Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológií

Ilkovičova2, 842 16 Bratislava 4

AugReality

Dokumentácia projektu-Inžinierske dielo

Vedúci tímu: Ing. Kapec Peter Phd.

Členovia tímu: Bc. Filip Škultéty,

Bc. Peter Belai,

Bc. Martin Mokrý,

Bc. Lukáš Hagara,

Bc. Patrik Berger,

Bc. Marek Roštár,

Bc. Michal Galbavý

Akademický rok: 2016/2017

Obsah

1. Úvod

Dokument inžinierske dielo opisuje technickú dokumentáciu projektu Vizualizácia informácií v obohatenej realite. Opisuje architektúru a postup rozširovania projektu. Základná architektúra je prebratá z minuloročných projektov.

V druhej kapitole sú opísané globálne ciele pre zimný semester, ktoré sa budeme snažiť splniť počas prvých 5 šprintov. Ciele sú rôznorodé, napríklad opravovanie warningov, zobrazenie modelu dlaní v scéne alebo vytvorenie datasetu zo zariadenia kinect 2. V tretej kapitole sa nachádza celkový obraz systému, ktorý pomocou UML diagramov opisuje systém. V 4. kapitole sú bližšie opísané moduly systému, konkrétne modul LeapAR, modul Kinect 2.

1.1. Podiel práce na dokumentácií k inžinierskemu dielu

Kapitola	Autor
Úvod	Filip Škultéty
Globálne ciele pre ZS	Filip Škultéty
Celkový pohľad na systém	Martin Mokrý
Celkový pohľad na systém: Kinect 2	Lukáš Hagara
Moduly systému: Modul LeapAR	Martin Mokrý
Moduly systému: Modul Kinect 2	Peter Belai

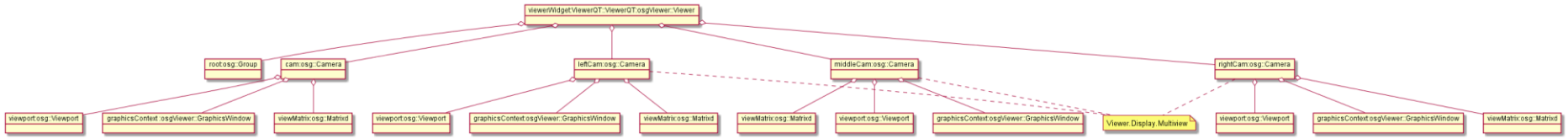
2. Globálne ciele pre ZS

V tejto kapitole sa nachádza zoznam globálnych cieľov pre zimný semester. V predmete tímový projekt rozširujeme už existujúci projekt, ktorý disponuje s viacerými funkcionalitami rôznych zariadení. Chceme preto pokračovať v tomto diele a splniť nasledovné ciele:

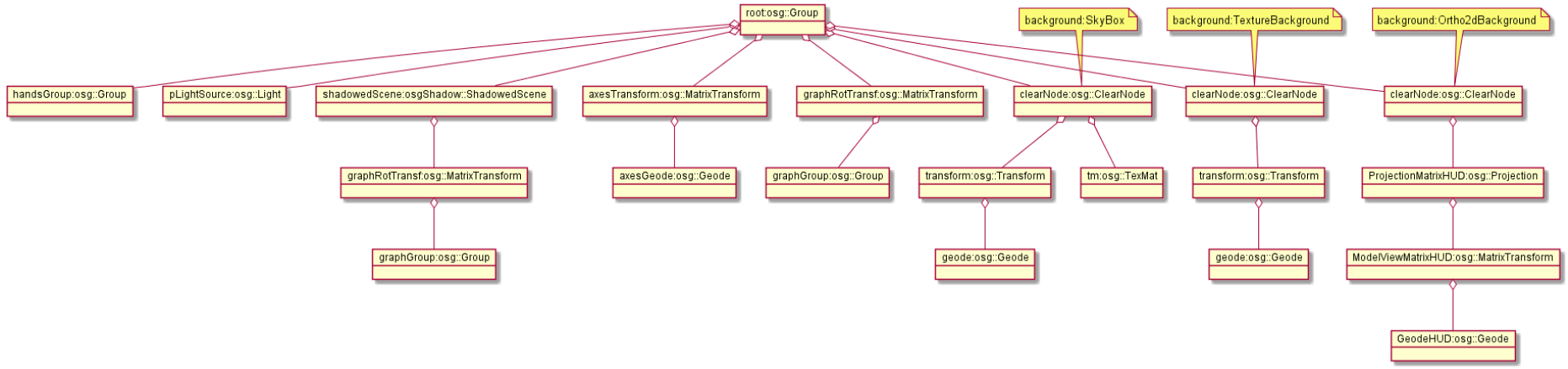
1. **Refaktorizácia zdrojového kódu**
   V projekte sa nachádzajú zoskupenia tried, ktoré by bolo vhodné refaktorovať. Modul, ktoré plánujeme refaktorovať: LuaGraph (dôvod: nedostatočné zapuzdrenie). V rámci refaktorizácie odstránime warningy z kompilátora, cppcheck a cpplint.
2. **Zobraziť model rúk v scéne grafu**
   Pomocou senzoru Leap chceme zobraziť pohyby a gestá používateľov v scéne grafu v reálnom čase. V projekte sa nachádza čiastočná implementácia leap senzoru, ktorú je potrebné dokončiť.
3. **Dataset zo zariadenia Kinect 2**
   Doplniť podporu zariadenia Kinect 2. Kinect 2 bude použitý na získanie datasetu, ktorý budú môcť využívať aj ostatní študenti fakulty. Zabezpečiť, aby si aplikácia vybrala medzi ovládačmi Kinect 1 alebo Kinect 2 podľa pripojeného zariadenia.
4. **Analýza marker less knižníc**
   Nahradenie Aruco knižnicou, ktorá vie pracovať bez markeru. Podmienky na knižnicu sú: jazyk C++, open source, multiplatform, postavená na OpenCV/PCL.

