SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE

Fakulta informatiky a informačných technológií

INŽINIERSKE DIELO

Tímový projekt VizReal

Predmet:	Tímový projekt II.
Členovia tímu:	Bc. Matej Brečka
	Bc. Richard Galeštok
	Bc. Richard Kubík
	Bc. Daniel Kuľbak
	Bc. Jakub Lackanič
	Bc. Viera Maronová
	Bc. Martin Redžepovič
Vedúci projektu:	Ing. Peter Kapec, PhD.
Akademický rok:	2018/2019

Obsah

1 O projekte VizReal

Táto kapitola obsahuje úvod do projektu, definované globálne ciele pre projekt, celkový pohľad na systém týkajúci sa architektúry a opis všetkých technológií, ktoré boli použité.

1.1 Úvod

Keď si predstavíme projekt, ktorý je napísaný v programovacom jazyku, vybavia sa nám funkcie, premenné a závislosti týchto funkcií. Práve tieto dve veci, teda funkcie a závislosti sa dajú zobrazovať graficky vo forme grafov, kde funkcie predstavujú vrcholy a ich volania zase hrany. Náš tím sa venujeme zobrazovaniu práve takýchto grafov pomocou virtuálnej alebo rozšírenej reality. Vizualizovanie týchto grafov nám dokáže pomáhať pri analýze kódu, chápaní závislosti jednotlivých modulov alebo na prezentačné účely.

Cieľom nášho projektu je teda poskytnúť užívateľovi väčší prehľad v napísanom kóde, či už je to vo virtuálnej realite kde kód predstavuje graf, alebo v rozšírenej realite kde je kód zobrazený ako metafora mesta.

1.2 Globálne ciele projektu na zimný semester

Keďže sa jedná o prerobenie projektu, cieľom na zimný semester je viac-menej opraviť chyby, ktoré sa vyskytli pri predošlej implementácií. Keďže sa v našom projekte vyskytuje viacero jazykov, je potrebné opraviť rozhrania pre komunikácie medzi nimi. Na overenie novo vzniknutých funkcionalít sa zase musia písať testy, ktoré treba doplniť aj do niektorých starších modulov.

V projekte sa taktiež nachádzajú zastaralé knižnice, ktoré je potrebné nahradiť inými. Ak si to zhrnieme, cieľom na zimný semester je refaktorovanie projektu, pridanie nových funkcionalít, zjednodušiť používanie a zlepšiť dokumentáciu projektu. Taktiež sme v projekte pridali nový modul na zobrazovanie grafov - unity. Je preto potrebné preniesť vlastnosti grafu a zároveň implementovať nástroj na vytvorenie schémy grafu.

1.2.1 Splnené ciele

V tomto semestri sa nám podarilo splniť čo sme si stanovili na jeho začiatku. Nepodporované knižnice sa nahradili novými, upravili sa rozhrania a napísali sa testy tam, kde chýbali ale aj pre novovzniknuté implementácie.

Build projektu je jednoduchší a intuitívnejší. Bol pridaný modul unity, do ktorého sa nám podarilo preniesť vlastnosti grafu ako je farba alebo veľkosť, a ten ho dokáže vykresliť na scénu.

1.3 Globálne ciele projektu na letný semester

Počas letného semestra nadväzovať na základnú implementáciu, ktorú sme vykonali minulý semester. Chceme sa zamerať na základné techniky interakcie s grafom a navigácie v scéne. Budeme vylepšovať už existujúce zobrazenie grafu a zlepšíme aj prenos vizuálnych vlastnosti grafu (uzlov a hrán) z LUA do Unity.

V druhej polovici sme si za cieľ stanovili rozšíriť existujúcu implementáciu o podporu rozšírenej reality. Na snímanie okolia nám poslúži stereocamera - ZED kamera. Na manipuláciu s grafom použijeme Leap Motion.

1.3.1 Splnené ciele

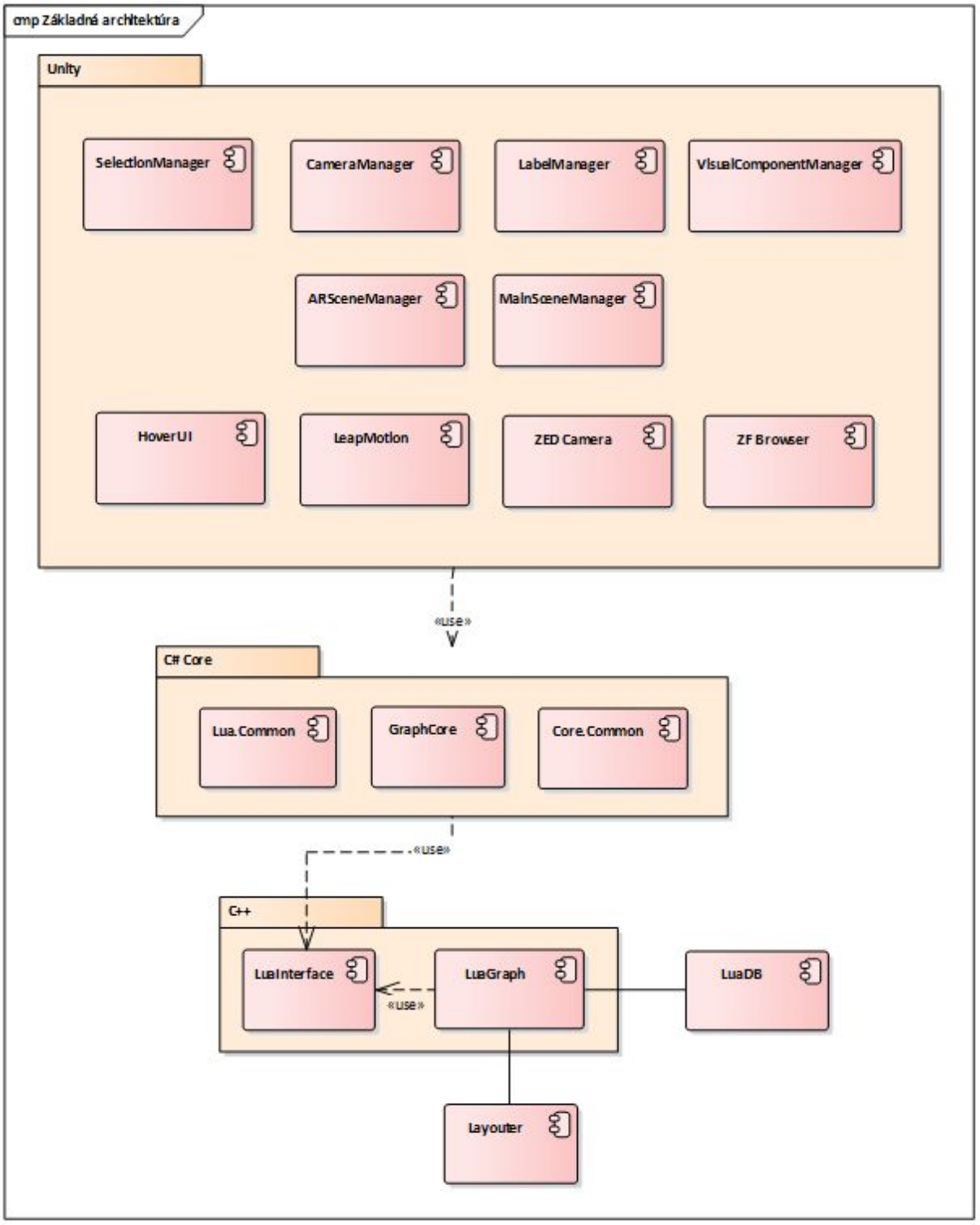
Počas letného semestra sa nám úspešne podarilo do projektu implementovať základnú navigáciu a interakciu s grafom. Vylepšili sme aj vizuálne mapovanie vlastnosti softvéru na graf. Každá vizuálna vlastnosť je teraz reprezentovaná samostatným Unity komponentom. S využitím ZED kamery sme dokázali graf umiestniť aj do rozšírenej reality. Pre lepšiu manipuláciu so scénou a grafom vo virtuálnej a rozšírenej reality sme úspešne implementovali Menu pomocou knižnice HoverUI, ktoré môže používateľ ovládať prostredníctvom Leap Motion.

