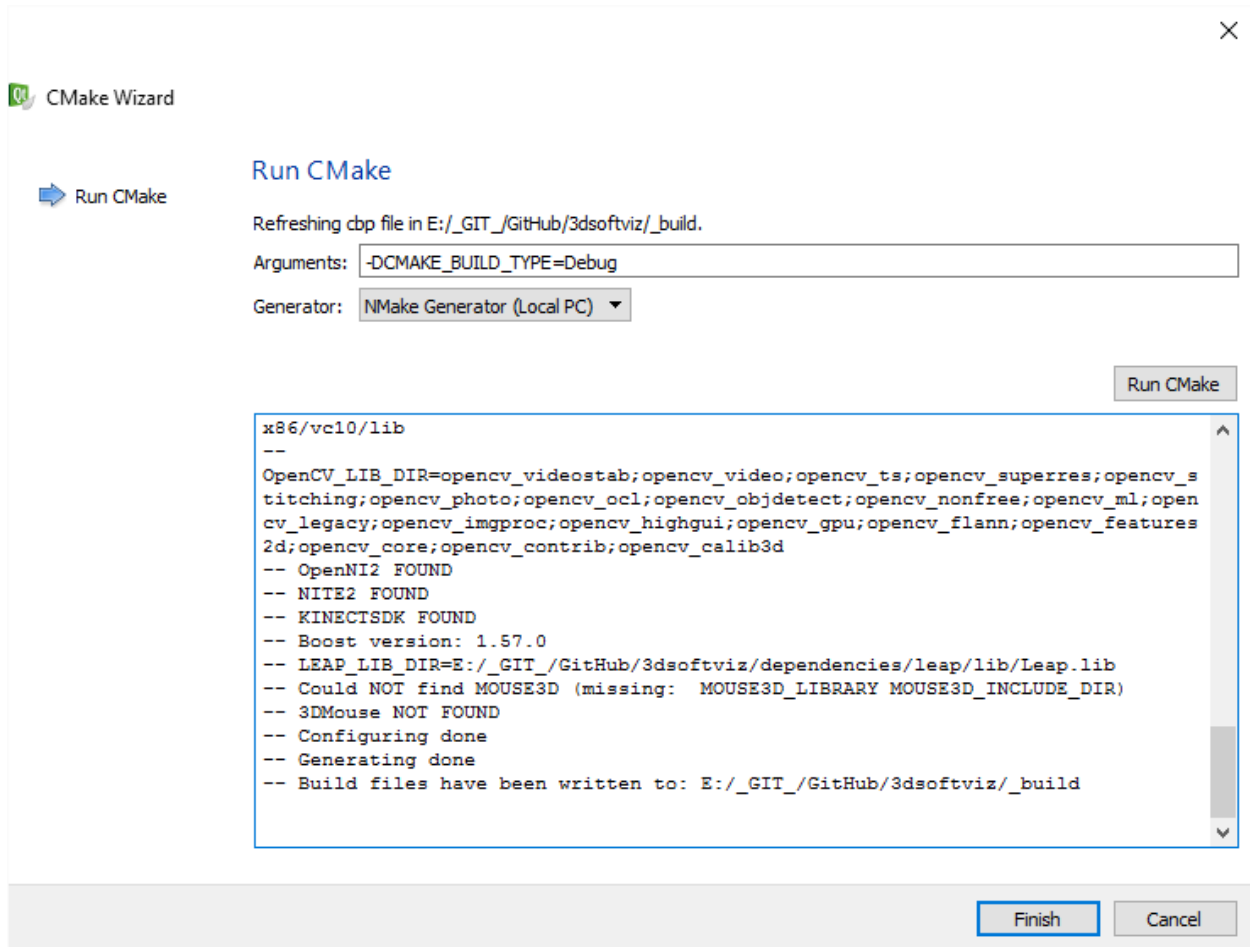


Fig. 1.10: QtC CMake wizard

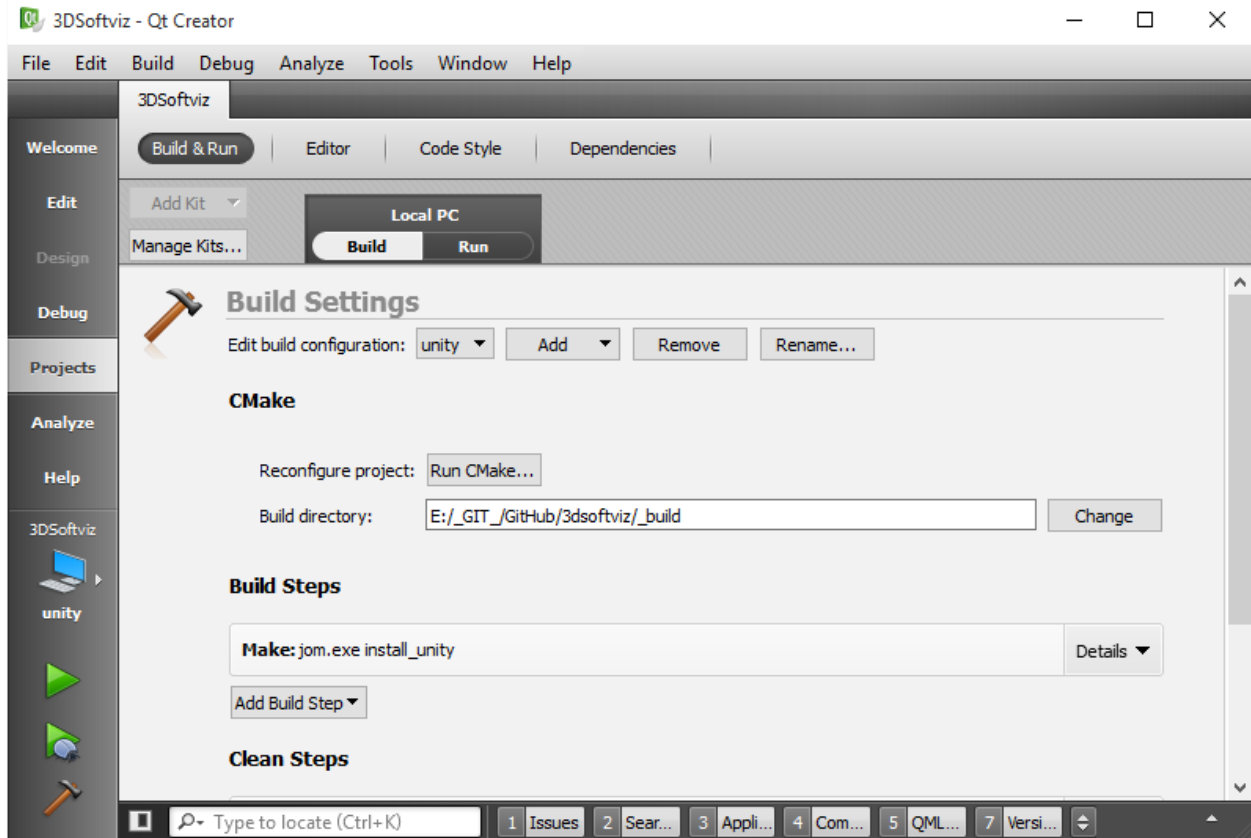


Fig. 1.11: QtC build project

Project: 3DSoftviz	
Kit: Local PC	
Deploy: Deploy locally	
Build	Run
all	3DSoftviz
unity	3DSoftviz_unity
	Run E:\_GIT_\GitHub\3dsoftviz\_install\bin\3DSoftviz.exe

Fig. 1.12: QtC build config



14. Po úspešnom zbuildovaní vybrať Projects > Build & Run > Run, v časti Run pridať Add > Custom Executable a nastaviť:
  - (a) executable: `%3DSoftViz%/_install/bin/3DSoftviz.exe`
  - (b) working directory: `%3DSoftViz%/_install/bin/`

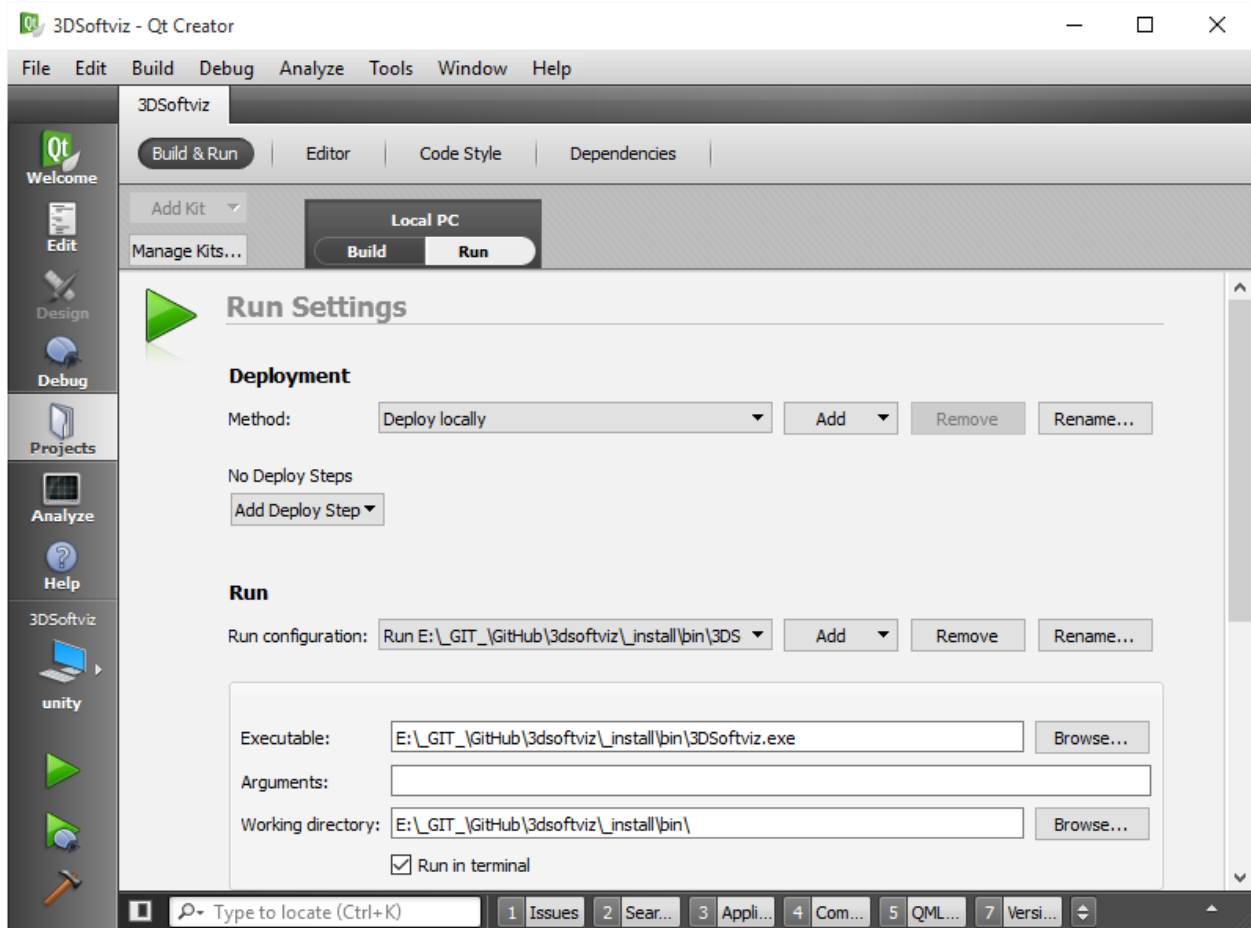


Fig. 1.13: QtC run project

15. Skontrolovať nastavenie run config – zadaná cesta

16. Spustiť program pomocou zeleného tlačidla Run

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

### 1.1.4 Rozšírenie 3DSoftviz o Kinect

1. Nainštalovať Kinect for Windows
2. Skontrolovať v RapidEE či sa vytvorila premenná `%KINECTSDK10_DIR%`

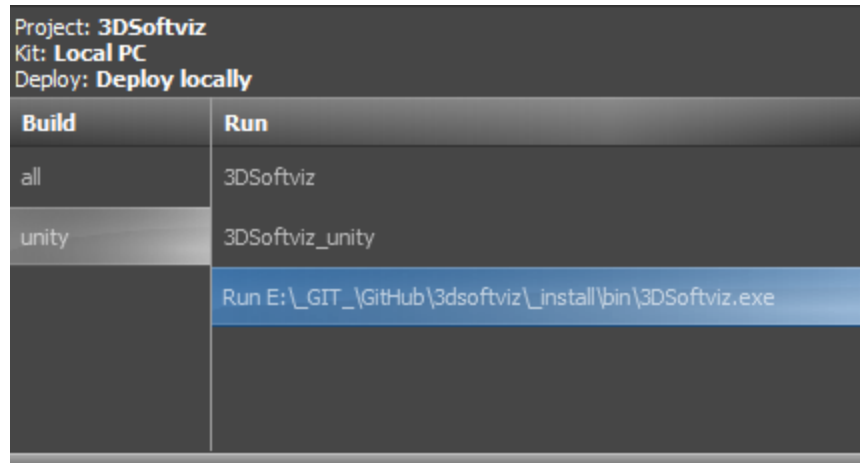


Fig. 1.14: QtC run config

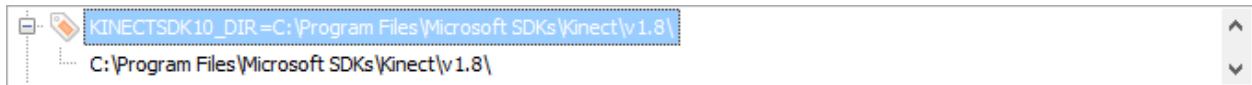


Fig. 1.15: KINECTSDK10\_DIR premenná

3. Nainštalovať OpenNI2 (OpenNI-Windows-x86-2.2.msi)
  - **x86!** – inak sa môžu vyskytnúť problémy s linkovaním
4. Skontrolovať v RapidEE či sa vytvorili premenné:
  - (a) `%OPENNI2_INCLUDE%`
  - (b) `%OPENNI2_LIB%`
  - (c) `%OPENNI2_REDIST%`
  - (d) `%OPENNI2_ROOT%`

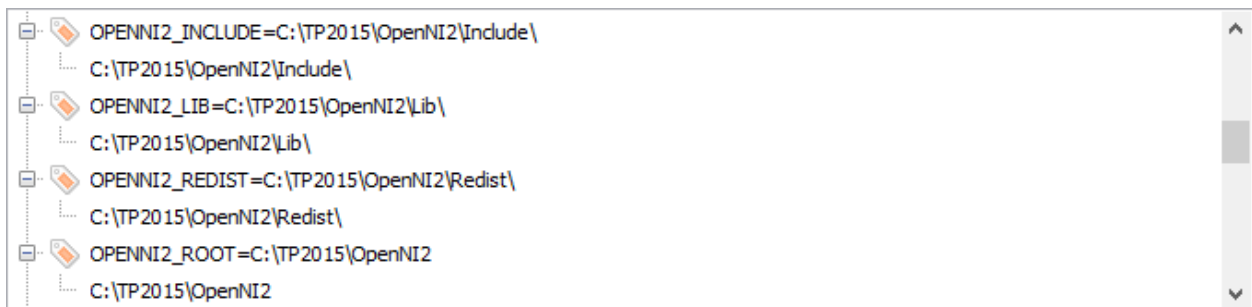


Fig. 1.16: NITE2 premenné

5. Nainštalovať NiTE2 (NiTE-Windows-x86-2.2.msi)
  - **x86!** – inak sa môžu vyskytnúť problémy s linkovaním
6. Skontrolovať v RapidEE či sa vytvorili premenné:
  - (a) `%NITE2_INCLUDE%`

- (b) `%NITE2_LIB%`
- (c) `%NITE2_REDIST%`
- (d) `%NITE2_ROOT%`

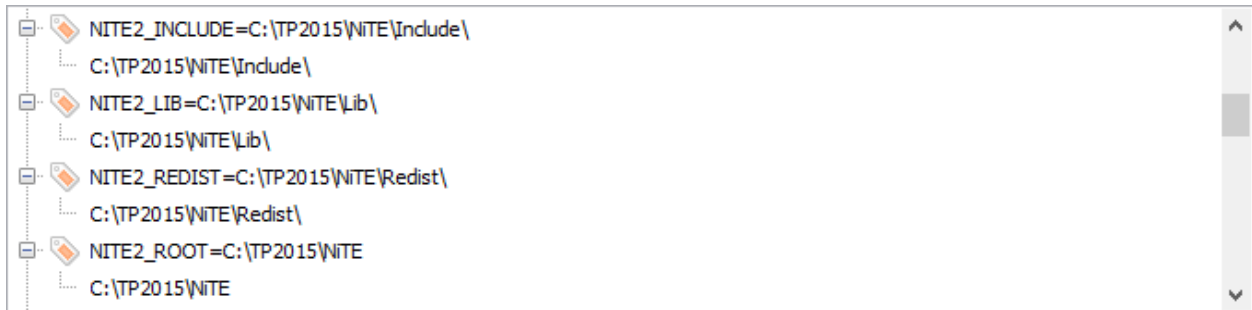


Fig. 1.17: OPENNI2 premenné

7. Pridať do premennej CMAKE\_INCLUDE\_PATH:

- (a) `%OPENNI2_INCLUDE%`
- (b) `%NITE2_INCLUDE%`

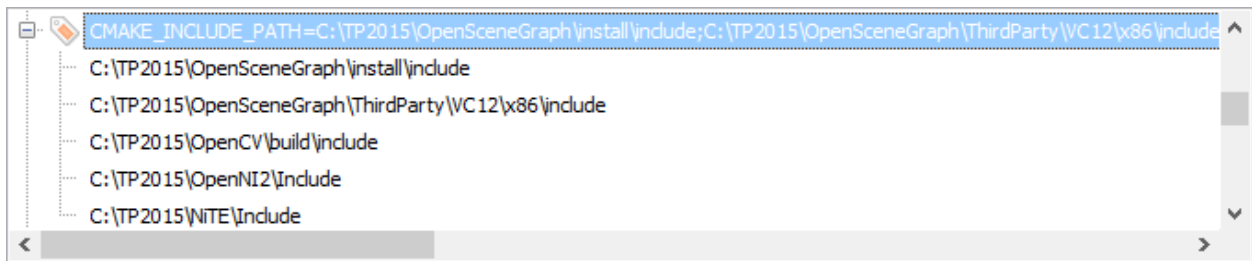


Fig. 1.18: OPENNI2 premenné

8. Pridať do premennej CMAKE\_LIBRARY\_PATH:

- (a) `%OPENNI2_ROOT%/Driver`
- (b) `%OPENNI2_REDIST%`
- (c) `%OPENNI2_REDIST%/OpenNI2/Drivers`
- (d) `%OPENNI2_LIB%`
- (e) `%NITE2_ROOT%/Samples/Bin/OpenNI2/Drivers`
- (f) `%NITE2_LIB%`

9. Pridať do premennej PATH:

- (a) `%OPENNI2_REDIST%/OpenNI2/Drivers`
- (b) `%OPENNI2_REDIST%`
- (c) `%NITE2_REDIST%`
- (d) `%NITE2_ROOT%/Samples/Bin`

10. Spustiť CMake a skontrolovať vo výpise:

```
CMAKE_LIBRARY_PATH=C:\TP2015\OpenSceneGraph\build\lib;C:\TP2015\OpenSceneGraph\install\lib;C:\TP2015\OpenSceneGr
C:\TP2015\OpenSceneGraph\build\lib
C:\TP2015\OpenSceneGraph\install\lib
C:\TP2015\OpenSceneGraph\ThirdParty\VC12\x86\lib
C:\TP2015\OpenCV\build\x86\vc12\lib
C:\TP2015\OpenNI2\Driver
C:\TP2015\OpenNI2\Redist
C:\TP2015\OpenNI2\Redist\OpenNI2\Drivers
C:\TP2015\OpenNI2\Lib
C:\TP2015\NITE\Samples\Bin\OpenNI2\Drivers
C:\TP2015\NITE\Lib
```

Fig. 1.19: OPENNI2 premenné

```
PATH=C:\TP2015\CMake\bin;C:\TP2015\Qt\bin;C:\TP2015\Qt\Qtcreator\bin;C:\TP2015\OpenSceneGraph\build\bin;C:\TP2015\
C:\TP2015\CMake\bin
C:\TP2015\Qt\bin
C:\TP2015\Qt\Qtcreator\bin
C:\TP2015\OpenSceneGraph\build\bin
C:\TP2015\OpenSceneGraph\ThirdParty\VC12\x86\bin
C:\TP2015\OpenCV\build\x86\vc12\bin
C:\TP2015\OpenNI2\Redist\OpenNI2\Drivers
C:\TP2015\OpenNI2\Redist
C:\TP2015\NITE\Redist
C:\TP2015\NITE\Samples\Bin
```

Fig. 1.20: OPENNI2 premenné

- (a) OpenNI2 FOUND
- (b) NITE2 FOUND
- (c) KINECTSDK FOUND

### 1.1.5 Nastavenie debugera v QtCreator

1. Nainštalovať WinDbg
2. Skontrolovať v QtCreator Tools > Options > Build & Run > záložka Debuggers či sú autodetected

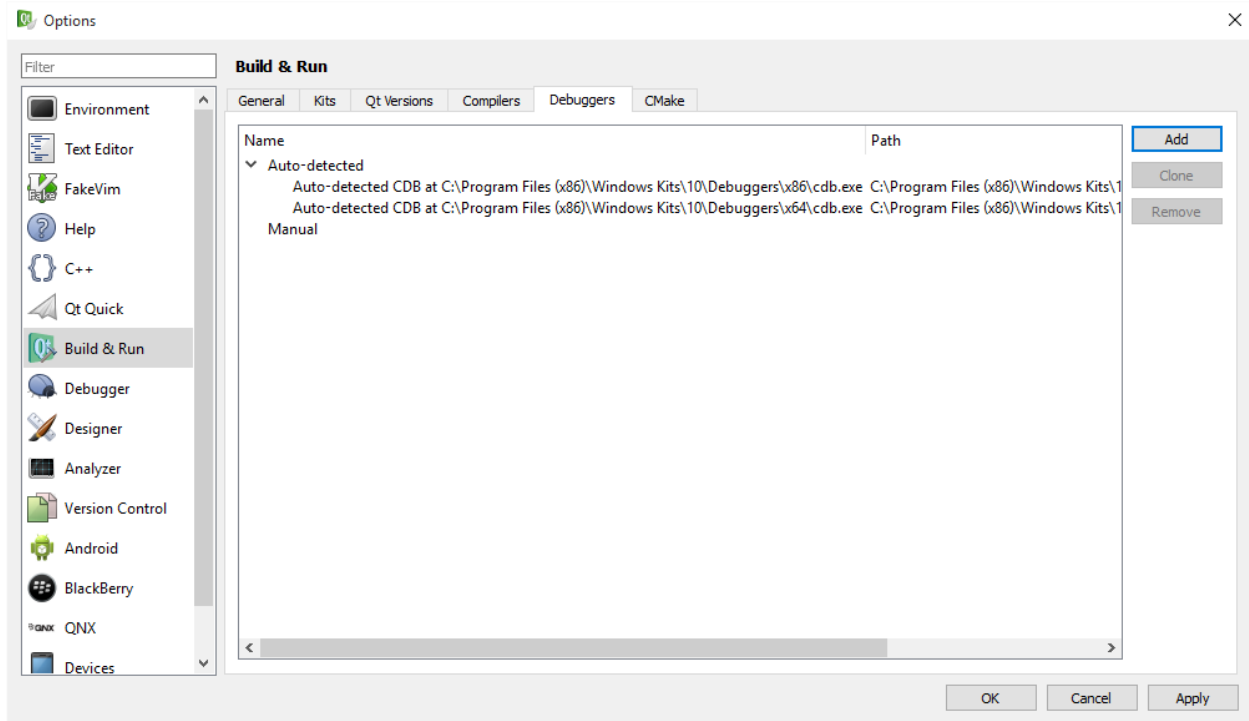


Fig. 1.21: QtC debugger nastavenia

3. Pridať do QtCreator Tools > Options > Build & Run > záložka Kits pre vytvorený profil položku Debugger (x86)
4. Spustiť CMake (-DCMAKE\_BUILD\_TYPE=Debug)
5. Zvoliť možnosť Debug (vľavo dole medzi Run a Build)

### 1.1.6 Portable zip s projektom (Windows)

1. Pri bilde projektu treba nastaviť build install\_unity a CMAKE\_BUILD\_TYPE = Release.
2. Nastavenie v Qt Creator 4.2 a vyššie
3. V starsich verziach treba kliknúť na Run CMake a do pola argumentov zadať: -DCMAKE\_BUILD\_TYPE=Release