3. Celkový pohľad na systém

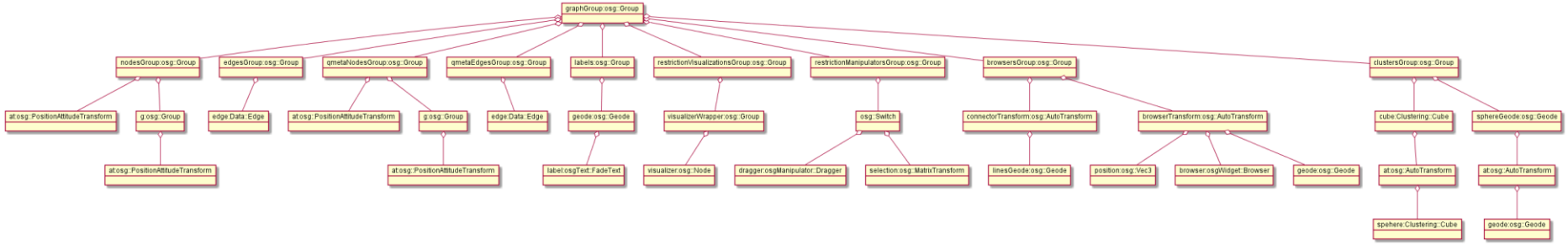
3.1. 0. úroveň abstrakcie(kamery a root)



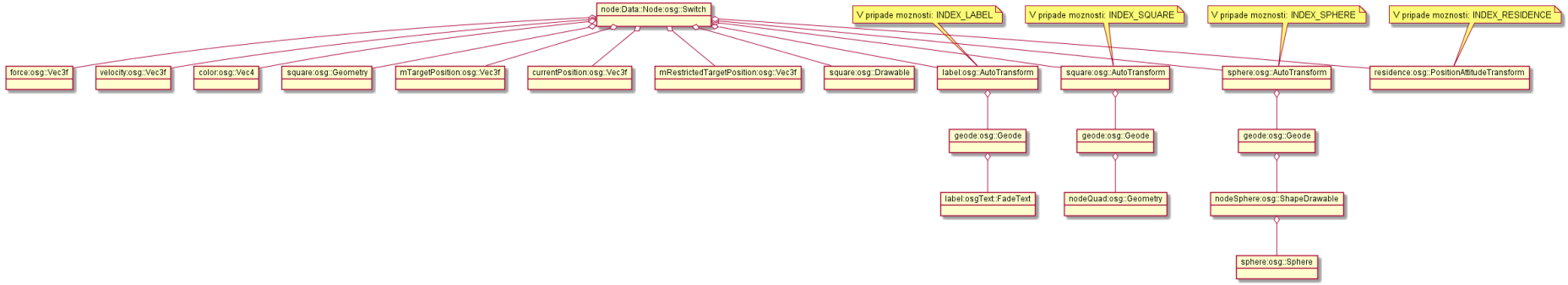
3.2. 1. úroveň abstrakcie(obsah rootu)



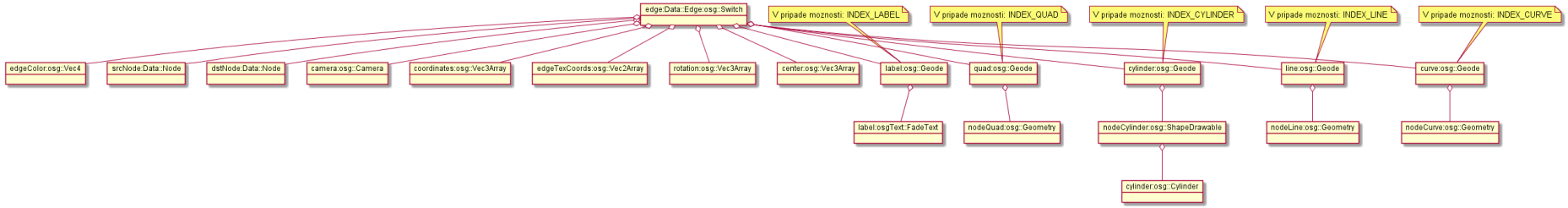
3.3. 2. úroveň abstrakcie(obsah graphGroup)