1.4 Celkový pohľad na systém

Kapitola obsahuje opis základnej architektúry spolu s obrázkom, na ktorom je možné vidieť všetky komponenty.

1.4.1 Základná architektúra

Na obrázku nižšie je možné vidieť základnú architektúru projektu. Od predošlej implementácie pribudol modul Unity, ktorý slúži na zobrazovanie grafov.



Obrázok č.1 - Základné komponenty systému

1.5 Technológie

V našom projekte využívame tieto zariadenia:

- **LeapSensor**

Hardvérový senzor pre počítač, ktorý podporuje zaznamenávanie pohybu rúk a prstov a transformovanie na vstup, podobný ako pri myši, pričom nepotrebuje žiaden dotyk rúk.

- **HTC Vive**
  Náhlavný systém pre virtuálnu realitu, podporujúci taktiež interakciu s virtuálnym prostredím pomocou ovládačov uchopených v ruke.
- **Oculus Rift**
  Náhlavný systém pre virtuálnu realitu inej spoločnosti, takisto podporujúci interakciu pomocou ovládačov uchopených v ruke
- **ZED 3D camera**
  Stereo kamera využiteľná na hĺbkové skenovanie prostredia a snímanie pohybu.

2 Moduly systému

Kapitola obsahuje podrobný opis všetkých modulov systému Softviz pre každú z vrstiev spolu s použitými algoritmami a technológiami.

2.1 Layouter

Tento modul má zodpovedá za výpočet rozloženia grafu a správu layoutov grafu.

2.1.1 Layout Manager

Zodpovedá za spravovanie layoutov grafu. Každý graf má svojho vlastného layout managera. Obsahuje základne funkcie pro kontrolu layoutovania ako sú štart, stop, pauza. Zabezpečuje to, ktorý algoritmus je v súčasnosti aktívny a zabezpečuje prenos dát z a do algoritmu.

Vo funkcii *updateNodes* dochádza k aktualizovaniu uzlov LuaDB grafu z práve aktívneho algoritmu. Z algoritmu sa načíta pozícia každého uzla, prípadne ďalšie atribúty ohľadom uzla, ktoré algoritmus môže obsahovať ako napríklad veľkosť uzla, a táto pozícia sa nastaví zodpovedajúcemu uzlu v LuaDB grafe.

2.1.2 Fruchterman-Reingoldov algoritmus

Jeho implementácia je v jazyku terra, ktorého rýchlosť je porovnateľná s jazykom C. Algoritmus si na začiatku vytvorí kópiu LuaDB grafou. Z lua tabuliek obsahujúce uzly a hrany si vytvorí pole terra štruktúr *TNode* a *TEdge*, ktoré reprezentujú graf a obsahujú všetky atribúty, ktoré sú potrebné pre výpočet rozloženia grafu. Vo funkcií initialize, sa vypočíta počiatočné rozloženie grafu. Po zavolaní funkcie *runlayouting* sa vytvorí nové vlákno, kde iteratívne prebieha výpočet nového rozloženia grafu.

2.2 LuaDB

Modul je napísaný v programovacom jazyku Lua. Hlavnou úlohou je analýza zdrojového kódu, ktorý je modulu poskytnutý zatiaľ zadaním priamej cesty, analyzuje volania funkcií,

komplexitu, a rôzne metriky, na základe ktorých sa neskôr ostatné moduly rozhodujú o vlastnostiach vykreslenia uzlu, napríklad veľkosť uzlu.

2.3 LuaInterface

Má na starostí komunikáciu s Lua časťou kódu využitím Sol2 knižnice. Je naprogramovaný v jazyku C++. Poskytuje základné funkcie pre vykonávanie Lua kódu, volanie Lua funkcii, a získavanie informácií z modulov v jazyku Lua. Každá funkcia, ktorá má byť prístupná v Lua.Common module by mala mať svoj ekvivalent v príslušnom ExportC súbore. Napríklad pre LuaInterface.cpp je to LuaInterfaceExportC.cpp. Tieto funkcie sa musia nachádzať v *extern "c"* bloku a musia mať definíciu kompatibilnu s jazykom C.

2.4 LuaGraph

Úlohou tohto modulu je s využitím modulu LuaInterface načítať LuaDB graf z jazyka Lua do jazyka C++. Jeho hlavná logika sa nachádza vo funkcii *LuaGraph::loadGraph(int id).* Podobne ako LuaInterface, každá pridaná funkcia, ktorá má byť prístupná v Lua.Common by mala byť definovaná aj v príslušnom ExportC súbore.

2.5 Lua.Common

Tento modul využíva PInvoke pre prístup k funkciám exportnutých v C++ DLL LuaGraph a LuaInterface. Jeho jedinou úlohou je zjednodušiť prístup k týmto funkciám. Hlavnými triedami sú *LuaInterface* a *LuaGraph*, ktoré volajú príslušné C++ funkcie a v prípade potreby vykonávajú konverziu dát na príslušné dátové typy v jazyku C#. Pre zjednodušenie načítania poľa, štruktúr a stringov bola vytvorená trieda *MarshallingUtils,* ktorá obsahuje pomocné metódy na prevod týchto dátových typov z jazyka C do C#.

Pre reprezentáciu akéhokoľvek objektu v jazyku Lua slúži trieda *LuaObject*, ktorá implementuje aj konverziu lua typu na C# typ. Keďže väčšia tried v tomto module pracuje priamo s pamäťou, je potrebné myslieť na to, že túto pamäť treba manuálne dealokovať pretože nebude odstránená automaticky garbage collectorom. Pre zjednodušenie je každá takáto trieda implementovaná návrhovým vzorom *Dispose.* Je na to potrebné myslieť najmä pri volaní metód LuaObject.GetObject a LuaInterface.GetObject (a každej inej, ktorá priamo vracia LuaObject), pretože dealokacia pamäte, ktorú si alokuje *LuaObject, je* na tom, kto túto funkciu zavolal.