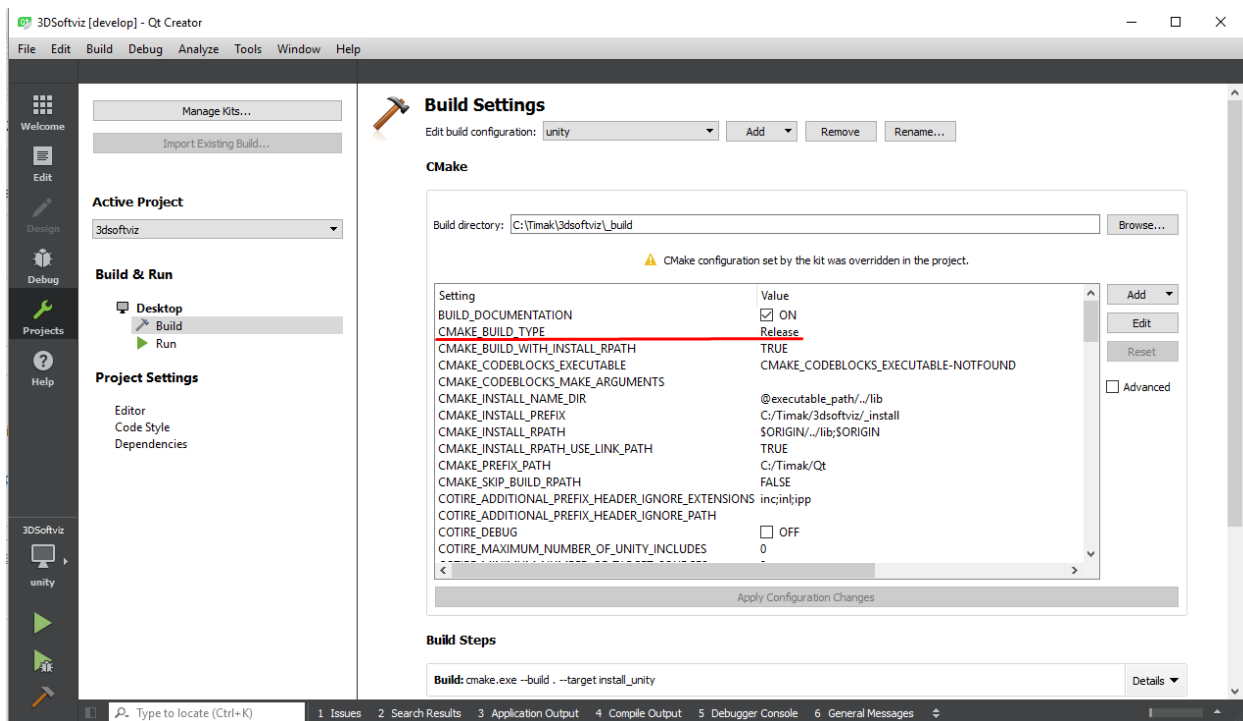
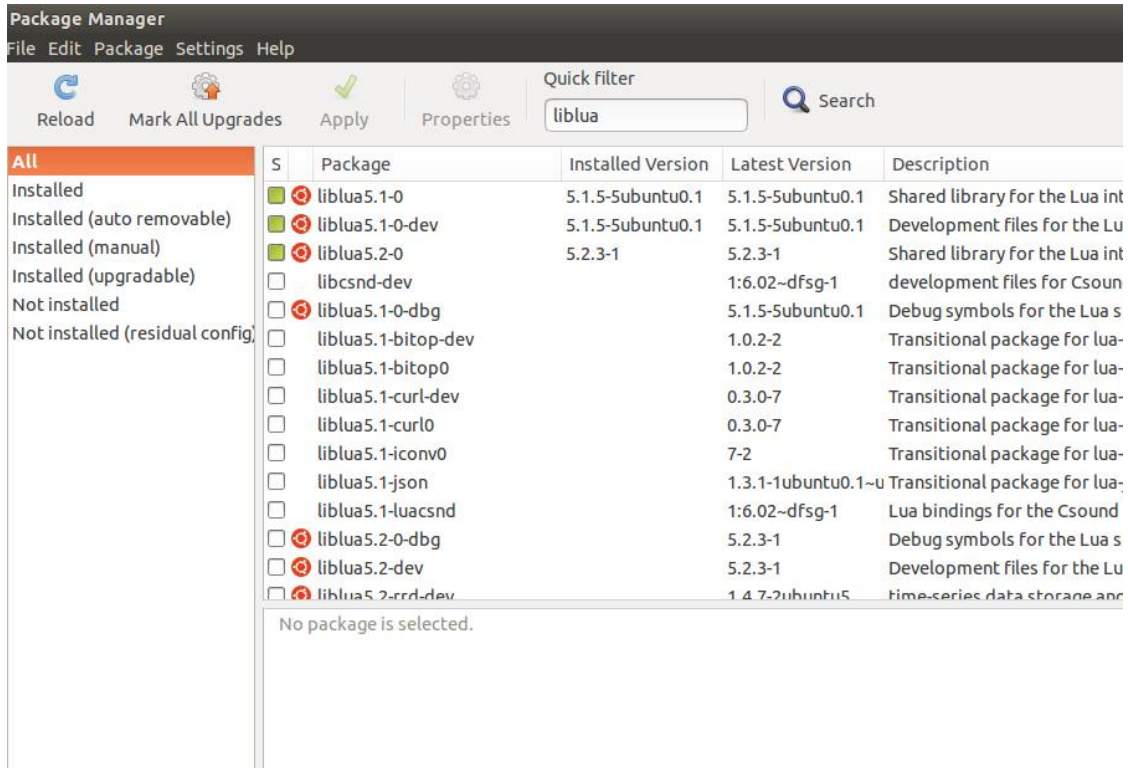


Fig. 1.22: Qt creator build

4. Build projektu.
5. Stiahnut `_install.zip` . V zipe su popripravane dll subory, ktore program potrebuje.
6. Rozbalit zip na disk.
7. Do rozbaleného priečinku nakopirovať obsah z adresára “3dsoftviz/\_install/” (po buildovani)
8. Spustiť `_install/bin/3DSoftviz.exe`.

## 1.2 Návod pre Linux

1. Nainštalovať Synaptic Package Manager



2. Nainštalovať prostredníctvom Synaptic nasledujúce programy

- **Git** - git
- **OpenSceneGraph** - libopenscenegraph-dev
- **Qt4** - libqt4-dev
- **QtCreator** - qtcreator
- **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

---

**Note:** Názvy knižníc sú ukázané pre Ubuntu

---

3. Klonovať cez Git projekt 3dsoftviz (AlphaReach klonuje z repozitára Cimox)

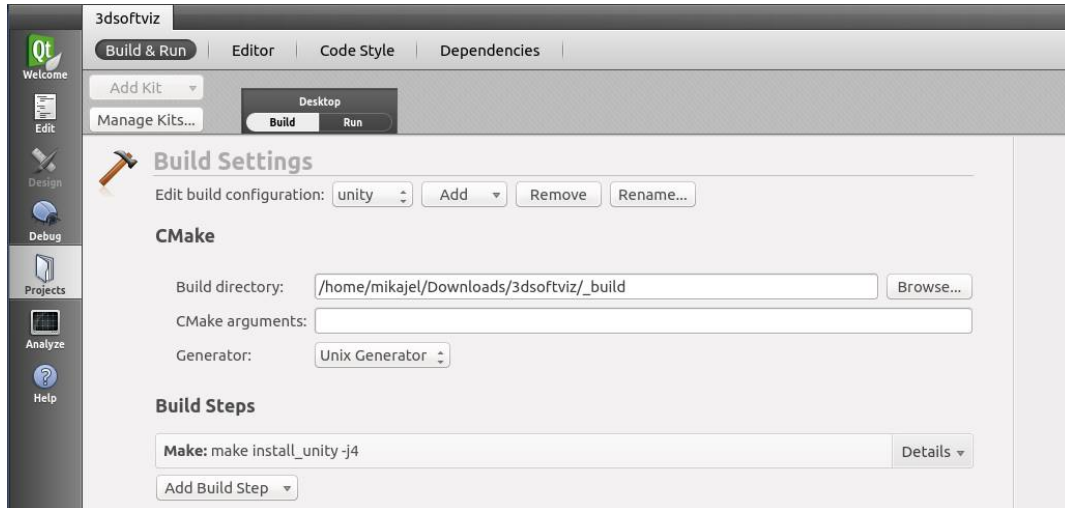
- git clone \*\* **url repozitára** \*\*
- git submodule update –init –recursive

4. 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

### 5. 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

### 6. Pridanie Kinectu do projectu

- Nainštalovať **NiTE** – pridať systémové premenné
- Nainštalovať **OpenNI** alebo cez `sudo apt-get install` – pridať systémové premenné
- Nainštalovať **Freenect2** - postupovať podľa <<https://github.com/OpenKinect/libfreenect2/blob/master/README.md#linux>> -je potrebné aby fungoval aj OpenNI test

---

**Note:** Ak je potrebný prístup k knižniciam kvôli vývoju tak je potrebné nastaviť premenné `FREENECT2_INCLUDE` a `FREENECT2_LIBRARY`

---

---

**Note:** Pre použitie je nutný Kinect for XBOX

---

**Attention:** Na OSX, ak je found OpenNI2 a NiTE2, aplikacia crashne pri spustani s chybovou hlaskou:

- *not found libNiTE2.dylib*
- *not found libOpenNI2.dylib*

Treba tieto kniznice skopirovat z ich domovskych priecinkov (z lib alebo redistrib).



## 7. Inštalácia drivera pre 3dmyš

- Nainštalovať Motif3 pomocou Synaptic Package Manager
- **Pre uistenie, že 3dmyš funguje, skúsiť xapp aplikáciu z drivera**
  - Pre spustenie drivera (ktory musí bezat pre pouzivanie mysy) spustite prikaz

```
sudo /etc/3DxWare/daemon/3dxsrv -d usb
```

- Pre podrobnejsie instrukcie ohladne instalacie a pouzivania drivera otvorit navod prilozeny v suboroch oficialneho drivera pre Linux “**InstallationInstructions\_Linux.txt**“

## 1.3 Používateľská príručka pre 3D Softviz

### 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 - pušť a základné grafy pre rýchle testovanie (100-uzlový, 500-uzlový, Veolia, Lua-graph)
- hlavné okno - zobrazuje graf a umožňuje s ním používateľovi interagovať
- ovládací panel – nástroje pre prácu s grafom

### 1.3.1 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 metrick). Automatické použitie metrick 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

### 1.3.2 Záložka GRAPH



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



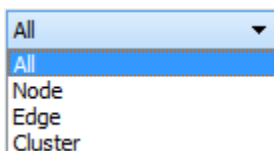
- výber jedného prvku grafu (single-select mód)

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

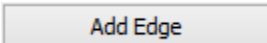


- pridanie meta uzla do grafu

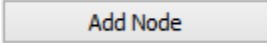
 - odstránenie vybraných meta uzlov z grafu

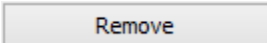
 - ukotvenie vybraných uzlov na aktuálnej pozícii, t. j. nebudú sa pohybovať v závislosti od pôsobenia síl ostatných uzlov

 - uvoľnenie ukotvených uzlov

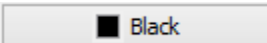
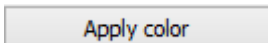
 - pridanie hrany medzi dvomi vybranými uzlami

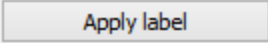
Umožňuje pridať hranu medzi dvoma vybranými uzlami, kde ešte nie je hrana, inak končí akcia chybovou hláškou. Ideálne je čiernou šípkou vybrať jeden uzol a bielou šípkou ho nastaviť na také miesto, kde sa ho dá spojiť s druhým uzlom - je potrebné mať nastavenie Node v takomto prípade spolu s Multi-select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

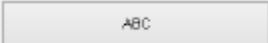
 - pridanie uzla do stredu pohľadu

 - 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)

### 1.3.3 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.

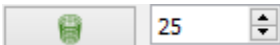


- aplikovanie priestorového ohraničenia: kužeľový strom

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




- aplikovanie priestorového ohraničenia: povrch valca


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


   - aplikovanie priestorového ohraničenia: povrch kužel'a

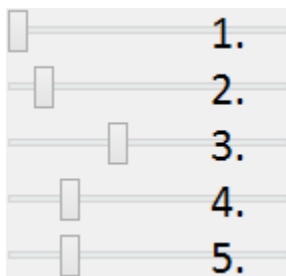
Vloží do scény tzv. bod záujmu, od ktorého sú uzly zobrazovaného grafu odtláčané do tvaru kužel'a. Polomer kužel'a sa dá nastaviť pomocou číselníka. Veľkosť kužel'a sa nastavuje automaticky podľa toho, kam sa používateľ prostredníctvom kamery pozerá.

 - 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

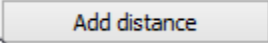


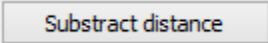
1. nastavenie veľkosti rozmiestnenia
2. nastavenie priehľadnosti rozmiestnenia
3. nastavenie počtu zobrazených gúľ
4. nastavenie faktora zosilnenia odpudivých síl v radiálnom rozmiestnení pre uzly, ktoré nie sú na rovnakej vrstve
5. nastavenie faktora zosilnenia odpudivých síl v radiálnom rozmiestnení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

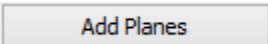
Veľkosť radiálneho zobrazenia sa dá nastaviť v rozmedzí 0 – 300, parameter priehľadnosti 0 - 100 %, veľkosť faktora zosilnenia odpudivých síl sa nastavuje medzi hodnotami 1 - 5000.

**Vertigo zoom** - prepínač medzi normálnou a vertigo kamerou

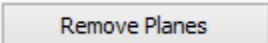
Tento mód kamery je vhodné použiť vtedy, keď chce používateľ meniť dva rôzne pohľady na graf: lokálny pohľad, pri ktorom môže používateľ s väčšou presnosťou skúmať jednotlivé uzly a vzťahy medzi nimi a globálny pohľad, pri ktorom môže používateľ skúmať vzťahy medzi uzlami a rozloženie uzlov v daných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

### 1.3.4 Záložka CLUSTERING



- zlúčenie vybraných uzlov

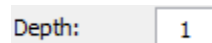
Umožňuje zlúčiť vybrané uzly do jedného spoločného uzla. Takýto uzol sa bude v pokračovaní zobrazovať modrou farbou.



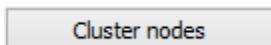
- zrušenie zlúčenia vybraných uzlov



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

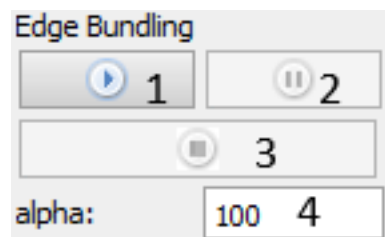


- nastavenie počtu rekurzií pre vybraný algoritmus



- 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 zhlukovania, sa odkryjú nasledujúce možnosti:

Opacity:

auto

selected

auto - automatická priehľadnosť - mení sa na základe vzdialenosti zhlukov od kamery

selected - priehľadnosť označeného zhluku – pomocou posuvníka(nižšie) sa mení priehľadnosť len označených zhlukov



- posúvaním upravíme priehľadnosť označených zhlukov

Cluster shapes:

0



8

- 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 zhluku sa odkryjú nasledujúce možnosti:

Restrict

- 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 lepíť na jeho stenu a posúvať spolu s ním. Ignoruje príťažlivé a odpudivé sily medzi ním a ostatnými uzlami grafu (posunutie zhluku bez obmedzovača spôsobí posun celého grafu za týmto zhlukom). Obmedzovač začína svoje pôsobenie ako kocka, je možné zmeniť jeho tvar natáhovaním a stlačením.

Restart Layout

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

Repulsive force

1,00



- upravenie odpudivej sily medzi uzlami v označenom zhluku

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 pust' ( 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

### 1.3.5 Záložka CONNECTIONS

Nick:

- napísanie mena, pod ktorým bude používateľ vystupovať v kolaborácii

- spustenie/zastavenie servera

Host:

- napísanie IP adresy servera

- pripojenie(odpojenie) ku(od) kolaborácii

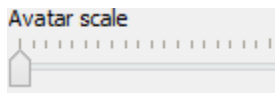
Collaborators:

- 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* (centrovat').

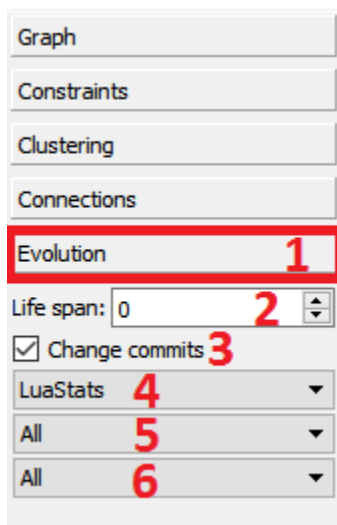
Po aktivovaní funkcie *Spy* získa používateľ pohľad iného používateľa, ktorý je priebežne aktualizovaný – znamená to, že pohybom sledovaného používateľa sa aktualizuje aj pohľad sledujúceho. Po aktivácii *Center* nasmeruje pohľad používateľa tak, aby v jeho strede bol iný používateľ. Pri centrovaní platí to isté, čo pri špehovaní – teda pri aktualizácii polohy centrovaneho 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.



- nastavenie veľkosti avatarov v scéne

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

### 1.3.6 Záložka EVOLUTION



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

2. *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 zotrývajú o verzie dlhšie podľa nastavenej hodnoty
3. *Change commits* - prepínač spracovania Git repozitáru. Ak je zaškrtnutý, inicializuje sa spracovanie na úroveň grafu volaní. V opačnom prípade - na úroveň histórie Git repozitáru
4. **Kombo box s výberom vizualizácie - prepínanie sa medzi jednotlivými možnosťami vizualizácie grafu volaní**

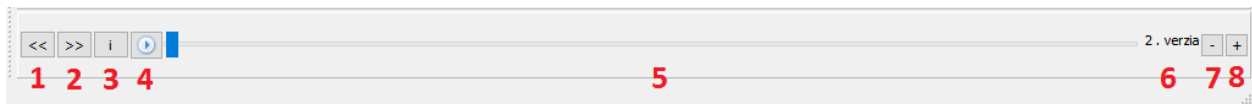
- *LuaStat* - vizualizácia softvérových metrík pomocou analýzy Lua zdrojového kódu
- *Difference* - pohľad na zmeny, ktorými softvér prešiel pri prechode na novú verziu
- *Changes* - aktivovanie filtrovania nad práve aktívnou vizualizáciou

#### 5. 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

#### 6. Kombo box zoznamu možností - možnosti závisia od vybraného filtra

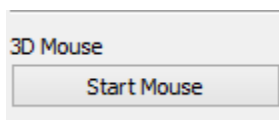
- zoznam autorov s možnosťou zobrazenia zmien všetkých autorov - *All*
- štruktúra - *Files* (zobrazí v grafe volaní len uzly reprezentujúce adresáre a súbory), *Local Functions* (zobrazí rozšírenú možnosť Files spolu s uzlami lokálnych funkcií), *Global Functions* (zobrazia sa uzly možnosti Local Functions spolu s uzlami globálnych funkcií) a *Modules* (zobrazí všetky štruktúry, ktoré sa v grafe nachádzajú)



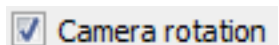
- panel ovládania evolúcie

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

### 1.3.7 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 - povolí uje 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

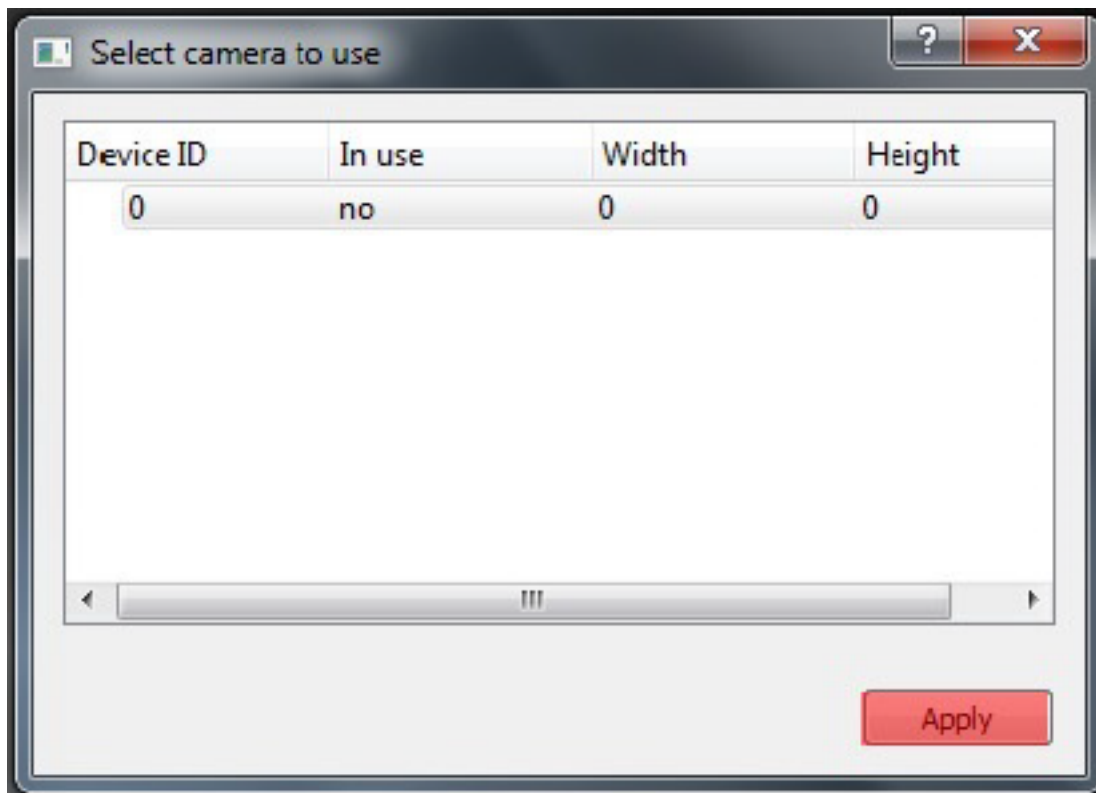
---

Start Leap - zapnutie ovládania pomocou Leap Sensor-u

### Okno pre prácu s kamerou

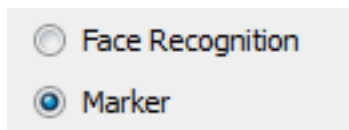
Face Recognition  
 Marker - prispôbenie ľavej strany okna pre ovládanie funkcionality rozpoznávania tvare (pri zapínaní treba zaškrtnúť Camera rotation a Camera enabled)

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.

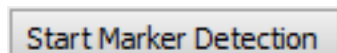


výber snímacieho zariadenia

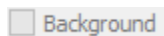
- okno pre



- 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 snímání ako pozadie pre graf

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

**Marker is behind** - prepínanie medzi pohybom podľa značky ako keby sa kamera pozerala na používateľa a naopak

**Correction** - 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

**NoVideo** - vypnutie/zapnutie zobrazenia videa

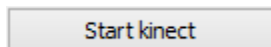
Toto prepínanie a vypnutie zobrazenia videa má vplyv len na zobrazenie v rámci tohto ovládacieho okna „Face Recognition“ and „Marker Detection“ a neovplyvňuje to ani voľbu kamery pre video pozadie.



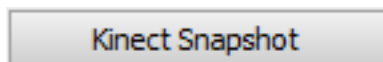
- Interakcia s vizualizáciou v obohatenej realite

- 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



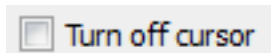
- zapnutie detekcie



- 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 hĺbky, nie vertikálne alebo horizontálne)



Turn off zoom

- vypne možnosť približovania

 Aruco

- nastavenie práce s arucom

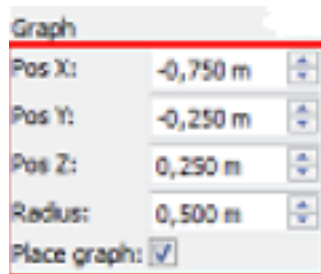
- zobrazenie okna projekčného zobrazenia

Projector		
FOV:	30,00 Å°	<input type="button" value="↑"/>
Pos X:	-0,665 m	<input type="button" value="↑"/>
Pos Y:	-1,345 m	<input type="button" value="↑"/>
Pos Z:	0,825 m	<input type="button" value="↑"/>
Dir X:	-0,085 m	<input type="button" value="↑"/>
Dir Y:	1,345 m	<input type="button" value="↑"/>
Dir Z:	-0,587 m	<input type="button" value="↑"/>

- v projekčnom 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°	<input type="button" value="↑"/>
Pos X:	-1,880 m	<input type="button" value="↑"/>
Pos Y:	-0,950 m	<input type="button" value="↑"/>
Pos Z:	1,720 m	<input type="button" value="↑"/>
Dir X:	1,130 m	<input type="button" value="↑"/>

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



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



- 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"

### 1.3.8 Hlavné okno



- filtrovanie hrán

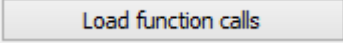


- filtrovanie uzlov


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'"

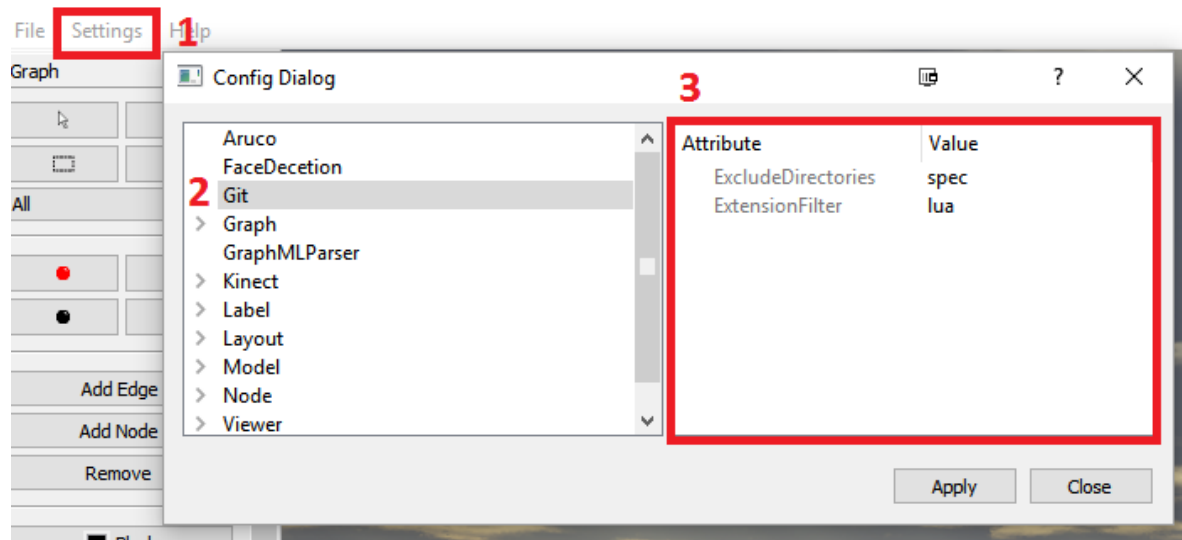
Filter je navrhnutý pre grafovú vizualizáciu softvéru s využitím softvérových metrick jazyka Lua a je vyhodnotený po stlačení klávesu „Enter“.