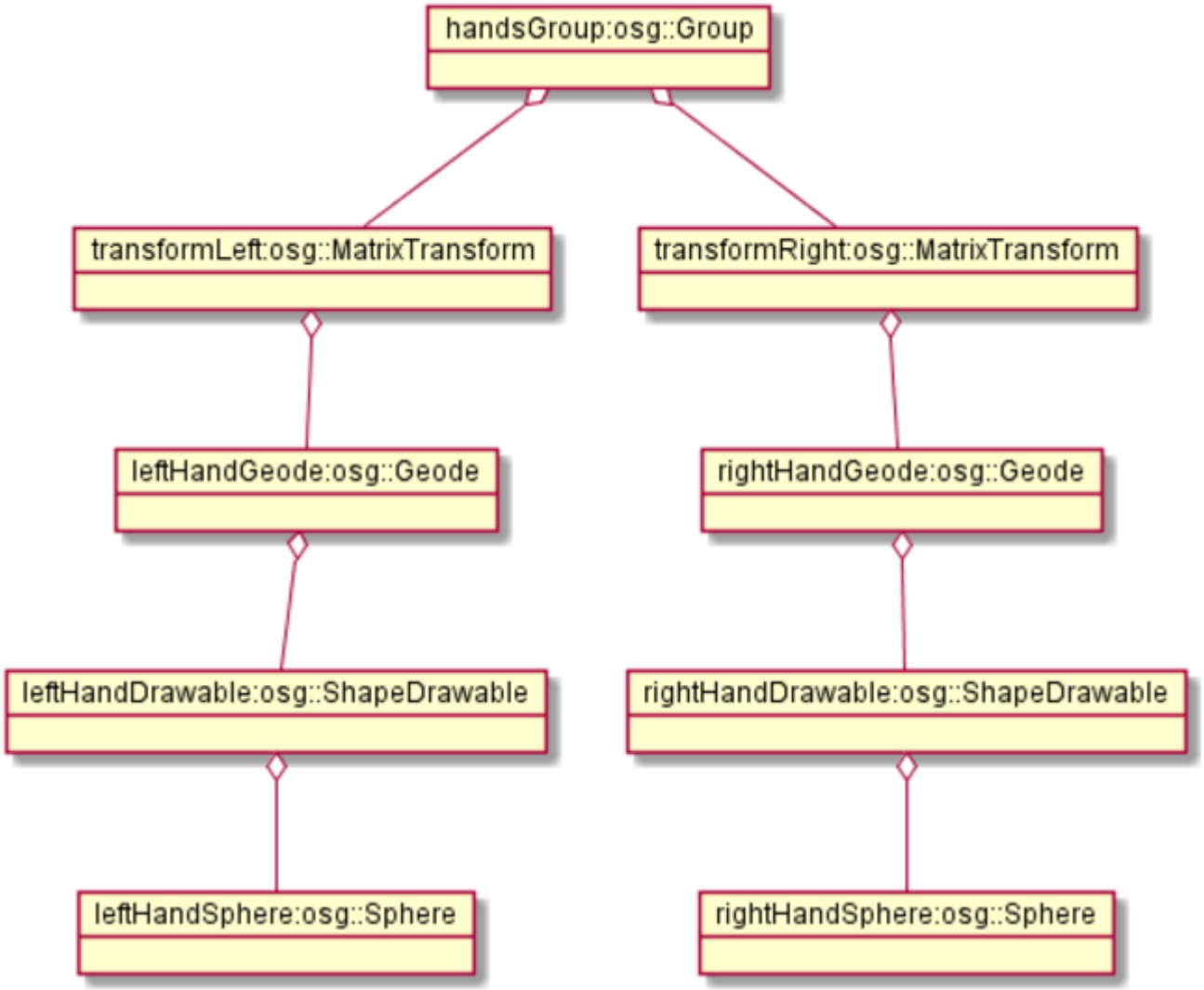
3.4. 3. úroveň abstrakcie(obsah graphGroup)



3.5. Edge(Data::Edge:osg::Switch):

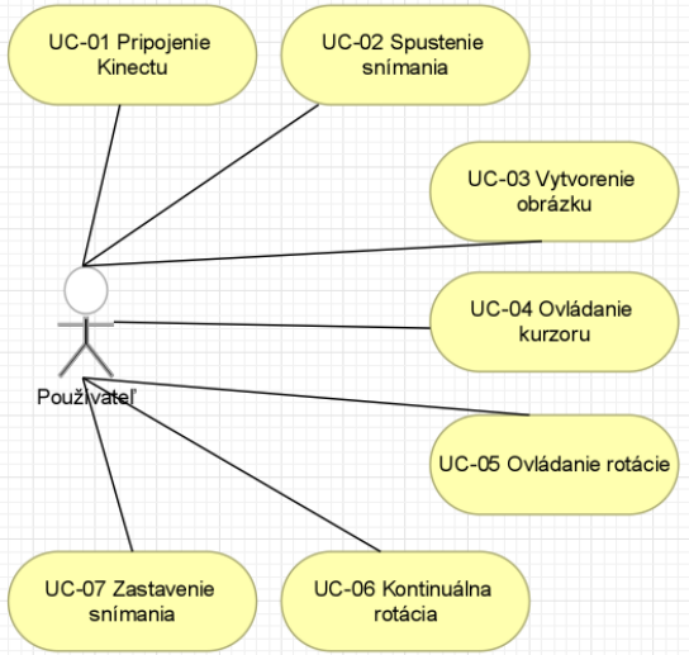


3.6. handsGroup(osg::Group)



3.7. Kinect 2

3.7.1. Use case Kinect 2



Obrázok 3.7.1 Kinect 2 use case

UC-01 Pripojenie Kinectu

1. Používateľ pripojí Kinect do elektrickej siete
2. Používateľ pripojí Kinect do USB
3. Používateľ spustí aplikáciu
4. Systém zobrazí potvrdenie o inicializácii Kinectu

UC-02 Spustenie snímania

1. Používateľ zvolí možnosť zapnutia Kinectu
2. Systém zobrazí okno pre Aruco a Kinect
3. Používateľ zvolí možnosť zapnutia Kinectu
4. Systém zobrazí záznam z live feedu

UC-03 Vytvorenie obrázku

1. Používateľ zvolí možnosť vytvorenia obrázku
2. Systém vytvorí obrázok pomocou normálnej kamery
3. Systém vytvorí obrázok pomocou hĺbkovej kamery
4. Systém vytvorí súbor s hĺbkovou informáciou
5. Systém uloží vytvorené súbory na pevný disk