Pre zjednodušenie načítania dát pre uzly a hrany z tabuľky data v LuaDb grafe bol vytvorený atribút *LuaParameterAttribute.* Každý údaj v tabuľke data, ktorý chceme načítať by mal mať svoj ekvivalent v triede *LuaNodeData/LuaNodeEdge* alebo *GraphObjectData*, ak ide o spoločnú vlastnosť ako je napríklad farba. Každá premenná tejto triedy by mala byť označená atribútom *LuaParameter*, kde v parametri *LuaName* by mal byť názov tejto premennej v data tabuľke. Všetky premenné označené týmto atribútom su automatický pri vytvoreni objektu typu *GraphObject* načítané z LuaDB grafu vo funkcii GraphObject.*LoadData.* Pri odstránení objektu typu *GraphObject* je automatický dealokovaná všetka pamäť, ktorú tieto dáta používajú, takže nie je potrebné volať metódu Dispose na žiadnej premennej označenej týmto atribútom.

2.6 GraphCore

V tomto module je obsiahnutá všetka logika týkajúca sa grafu. V metóde *Graph.InitializeGraph* dochádza k vytvoreniu grafu z Lua.Common.Graph. K ďalšiemu aktualizovaniu tohto grafu z LuaDb grafu dochádza v metóde *Graph.UpdateNodes*, ktorá sa prostredníctom *GraphManager*-a volá v pravidelných intervaloch z unity herného objektu *GraphLoader*. Súčasťou modulu je trieda *LayoutManager,* ktorý je reprezentáciou layout managera, ktorý sa nachádza v module layouter.

Vlastnosti objektov typu *GraphCore.Node* a *GraphCore.Edge*, ktoré sa pravidelne menia ako napríklad farba, pozícia, tvar a podobne, by mali obsahovať aj príslušnú udalosť, ktorá signalizuje zmenu tohto atribútu, aby bolo možné podľa neho následne aktualizovať vizuálnu reprezentáciu grafu v Unity.

2.7 Unity

Modul reprezentuje prezentačnú vrstvu celkového projektu. Vizualizuje graf analyzovaného zdrojového kódu a umožňuje manipuláciu grafu ako celku a aj s jeho jednotlivými vrcholmi a prepojeniami. Graf zobrazuje komplexitu analyzovaného zdrojového kódu a prepojenia medzi modulmi. Rámec Unity si vyžaduje implementáciu skriptov v C#.

Súčasťou tohto modulu sú dodatočné moduly ako podpora vykresľovania vo virtuálnej realite na Oculus Rift alebo HTC Vive a interakciu s grafom pomocou senzoru rúk Leap Motion, vykresľovanie v rozšírenej realite a manipuláciu s grafom sledovaním špeciálnych symbolov pomocou bežnej kamery.

2.7.4 AR Technológie

Na spustenie AR scény sú nutné okrem VR/AR okuliarov 3 technológie. Tie sú:

- **LEAP Motion Senzor**
- **ZED Kamera**
- **HoverUI Kit**

ZED Kamera poskytuje zobrazenie priestoru okolo používateľa. V projekte je použité **SDK v2.8.** Pred používaním je nutné SDK najprv nainštalovať. To je dostupné na stránke: https://www.stereolabs.com/developers/. Kamera dokáže okrem snímania aj vnímať okolie ako 3D priestor vďaka hĺbkovému snímaniu. V projekte sa preto používa primárne na dve veci. Prvou je že dokáže vykonávať **“spatial mapping”**, čo je vytvorenie digitálneho meshu z prostredia a následne ho aj ukladať. Druhou vlastnosťou je **“depth perception”,** pri ktorej ZED kamere zadáme dvojrozmerné súradnice a ona nám dokáže vrátiť vzdialenosť kamery od body, na ktorý sa pozeráme. Táto funkcia sa používa na umiestňovanie grafu na vybraný bod.

Do scény sa po nainštalovaní pridá modul, ktorý sa nazýva **Zed_Rig_Stereo.** Ten slúži namiesto kamery v scéne a taktiež obsahuje komponent *ZED Manager,* čo je knižnica funkcií ZED kamery. Pokiaľ potrebujeme využiť funkciu tejto knižnice, v skripte si vytiahneme práve tento komponent.

LEAP Motion zabezpečuje sledovanie pohybu rúk. Použite je **SDK v2.3.1**. Pre inštaláciu je nutné stiahnuť vyššie spomínaný SDK zo stránky: https://developer.leapmotion.com/get-started/ a nainštalovať. Parametre sú v Unity už prednastavené. LEAP je v tejto scéne primárne kvôli interaktívnemu menu HoverUI.

Na sprístupnenie LEAP senzoru v AR scéne je potrebné na kameru pripnúť *LEAP XR Service Priovider* a na sprístupnenie rúk treba pridať objekt *LEAP Rig* do scény.

HoverUI Kit slúži ako interaktívne menu, ktoré je pripnuté na rukách pomocou LEAP senzoru. Inštalácia nie je potrebná, všetky potrebné prostriedky sú už v projekte. Aktivuje sa obrátením ľavej ruky smerom nahor a naviguje sa ním pomocou ukazováku pravej ruky. Pomocou tohto menu sa interaguje so samotným grafom, teda funkcie ako zapnúť layoutovač, zmeniť typy uzlov/hrán alebo sa pomocou neho umiestni graf na miesto, na ktoré sa momentálne používateľ pozerá.

Na pripnutie Hover UI menu na LEAP ruky je potrebné na Leap Rig pripnúť *Hover Input Leap Motion.*

2.7.1 Input Handler

Tento modul vytvára jednotný systém pre spracovanie vstupov. Desktopová časť zaobaľuje Unity input systém a AR časť zaobaľuje Leap input systém.

Modul pracuje na základe Observer patternu, kde skript, ktorý chce používať Input sa prihlási na vstup metódou *InputHandler.Subscribe.* Akcie pre vstupy sú zadefinované v *IBaseInputAction*. Konkrétne definície vstupu si definujú jednotlivé spôsoby vstupov. (napr. *DesktopInputAction*, *LeapInputAction*). *InputHandler.Subscribe* zároveň vracia *InputSubscribtion*, s ktorou je možné input zrušiť cez metódu *InputHandler.Unsubscribe*.

Callbacky pre inputy zavolá *InputHandler*, pričom dozerá na to, aby zároveň neboli zavolané callbacky pre rovnaké vstupy (napr. Ctrl+Mouse0 a zároveň Mouse0).