 - 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

### 1.3.9 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 ”,”

## 1.4 Používateľská príručka pre Vuzix a Leap

### 1.4.1 Stereoskopické 3D s Vuzix okuliarmi

#### Okuliare

Ako prvé je potrebné pripojiť Vuzix okuliare k počítaču (HDMI a 2x USB)

Okuliare by mali byť rozpoznané ako zobrazovacie zariadenie, pre ktoré nastavte duplikáciu obrazu z monitora, na ktorom beží 3Dsoftviz.

Nakoniec pomocou ovládacieho zariadenia vyvolajte menu a nastavte režim zobrazovania obrazu na side-by-side 3D alebo top-bottom 3D, poprípade upravte aj jas a kontrast obrazu.

(menu sa zobrazí len priamo na displejoch okuliarov)

#### 3Dsoftviz

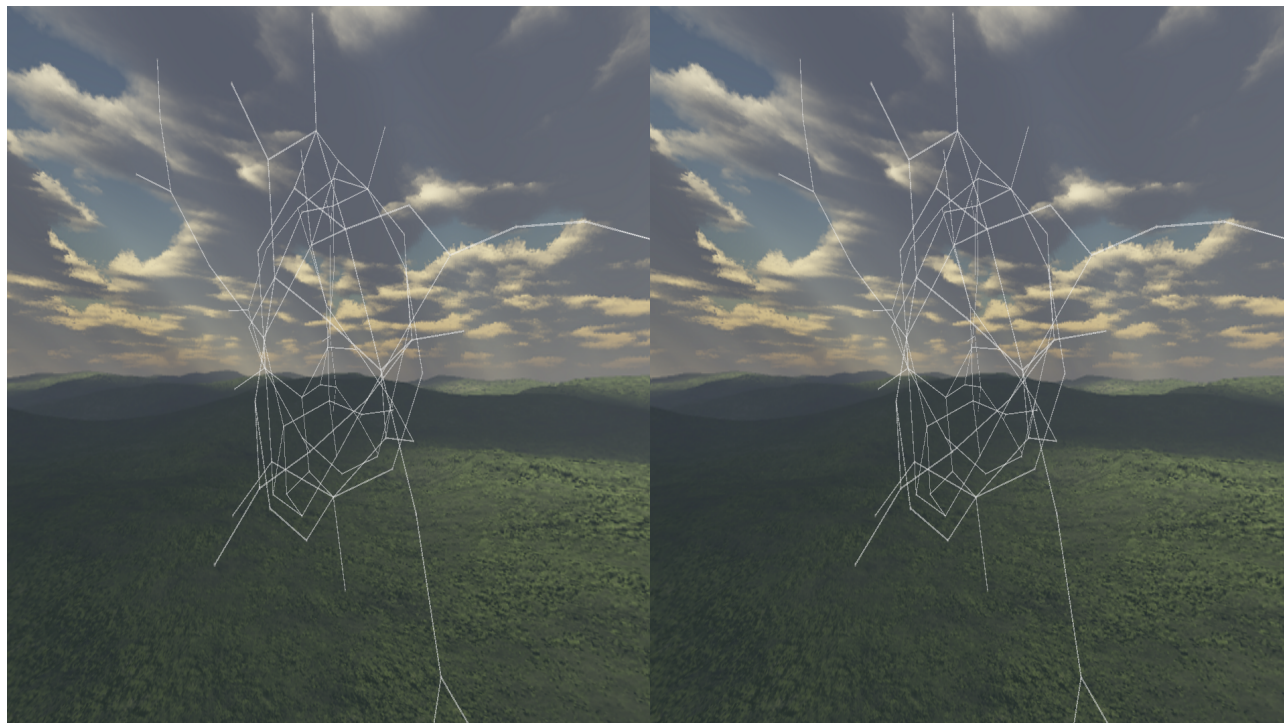
Pre správne zobrazenie je potrebné nastaviť 3Dsoftviz do režimu plnej obrazovky stlačením klávesy L a skryť panel s nástrojmi stlačením klávesy T

Následne je možné pomocou klávesy G prepínať medzi režimami:

- normálne zobrazenie
- top-bottom 3D zobrazenie
- side-by-side 3D zobrazenie

Vyberte zobrazenie korešpondujúce s nastavením okuliarov.

Klávesami H a J je možné upravovať nastavenie vzdialenosti očí, ktorá mení posun medzi obrazmi ktorým sa dosahuje 3D efekt. Úprava je po 0.01m a aktuálna hodnota sa zobrazuje v konzole.



## 1.4.2 Ovládanie rukami s Leap senzorem

### Leap:

Ako prvé je potrebné pripojiť Leap senzor k počítaču (1x USB). Ďalej je potrebné nainštalovať oficiálny softvér k Leap senzoru z oficiálnej stránky.

### 3Dsoftviz:

V ľavej lište s nástrojmi je v karte *More features* možnosť *Start Leap*, po stlačení ktorej začne snímanie rúk nad senzorom.

Podporované je nasledovné ovládanie:

- Ľavá ruka s vystretými prstami – pohyb kamery dopredu
- Ľavá ruka so skrčenými prstami – zastavenie pohybu kamery
- Pravá ruka s vystretými prstami – kamera sa otáča podľa a naklonenia dlane

Snímanie je možné zastaviť opätovným stlačením rovnakého tlačidla ktoré ma počas snímania text *Stop Leap*.



## PRE VÝVOJÁROV

Pomocné materiály pre vývojárov, pracujúcich na projekte.

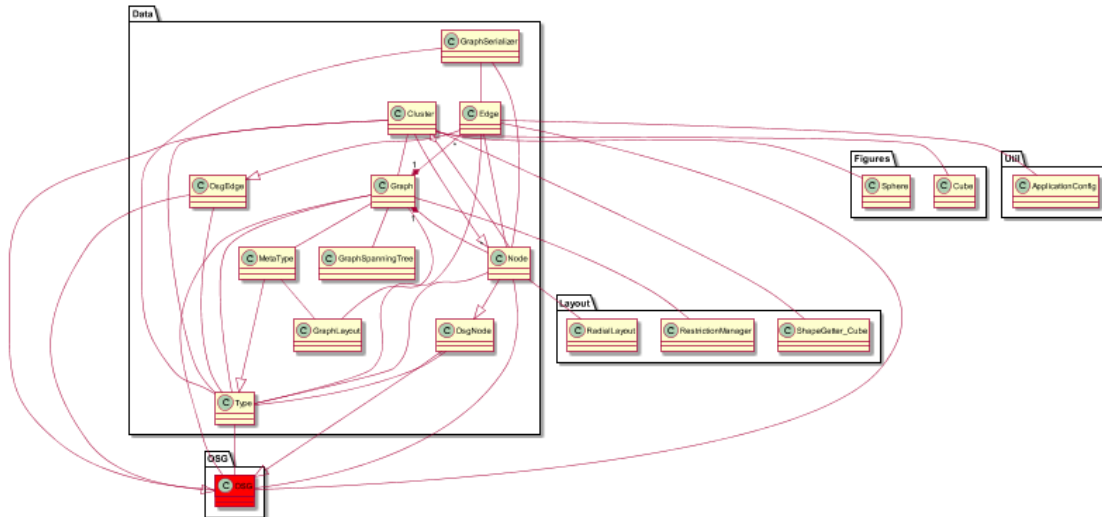
### 2.1 UML diagramy

#### 2.1.1 Structural

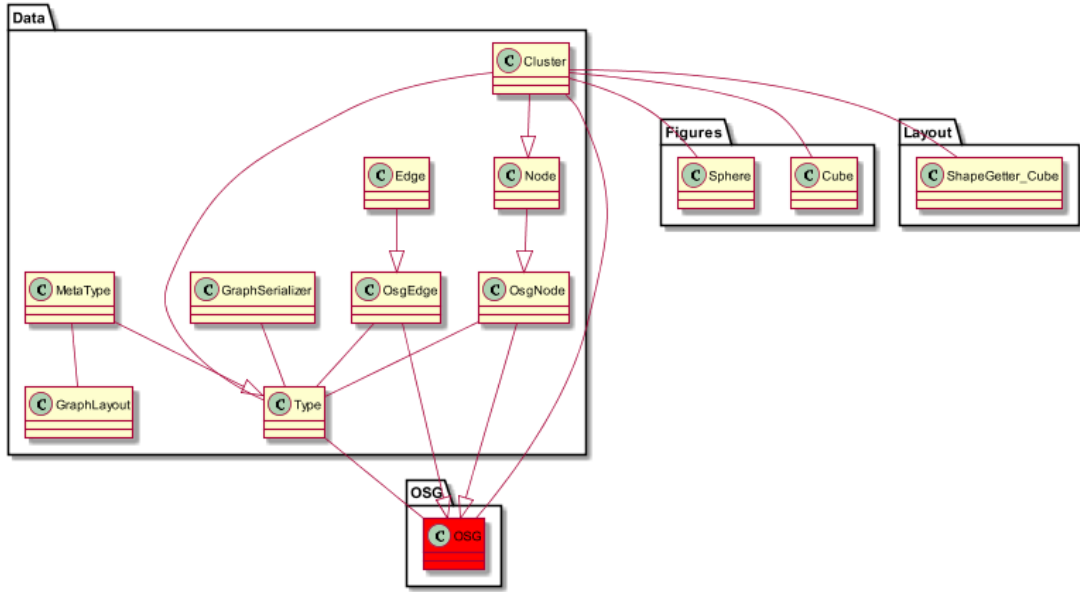
##### Class Diagrams

##### Data

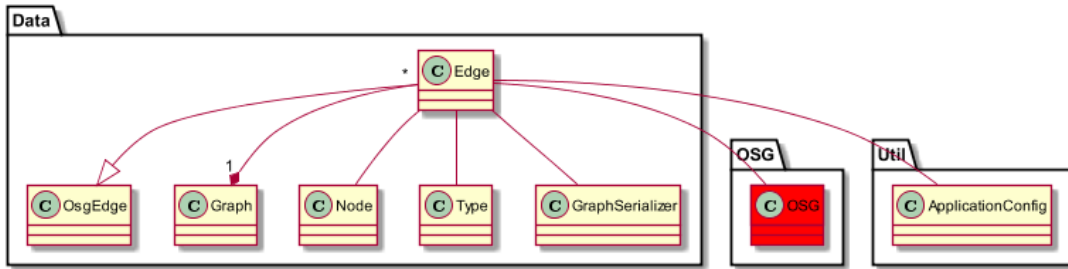
Data:



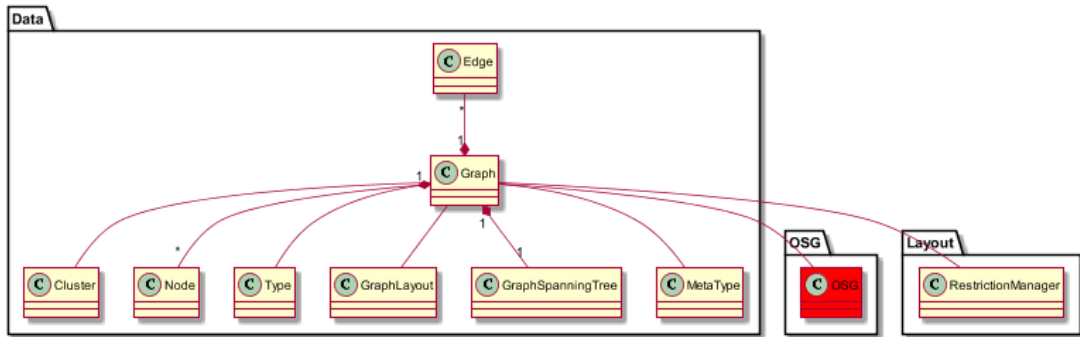
Data1:



Data.Edge:

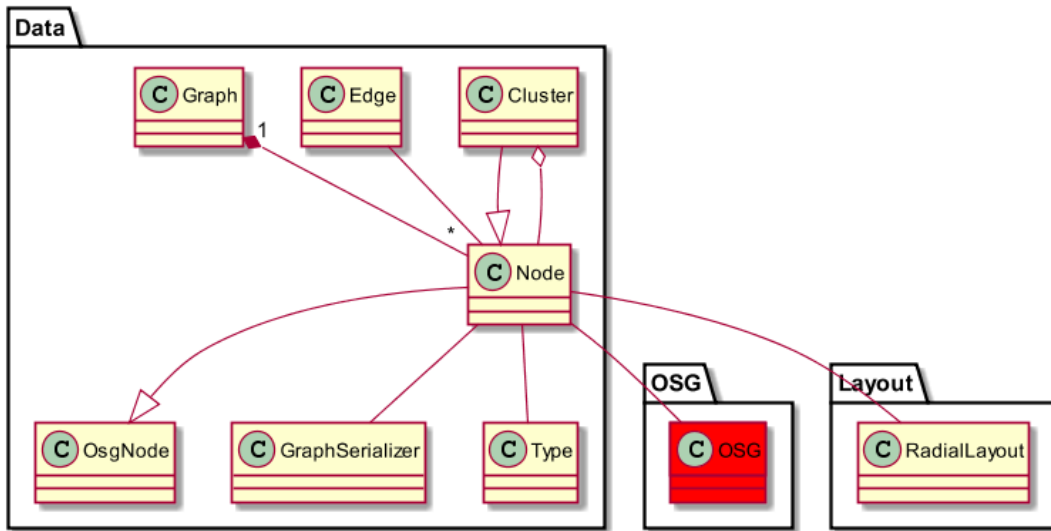


Data.Graph:



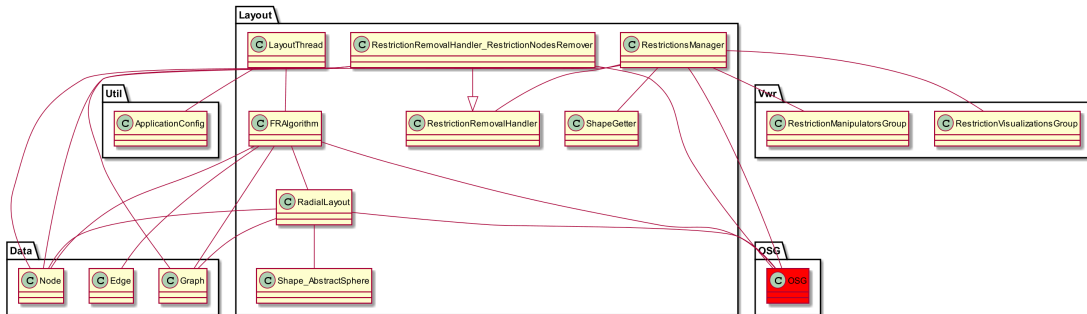
Data.Node:



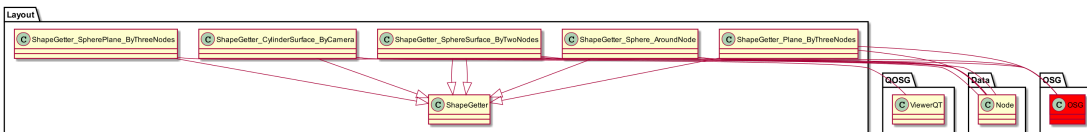


## Layout

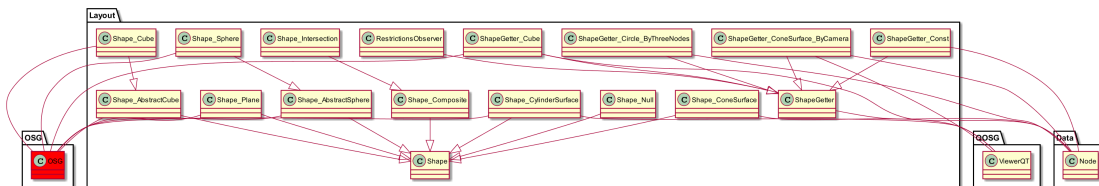
Layout1:



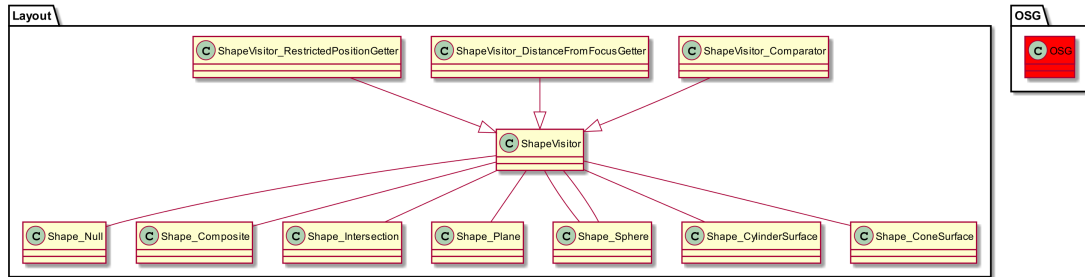
Layout2:



Layout3:



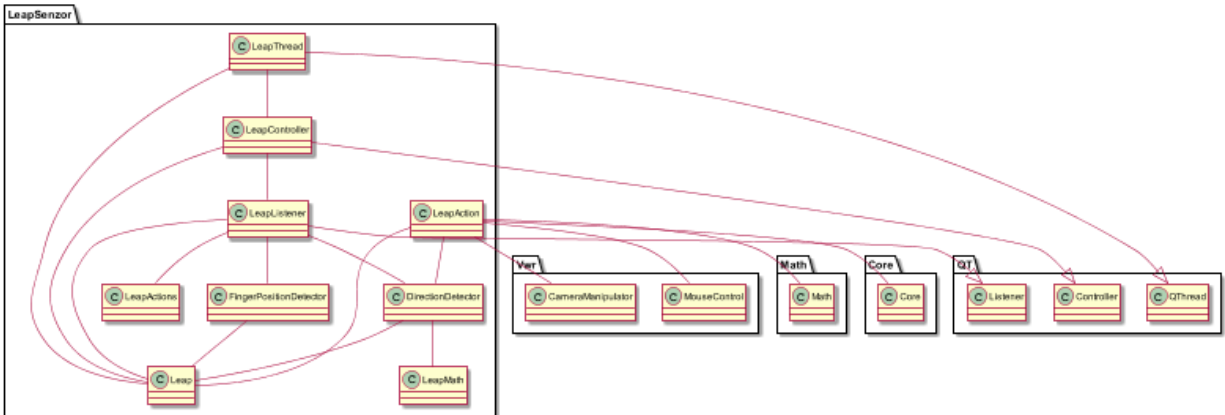
Layout-Visitor:



Layout - complete:



## LeapSensor



LeapThread dedi of QThread - include Leap, LeapController - vytvara samostatny thread pre leap senzora

LeapListener dedi od Listener - include Leap, LeapActions, DirectionDetector, FingerPositionDetector - prekonava funkcie Listenera ktory sa staraju o zmenu stavu leap senzora

LeapController dedi od Controller - include Leap, LeapListener - spusta a zastavuje pocuvanie leap senzora

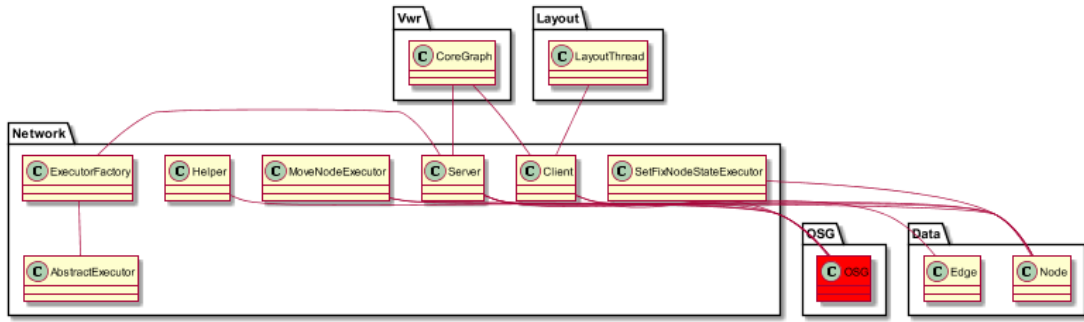
LeapAction - include DirectionDetector, MouseControl, CameraManipulator, Leap, Math, Core - Obsahuje funkcie ktore sa vykonaju na zaklade detekovaneho gesta

FingerPositionDetector - include Leap - obsahuje funkcie na pracu s poziciou prstov

DirectionDetector - include Leap, LeapMath - obsahuje funkcie na pracu s natocenim roznych casti ruky

## Network

Network:

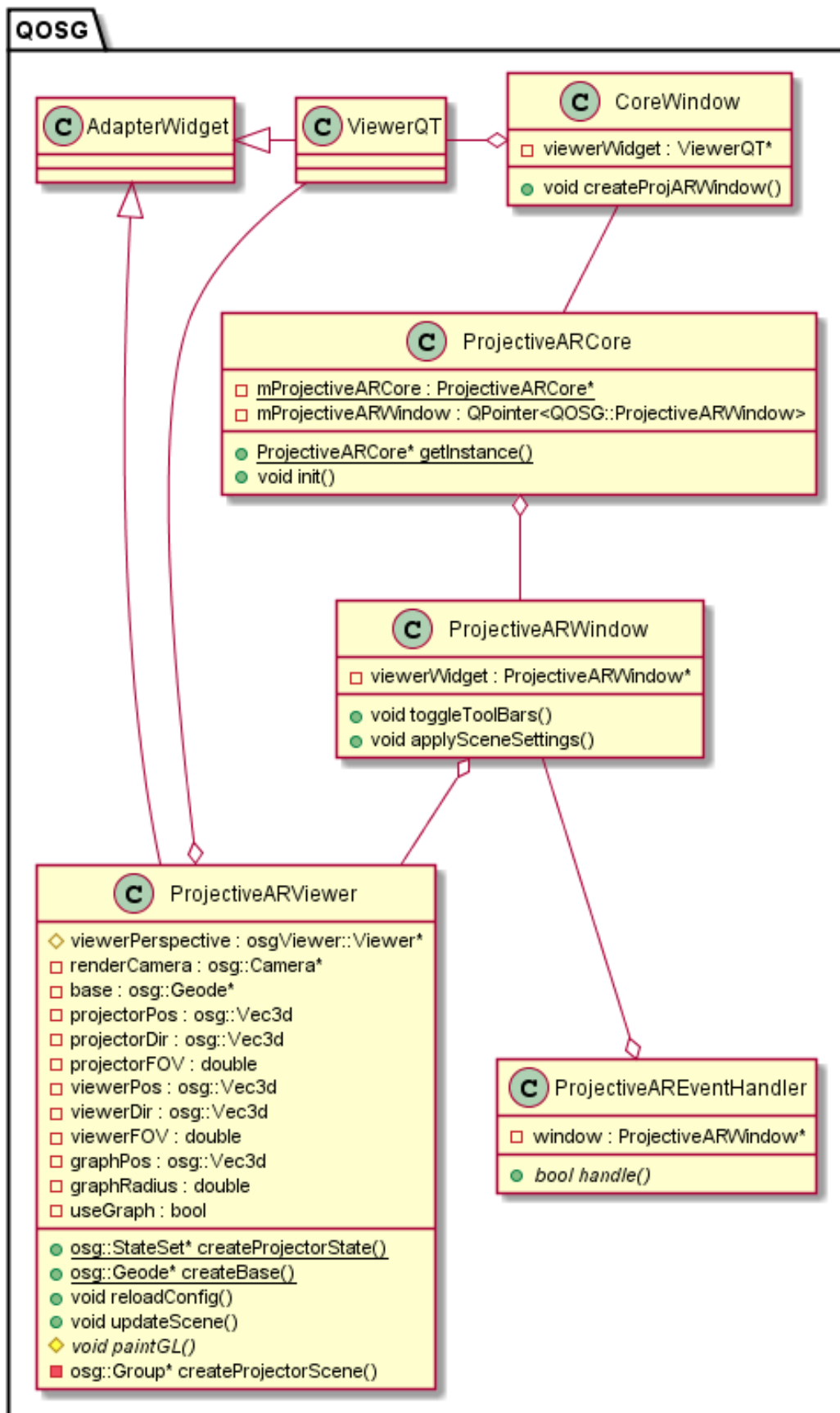


Network (complete):