UC-04 Ovládanie kurzoru

1. Používateľ sa umiestni pred Kinect
2. Používateľ zdvihne ruku s otvorenou dlaňou oproti kamere

3. Používateľ pomocou dlane imituje gesto potlačenia
4. Systém rozpozná dlaň a presunie kurzor podľa jej pozície

UC-05 Ovládanie rotácie

1. Používateľ zvolí možnosť na vypnutie kurzora
2. Používateľ sa umiestni pred Kinect
3. Používateľ zdvihne ruku s otvorenou dlaňou oproti kamere
4. Používateľ pomocou dlane imituje gesto potlačenia
5. Systém rozpozná dlaň a nastaví rotáciu podľa jej pozície

UC-06 Kontinuálna rotácia

1. Používateľ umiestni dlaň na okraj záberu kamery
2. Systém vykoná kontinuálnu rotáciu podľa zvoleného okraju
3. Používateľ umiestni dlaň od okraju kamery
4. Systém zastaví kontinuálnu rotáciu

UC-07 Zastavenie snímania

1. Používateľ zvolí možnosť zastavenie Kinectu
2. Systém zastaví snímanie

4. Moduly systému

4.1. Modul LeapAR

Modul LeapAR slúži na zobrazovanie modelov rúk v obohatenej realite. Modely rúk simulujú pohyby a gestá vykonávané rukami používateľov v reálnom čase na granularite jednotlivých článkov prstov a dlane. Na svoje fungovanie využíva Leap senzor a jeho podpornú knižnicu.

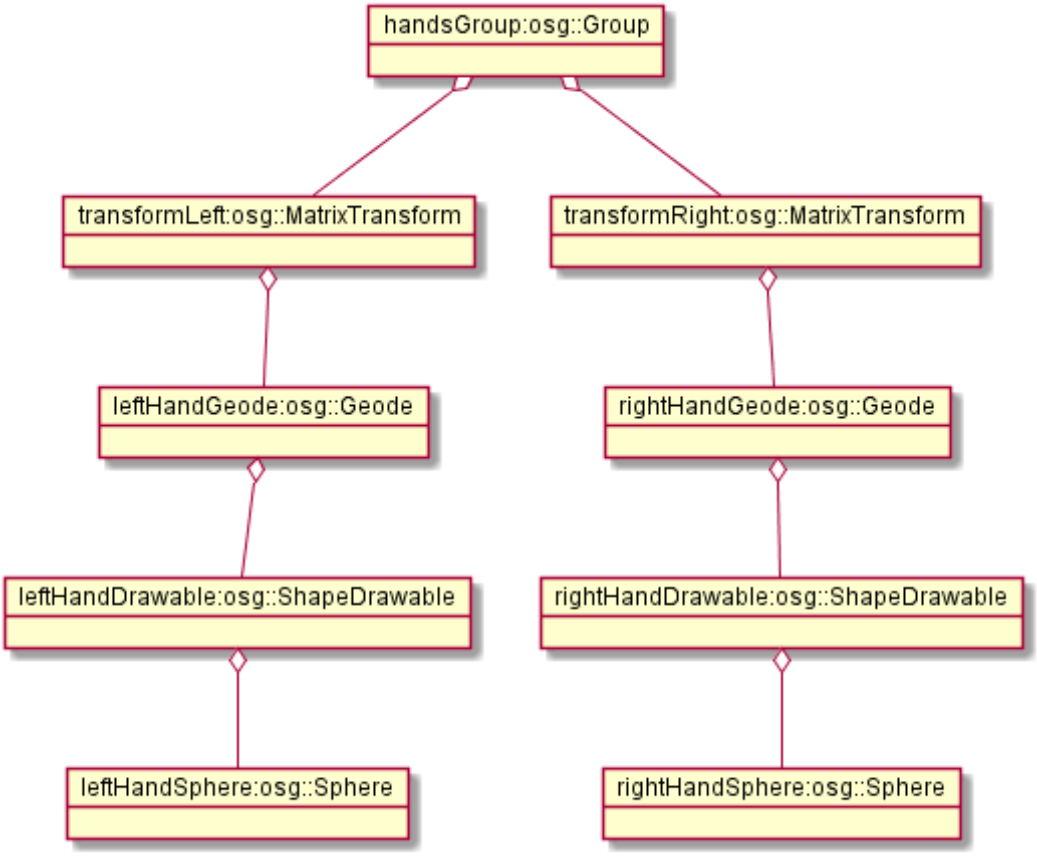
4.1.1. Analýza

V čase vykonávania analýzy existovala základná implementácia tohto modulu, ktorá zachytávala pozície dlaní rúk a prenášala ich do scény vo forme gúľ (sphere), ktoré sa v scéne pohybujú v závislosti od zachytávanej pozície ruky. Pred vykonaním prvotného návrhu bolo potrebne zanalyzovať:

1. Súčasnú štruktúru rúk v scéne
2. Súčasný model adaptéru pracujúceho s Leap knižnicou
3. Triedy a metódy poskytované Leap knižnicou