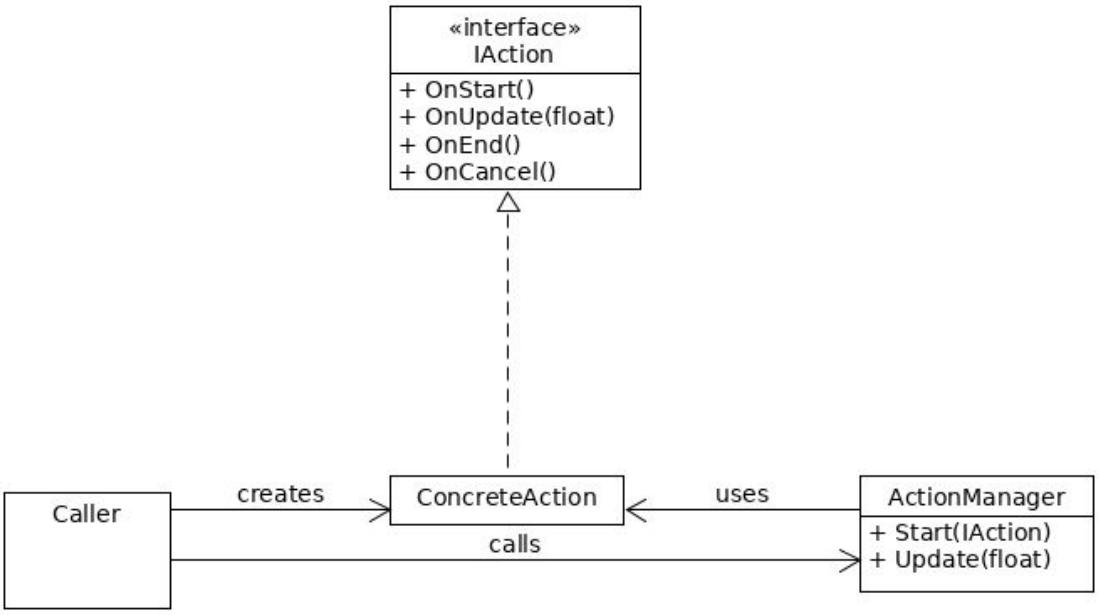
Jedna vstupová akcia resp. vstupová udalosť je definovaná ako *InputEvent,* pričom sa event skladá z jedného alebo viacerých *InputElement*. InputElement sa dá predstaviť ako jedna klávesa alebo axis (napr. stlačenie Mouse0) a InputEvent ako kombinácia týchto elementov (napr. Ctrl+Mouse0).

Desktopová časť inputov zaobaľuje Unity konštanty a metódy:

- *Input.Desktop.Key* - zaobaľuje *UnityEngine.KeyCode*
- *Input.Desktop.Axis* - zaobaľuje *UnityEngine.Axis,* resp. *string*

2.7.2 Actions

Tento modul vytvára jednotný systém pre spúštanie znovupoužiteľných kusov kódu spúštané z grafického rozhrania. Je to naša implementácia Command patternu.



Obrázok č. 2 - Diagram znázorňujúci Command pattern

Akcie sa púštajú pomocou metódy *ActionManager.Start*. *IAction* má svoj životný cyklus *OnStart*, *OnUpdate*, *OnEnd*, *OnCancel*. Taktiež určuje svoju *IActionGroup*. V jeden moment nemôžu prebiehať viaceré akcie z rovnakej *IActionGroup*. Ak sa začne iná akcia, tak sa predošlá zruší a zavolá sa jej *OnCancel*. Akcie majú predvolenú skupinu *ActionGroup.None*, pre ktorú tieto vzájomné vylučovanie neplatí.

Taktiež je zadefinovaná *OneTimeAction*, ktorá je špeciálny prípad *IAction*, ktorá nemá svoj životný cyklus a vykoná sa iba jedenkrát.

2.7.3 HoverUI Kit

K modulu HoverUI Kit[1] patria, v rámci AR scény, herné objekty:

HoverCast

Herný objekt obsahuje všetky vizuálne vlastnosti, komponenty a definície animácií, ktoré sa dejú pri používaní menu. Vyskytujú sa v ňom 4 objekty:

- **OpenItem** - definícia tlačidla na otváranie/zatváranie menu.
- **TitleItem** - obsahuje nastavenie textu pre názov menu.
- **BackItem** - definícia tlačidla pre krok späť, ktoré sa nachádza sa v strede dlane.
- **Rows** - aktuálne rozloženie hlavného menu, ktoré môžeme upravovať (prípadne prídávať nové položky).

HoverMock

Dcérsky herný objekt Mock obsahuje makety všetkých nami používaných typov komponentov v hlavnom menu. Pri vytváraní nových komponentov je možné tieto herné objekty kopírovať, čím sa môžeme vyhnúť zdĺhavému a komplikovanému manuálnemu vytváraniu komponentov prostredníctvom externých HoverBuilderov. Pri nakopírovaní

[1] https://github.com/aestheticinteractive/Hover-UI-Kit/wiki

netreba zabudnúť na zmenu názvu herného objektu a jeho vlastností *ID* a *Label* (prípadne *Navigate To Row* pre typ **RowSelector,** *Radio Group ID* pre typ **Radio,** *Range Min* a *Range Max* pre **Slider**).

Tento objekt obsahuje makety komponentov:

- **Row** - riadok vnoreného menu po kliknutí na RowSelector, ktorý obsahuje konkrétne komponenty.
- **RowSelector** - komponent typu Selector, ktorý sa stará o navigáciu do vnoreného menu (Row).
- **Back** - komponent typu Selector, ktorý má za úlohu o navigáciu do rodičovského menu (krok späť).
- **Selector** - komponent, ktorý si správa ako klasické tlačidlo - neuchováva si stav.
- **Radio** - komponent, ktorý sa spojí s ostatnými komponentami tohto typu prostredníctvom Group ID - v jednej skupine môže byť označené maximálne jedno Radio tlačidlo.
- **CheckBox** - tlačidlo, ktoré si uchováva stav (vieme určiť, či bol stlačený alebo nie).
- **Slider** - posúvnik, ktorý sa je možné nastaviť na jednu z hodnôt z intervalu <min,max>.
- **TextItem** - komponent, ktorý zobrazuje text.

HoverUIManager

Tento herný objekt je jedným z manažérov v AR scéne, ktoré sú uložené v objekte *ManagersAR*. Obsahuje referencie na všetky komponenty HoverUI menu a skript HoverUIManager, ktorý na začiatku životného cyklu priradí komponentom event handlery, ktoré budú pri interakcii spúšťať jednotlivé akcie z *Action Pattern*-u.

```
public class HoverUIManager : MonoBehaviour {

    private GraphLoader graphLoader;
    // ...
    public HoverItemDataCheckbox pauseLayouting;

    public void Start()
    {
        graphLoader = GameObject.FindGameObjectWithTag(GameObjectTags.GraphLoader).GetComponent<GraphLoader>();

        // ...

        pauseLayouting.OnSelected += (sender) =>
        {
            if (pauseLayouting.Value)
            {
                ActionManager.Instance.Start(new PauseLayoutingAction(graphLoader.GraphManager.Graphs));
            }
            else
            {
                ActionManager.Instance.Start(new ResumeLayoutingAction(graphLoader.GraphManager.Graphs));
            }
        };
    }

    // ...
}
```

Obrázok č. 3 - Ukážka priradenia eventu pre komponent typu CheckBox

Ukázku priradenie eventu v kóde zobrazuje obrázok č. 2 a sumár komponentov s ich udalosťami tabulka č. 1.