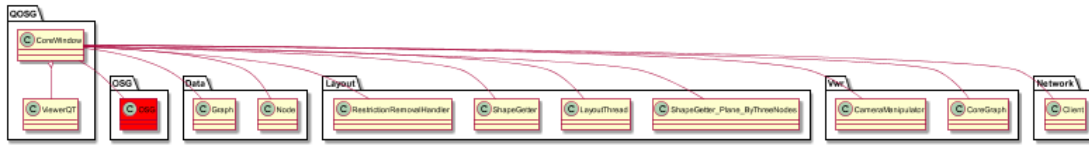


## QOSG

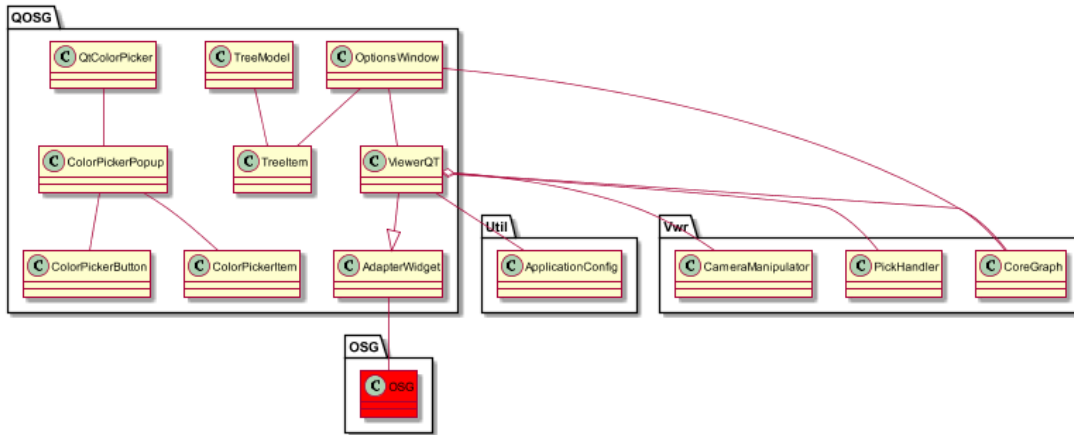
Nove pridane triedy:



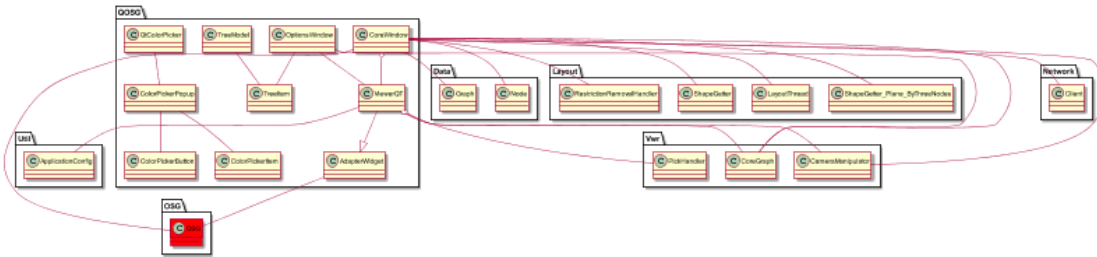
CoreWindow:



QOSG:

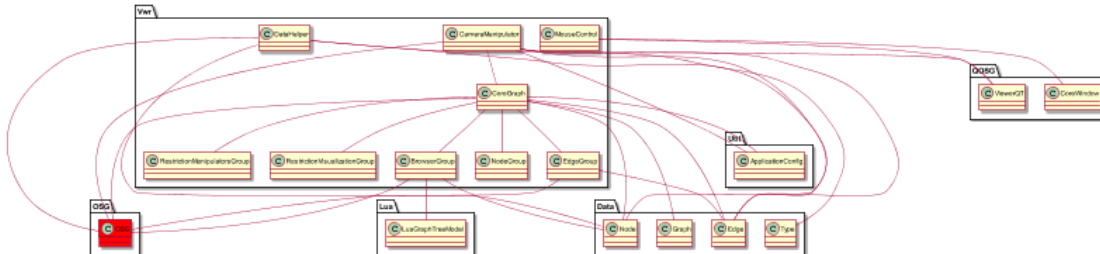


QOSG complete:

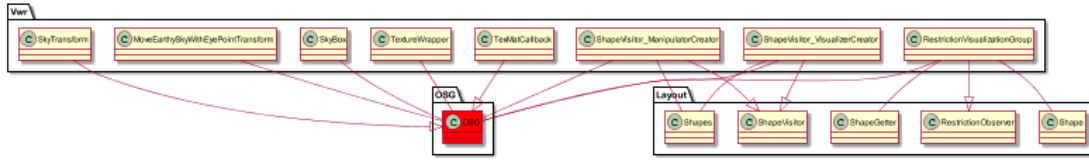


Viewer

Viewer1:



Viewer2:



Viewer3:



Viewer complete:



## GitLib

GitLib a 3DSoftViz architektura:

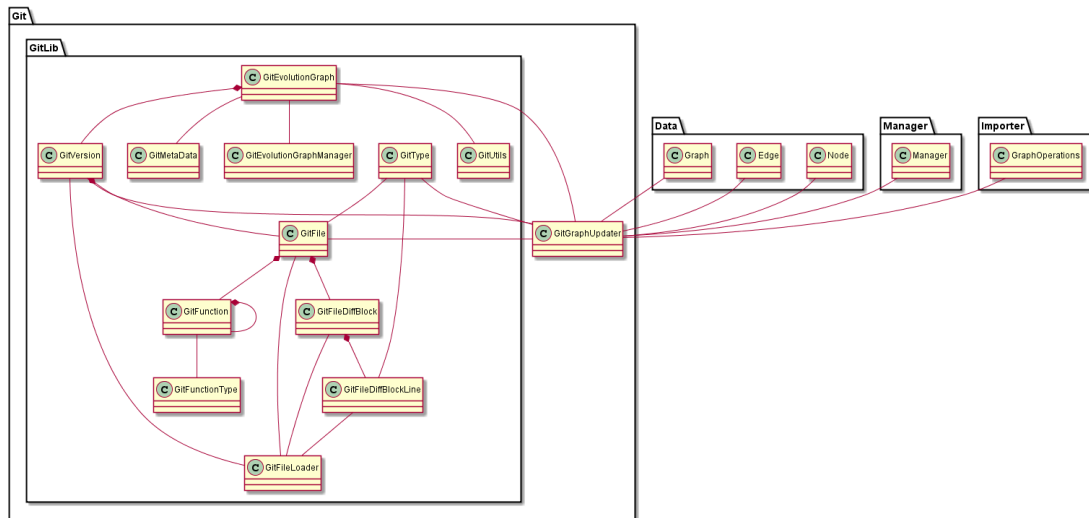
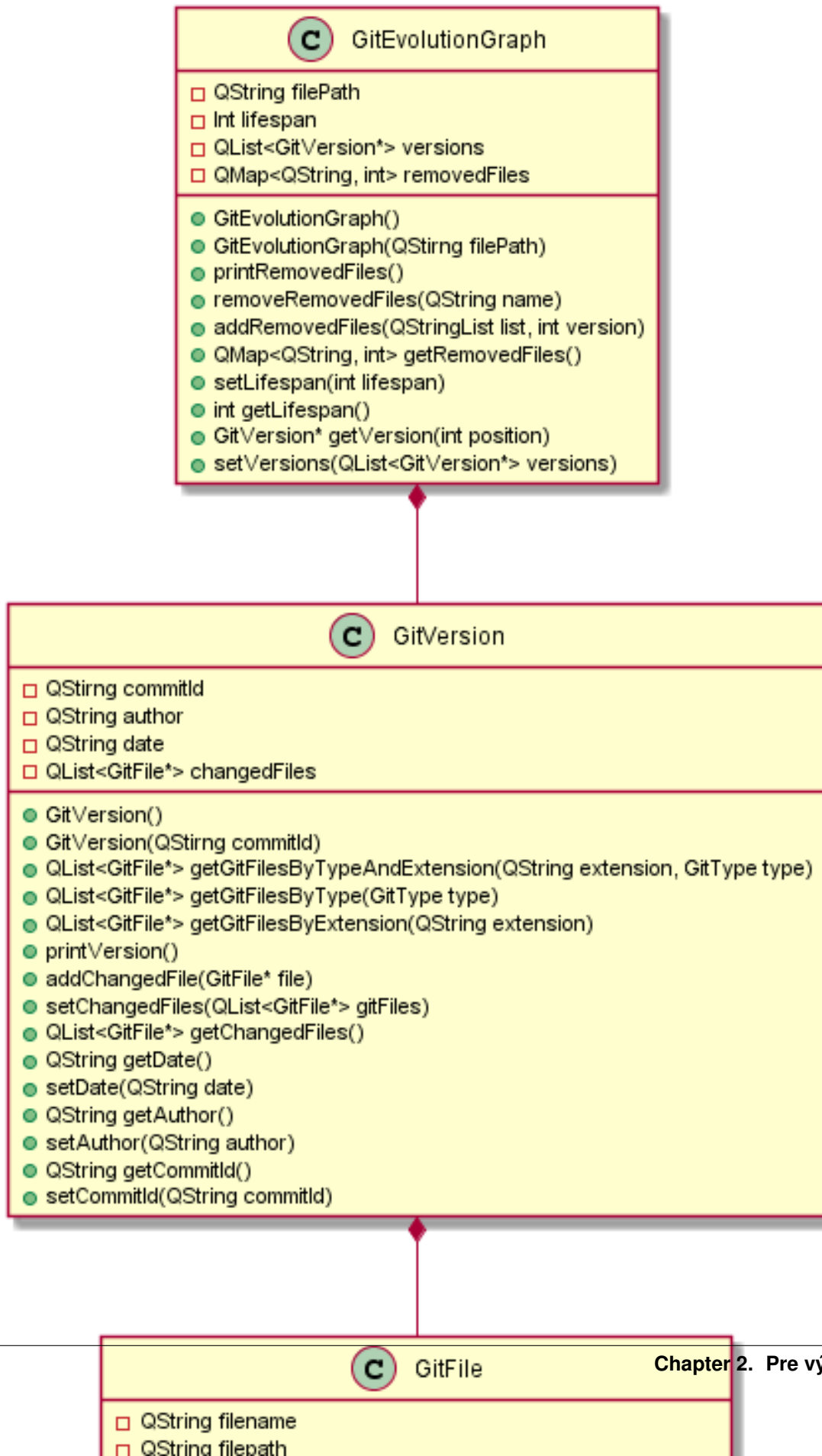


Diagram tried GitLib knihnice:

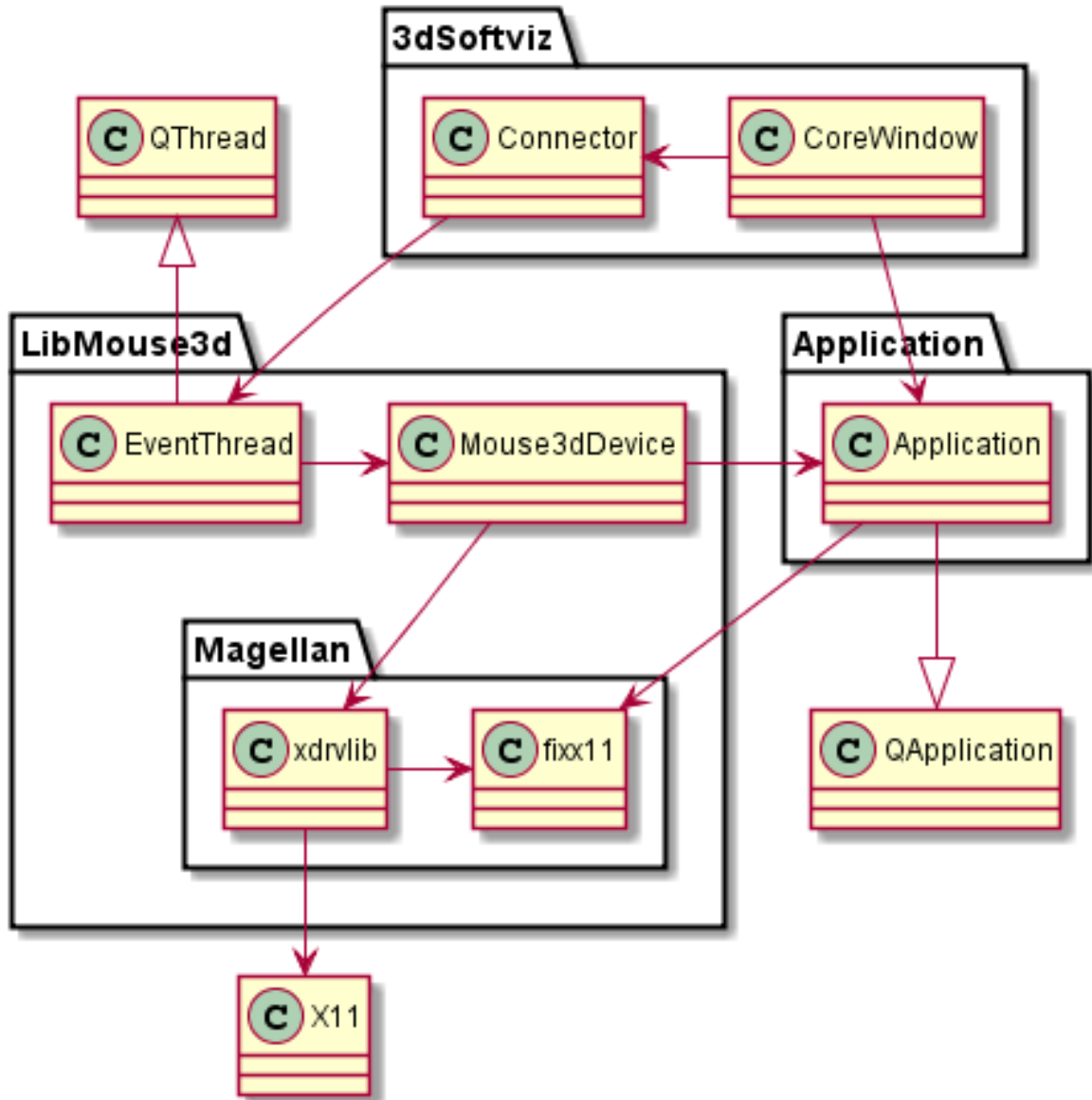




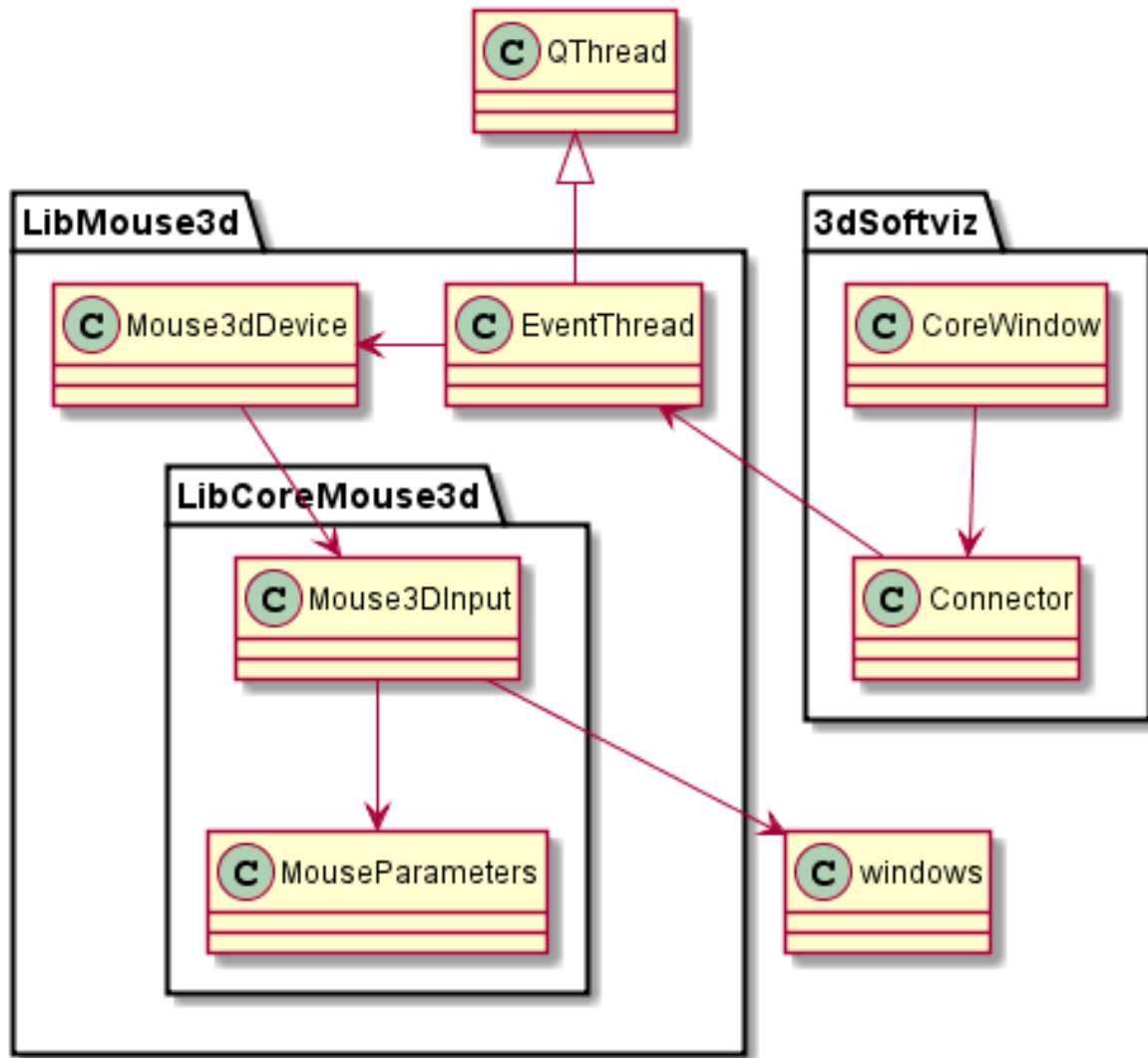


## 3D Mouse

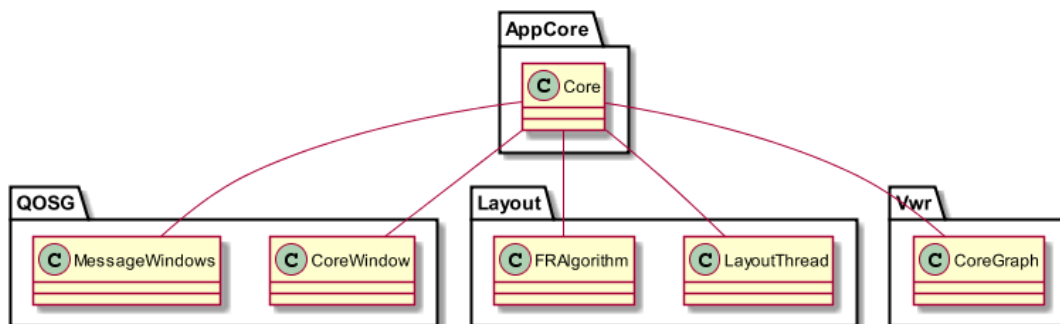
Linux:



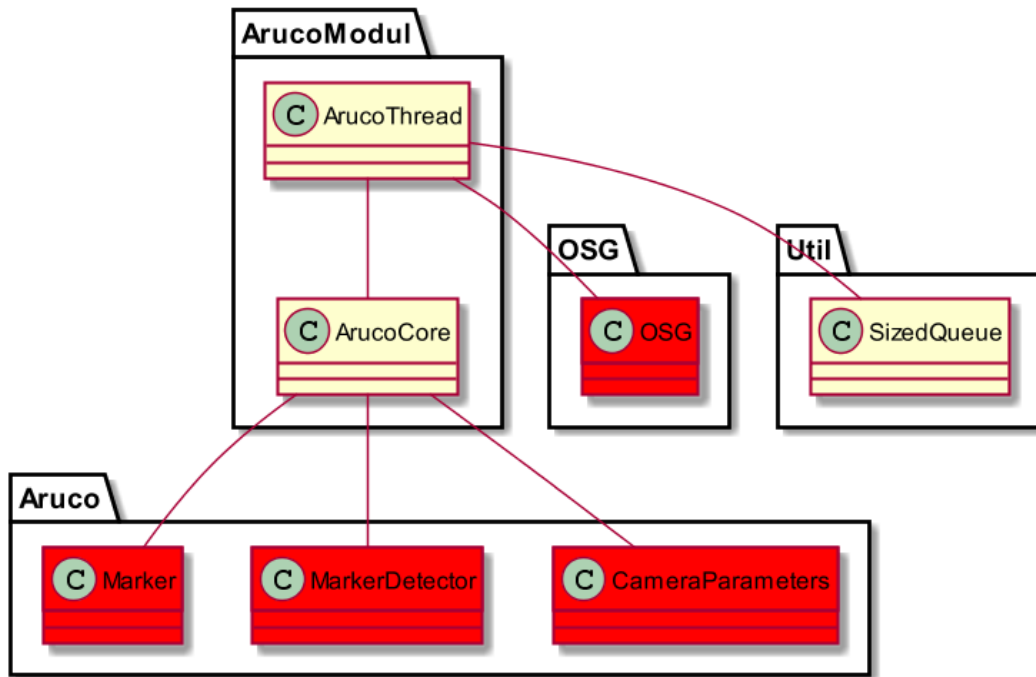
Windows:



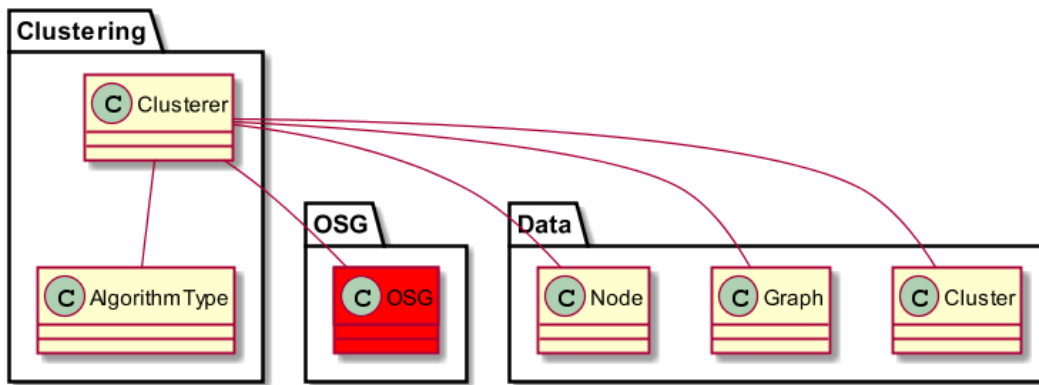
AppCore:



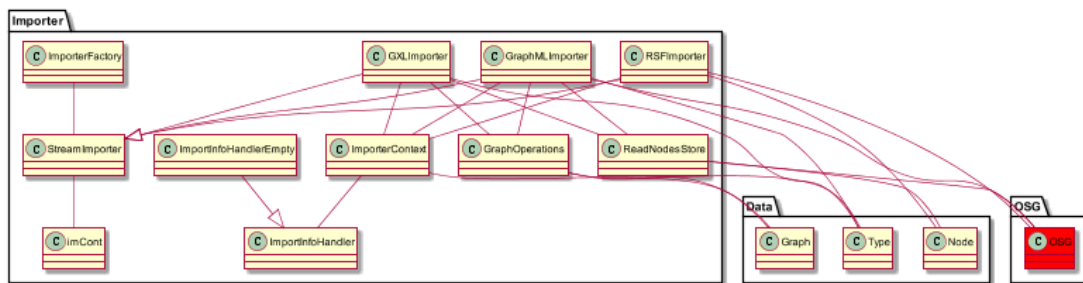
ArucoModul:



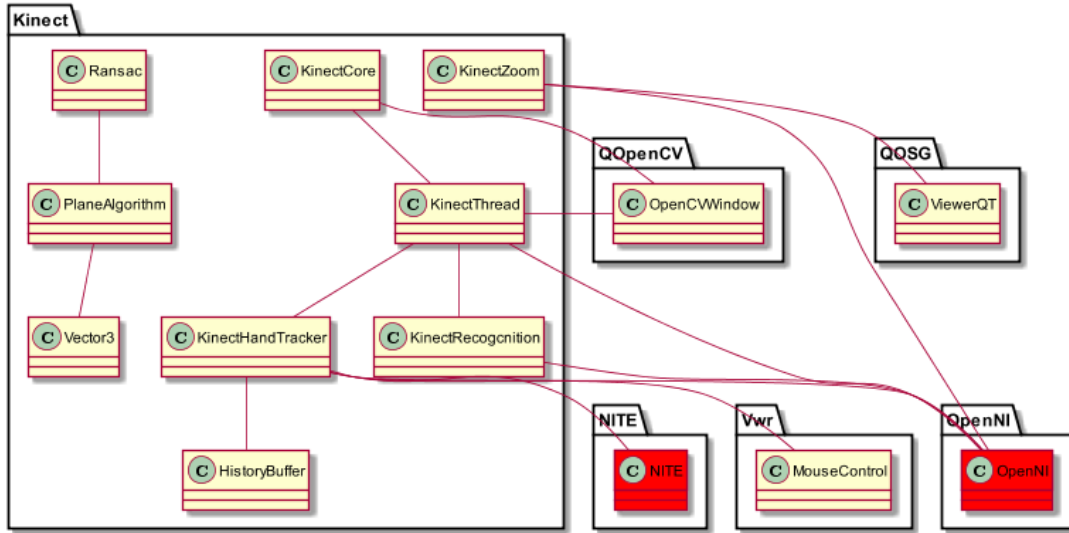
Clustering:



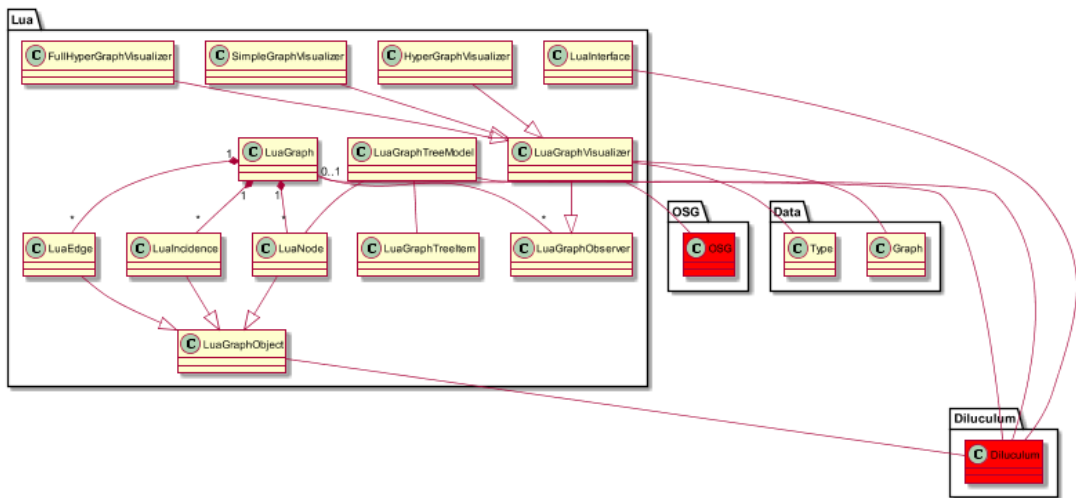
Importer:



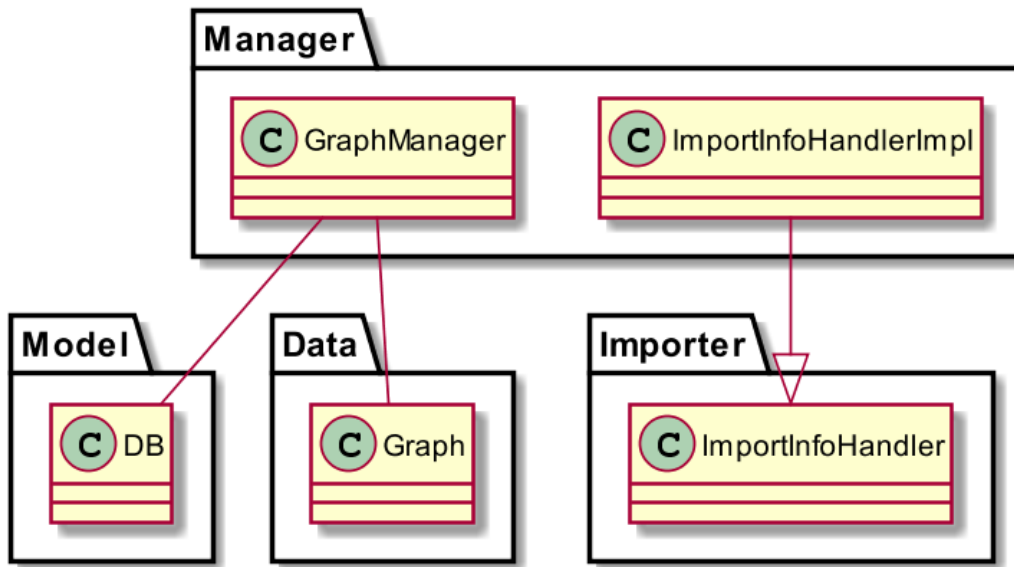
Kinect:



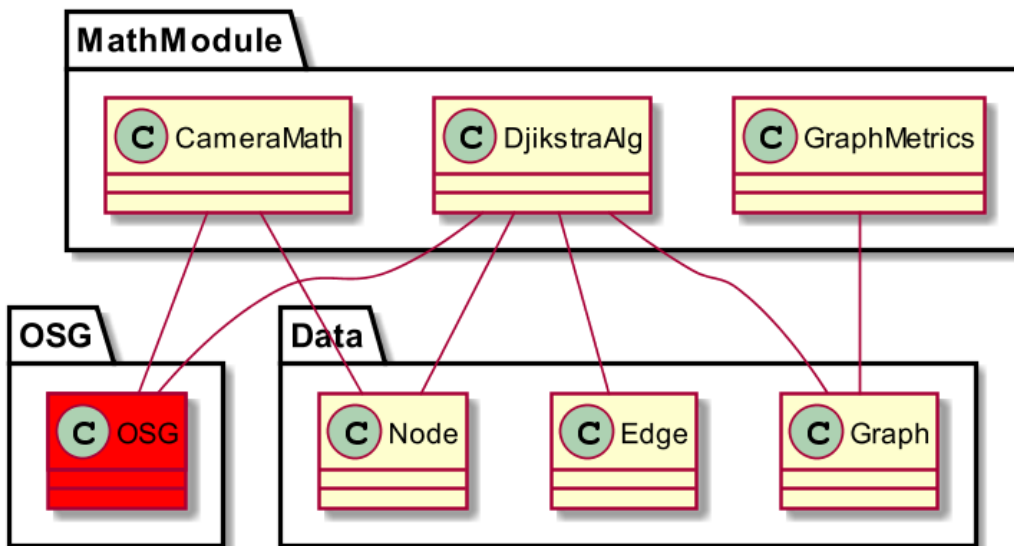
Lua:



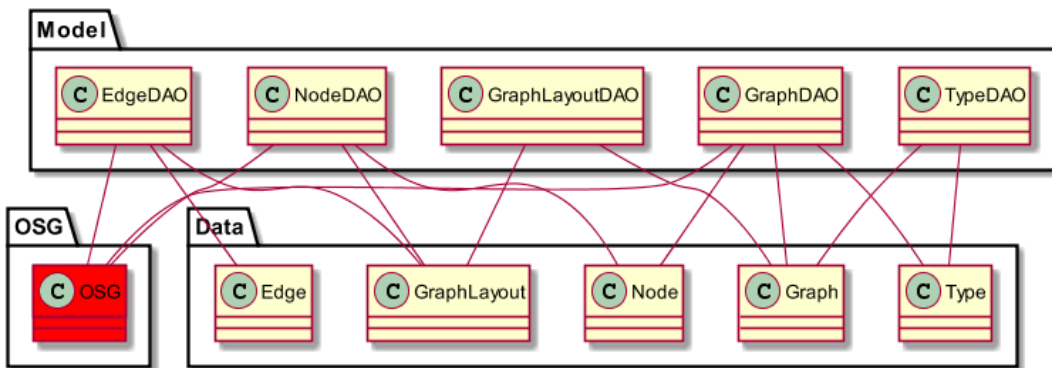
Manager:



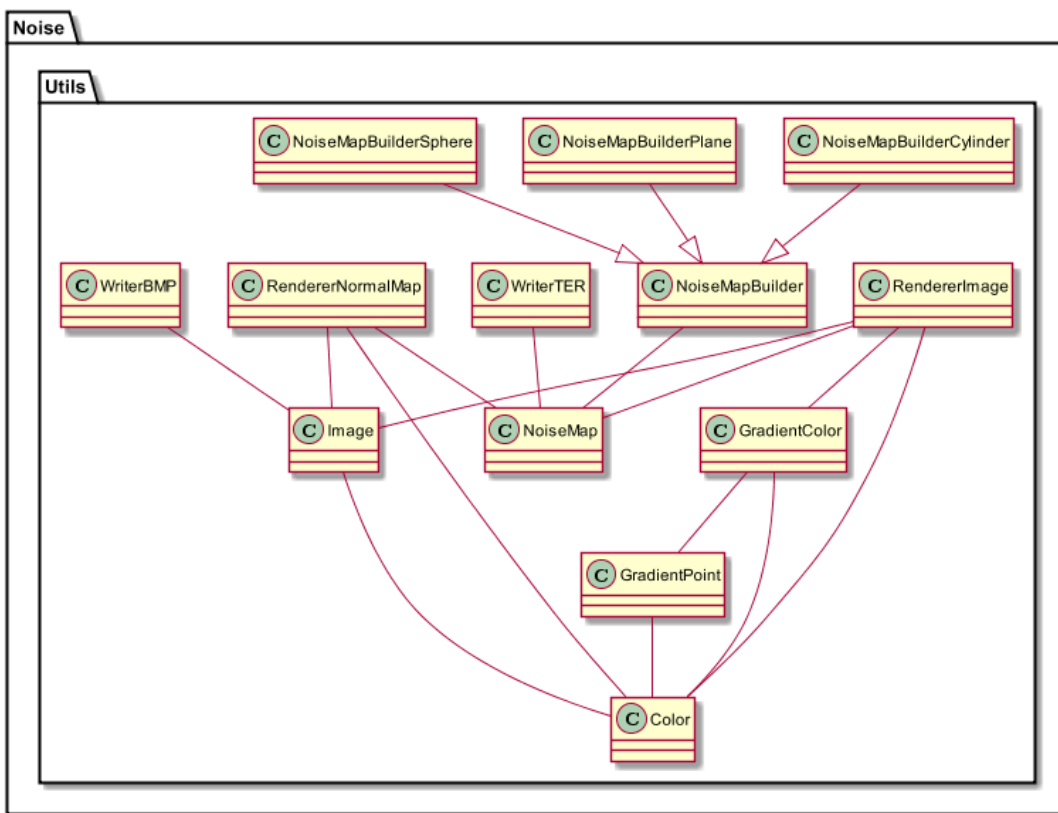
MathModul:



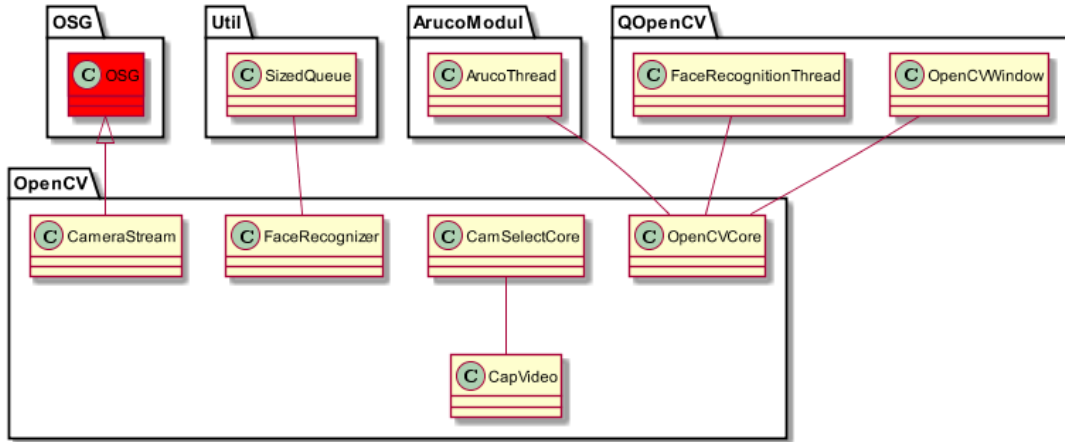
Model:



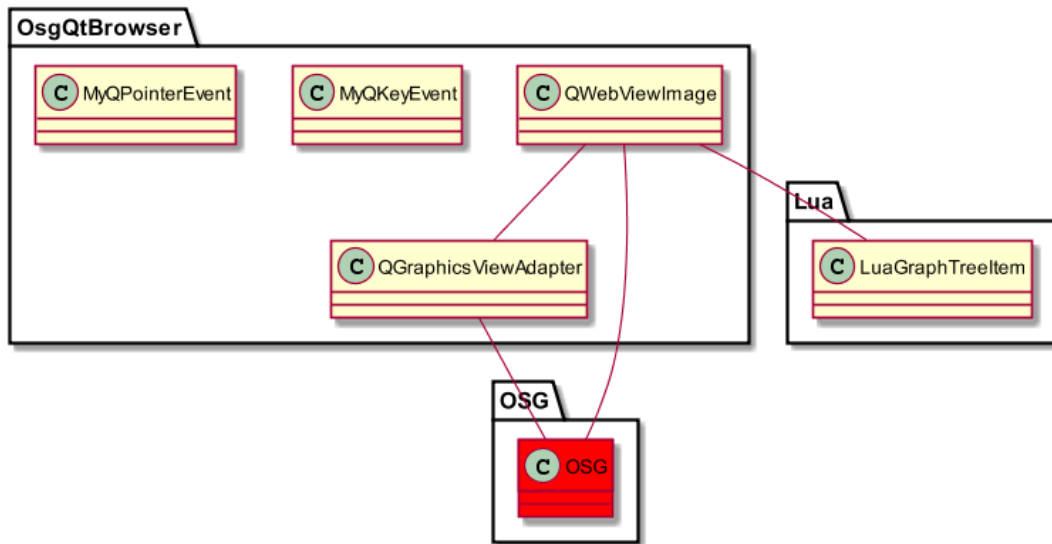
Noise:



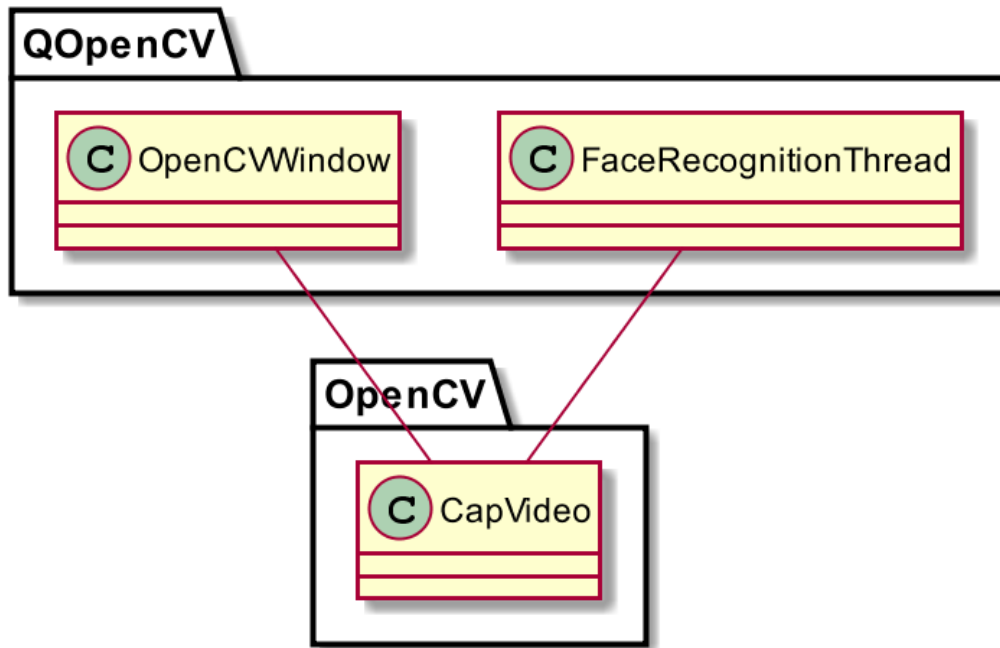
OpenCV:



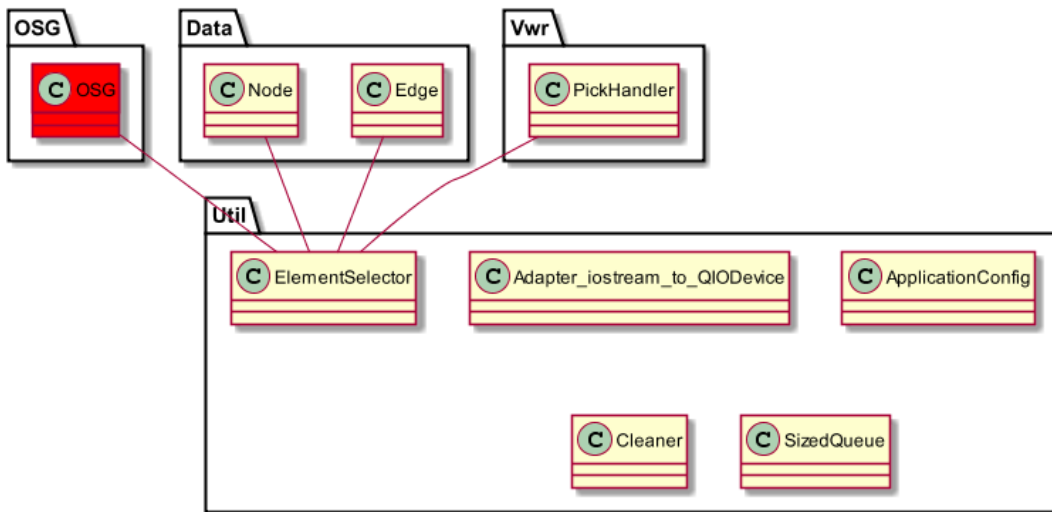
OsgQtBrowser:



QOpenCV:



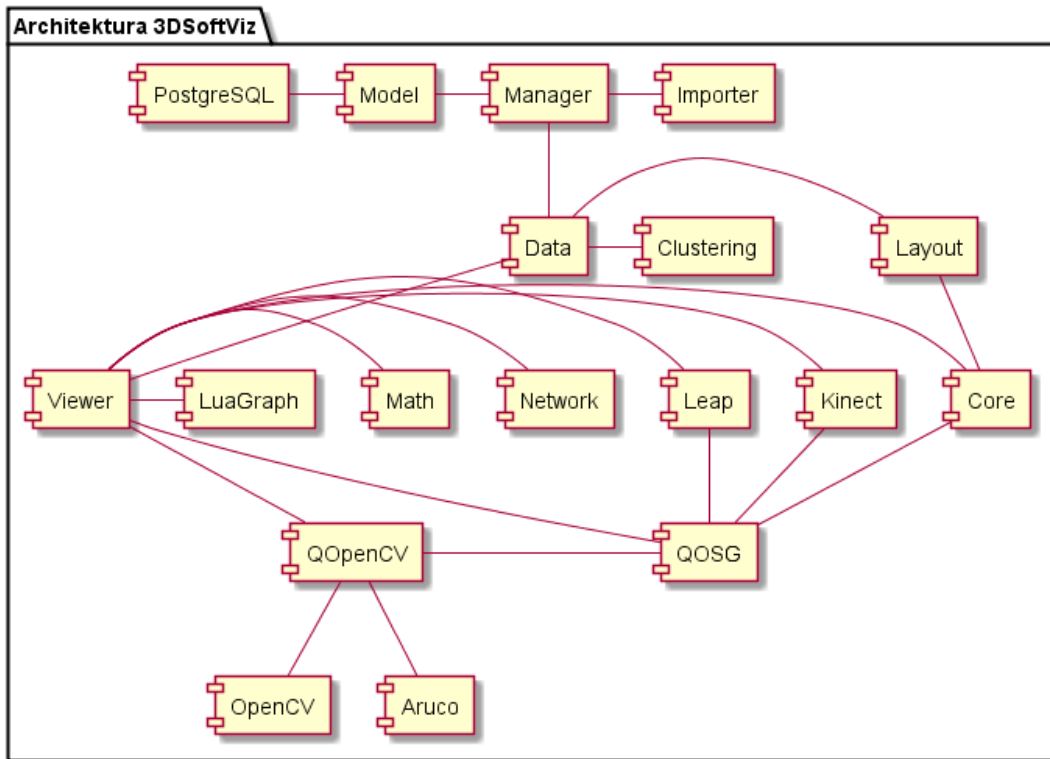
Util:



## Component diagrams

Architektura 3dsoftviz:





## 2.1.2 Behavioral

### Activity diagrams

- 

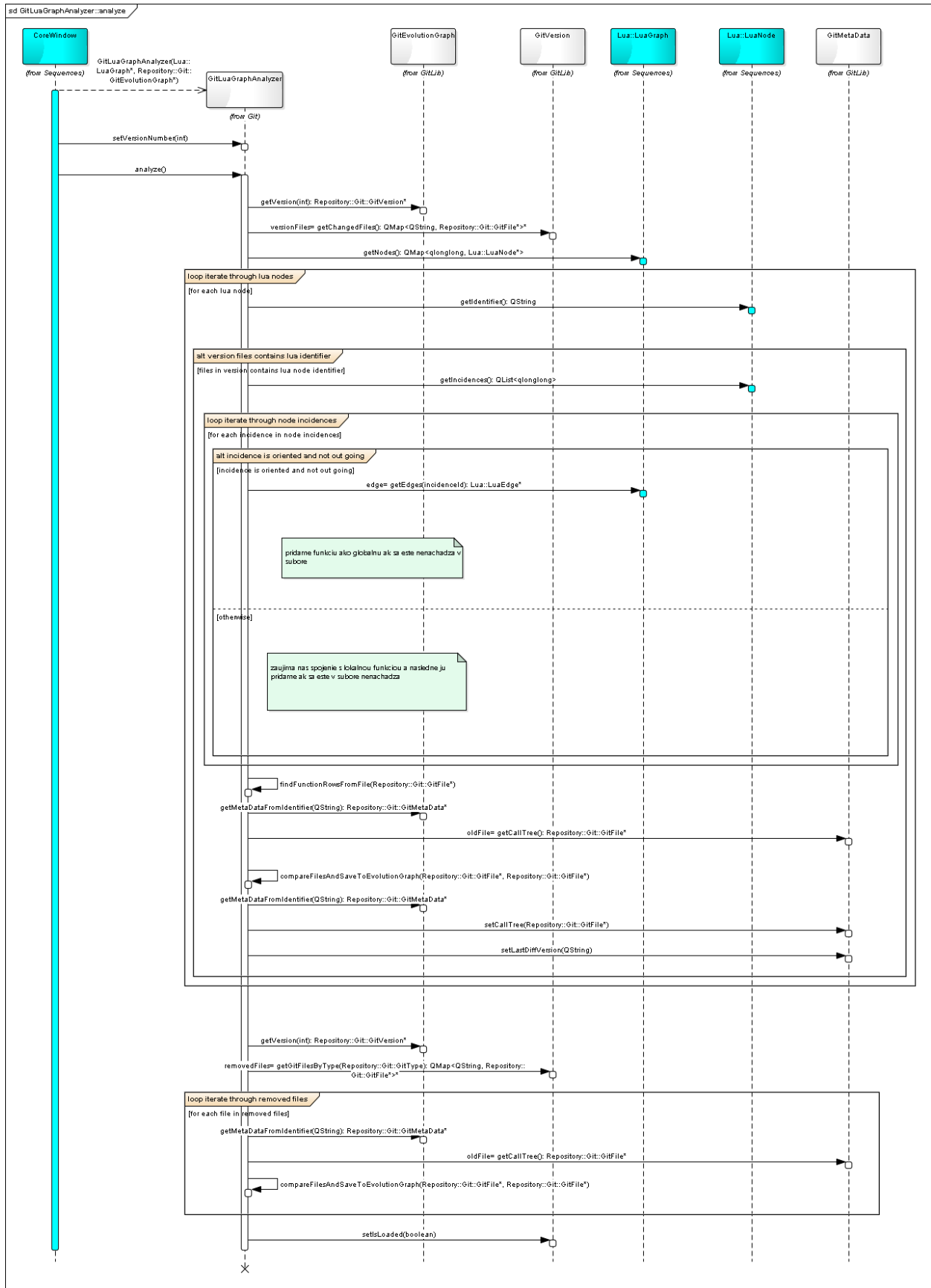
### Use-case diagrams

- 

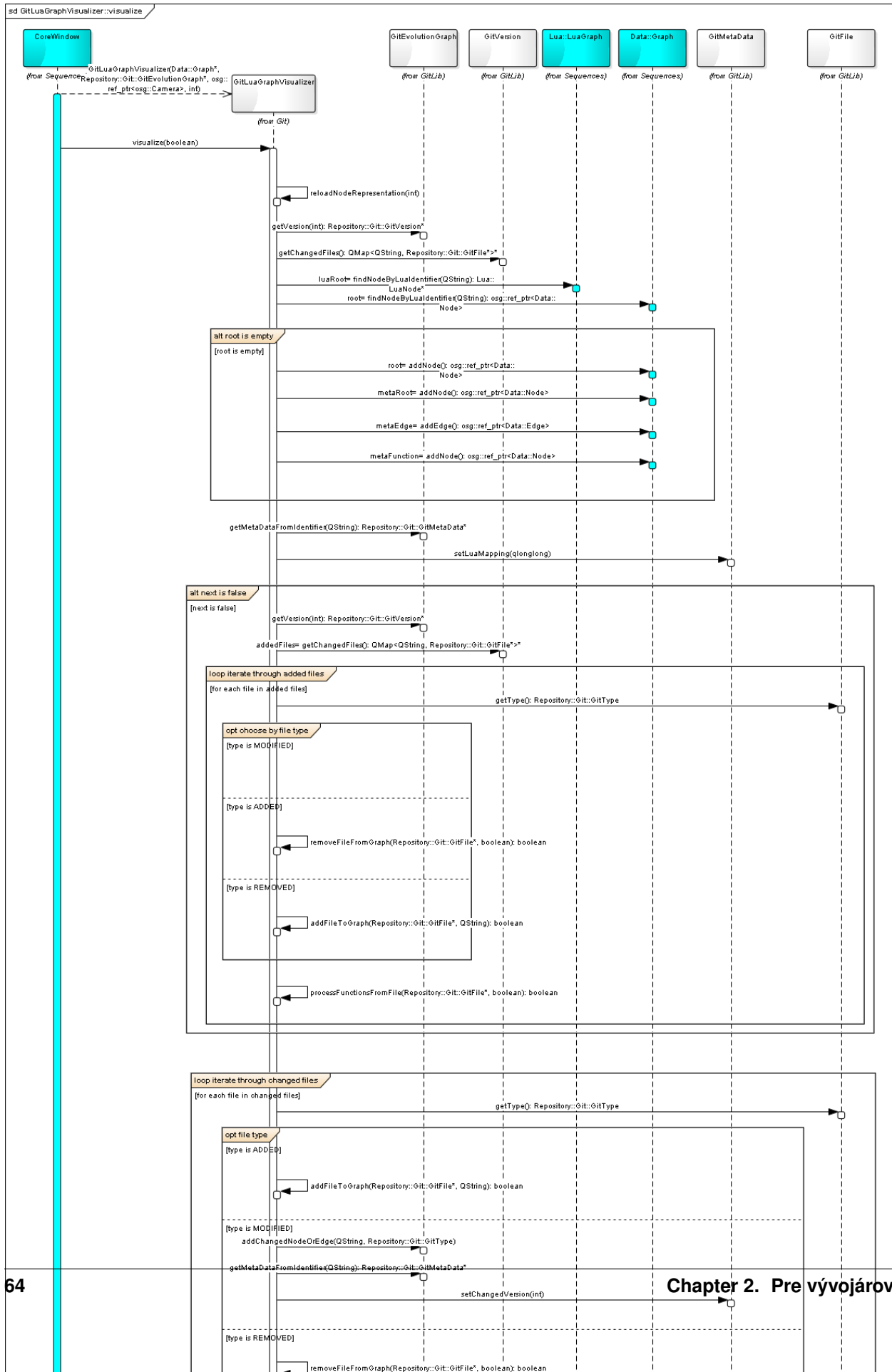
### Sequence diagrams

### GitLua

Git Lua Graph Analyzer:

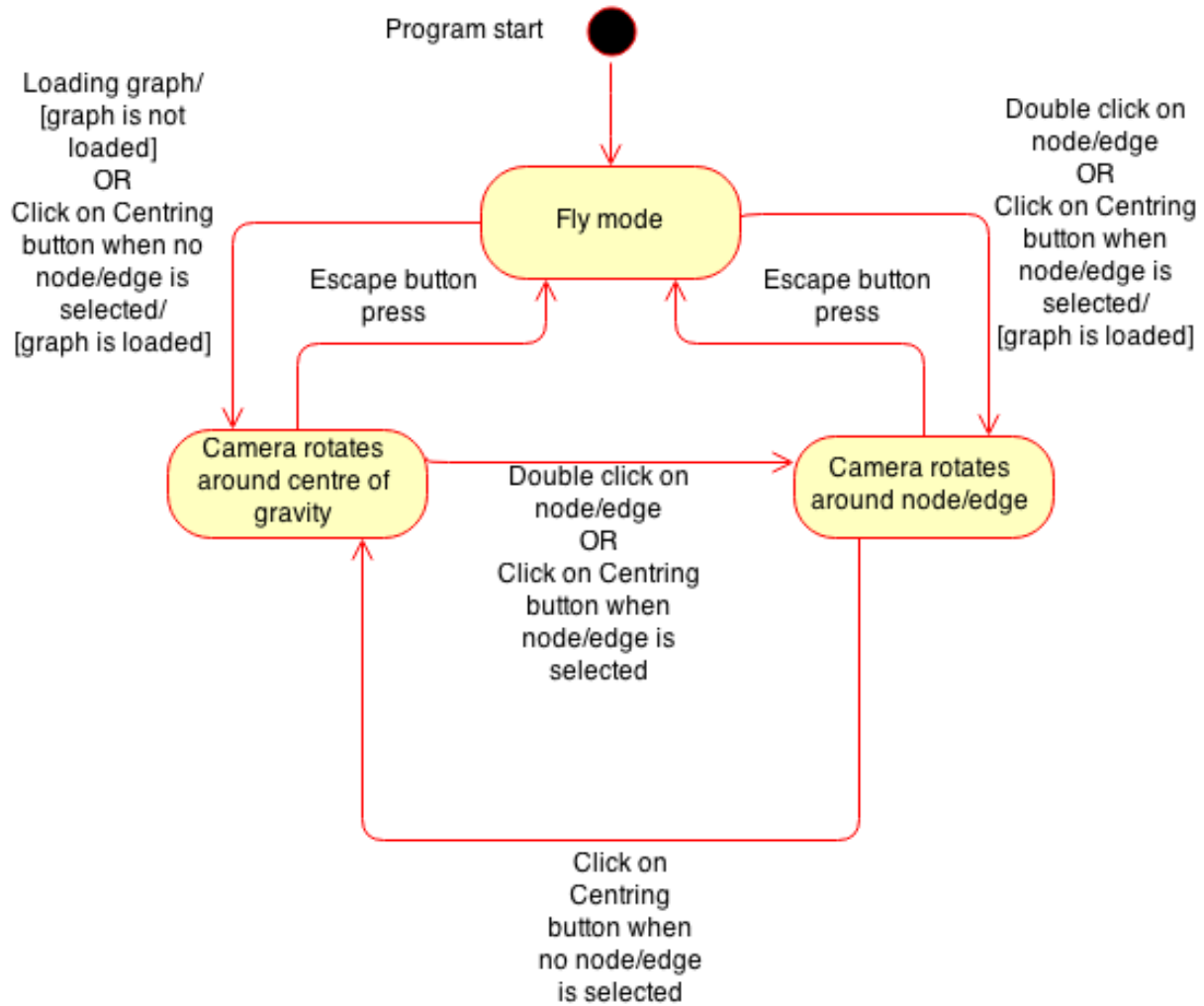


Git Lua Graph Visualizer:



### Statechart diagrams

Camera Movement:



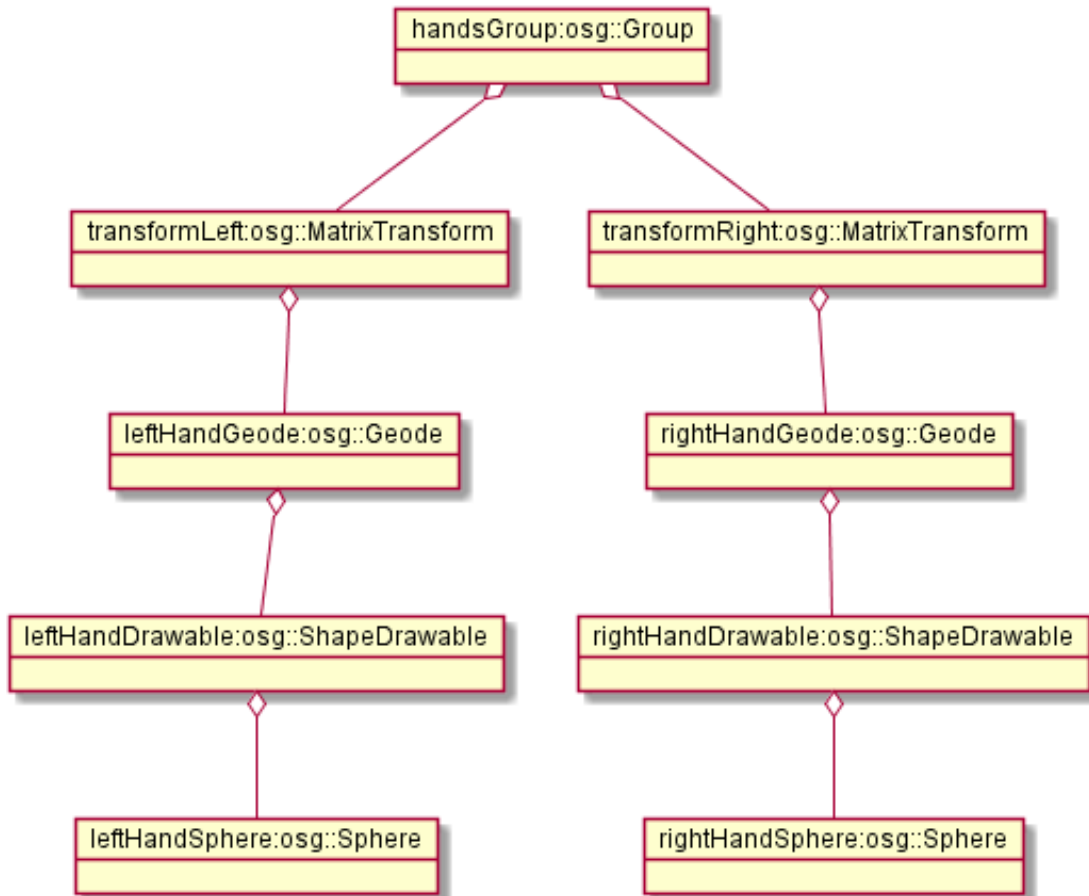
### Object diagrams

Graph structure layer0:

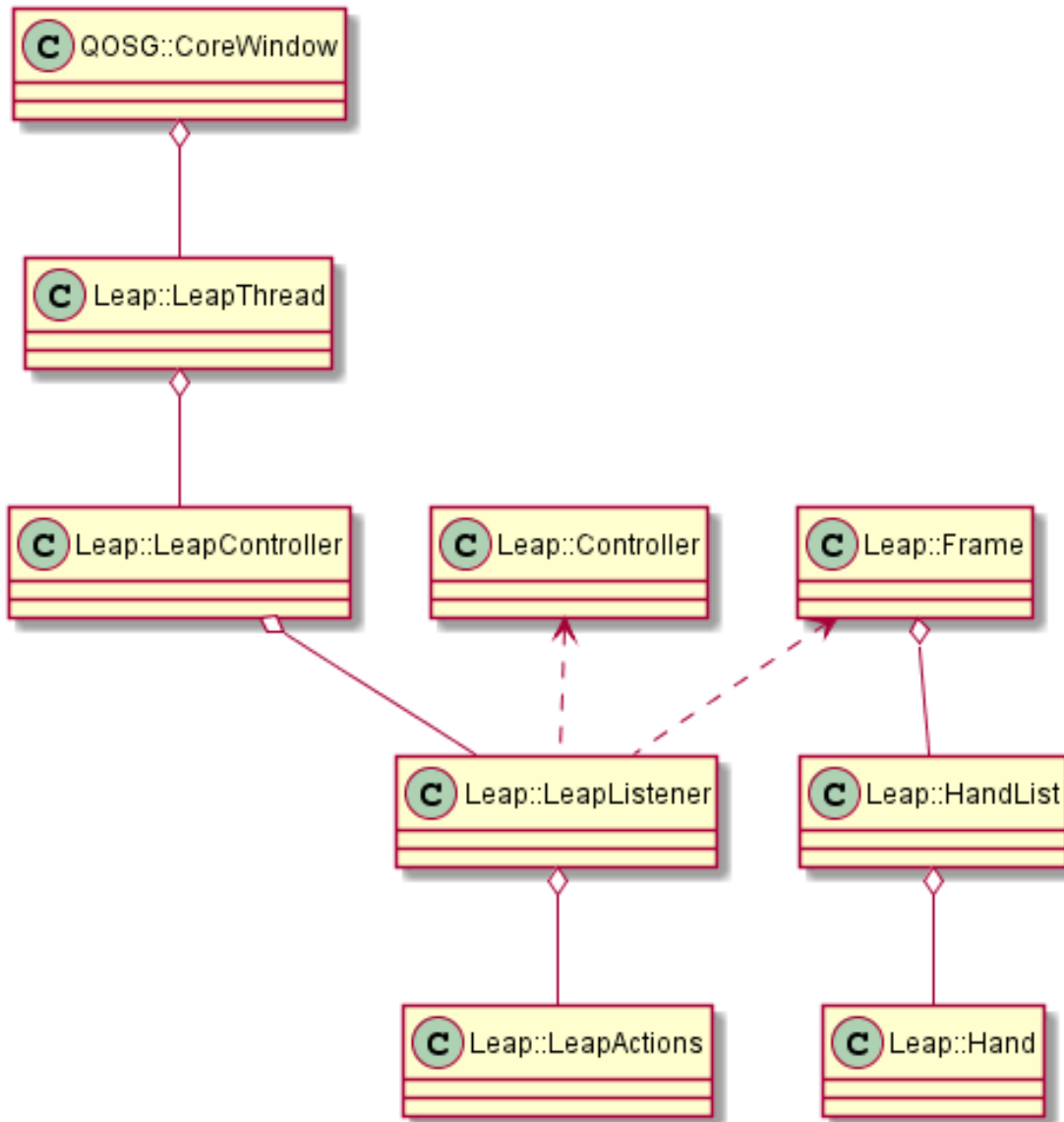


Graph structure layer1:



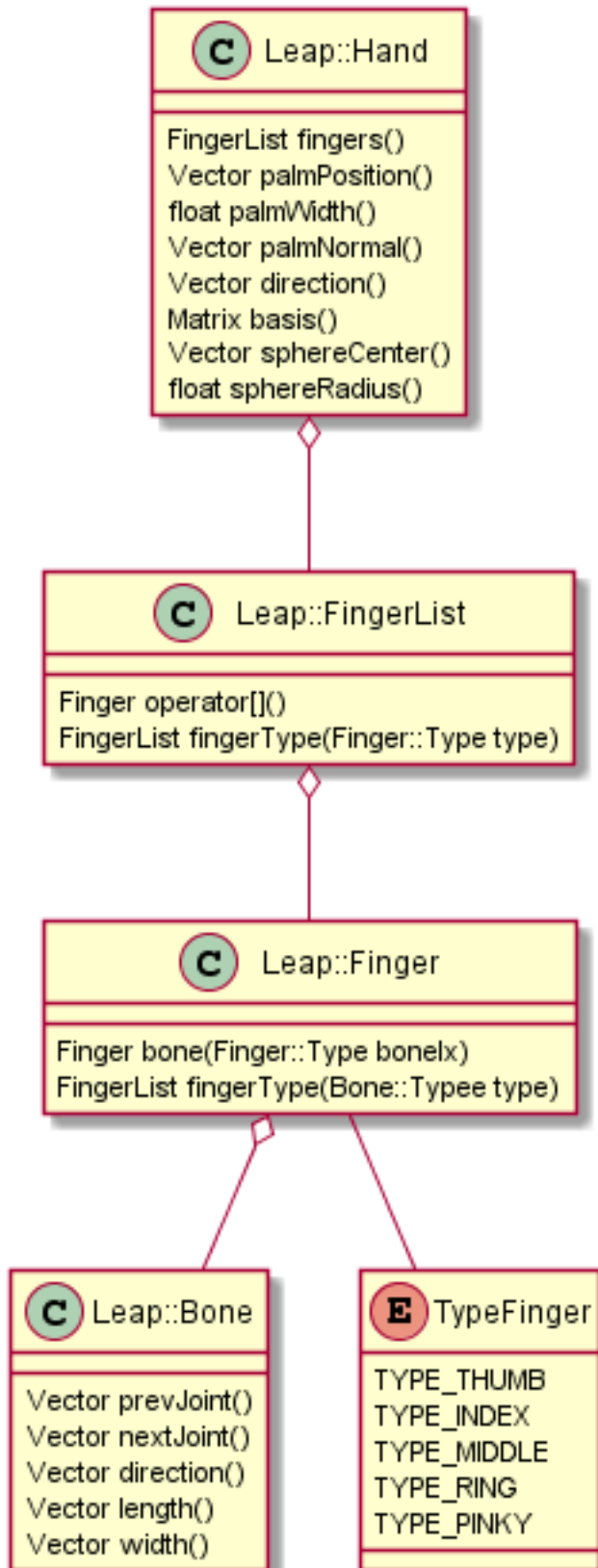


LeapAR Adapter:

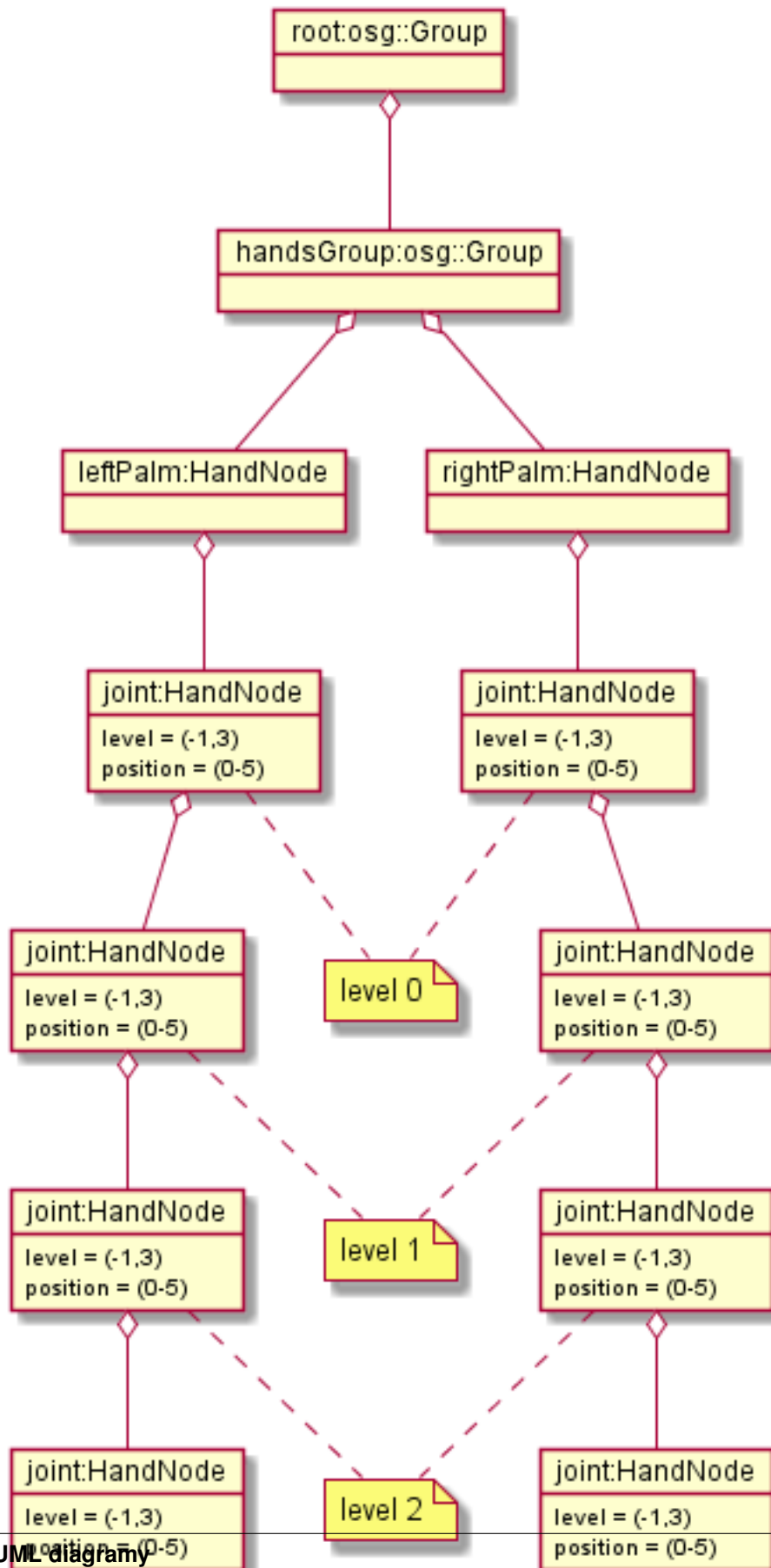


Api leap library:

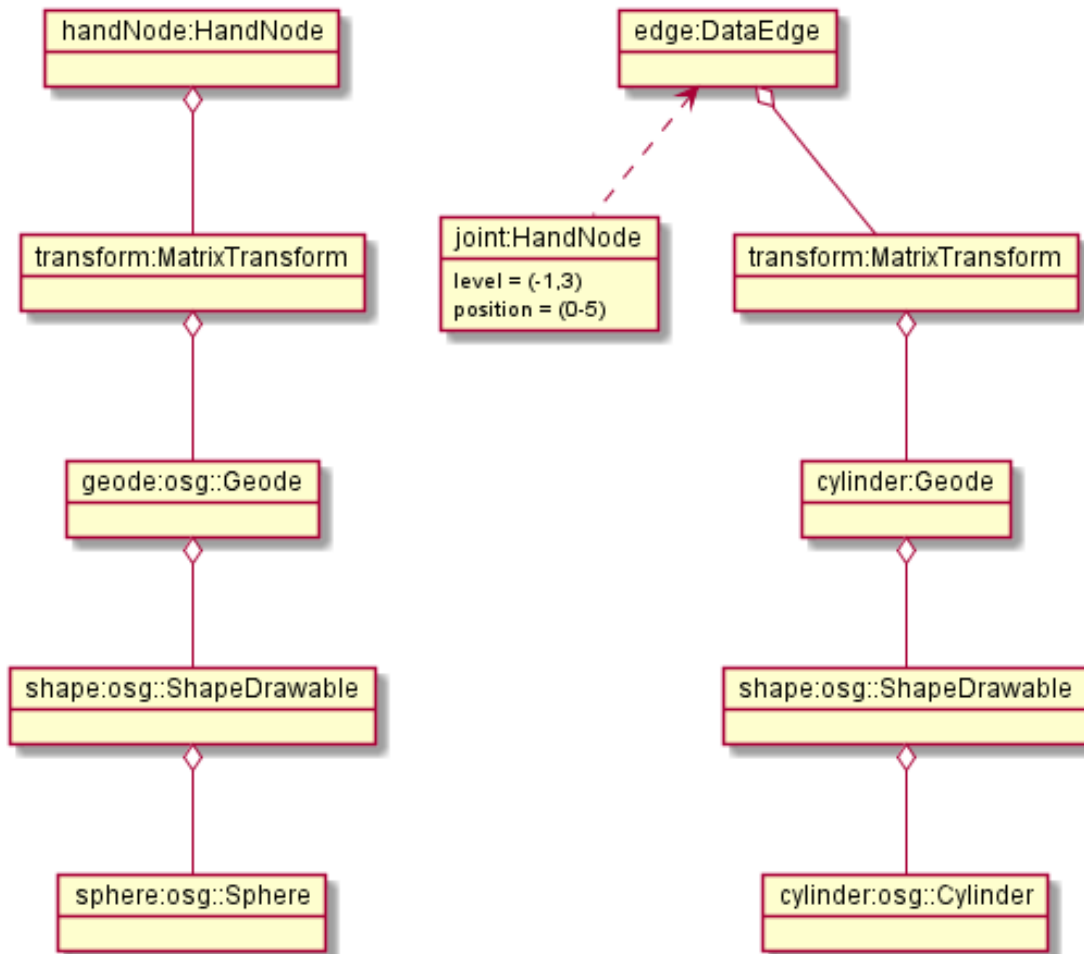




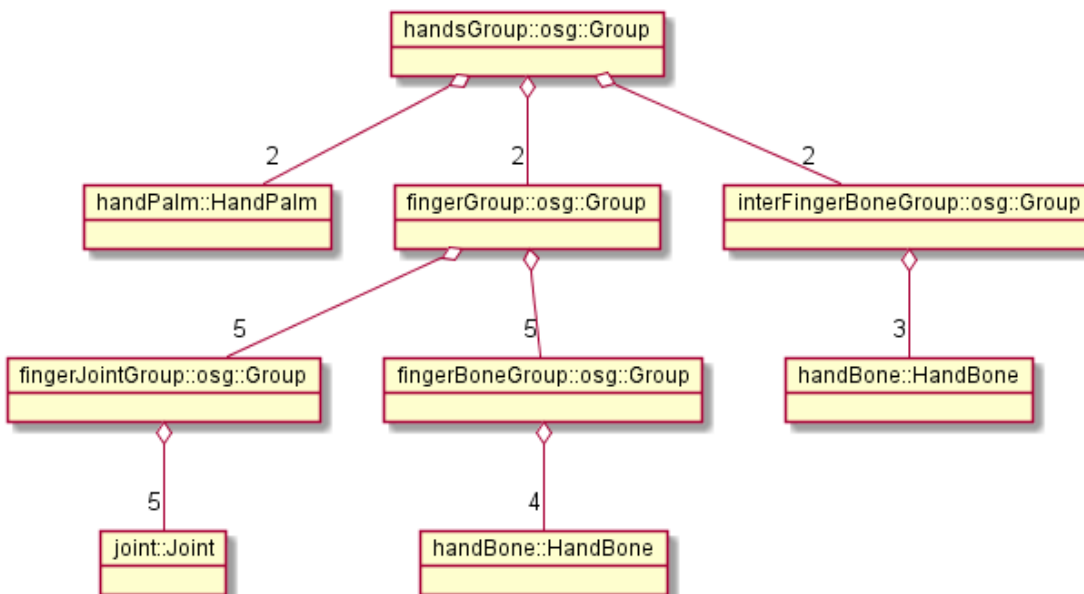
Navrh modelu ruky layer0:



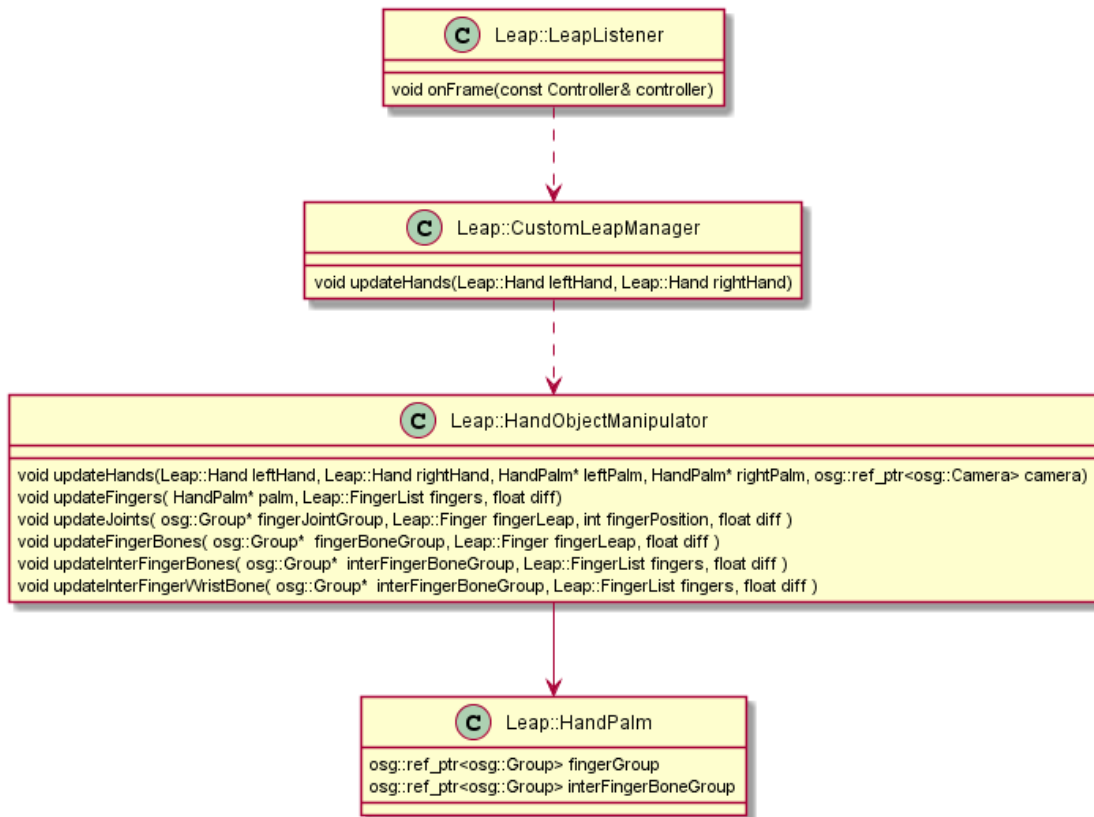
Navrh modelu ruky layer1:



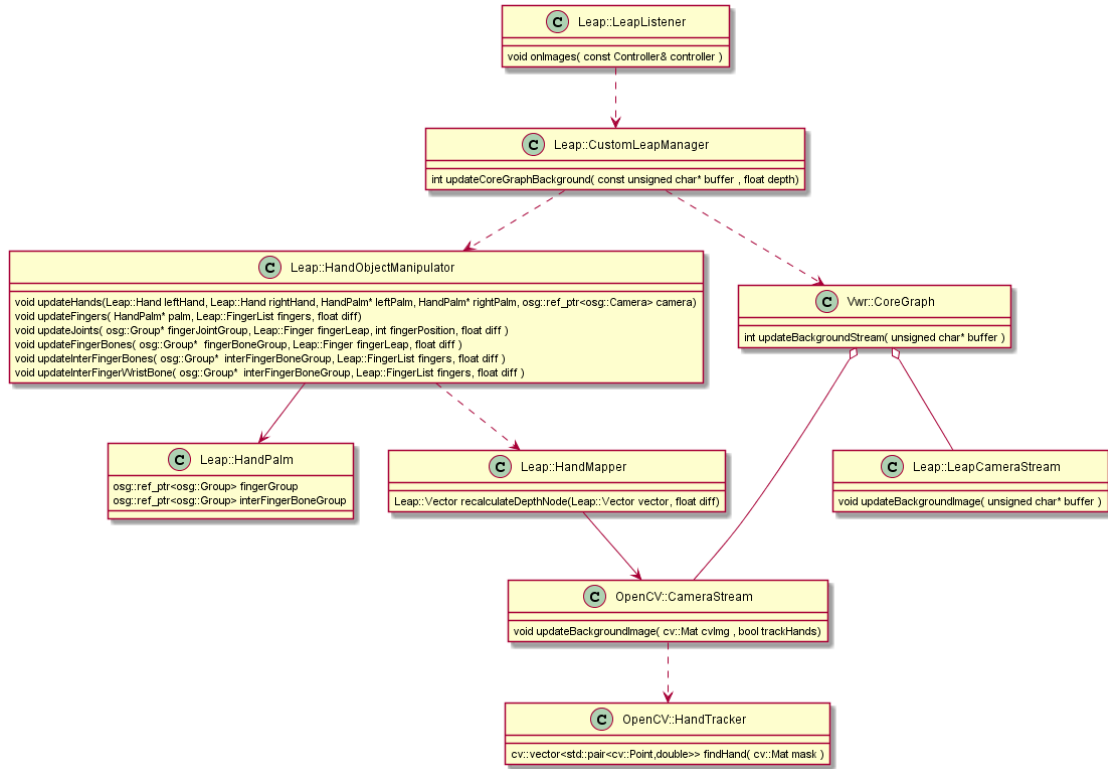
Implementovany model ruky :



Volania pri update ruk v scene pri LeapAR:







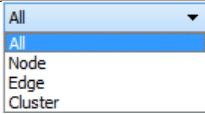




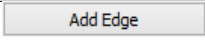
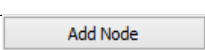
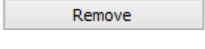

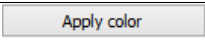
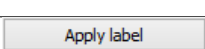




Mapovanie ruk a update pozadia pri LeapAR :





## 2.2 Akceptačné testovanie









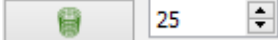
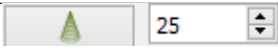


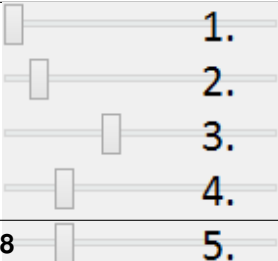
### 2.2.1 Záložka GRAPH

Zoznam ovládacích prvkov	Opis funkcionality	Status	Opis závady
	manipulácia s prvkami grafu, nič neoznačuje	funkčné	
	výber jedného prvku grafu	funkčné	
	výber viacerých prvkov grafu	funkčné	
	centrovanie pohľadu vzhľadom na vybraný prvok grafu	funkčné	
	typ výberu prvku grafu: všetko, iba uzly, iba hrany, klastre	funkčné	
	pridanie meta uzla do grafu	funkčné	
	odstránenie vybraných meta uzlov z grafu	ne-funkčné	meta uzol sa nedá odstrániť zo scény
	ukotvenie vybraných uzlov na aktuálnej pozícii	funkčné	
	uvolnenie ukotvených uzlov	funkčné	
	pridanie hrany medzi dvomi vybranými uzlami	funkčné	
	pridanie uzla	funkčné	
	odstránenie vybraných elementov (uzly alebo hrany)	funkčné	
	výber farby pre zafarbenie uzla	ne-funkčné	nedá sa vybrať iná farba ako čierna
	aplikovanie vybranej farby na vybraný uzol	funkčné	
	aplikovanie textového označenia na vybrané uzly podľa textu	ne-funkčné	uzol sa po aplikovaní zmenší a táto akcia sa nedá vrátiť späť
	zapnutie/vypnutie zobrazovania popisov uzlov a hrán	funkčné	
	spustenie animovania rozmiestňovania uzlov grafu	funkčné	
	zastavenie animovania rozmiestňovania uzlov	funkčné	
	zmena odpudivých síl pôsobiacich medzi uzlami	funkčné	



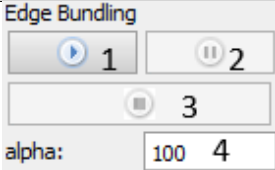






## 2.2.2 Záložka CONSTRAINTS

Zoznam ovládacích prvkov	Opis funkcionality	Status	Opis závady
	aplikovanie priestorového ohraničenia: povrch gule	funkčné	
	aplikovanie priestorového ohraničenia: obsah gule	funkčné	
	aplikovanie priestorového ohraničenia: rovina	funkčné	
	aplikovanie priestorového ohraničenia: prienik gule a roviny	funkčné	
	aplikovanie priestorového ohraničenia: kružnica	nefunkčné	nedá sa zmeniť priemer kružnice, po pokuse zmeniť priemer, nefunguje odstránenie ohraničenia
	aplikovanie priestorového ohraničenia: kužeľ	funkčné	
	aplikovanie priestorového ohraničenia: kužeľový strom	funkčné	
	odstránenie vybraných priestorových ohraničení	funkčné	
	aplikovanie priestorového ohraničenia: povrch valca	funkčné	
	aplikovanie priestorového ohraničenia: povrch kužeľa	funkčné	
	aplikovanie radiálneho rozmiestnenia na označené uzly	funkčné	
	výber módu vykreslenia radiálneho rozmiestnenia (drôtený, plný)	funkčné	
	nastavenie módu 2D/3D radiálneho rozmiestnenia	funkčné	
			
78	1. nastavenie veľkosti rozmiestnenia 2. nastavenie	funkčné	Chapter 2. Pre vývojárov

### 2.2.3 Záložka CLUSTERING

Zoznam ovládacích prvkov	Opis funkcionality	Status	Opis závady
	zlúčenie vybraných uzlov	funkčné	
	zrušenie zlúčenia vybraných uzlov	funkčné	
Adjacency ▾	definovanie algoritmu, ktorým sa bude zhlukovať graf	funkčné	
Depth: 1	nastavenie počtu rekurzií pre vybraný algoritmus	funkčné	
Cluster nodes	spustenie zhlukovania nad aktívnym grafom	funkčné	
Edge Bundling 	<ol style="list-style-type: none"> <li>1. spustenie algoritmu na zväzovanie hrán</li> <li>2. pozastavenie algoritmu na zväzovanie hrán</li> <li>3. úplne zastavenie algoritmu na zväzovanie hrán a zobrazenie pôvodného grafu</li> <li>4. vstupné pole na zadanie konštanty, určujúcej silu akou sú hrany k sebe počas zväzovacieho algoritmu prítahované</li> </ol>	funkčné	

Po použití funkcie zhlukovania, sa odkrývajú nasledujúce možnosti:

Zoznam ovládacích prvkov	Opis funkcionality	Status	Opis závady
Opacity: <input type="checkbox"/> auto <input type="checkbox"/> selected	auto - automatická priehl'adnosť - mení sa na základe vzdialenosti zhlukov od kamery selected - priehl'adnosť označeného zhluku	funkčné	
	pomocou posuvníka sa mení priehl'adnosť len označených zhlukov	funkčné	
Cluster shapes: 0  8	posúvaním sa mení prahová hodnota, pri ktorej sa menia tvary zhlukov - spodné číslo udáva, koľko uzlov obsahuje daný zhluk	funkčné	

Pri označení konkrétneho zhluku sa odkrývajú nasledujúce možnosti:

Zoznam ovládacích prvkov	Opis funkcionality	Status	Opis závady
<input type="button" value="Restrict"/>	kliknutím zmeníme označený zhluk na obmedzovač	nefunkčné	pri pokuse o otestovanie program spadne
<input type="button" value="Restart Layout"/>	znovurozmiestnenie uzlov v priestore po tom, ako sa nalepia na hranu – obmedzovača	neotestované	nie je možné otestovať kvôli problému s Restrict
Repulsive force <input type="text" value="1,00"/>	upravenie odpudivej sily medzi uzlami v označenom zhluku - čím je hodnota väčšia, tým budú uzly ďalej od seba	neotestované	nie je možné otestovať kvôli problému s Restrict

## 2.2.4 Záložka CONNECTIONS

Zoznam ovládacích prvkov	Opis funkcionality	Status	Opis závady
Nick: <input type="text" value="Nick"/>	meno, pod ktorým bude používateľ vystupovať v kolaborácii	funkčné	
<input type="button" value="Host session"/>	spustenie/zastavenie servera	funkčné	
Host: <input type="text" value="localhost"/>	IP adresa servera	funkčné	
<input type="button" value="Connect to session"/>	pripojenie(odpojenie) ku(od) kolaborácii	funkčné	
Collaborators: <input type="text"/>	zoznam používateľov	funkčné	
<input type="checkbox"/> Spy <input type="checkbox"/> Center <input type="checkbox"/> Shout	Spy - získa používateľ pohľad iného používateľa Center - nasmeruje pohľad používateľa tak, aby v jeho strede bol iný používateľ Shout - iným používateľom sa v scéne zobrazí pri vašom mene ikona znázorňujúca, že sa pokúšate upútať pozornosť	funkčné	
Avatar scale <input type="range"/>	nastavenie veľkosti avatarov v scéne	funkčné	



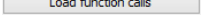

## 2.2.5 Záložka MORE FEATURES

Zoznam ovládacích prvkov	Opis funkcionality	Status	Opis závady
<input checked="" type="checkbox"/> Camera rotation	ak je zaškrtnuté, kamera nasmerovaná a graf sa pohybuje na základe pohybu tváre, značky alebo rúk, inak sa na základe týchto akcií rotuje samotný graf	funkčné	
<input checked="" type="checkbox"/> Camera enabled	povoľuje použitie kamery	funkčné	
<input type="button" value="Start camera"/>	otvorenie okna pre prácu s kamerou	funkčné	
<input type="button" value="Start Speech"/>	otvorenie okna pre prácu s hlasovým ovládaním	neotestované	Speech je momentálne vylúčený z projektu
<input type="button" value="Start Leap"/>	zapnutie ovládania pomocou Leap Senzoru	funkčné	

Možnosti okna otvoreného po kliknutí na vyššie uvedené tlačidlá:

Zoznam ovládacích prvkov	Opis funkcionality	Status	Opis závady
<input checked="" type="radio"/> Face Recognition <input type="radio"/> Marker	prispôsobenie ľavej strany okna pre ovládanie funkcionality rozpoznávania tváre	funkčné	treba pri zapínaní zaskrtnúť Camera rotation a Camera enabled
Start Face Recognition	zvolenie kamerového zariadenia a následným potvrdením objavenie záberu z kamery s rozpoznávaním tváre (graf sa pohybuje na základe pohybu tváre)	funkčné	
<input type="radio"/> Face Recognition <input checked="" type="radio"/> Marker	prispôsobenie ľavej strany okna pre ovládanie funkcionality rozpoznávania značky	funkčné	
Start Marker Detection	zvolenie kamerového zariadenia a následným potvrdením objavenie záberu z kamery určenej pre rozpoznávanie značky a graf sa začne otáčať a pohybovať so značkou	funkčné	
<input type="checkbox"/> Background	nastavenie aktuálne snímanie ako pozadie pre graf	funkčné	
<input type="checkbox"/> Marker is behind	prepínanie medzi pohybom podľa značky ako keby sa kamera pozerala na používateľa a naopak	neotestované	neodporilo sa vykonať test
Update cor. param.	nastavenie korekčných parametrov	neotestované	neodporilo sa vykonať test
Change Markers	zmena spôsobu použitia značky v prípade, že používateľ má k dispozícii len jednu značku	funkčné	
<input checked="" type="checkbox"/> NoVideo	vypnutie/zapnutie zobrazenia videa	funkčné	
Start kinect	zapnutie detekcie Kinectom	funkčné	
Kinect Snapshot	zachytenie kádra s následnou možnosťou dať ho na pozadie	neotestované	neodporilo sa vykonať test
<input type="checkbox"/> Turn off cursor	prepínanie medzi detekovaním ruky pre manipuláciu grafu alebo kamery v podobe rotovania a medzi detekovaním ruky pre funkciu "klik" (pohyb ruky do hĺbky, nie vertikálne alebo horizontálne)	funkčné	
<input type="checkbox"/> Turn off zoom	vypnutie možnosti približovania	funkčné	
<input checked="" type="radio"/> Aruco	nastavenie práce s arucom (manipulácia grafu pomocou značiek)	funkčné	

## 2.2.6 Hlavné okno

Zoznam ovládacích prvkov	Opis funkcionality	Status	Opis závady
	filtrovanie hrán	funkčné	nefunguje na operačnom systéme Windows
	filtrovanie uzlov	funkčné	nefunguje na operačnom systéme Windows
	zobrazí dialóg pre výber súborov; po vybratí vykreslí do pol'a pod tlačidlom graf volaní funkcií týchto súborov	funkčné	nefunguje na operačnom systéme Windows
	prepínanie medzi zobrazovaním jedného prehliadača pre každý uzol a zobrazovaním jedného prehliadača pre všetky vyznačené uzly	funkčné	nefunguje na operačnom systéme Windows
File – načítanie grafu zo súboru	načítanie grafu do scény	funkčné	
File - načítanie grafu z databázy	načítanie grafu do scény	neotestované	nie sú k dispozícii grafy v databázovej podobe
File - uloženie grafu	uloží graf	neotestované	možnosť bola implementovaná pre databázové riešenie
File - uloženie layout	ponúkne možnosť nazvať layout a uloží ho	neotestované	možnosť bola implementovaná pre databázové riešenie
File - ukončenie aplikácie	ukončí aplikáciu	funkčné	
Settings	nastavenia jednotlivých parametrov aplikácie	funkčné	

## 2.3 Logovacie a benchmarkingové knižnice

### 2.3.1 Logovacie knižnice

Pri použití Easylogging++:

- #include “easylogging++.h” len tam kde treba použiť
- INITIALIZE\_EASYLOGGINGPP je len raz v main-e

#### Zdroje:

- Loguru
- Logog
- Spdlog
- Plog
- Easyloggingpp



- G3log
- Glog
- log4cpp
- log4cplus
- log4cxx
- BOOSTlog

## 2.3.2 Benchmarkingové knižnice

Celero je najlepším výberom, len potrebuje C++11 compiler.

Pre iné platformy ako Windows je možné použiť Hayai.

### Zdroje:

- Hayai
- Celero
- Nonius
- Googlebenchmark

## 2.4 Metodiky

### 2.4.1 Ako programovať v C++

#### Ako písať headre a cpp súbory

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

- headre z projektu
- headre z Qt a OSG
- systemové headre

v .cpp je prvý header príslušne .h-čko toho .cpp

---

**Important: pravidlo:** includujem najprv tie, ktoré môžu includnúť čo najviac

---

#### pravidlá pre písanie .h

- obsahujú iba definície metód, NIE implementáciu metód
- includujeme LEN čo je treba pre header, NIC navyše
- ak je v triede/metode použitý pointer typ, napr.  
Node\* n;  
tak stačí použiť doprednú deklaráciu, t.j.

```
class Node;
a netreba includovat Node.h
```

**Attention: toto nefunguje**, keď:

- sa dedi trieda
- to není pointer, t.j. je to Node n;
- keď je použitý `osg::ref_ptr` pointer, resp. šablony

### pravidla pre písanie .cpp

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

### ZLE praktiky, resp. čo nerobiť

NIKDY v headroch a cpp suboroch nepoužívať “using namespace”

- nepoužívať “using” keyword

ak už je použité, tak to treba upraviť na :

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

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

- použiť `nullptr` (ak kompilátor podporuje C++11), prípadne použiť `NULL`

### DOBRE praktiky, resp. čo robiť

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

- najme pointre
  - **skontrolovať ak je “new” v konštruktore, tak MUSÍ byť “delete” v deštruktore** (neplatí pre: `osg::ref_ptr`)
- inicializovať v takom poradí v akom sú zapísané v triede:
  - **usporiadať atribúty od “najväčších” (napr. pointer, trieda, double) po “najmenšie” (int, char, bool)** aj keď sa bude miešať `public/private`

“std::cout”

- pokiaľ je v kóde, ktorý používa Qt, tak prerobiť na `qDebug`
- resp. najst a používať nejakú externú knižnicu na logovanie

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

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

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

pravidelne sa robí štatistická analýza kódu pomocou nástrojov `cppcheck` a `cpplint`

- vid. metodiky cpcchceck, cpplint

pravidelne sa zdrojovy kod formatuje pomocou nastroja astyle

pravidelne sa merguju vsetky prace

## 2.4.2 CppCheck

Stiahni a extrahuj: <http://cppcheck.sourceforge.net>

pouzite cppcheck v 1.70

### Ako pouzivat v default GUI

1. v **Edit->Preferences nastavit**: General: check “Force checking all #ifdef configurations” Paths: pridat cestu do “3dsoftviz/include” Advanced: check “Show inconclusive errors” !!! toto hlasi dost false-positive, ALE obcas najde dolezite veci !!!
2. Check->Directory a vybrat “3dsoftviz/src”

### Ako pouzivat v QTCreatore (Windows)

1. Pridaj cestu k cppcheck-u do systemovej premennej PATH: (napr. d:/timak/cppcheck)
  2. V run\_cppcheck.bat (root adresar projektu) zmen set cppcheck\_path= ../cppcheck.exe na celu cestu k cppcheck.exe
  3. Spusti QTCreator
  4. Projects -> Build & Run -> Build
  5. V Build Settings -> Edit Build configuration klikni Add -> clone selected a zadaj “cppcheck”
  6. V Build steps rozklikni Details a v Targets oznac “cppcheck”. Ak su oznacene aj ine targety, tak ich je potrebne odznacit(Vysledok: Make: jom.exe cppcheck).
  7. Klikni na kladivko v lavom dolnom rohu (Build).
  8. Vystupom je subor cppcheck-report.txt v podadresari \_build
- poznanky k reportom: !!! ak je nieco nejasne, treba sa opytat !!!

**!!! ignorovat hlasky pre externy kod: !!!** noiseutils, qtcolorpicker

### dolezite hlasky typu:

Technically the member function XYZ can be const.

- najma funkcie typu “getter” mozu byt const, napr.:

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

Technically the member variable XYZ can be const.

- treba skontrolovat!!!

The class ‘RestrictionsManager’ does not have a constructor.

- kazda trieda by mala mat konstruktor (kompiler nam sice vytvori default, ale... vid. nizsie)
- ‘class Type’ does not have a copy constructor which is recommended since the class contains a pointer to allocated memory.
- zavisi, ci sa “copy constructor” v kode pouziva - nutna kontrola

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

- to je jasne

C-style pointer casting

- to je riesene aj cez cplint v samostatnej hotfix branch
- v gcc da sa zapnut -Wold-style-cast -> momentalne hlasi velmi vela warningov
- je to kvoli citatelnosti, ALE ma to svoje opodstatnenie

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

- vsetky class variables by mali byt inicializovane v konstruktore cez “initialization list”

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

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

!!! treba skontrolovat - indikuje zavanu chybu !!!

Uninitialized variable: newGraph

!!! treba skontrolovat - indikuje zavanu chybu !!!

### ostatne hlasky:

- treba skontrolovat - a mali by sa opravit

## 2.4.3 Cplint

Cplint

je to Python script

- pokial by bol problem spustat, mozem to riesit ja

### ako pouzivat v QtCreator (Windows)

1. Stiahni a nainstaluj python (napr. verziu 2.7)
2. Ako administratorspusti prikaz: pip install cplint
3. Najdi lokaciju kde sa to nainstalovalo. Pravdepodobne: c:/Python27/Scripts/cplint.exe V cplint.bat (root adresar projektu) zmen set cplint\_path= ../cplint.exe na celu cestu ku cplint.exe
4. Spusti QtCreator
5. Projects -> Build & Run -> Build

6. V Build Settings -> Edit Build configuration klikni Add -> clone selected a zadaj "cpplint"
7. V Build steps rozklikni Details a v Targets oznac "cpplint". Ak su oznacene aj ine targety, tak ich je potrebne odznacit(Vysledok: Make: jom.exe cpplint).
8. Klikni na kladivko v lavom dolnom rohu (Build).
9. Vystupom je subor cpplint-report.txt v podadresari \_build

### ako pouzivat (Linux)

pouzite testy su nastavene v CPPLINT.cfg

dostupne testy sa vypisu: cpplint.py -filter=

**spustit v include adresari:** cpplint.py Aruco/\* Core/\* Data/\* Importer/\* Layout/\* Math/\* Network/\* Network/executors/\* OpenCV/\* QOpenCV/\* Speech/\* Viewer/\* Clustering/\* Clustering/Figures/\* Fglove/\* Kinect/\* Kinect/RansacSurface/\* Manager/\* Model/\* OsgQtBrowser/\* QOSG/\* Util/\* 2>&1 | tee report-include.txt

**spustit v src adresari:** cpplint.py Aruco/\* Core/\* Data/\* Importer/\* Layout/\* Math/\* Network/\* Network/executors/\* OpenCV/\* QOpenCV/\* Speech/\* Viewer/\* Clustering/\* Clustering/Figures/\* Fglove/\* Kinect/\* Kinect/RansacSurface/\* Manager/\* Model/\* OsgQtBrowser/\* QOSG/\* Util/\* 2>&1 | tee report-src.txt

### poznacky k reportom

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

**ignorovat hlasky pre externy kod:** noiseutils, qtcolorpicker  
ciastocne pre cameramanipulator, QGraphicsViewAdapter

### dolezite hlasky typu:

Constructors callable with one argument should be marked explicit

- nastudovat, asi staci pridat keyword explicit

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

- nastudovat (ci to nieje false-positive)

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

- nastudovat

Do not use namespace using-directives.

- odstranit "using namespace" (okrem externeho/cudieho kodu )

Consider using rand\_r(...) instead of rand(...) for improved thread safety.

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

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

!!! indikuje vazny problem !!!

flase-positive:

**Clustering/Figures/Cube.h:28: Add #include <algorithm> for transform [build/include\_what\_you\_use] [4]**  
Cube ma metodu tranform -> netreba include

#### ostatne hlasky:

- treba skontrolovat - a mali by sa opravit (resp. uz som ich opravil ;-)

## 2.4.4 CodeReview

### Ako skontrolovat vytvoreny kod

#### Kontrola kodu zahrna aplikaciu predchadzajucich metodik:

- Ako programovat v C++,
- CppCheck,
- CppLint,
- upravu kodu pomocou astyle

Vysledny kod musi byt skompilovatelny a CppCheck a CppLint nesmia zistit ziadne problemy.

## 2.4.5 AStyle

### Ako pouzivat v QtCreator (Windows)

1) Stiahni [AStyle](#) do priecinku s programmi tykajucim sa projektu \$ASTYLE\_PATH. 1) Pridaj cestu k astyle-u do systemovej premennej PATH: \$ASTYLE\_PATH/bin (napr. d:/timak/AStyle/bin) 3) Spusti QtCreator 4) Projects -> Build & Run -> Build 5) V Build Settings -> Edit Build configuration klikni Add -> clone selected a zadaj "style" 6) V Build steps rozklikni Details a v Targets oznac "style". Ak su oznacene aj ine targety, tak ich je potrebne odznacit (Vysledok: Make: jom.exe style). 7) Klikni na kladivko v lavom dolnom rohu (Build).