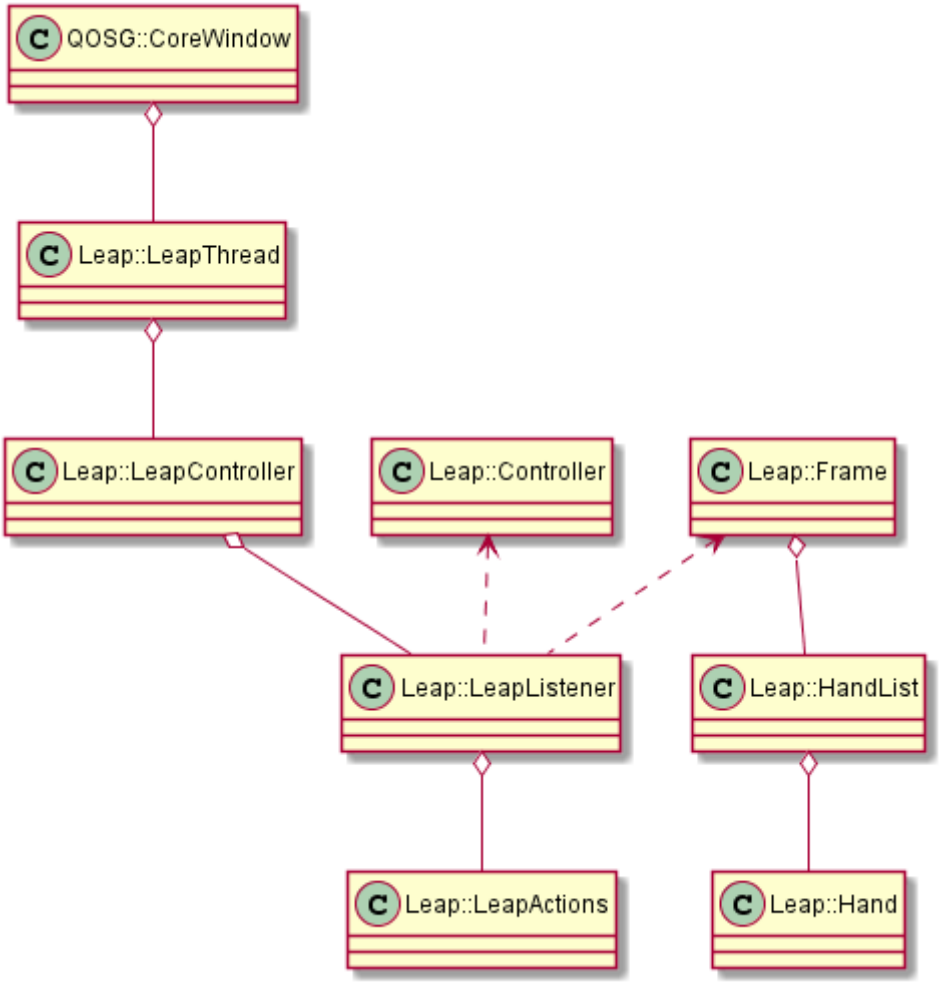
1. Súčasná štruktúru rúk v scéne:

Každá ruka pozostáva z Transformačnej matice, ktorá obsahuje geódu v tvare gule. Pozícia gule v scéne je menená na základe pozície dlane reálnej ruky.



Obrázok 4.1.1 Súčasný stav štruktúry rúk v scéne

2. Súčastný model adaptéru pracujúceho s Leap knižnicou



Obrázok 4.1.2 Súčasný stav štruktúry adaptéru pre Leap senzor.

Leap::LeapThread - predstavuje samostatné vlákno, na ktorom sa získavajú dáta z Leap senzoru.

Leap::LeapController - riadi spúšťanie a vypínanie načítavania dát zo senzoru na vlákne.

Leap::LeapListener - dedí od triedy Listener z knižnice Leapu - ma dôležitú metódu onFrame(), ktorá sa vykonáva ak aplikácia beží v popredí.

Leap::LeapAction – získava štruktúrované dáta z Leap knižnice a upravené ich zasiela do objektu na aplikovanie transformácie.

Leap::Frame – trieda knižnice Leapu, ktorá v sebe zaobaľuje Leap::HandList a ten následne Leap::Hand.

3. Triedy a metódy poskytované Leap knižnicou

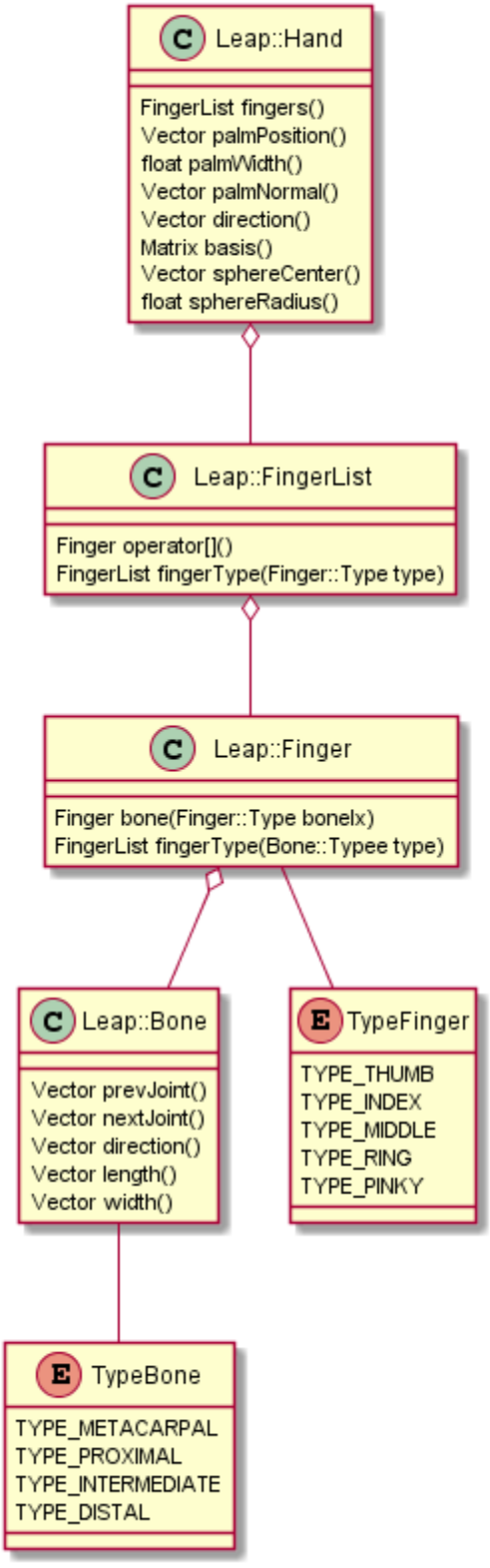
Nižšie uvedený diagram predstavuje zjednodušený pohľad na triedy poskytované Leap knižnicou, pričom zobrazuje triedy a metódy, ktoré sú priamo spojené s pozíciou jednotlivých článkov ruky v scéne.