Tabuľka č. 1 - Sumár komponentov a jeho eventov

Komponent	**Udalosť**
HoverItemDataSelector	*OnSelected*
HoverItemDataCheckbox	*OnSelected, OnDeselected*
HoverItemDataRadio	*OnSelected, OnDeselected*
HoverItemDataSlider	*OnValueChanged*

2.7.5 Camera Manager

Skript zahŕňa funkcionalitu používanú pri interakcii s kamerou. Zabezpečuje efektívne manipulovanie a pohyb vo vizualizovanom grafe. Medzi hlavnú funkcionalu tejto časti systému patrí približovanie sa ku konkrétnym uzlom (*FocusObject*), orbitovanie okolo jedného alebo skupiny uzlov (*OrbitalMode*) a automatické oddialenie kamery do vzdialenosti potrebnej na zobrazenie kompletného grafu (*ZoomToFit*).

- **FocusObject** - funkcionalita zabezpečujúca priblíženie kamery ku konkrétnemu uzlu po vykonaní akcie používateľa. Vyvolanie samotnej akcie zabezpečuje metóda *ZoomInTo(clickedObjectTransform)*, ktorej parameter *clickedObjectTransform* nesie používateľov zvolený objekt. V procese približovania má kamera prednastavené dva stavy - *isZoomed* a *isZooming* v závislosti od toho, či je kamera k uzlu priblížená, alebo je v režime približovania *(UpdateZoomInTo).* Pri tomto režime sú použité štandardné operácie pre interpoláciu a rotáciu, ktorými Unity disponuje.

- **OrbitalMode** - orbitálny mód, ktorého úlohou je centrovanie kamery na žiadaný bod a následné orbitovanie. Nemusí ísť nutne o uzol v grafe, funkcionalita ponúka aj orbitovanie okolo ťažiska viacerých vybraných uzlov používateľom.
    - **Selekcia uzla/uzlov**: Poskytuje *SelectionManager,* ktorého metóda *GetSelectedObjects()* vracia zoznam označených uzlov.
    - **Výpočet ťažiska**: Zo zoznamu označených uzlov vykonáva výpočet pozície ťažiska.
    - **Interakcia**: Prvým krokom je vycentrovanie kamery na žiadaný bod. Držaním zvoleného tlačidla je možné myšou orbitovať po X-ovej alebo Y-ovej osi.

- **ZoomToFit** - funkcionalita, ktorá po jej zavolaní oddiali, alebo priblíži kameru od grafu tak, aby sa zmestil na obrazovku. Vyvolanie samotnej akcie zabezpečuje metóda *ZoomToFit().* Táto metóda ráta cieľovú pozíciu a natočenie kamery, ktoré musí kamera dosiahnuť, aby sa dosiahol výsledný efekt. V procese približovania / oddiaľovania má kamera prednastavené dva stavy - *isZoomed* a *isZooming* v

závislosti od toho, či je dosiahnutý cieľový stav kamery, alebo je v režime pohybu k tomuto stavu (*UpdateZoomToFit*).

Všetky akcie sú aplikované za pomoci Action Patternu a ich vstupy sú spracovávané Input Handlerom.

2.7.6 Label Manager

Tento skript obsahuje konštanty a funkcie, ktoré sú potrebné pre zobrazovanie jednotlivých popiskov pre uzly alebo hrany. Skript konkrétne obsahuje:

- konštanty:
  - **LabelCharSet** - hodnota typu float, ktorá ovplyvňuje všeobecnú veľkosť popiskov.
  - **showAllLabels** - indikuje, či sa majú popisky zobraziť ihneď po načítaní grafu.
- funkcie:
  - **ToggleAllLabels** - prepína všetky popisky medzi stavmi viditeľné a neviditeľné.
  - **ShowLabel** - zobrazí popisok pre určitý herný objekt.
  - **HideLabel** - skryje popisok pre určitý herný objekt.
  - **ShowLabels** - zobrazí popisky pre poskytnuté herné objekty.
  - **HideLabels** - skryje popisky pre poskytnuté herné objekty.

2.7.6 Selection Manager

Tento skript slúži na označovanie uzlov a hrán v grafe. Ponúka tieto funkcie:

- **Select -** po kliknutí na uzol/hranu sa daný objekt označi, a pridá sa do zoznamu označených objektov
- **MultiSelect -** pri držaní CTRL a označovaní uzlov/hrán sa pridávajú označené objekty do *ISet*
- **Unselect -** ak je uzol/hrana už označený a opätovne naň klikneme, objekt sa odznačí
- **UnselectAll -** pokiaľ klikneme mimo grafu alebo máme označný objekt/-y a klikneme na iný objekt bez držania CTRL, ostatné označené objekty sa odznačia

Všetky grafové objekty, ktoré sú označené sa udržujú v *ISet* **selectedObjects**, ktorý si dokážeme getnúť ak by sme k ním potrebovali pristúpiť zvonku. Po vizuálnej stránke v projekte vidíme označené objekty pomocou oranžového obrysu.

3 Infraštruktúra

Repozitáre 3dsoftvis_remake, LuaGraph a LuaInterface majú plne vybudovanú CI infraštruktúru. Táto infraštruktúra sa stará o to, aby sme vyvíjaný projekt vždy vo funkčnom stave a vždy bol pripravený na dodanie zákazníkovi.

Infraštruktúra sa stará o build projektu a všetkých jeho variant, púšťaním automatizovaných jednotkových testov, inštrumentálnych testov, analyzovaním pokryia kódu testami a zverejňovaním ich výsledkov na GitLabe a na stránke tímového projektu vo forme

artefaktov.Taktiež analyzuje zdrojový kód a jeho kvalitu pomocou Lintu a ohlasuje možné sémantické problémy. Tiež sa stará o generovanie dokumentácie zo zdrojových kódov pomocou Doxygenu.

Všetky vygenerované artefakty sú prístupné na webovej stránke tímu http://team07-18.studenti.fiit.stuba.sk/ci-artifacts/.
Staré artefakty sú každodenne automaticky mazané, pričom sa zanechávajú len najnovšie pre jednotlivé aktívne vetvy v repozitári.

Pipeline v repozitári 3dsoftvis_remake sa skladá z viacerých úrovní (angl. stage), pričom každá úroveň má na starosti inú časť procesu deploymentu systému. Pipeline sa pre šetrenie spúšta len pre vetvy s vytvoreným merge requestom.

build
Táto úroveň vykonáva základný build samostatných modulov, ktoré nie sú závislé na prezentačnej vrstve Unity.

build_modules
Stará sa o buildovanie C# modulov.

build_unity_player
Táto úroveň vykonáva build Unity playeru (spústiteľnú binárku), vytvorené Unity moduly a použité knižnice Leap Motion, Zed Camera, HoverUI, ZF Browser.

build_unity_windows
Builduje unity player pre platformu Windows. Táto úloha je povinná.

build_unity_linux
Builduje unity player pre platformu Linux. Táto úloha je volitelná.

build_unity_mac
Builduje unity player pre platformu Mac OS. Táto úloha je volitelná.