## 2.4.6 Metodika tvorby a údržby UML diagramov prostredníctvom PlantUML

### PlantUML

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

PlantUML je voľne dostupný na [stiahnutie](#) z oficiálnej stránky, prípadne je možné na otestovanie použiť aj jednoduchú [web aplikáciu](#).

Pre plnohodnotné využitie je potrebné mať taktiež nainštalovaný [Graphviz](#).

Tiež ponúka možnosť [integrácie](#) s viacerými textovými editormi a wiki stránkami.

### Pravidlá pre tvorbu súborov

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

3. Názvy súborov sú po anglicky.

## Užitočné príkazy a postupy

### Odstránenie duplicity pomocou Preprocesoru

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

```
@startuml
'Bez preprocesoru

package "class Filter representation" {
    class ObjectStructure
    class Element {
        +{abstract}register(Visitor v)
    }
    class Mapper {
        +register(Filter f)
    }
    class Klient
    class Visitor {
        +{abstract}visitMapper(Mapper m)
    }
    class Filter {
        +visitMapper(Mapper m)
    }

    ObjectStructure -down-> Element
    Mapper -up-|> Element

    ObjectStructure <-left- Klient

    Klient -down-> Visitor
    Filter -up-|> Visitor
}
@enduml
```

```
@startuml
'S preprocesorom
!define o(x) ObjectStructure
!define e(x) Element
!define m(x) Mapper
!define k(x) Klient
!define v(x) Visitor
!define f(x) Filter

package "class Filter representation" {
    class o()
    class e() {
        +{abstract}register(v(x) v)
    }
    class m() {
        +register(Filter f)
    }
}
```

```

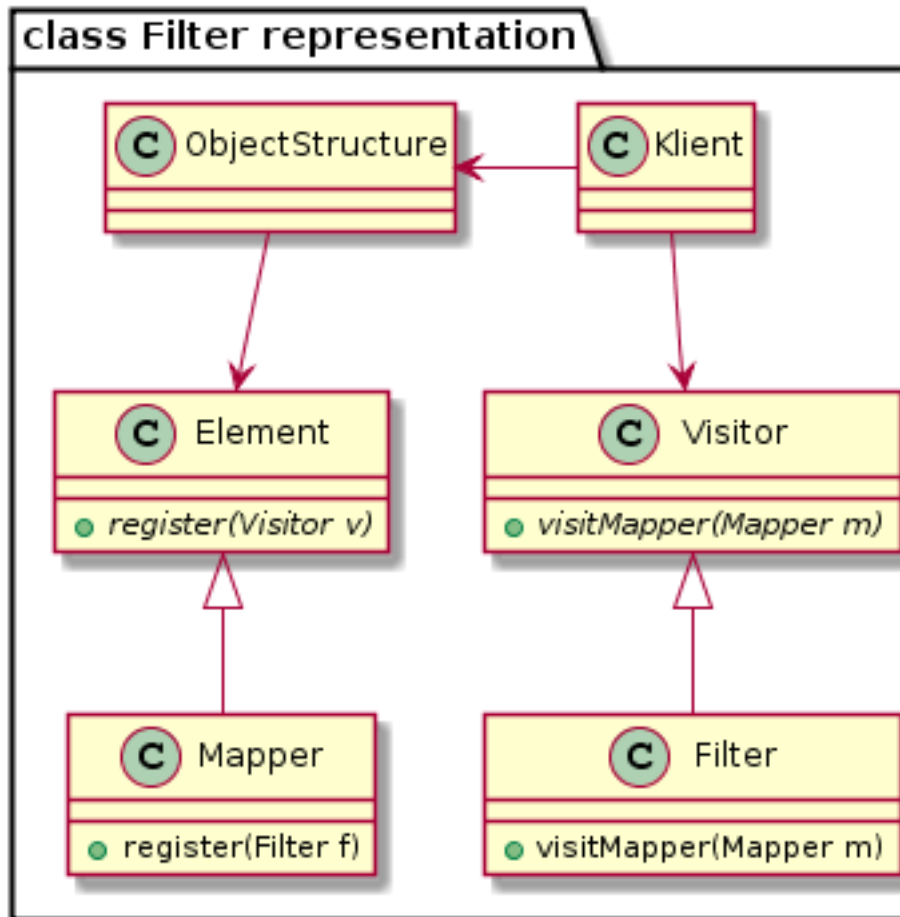
class k()
class v() {
    +{abstract}visitMapper(m(x) m)
}
class f() {
    +visitMapper(m(x) m)
}

o() -down-> e()
m() -up-|> e()

o() <-left- k()

k() -down-> v()
f() -up-|> v()
}
@enduml

```



*V oboch prípadoch bude výsledok nasledovný:*

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



## Použitie aliasov v sekvenčnom diagrame

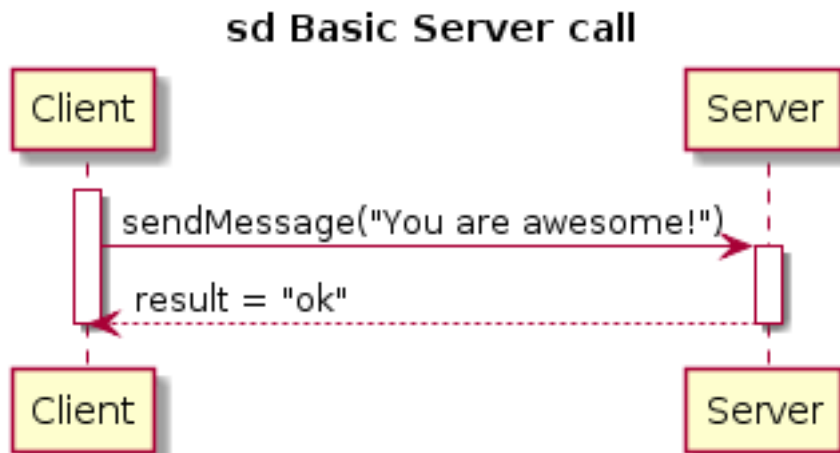
V sekvenčných diagramoch odporúčame pri definovaní volaní medzi objektami používať aliasy (skratky). Ich princíp je analogický s predchádzajúcim makrom avšak sú ešte o niečo prehľadnejšie. Aliasy nie sú však podporované v class diagrame.

```
@startuml
participant Client as c
participant Server as s

title sd Basic Server call

activate c
  c -> s: sendMessage("You are awesome!")
  activate s
    s --> c: result = "ok"
  deactivate s
deactivate c

@enduml
```



*Výsledok:*

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

## Pravidlá pre súborovú štruktúru

Samotné UML diagramy je potrebné rozdeliť do prehľadnej súborovej štruktúry:

- projekt (názov projektu, napr. 3dsoftviz)
  - doc (inštalačná dokumentácia, vygenerovaná dokumentácia atď.)
  - uml
    - \* structural
      - class diagrams (korešpondujúce s reálnym kódom)
      - component diagrams
    - \* behavioral
      - activity diagrams

- use-case diagrams
- sequence diagrams
- state diagrams

### 2.4.7 Gitflow metodika

#### Forkovanie na GitHub-e

Fork na GitHub-e neprenesie tag-y do forknutého repozitára, treba ich ručne preniesť, v novom repozitári:

- `git remote add povodny-repozitar git@github.com:povodny-repozitar/nazov_repo.git`
- `git fetch povodny-repozitar 'refs/tags/:refs/tags/'`
- `git push --tags`

#### Vetvy

- Master - hlavný projekt
- **Develop** - branchnutá z mastra, každý sprint má vlastnú Develop vetvu, na konci sprintu sa merge späť do mastra, **!!! pred mergeom treba spraviť komplet build (nie len unity)**
- Feature - branchnutá z developu, každý nový kus funkcionality (task v issue tracking nastroji), ktorý sa kódi musí mať vlastnú
  - Feature vetvu... po dokončení a validácii kódu sa merge späť do Developu, **NEINTERAGUJE S MASTER VETVOU**
- Hotfix - vetva na rýchly fix priamo z mastra, merge sa do mastra a do developu, navyšuje aktuálnu verziu

Vždy mergeujeme cez Shell a s prepínancom **--no-ff**

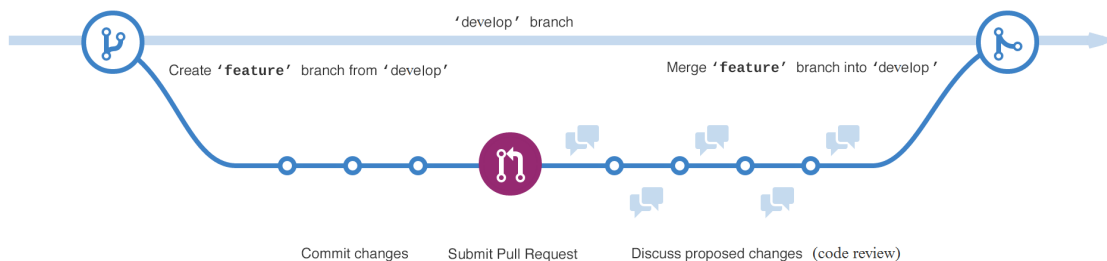
Cheat sheet so všetkými základnými commandmi: <https://training.github.com/kit/downloads/github-git-cheat-sheet.pdf>

#### Pull requesty

Po dokončení práce, keď sme ready to review sa dáva pull request na vetvu, do ktorej sa bude mergeovať.

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

Po odsúhlasení Pull requestu sa potom pristúpi k mergeu.



Obrazok ilustruje vytvorenie feature branch z develop vetvy, implementáciu rozdelenú do znázornených commitov, následný pull request predstavujúci žiadosť o code review a finálny merge do develop vetvy daného sprintu.

## Commit messages

v Commit messages používame tagy a ID ulohy na zaciatok:

- [FIX] - fixli sme nejaku chybu z minula, bugfix, hotfix a podobne
- [ADD] - pridali sme novu funkcionalitu, subor, ...
- [DOC] - pridali sme dokumentaciu, komentary...
- [REF] - pre refactoring
- [FMT] - formatovanie textu, uprava
- [TEST] - pre testy

Za tym velmi strucne (a vystizne) opiseme, ake zmeny sme spravili. Message by mali byt kratke, no pokrывать vsetko, co sme v commite spravili. **\*\*!!!** vseobecny tvar: “[tag] #taskId Popis vykonanej zmeny” **\*\*** Napr. [DOC] #3654  
*Pridanie uvadzania ID ulohy do gitflow metodiky*

## Useful commands

- **\$ git submodule update –init –recursive**
  - update submodulov (dependencies)
- **\$ git checkout -f meno\_branch**
  - checkout branche aj napriek lokalnym zmenam, budu zahodene
- **\$ git status**
  - vypise vsetky vykonane zmeny
- **\$ git stash / \$ git stash pop**
  - ulozi stav projektu do stashu, z ktoreho sa da potom tento stav pop-nut, dobre na prenos zmien medzi vetvami

## Tvorba feature branch-u:

- `$ git checkout -b “feature/meno-feature” develop //Switched to a new branch “feature/meno-feature”`

## Mergovanie hotoveho feature:

- `$ git checkout develop //Switched to branch ‘develop’`
- `$ git merge –no-ff meno-feature`
- `$ git push origin develop`

## Tvorba hotfix branch-u:

- `$ git checkout -b “hotfix/nazov-co-fixujem” master //Switched to a new branch “hotfix-{cislo_verzie}”`
- `$ git commit -m “sprava, co som spravil”`

### Uzatvorenie Hotfix branchu:

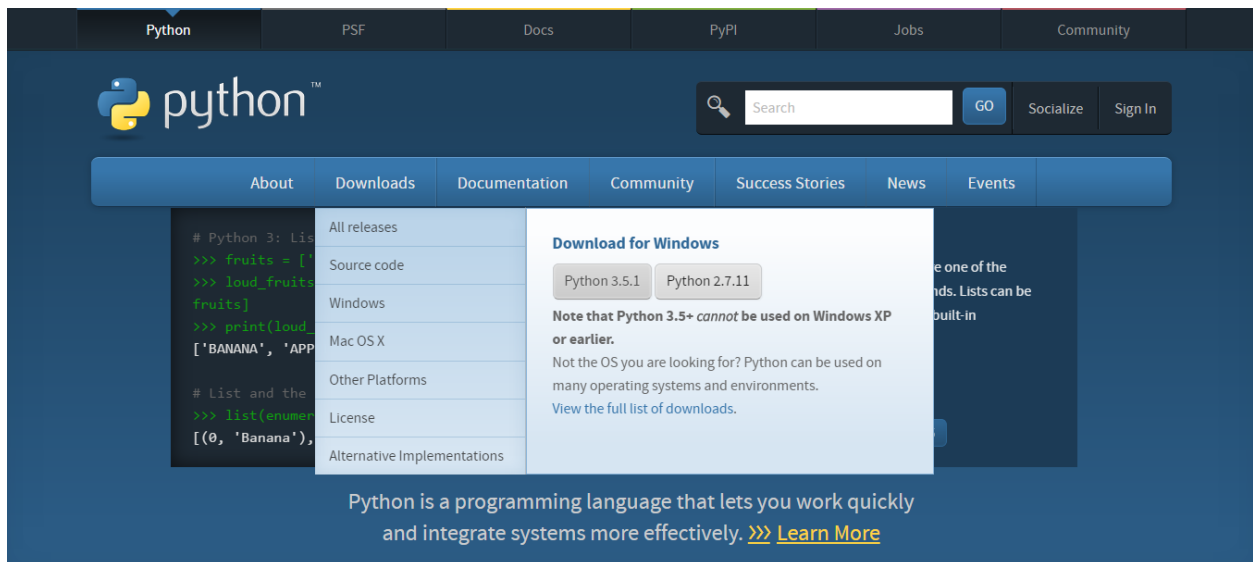
- \$ git checkout develop //Switched to branch 'develop'
- \$ git merge --no-ff "hotfix/nazov-co-fixujem"

## 2.4.8 Sphinx dokumentacia

### Instalacia

#### Python

- Pre prahu so Sphinxom treba mat nainstalovany Python.



**Note:** Python ponuka verzie 2.x a 3.x. Sphinx 1.3 moze bezat pod Python 2.6, 2.7, 3.3, 3.4, ale odporucana verzia je 2.7.

- Pre stahnutie a instalovanie externych kniznic pre Python existuje prikaz *pip*. Prikaz uz sa nachadza v oficialnych verziach Pythonu 3.4.0 alebo 2.7.9.

Ak prikaz sa nenainstaloval automaticky, treba ho stiahnut zo stranky <https://bootstrap.pypa.io/get-pip.py> a niekam ulozit. V prikazovom riadku treba prejsť do adresara s *get-pip.py* a spustit nasledovny prikaz:

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

### Sphinx

- Prejsť do priecku s dokumentaciou (tam kde index.rst sa nachadza) a pomocou prikazu pip nainstalovat Sphinx:

```
pip install sphinx
```

– ([sphinx-doc.org](http://sphinx-doc.org))

- Ak treba vytvorit novu dokumentaciu, pre nastavenie zdrojoveho adresara a vytvorenie potrebných suborov na pracu so Sphinx treba spustit prikaz

```
sphinx-quickstart
```

a odpovedat na otázky. Vyberte si všetky predvolené odpovede a po vyzve zadajte názov, autorov a verziu projektu.

- Týmto príkazom budú vygenerované súbory *Makefile*, *make.bat* a *conf.py.in*.
  - Všetky konfigurácie dokumentácie sú v *conf.py.in*.

**Attention:** *Sphinx-quickstart* a vytváranie týchto súborov generujú novú dokumentáciu! Ak súbory *index.rst*, *Makefile*, *make.bat* a *conf.py.in* už existovali, tak sa prepisu!

- Sphinx dokumentácia generuje výstup v rôznych formátoch zo súborov *.rst*. Podrobnejšie o [RestructuredText](#).

## HTML dokumentácia

- Súbor *make.bat* povolí vygenerovať dokumentáciu v tom formáte, ktorý potrebujete
- Pre generovanie HTML dokumentácie treba v príkazovom riadku prejsť do priečinku s ReST súbormi a *make.bat* súborom a spustiť príkaz

```
make html
```

- Inak generovanie dokumentácie sa dá spustiť pomocou CMake v QtCreatore

## PDF dokumentácia

Pre generovanie PDF dokumentácie potrebujeme najprv vytvoriť Latex dokumentáciu.

---

**Note:** Pre prácu s Latex treba mať [TeXlive](#)

---

Príkazom

```
make latex
```

vygeneruje sa Latex dokumentácia, ktorá následne sa môže konvertovať do PDF pomocou programu [TeXstudio](#).

---

**Note:** PDF dokumentácia generuje len pomocou príkazového riadku a externého programu, neda sa spustiť cez CMake!

---

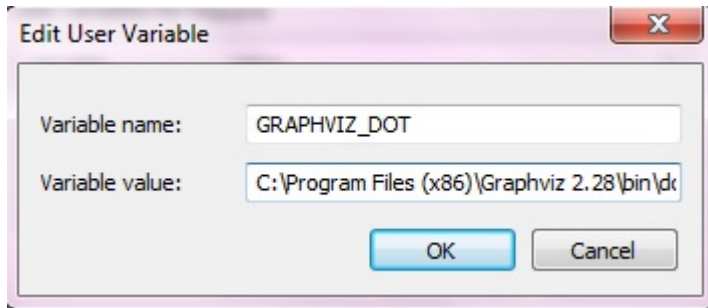
## PlantUML

Pre prácu s PlantUML nástrojmi v Sphinx treba:

- nainštalovať Javu
- pridať Javu do premenných prostredí (environment variable)
- nainštalovať [Graphviz](#)

– odporucana verzia je 2.28

- pridať Graphviz do premenných



---

**Note:** Hodnota premennej ma byť do *dot.exe*

---

- pridať Graphviz do extensions v `conf.py.in`:

```
extensions = ['sphinx.ext.graphviz']
```

- nainštalovať *sphinxcontrib-plantuml* zo stránky alebo príkazom

```
pip install sphinxcontrib-plantuml
```

- pridať plantuml do extensions v `conf.py.in`:

```
extensions = ['sphinxcontrib.plantuml']
```

- stiahnuť `plantuml.jar`

- pridať do `conf.py.in` príkaz

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

**Attention:** Dôležité je zmeniť túto cestu na správnu, akú máte aktuálnu na Vašom počítači!

- pridávať UML do dokumentácie je možné pomocou

```
.. uml::  
  
    !include /cesta/do/subor.wsd(txt)  
  
    alebo  
  
    @startuml  
    PlantUML kod  
    @enduml
```

## Excel tabulky

- Pre import Excel súborov do dokumentácie treba nainštalovať *exceltable* pomocou príkazu

```
pip install sphinxcontrib-exceltable
```

- Pridat *exceltable* do extensions v *conf.py.in*:

```
extensions = ['sphinxcontrib.exceltable']
```

- Importovat tabulky pridaním do *.rst* suboru:

```
.. exceltable:: caption
:file: path/to/document.xls
:header: 1
:selection: A1:B2
```

- Podrobnejšie o [Options](#)

## 2.4.9 TFS metodika

### Všeobecná metodika na manažment úloh v tíme

#### Pridanie novej úlohy

- Uviest' ohodnotenie úlohy, priority, opis, uviesť kedy je úloha hotová.
- Uviest' odhadovaný čas dokončenia.
- Opis musí byť podrobný, aby každému členovi bolo jasné, čo ma vykonať po pridelení úlohy.

#### Rozdeľovanie úloh

- Každý si vyberie (potiahne) úlohu/úlohy, ktorá/é majú najvyššiu prioritu.
- Ak ostanú nepridelené úlohy, študentský vedúci tímu pridelí členom zvyšné úlohy.

#### Kedy je úloha hotová

- Dokumentácia: Keď je znovu vygenerovaná.
- Kód: Potrebné spraviť code review a vykonať pull request do vetvy, ktorá sa bude mergovať.
- Testy: Keď je spravený report z testu.
- Zápisnica: Keď je nahratá vo formáte pdf na stránke tímu.

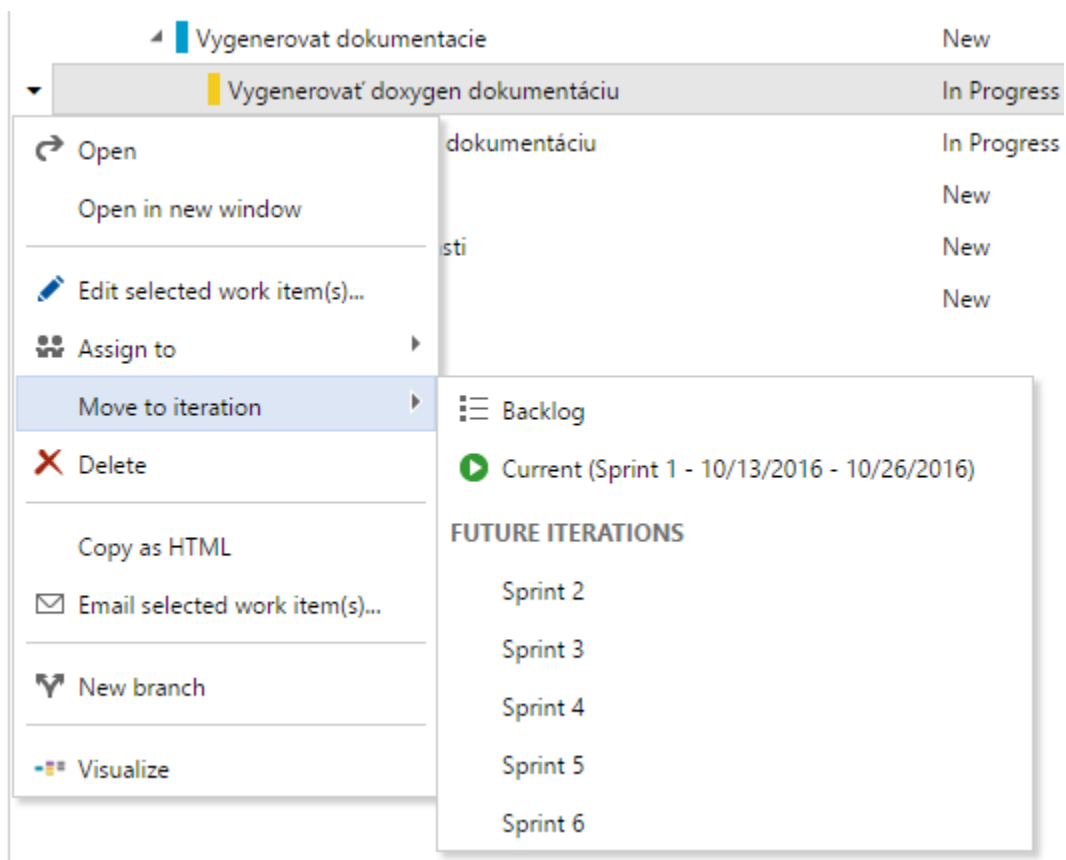
#### TFS metodika

- Adresa TFS: <https://tfs.fiit.stuba.sk:8443/tfs/> (Potrebné sa lognúť 2x)
- Projekt: AugReality / AugReality Team

## Pridávanie úloh

- Úlohy podobného charakteru priradíme do spoločného backlog itemu.
- Pri pridávaní úlohy sa automaticky nastaví stav 'To Do'.
- V prípade objavenia chyby, je potrebné vytvoriť novú úlohu typu Bug (Chyba)

Úlohy (tasky) sa môžu nachádzať v troch stavoch: \* To Do \* In Progress \* Done Nesplnené úlohy, ktoré sa nestihli dokončiť v danom šprinte, presunieme do nasledujúceho šprintu.



## Vytváranie exportov z TFS

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

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

Login: ako do AIS-u

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





## Úrovne logovania

Easylogging++ poskytuje tieto úrovne logovania:

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

## Čo a ako treba logovať

### Čo je potrebné logovať:

- Začiatok každej metódy
- Všetky možné error / warningy

### Ako treba logovať:

- Začiatok každej metódy logujeme v tvare

```
function() {  
    LOG( INFO ) << "MENO_BALIKU/MENO_TRIEDY/MENO_FUNKCIE (PARAMETRE) "  
}
```

Pozor, parametre používať iba v prípade, že je to vhodné

- Všetky možné error / warningy logujeme v tvare

```
LOG( WARNING/ERROR/FATAL ) << "MENO_BALIKU/MENO_TRIEDY/MENO_  
↪FUNKCIE (PARAMETRE) "
```

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

## 2.4.11 Metodika na písanie BDD testov

### Písanie testov

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

```
#include <igloo/igloo_alt.h>  
using namespace igloo;
```

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

```
Describe(a_foo_bar) {  
    ...  
}
```

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

```
It( foo_should_be_bar ) {
...
}
```

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

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

## Spustenie testov

1. Testy (všetky \*.cpp) pre vas modul umiestnujte do priečinka:

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

Testy fungujú tak, že si buildnete vlastný .exe súbor, ktorý otestuje danú funkcionálnosť. Test si buildnete následovne:

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

2. Teraz si potrebujete nakopirovať nejaký kód, minimálne by ste mali mať:

```
file( GLOB_RECURSE SRC_FOO_TESTS "tests/Foo/*.cpp" )
add_executable( test_foo_module ${TEST_RUNNER} ${SRC_FOO_TESTS} )
add_dependencies( test_foo_module igloo )
target_compile_options( test_foo_module PUBLIC ${FLAGS_FOR_DEBUG} )
```

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

```
add_test( testing_foo_module test_foo_module )
```

3. **Teraz musíte zmeny uložiť a nechať zbehnúť CMake**

- pravým na project -> Run CMake
- v QtCreatori zmeňte build/run target na Tests/test\_foo\_module

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