Leap::Hand – reprezentuje dlaň ruky, z ktorej sa dá získať jej pozícia, natočenie, šírka prstov , ale aj zoznam prstov.

Leap::FingerList – reprezentuje zoznam kostíjednotlivých prstov.

Leap::Finger – reprezentuje samotný prst, ktorý sa skladá z kostí (Leap::Bone) a udržiava si ku každému prstu jeho typ (palec, ukazovák, stredný prst, prstenník, malíček).

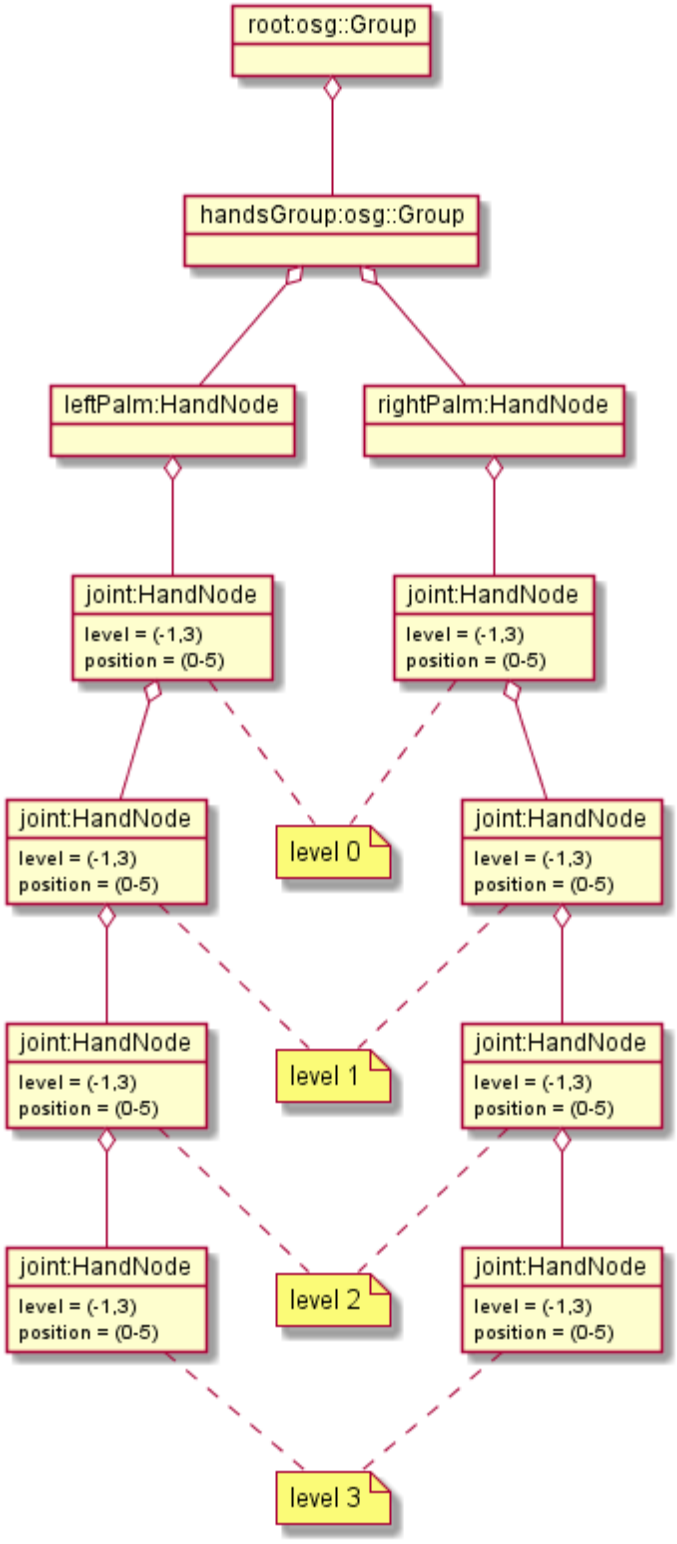
Leap::Bone – reprezentuje kosti nachádzajúce sa v prstoch. Udržiava si informácie o svojom strede, orientácii, dĺžke, šírke, ale aj oboch kĺboch na ktoré je napojená a type kosti.



Obrázok 4.1.3 Dôležité triedy a metódy poskytované Leap knižnicou

4.1.2. Návrh

Prvotný návrh modelu rúk je znázornený v Obrázok 4.1.4. Z pôvodného modulu sa nezmenil iba zaobaľujúci objekt handsGroup. Na rozdiel od knižnice Leapu nebude model ruka vyskladaný z kostí, ale z jednotlivých kĺbov, pričom kosti budú reprezentované ako hrany (spoje) medzi týmito vrcholmi.