QA
Táto úroveň vykonáva testovanie a udržiavanie kvality a zabránenie regresii.

run_tests
Táto úloha testuje C# moduly.

run_lua_tests
Táto úloha testuje Lua moduly.

docs
Táto úroveň vykonáva generovanie dokumentácie

make_doxygen
Úloha generuje dokumentáciu pomocou nástroju Doxygen pre C# moduly a Unity moduly.

4 Príručky

Táto kapitola dokumentuje a opisuje príručky, ktoré sú vhodné, či už pre vývojára alebo pre koncového používateľa systému.

4.1 Build modulu LuaInterface

Kapitola obsahuje opis krokov potrebných pre úspešný build modulu LuaInterface.

4.1.1 Úvodné nastavenia

1. Stiahnuť CMake Gui - posledná stabilná verzia (posledná stabilná verzia na *https://cmake.org/download/*)
2. Naklonovať repozitár LuaInterface
3. Nastavit SSH key pre Github/Gitlab

4.1.2 Pokyny pre build

1. Spustiť CMake
2. *Where is the source code* - priečinok s naklonovaným repozitárom LuaInterface
3. *Where to build the binaries* - vytvoriť nový priečinok LuaInterface_build (môže byť v rovnakom prečinku ako LuaInterface)
4. Configure
5. *Specify the generator for this project* - Visual Studio 15 2017 Win64 (nastaviť podľa používaného VS)
6. Nechať zvolené Use default native compilers
7. Configure spadne -> Zaškrtnúť možnosti:
    - **DOWNLOAD_AND_BUILD_SOL2**
    - **BUILD_LUAINTERFACE_TESTS**
    - **DOWNLOAD_AND_BUILD_LUA**
8. Configure
9. Generate
10. Open Project
11. Vo Visual Studio:
    - pravým na Solution LuaInterface -> Build Solution -> Success

4.2 Build modulu LuaGraph

Kapitola obsahuje opis krokov potrebných pre úspešný build modulu LuaGraph.

4.2.1 Úvodné nastavenia

1. Stiahnuť CMake Gui - posledná stabilná verzia (posledná stabilná verzia na *https://cmake.org/download/*)
2. Naklonovať repozitár LuaGraph
3. Nastavit SSH key pre Github/Gitlab

4.2.2 Pokyny pre build

1. Spustiť CMake
2. *Where is the source code* - priečinok s naklonovaným repozitárom LuaGraph
3. *Where to build the binaries* - vytvoriť nový priečinok LuaGraph_build (môže byť vedľa LuaGraph)
4. Configure

5. *Specify the generator for this project* - Visual Studio 15 2017 Win64 (nastaviť podľa používaného VS)
6. Nechať zvolené Use default native compilers
7. Configure spadne -> Zaškrtnúť možnosti:
    - **DOWNLOAD_AND_BUILD_LUAINTERFACE**
    - **BUILD_GRAPH_TESTS**
8. Configure
9. Generate
10. Open Project
11. Vo Visual Studio:
    - pravým na Solution LuaGraph -> Build Solution -> Failed
12. LuaGraph_build -> dependencies_install -> skopírovať lua.dll, luainterface.dll do LuaGraph -> Debug
13. Vo Visual Studio:
    - pravým na Solution LuaGraph -> Build Solution -> Success

4.3 Build projektu Softviz

Kapitola obsahuje opis krokov potrebných pre úspešný build projektu Softviz.

4.3.1 Úvodné nastavenia

- Pre buildenie projektu sa využíva cmake build systém
- Build systém zahŕňa build týchto častí:
    - Softviz (unity projekt - *Projects/3DSoftviz/UnityProject*)
    - SoftvizModules (moduly pre softviz - *Projects/3DSoftviz/CSProjects*)
    - Lua knižnice potrebné na extrahovanie grafu (luadb, lpeg, luametrics, ...)
    - LuaGraph, LuaInterface

4.3.2 CMake nastavenia

- **BUILD_UNITY**
    - Ak je hodnota *TRUE*, do buildu sa zahrnie aj build unity projektu.
    - Predvolená hodnota: *FALSE*
- **UNITY_TARGET_PLATFORM**
    - Určuje platformu, pre ktorú sa prevedie build Unity projektu. Podporované sú tri hodnoty - Windows, Linux, MacOS. Táto premenná má zmysel iba ak je premenná *BUILD_UNITY* nastavená na *TRUE*.
    - Predvolená hodnota: Windows
    - Výstupný adresár: ./_install/Softviz/
- **BUILD_SOFTVIZ_MODULES**
    - Ak je hodnota TRUE, do buildu sa zahrnie build solution, ktorá obsahuje Softviz moduly
    - Predvolená hodnota: *TRUE*
    - Výstupný adresár: ./_install/SoftvizModules/
- **BUILD_SOFTVIZ_MODULES_TESTS**

    - Ak je hodnota TRUE, do buildu sa zahrnie build testov pre softviz moduly
    - Predvolená hodnota: *FALSE*
    - Výstupný adresár: ./_install/SoftvizModules.Tests/
- **USE_TERRA_BINARIES**
    - Ak je hodnota TRUE, pri builde sa namiesto jazyku Lua použije jazyk Terra . Používajú sa oficiálny release *release-2016-03-25,* ktorý je dostupný na oficialnej stránke.
    - Predvolená hodnota: *TRUE*

4.3.3 Nastavenie cesty pre C knižnice (potrebné len pre Windows)

- je potrebné do premenných prostredia (enviroment variables) pridať premennú s názvom INCLUDE_PATH
- premenná musí obsahovať cesty k zložkam pre includovanie C headers súborov (niekde v priečinku kompilátora)
- pre visual studio 2017 Community na systéme Windows 10 sú cesty nasledovné:
    - *C:\Program Files (x86)\Windows Kits\10\Include\10.0.17763.0\ucrt*
    - *C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\VC\Tools\MSVC\14.16.27023\include*
    - *C:\Program Files (x86)\Windows Kits\10\Include\10.0.17763.0\um*
    - *C:\Program Files (x86)\Windows Kits\10\Include\10.0.17763.0\shared*
    - *C:\Program Files (x86)\Windows Kits\10\Include\10.0.17763.0\winrt*
    - *C:\Program Files (x86)\Windows Kits\10\Include\10.0.17763.0\cppwinrt*
- cesty sa môžu líšiť, preto je potrebné nájsť správne cesty pre konkrétnu verziu systému a použitý kompilátor
- Pre Visual Studio 2017 sa balík MSVC sa nachádza v Desktop development with C++ a balík Windows Kits v Universal Windows Platform development
- po nastavení môže byť potrebné reštartovať systém
- **Poznámka**: Pokiaľ sa premenná upravuje ako text, oddeľovač pre hodnotlivé cesty je znak ;

4.3.4 Pokyny pre build

1. Naklonujeme si repozitár pomocou príkazu `git clone`
2. Updatneme git submoduly pomocou príkazu `git submodule update --init`
3. Skontrolujeme si v *Projects/LuaDependencies* branche jednotlivých submodulov. Mali by byť nasledovné:
    - **luametrics** -> develop
    - **lua** -> lua-5.1
    - **luadb** -> develop
    - **lpeg** -> hotfix/msvc-support
    - **luacomments** -> TAG *0.2.1* alebo commit hash *047b26e035558ce458b9f9d14ef064df3f0e2db7*
4. Otvoríme si cmake v zložke s projektom
5. Vygenerujeme si build súbory bez cmake parametrov
6. Buildneme target/projekt *CopyExternalDataToUnity*

7. V unity projekte vykonáme build

4.4 Adresárová štruktúra

Všeobecná adresarová štruktúra repozitára:

Project/

- **./_install/** - obsahuje buildnuté artefakty
    - **Softviz/** - Plne funkčný softviz projekt
    - **SoftvizModules/** - Moduly pre Softviz projekt
    - **SoftvizModules.Tests/** - Testy pre Softviz moduly
- **./cmake/** - Obsahuje cmake moduly
- **./Documentation/** - Dokumentácia k projektu
- **./Projects/** - Obsahuje jednotlivé projekty Softvizu
    - **LuaDependencies/** - Lua knižnice, ktoré sú pridané do repozitára ako git submoduly
    - **3DSoftviz/**
        - **CSProjects/** - Softviz moduly (C#) spolu so svojimi testami, tie sú automaticky pribalené do unity projektu a obsahujú celú logiku
        - **UnityProject/** - Unity projekt pre Softviz, slúži iba ako vizualizačná vrstva. Všetka ostatná logika patrí do Softviz modulov
- **./resources/** - Dodatočné súbory k projektom (spoločné scripty, obrázky, ...)

Adresarová štruktúra pre Unity projekt:

- **./Assets/**
    - **Animations** - Skripty pre animácie
    - **Editor** - Skripty pre Unity Editor
    - **External** - Externé assety z Assets Store
    - **Plugins** - Externé knižnice
        - **ExternalModule** - Externé C# moduly, napr. Softviz moduly
        - **Lua** - Natívne (C, C++) knižnice pre prácu s Lua
    - **Resources** - Zdroje pre Unity - Prefabs, Textures, Materials, Fonts, ...
    - **Scenes** - Unity scény
    - **Scripts** - Skripty pre Unity objekty,
    - **StreamingAssets** - Assety, ktoré nie sú pri builde pribalené do projektu, ale iba prekopírované k projektu
        - **LuaScripts** - Všetky lua scripty, ktoré by malo byt možné dynamicky meniť aj po skompilovaní unity projektu
    - **UnityTests** - Testy pre Unity projekt
- **./ProjectSettings/**

!! POZOR !!

- Súbory v adresároch ./Assets/StreamingAssets/LuaScript a ./Assets/Plugins/ sú automaticky prepisované pri cmake builde pretože ide o externé závislosti,. Akekoľvek manuálne úpravy týchto súborov môžu byť zmazané. Tieto zložky sú zároveň aj v .gitignore, preto, akékoľvek zmeny nebudú commitnuté do repozitára. Pri ich úprave je potrebné nájsť zodpovedajúci projekt a vykonať zmenu tam!

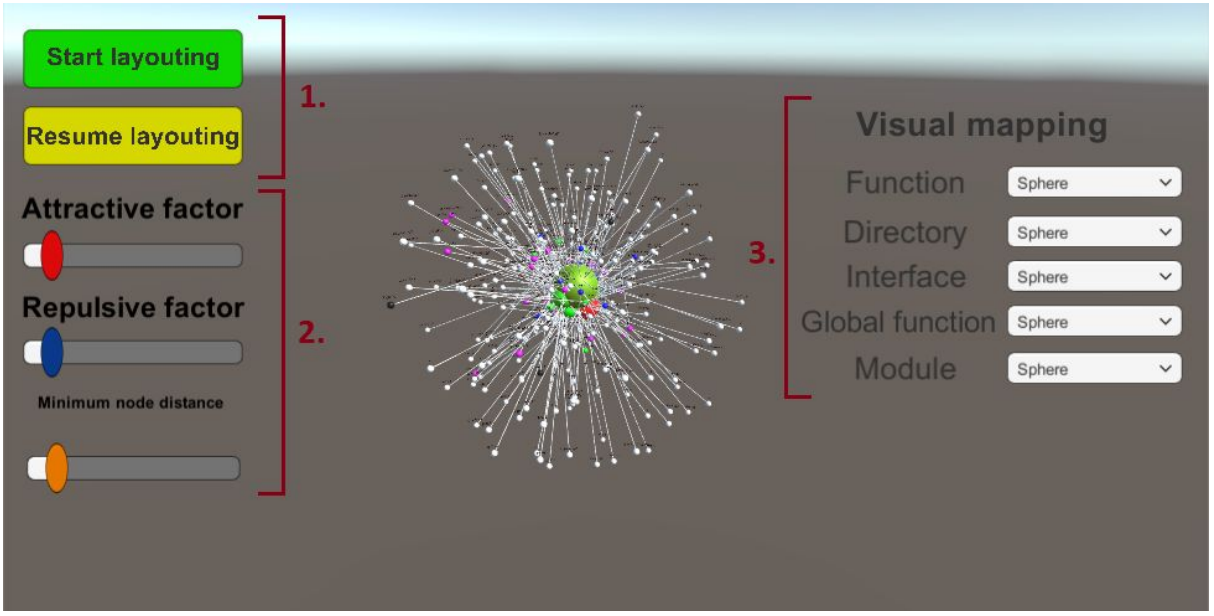
4.5 Používateľská príručka

Kapitola opisuje príručky vhodné pre koncového používateľa projektu Softviz.

4.5.1 Desktop

Na interakciu so scénou je možné použiť klávesnicu a myš. Pohyb po scéne je riešený klávesami W (dopredu), A (doľava), S (dozadu) a D (doprava). Otáčanie kamery je umožnené pohybom myši v smere želaného otočenia.

Táto scéna disponuje grafickým používateľským rozhraním. Jeho zobrazenie a skrytie sa vykonáva stlačením klávesy M. Toto používateľské rozhranie poskytuje viacero funkcií.



Obrázok č.4 - Legenda používateľského rozhrania

Vysvetlenie legendy:

1. **Layout control** - tlačidlá pre spustenie a zastavenie layoutovania grafu.
2. **Layout settings** - posuvníky nastavení pre Fruchterman-Reinfoldový layoutovací algoritmus (červený - nastavenie veľkosti príťažlivej sily, modrý - nastavenie odpudivej sily, oranžový - nastavenie minimálnej dĺžky hrán grafu).
3. **Visual mapping** - nastavenie vizuálneho mapovania jednotlivých typov uzlov. Pre každý je možné zvoliť rôzne geometrické útvary.

Pri zobrazení menu otáčanie kamery zablokované a myšou je možné:

1. **LMB** - označit objekt.

2. **CTRL + LMB** - označit viacero objektov.
3. **SRCOLL** - zoomovať k / od onačeného(ných) objekt(ov).
4. **MMB + pohyb myši** - krúženie okolo označeného(ných) objekt(ov).