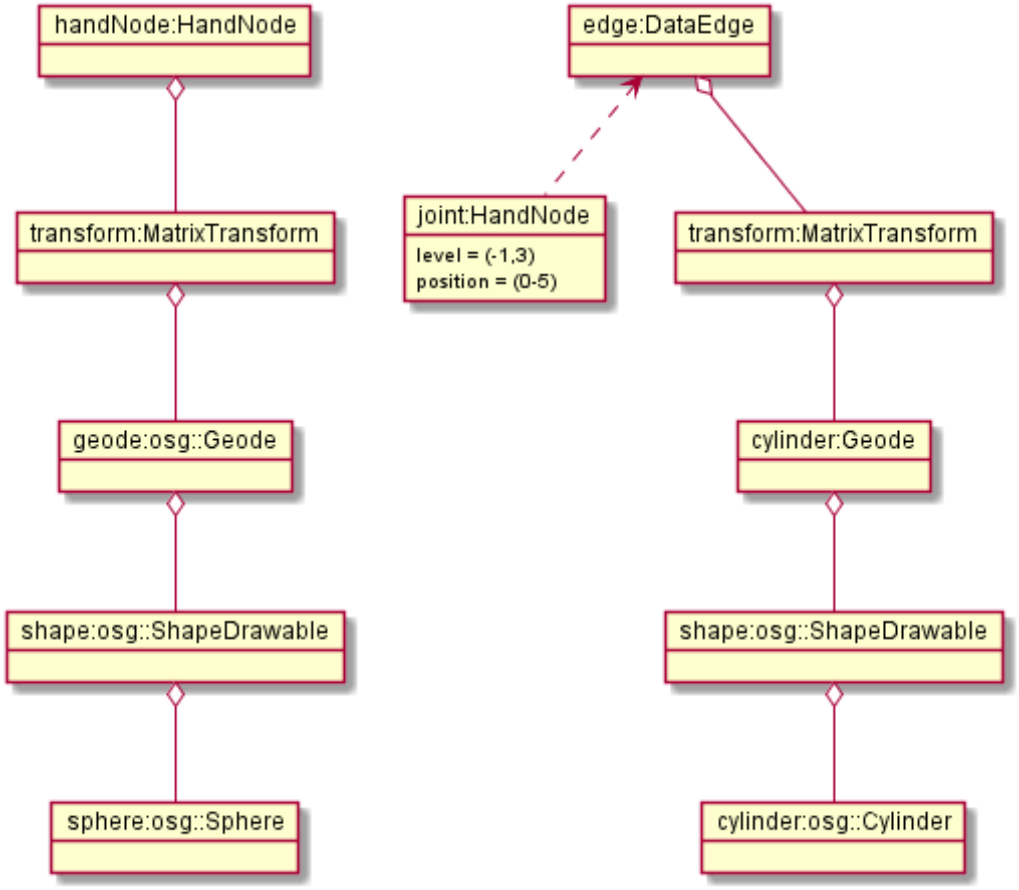
Obrázok 4.1.4. Diagram objektov navrhujúci štruktúru objektov tvoriacu model ruky.

Dlaň reprezentujúca stred ruky, bude znázorňovaná ako kĺb, s tým, že bude mať 5 potomkov typu HandNode, ktoré budú predstavovať začiatky jednotlivých prstov. Ďalší potomkovia týchto kĺbov budú kĺby jednotlivých článkov prstov. Každý HandNode (kĺb) si v sebe bude udržiavať informáciu o svojej pozícii v ruke pomocou atribútov:

- Level – poradie kĺbu v konkrétnom prste.
- Position – typ prstu, ku Ktorému kĺb patrí.

Kĺby budú vizuálne reprezentované pomocou gúľ (osg::Sphere) a kosti pomocou valcov (osg::Cylinder). Vizuálna reprezentácia týchto elementov bude v oboch prípadoch zaobalená do vlastných nadtried (viď Obrázok 4.1.5):

- Kĺb = HandNode
- Kosť = DataEdge



Obrázok 4.1.5. Diagram objektov navrhujúci zakomponovanie vlastných objektov do scény grafu.

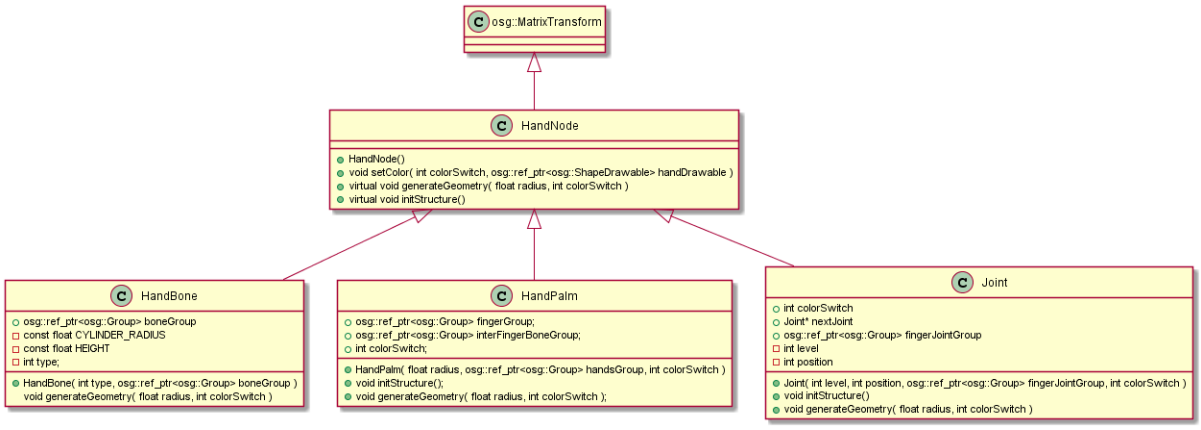
4.1.3. Implementácia

Pr implementácií bola zachovaná štruktúra adaptéru pre Leap senzor. Úprava na stala v striede *CustomLeapManager,* do ktorej boli pridané metódy:

```
void updateHands( Leap::Hand leftHand, Leap::Hand rightHand );
void updateFingers( HandPalm* palm, Leap::FingerList fingers );

void updateJoints( osg::Group*  fingerGroup, Leap::Finger finger,
int fingerPosition );
```

```
void updateFingerBones( osg::Group*  fingerGroup, Leap::Finger
finger, int fingerPosition );

void updateInterFingerBones( osg::Group*  interFingerBoneGroup,
Leap::FingerList fingers );

void updateInterFingerWristBone( osg::Group*  interFingerBoneGroup,
Leap::FingerList fingers );
```

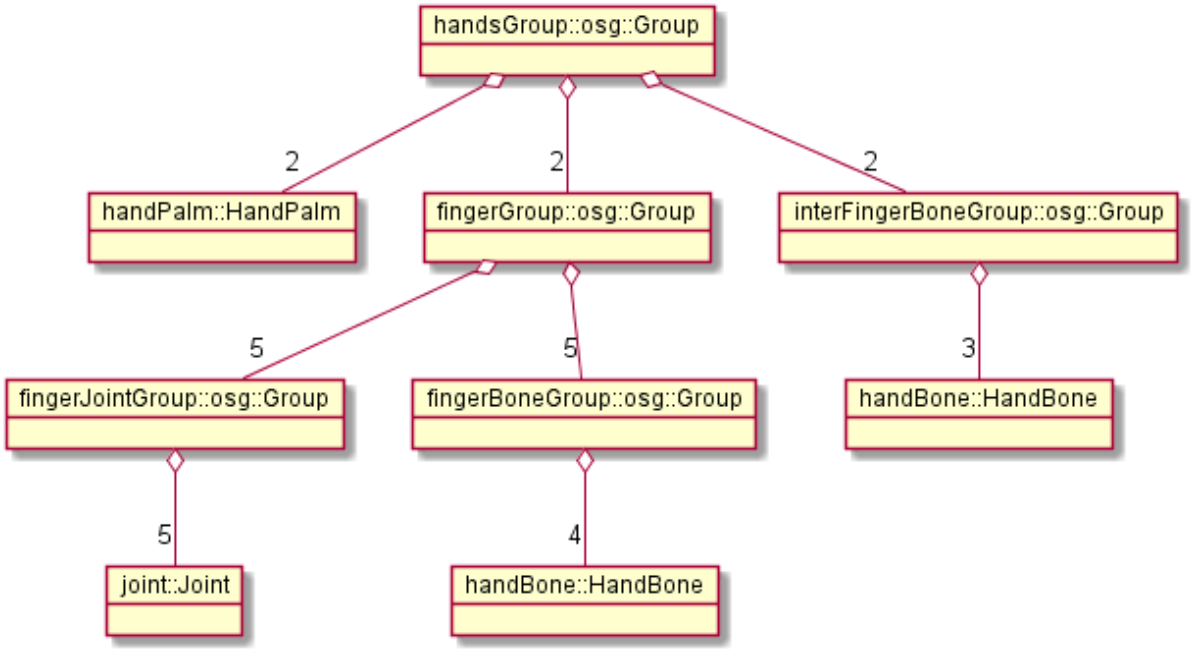
Privolaním `updateHands` sa automaticky postupne spustia aj ostatné a postupne preiterujú celú štruktúru objektov ruky v scéne. Model rúk je zložený z dlane(handPalm), kĺbov (Joint) a kostí (HandBone) viď Obrázok 4.1.6.



Obrázok 4.1.6. Diagram tried zobrazujúci štruktúru tired modelu ruky.

V scéne sú ruky reprezentované osg::Grupou – handsGroup. HandsGroup má v sebe pre každú ruku uloženú dlaň(handPalm), prsty(fingerGroup), medziprtové kosti (interFingerBoneGroup). Prsty osahujú osg:: Group pre každý jeden prst, s tým že prst je zložený z kĺbov a kostí. Medziprtové kosti

sú kosti medzi kĺbami v hánkach a kosť zápästia, ktoré sa v reálnej ruke nenachádzajú, ale dodávajú celistvejší výzor modelu ruky.



Obrázok 4.1.7. Diagram objektov zobrazujúci štruktúru objektov v scéne.

4.2. Modul Kinect

4.2.1. Analýza

Projekt bol doteraz uspôsobený na zariadenie kinect v1. Avšak nepodporuje v súčasnosti zariadenie kinect v2, ktoré máme aj na fakulte. Z toho dôvodu bolo potrebné zistiť možnosti, pomocou ktorých by bol náš projekt schopný fungovať aj s použitím kinect-u v2.

Na to sú potrebné tieto časti:

- libfreenect2
- openni2
- NiTe 2.2

4.2.2. Návrh

V súčasnosti sa ako najpoužívanejšie a najstabilnejšie riešenie javí opensource projekt na github:

`https://github.com/OpenKinect/libfreenect2`

Projekt je stále vo vývoji, avšak už v súčasnosti funguje pod MacOs X, Linuxom a aj Windowsom. Taktiež sa v projekte nachádza dokumentácia, ako tieto libfreenect2 spojiť s openni2 a Nite 2.2. V prípade nášho projektu bude potrebné meniť rozlíšenie kinect-u v2 na 640 x 480, pretože v opačnom prípade by bolo potrebné prerábať celý projekt. V súčasnosti taktiež zostáva otázne, či bude môcť projekt podporovať obe verzie kinectu súčasne.

4.2.3. Implementácia

4.2.4. Testovanie