Ďalšie klávesy akcií:

1. **P** - zapnutie a vypnutie gravitácie.
2. **G** - zapnutie a vypnutie hýbania grafu za uzol.
3. **F** - zapnutie a vypnutie fyziky.
4. **V** - zmena layoutovania grafu.
5. **B** - layoutovanie grafu na plochu.
6. **K** - prepínanie medzi vizualizovanim uzlov kockou a guľou.
7. **J** - prepínanie medzi vizualizovanim hrán kužeľom a hranolom.
8. **I** - skrytie hrán grafu.
9. **+** - zväčšenie grafu.
10. **-** - zmenšenie grafu.
11. **O** - označenie všektých popiskov.
12. **PgUp** - priblíženie sa k zdrojovému uzlu.
13. **PgDn** - priblíženie sa k cieľovému uzlu.
14. **Home** - priblíženie sa k označenému uzlu.
15. **Z** - zoom-to-fit (priblíženie / oddialenie kamery tak, aby bol viditeľný celý graf).

4.5.2 VR

V tejto scéne je možné sa pohybovať prirodzeným spôsobom, tak ako v reálnom prostredí. Headset sa stará o mapovanie pozície v reálnom prostredí do virtuálneho prostredia. Interakcia s prostredím je riešená sledovaním rúk používateľa a ich mapovaním do virtuálneho prostredia. Ovládacie menu je možné otvoriť rovnako ako je opísane nižšie pri AR scéne, teda natočením ľavej dlane smerom na kameru.

Interakcia

V scéne je zapnutá fyzická interakcia. To znamená, že rukou môžeme udierať, uchopiť a hádzať uzlami. Uchopenie je nutné vykonávať troma prstami - palec, ukazovák a prostredník.

Selekcia

Selekciu uzlov a hrán je možné vykonať gestom pištole. Teda natiahnutím ukazováku a prostredníku a stiahnutím prstenníka a malíčka. Palec pri tomto geste slúži na selekciu a deselekciu objektu, na ktorý týmto gestom ukazujeme. Zdvihnutím palca sa objekt označí, zložením sa odznačí.

Obmedzujúce objekty

Pre zlepšenie čitateľnosti je možné použiť obmedzujúce objekty, ktoré vedia držať a hýbať uzly vo svojom priestore a na ktorých povrch sa viažu uzly. Je možné ich vytvoriť pomocou položky “Spawn restriction” v HoverUI menu. Je možné pridať guľu, kocku, alebo plochu, ktorá sa pred nami vytvorí. Tento obmedzujúci objekt je potom možné presúvať a meniť jeho veľkosť.

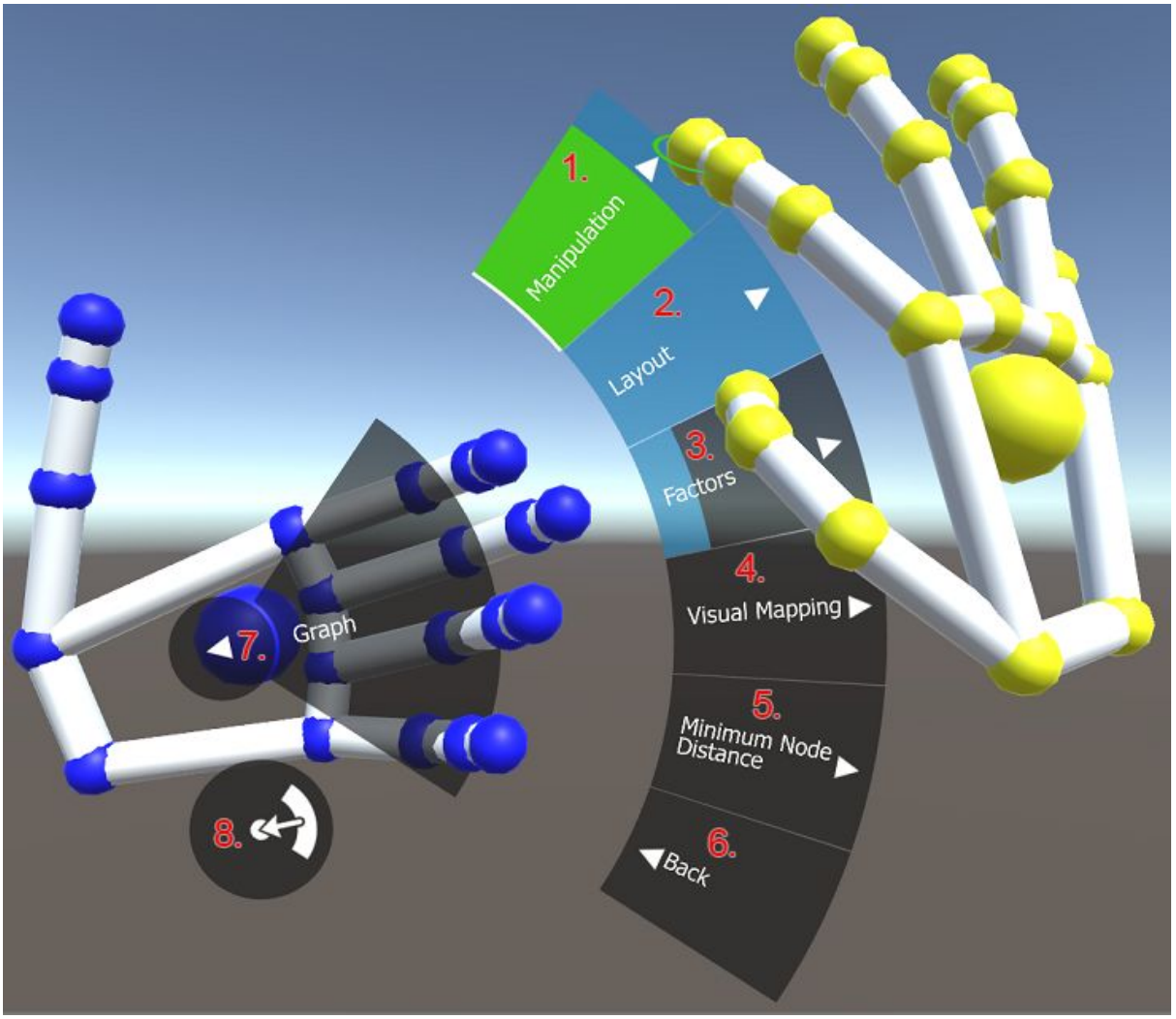
Pre pridanie uzlov do tohto objektu stačí jednoducho uzol chytiť a presunúť ho dovnútra. Pre pridanie viacerých uzlov naraz je potrebné si ich označiť. Pre označenie všetkých uzlov je možné tiež použiť funkciu poskytnutú v HoverUI menu.

4.5.3 AR

Pohyb v scéne a mapovanie rúk sú implementované rovnako ako vo VR scéne.

Ovládanie - HoverUI

HoverUI menu sa zobrazí len pri natočení ľavej dlane smerom na kameru. Stlačenie jednotlivých tlačidiel sa realizuje prostredníctvom priblíženia pravého ukazováka na tlačidlo. Ak tlačidlo zasvieti na svetlo-modro, znamená to, že kurzor na pravom ukazováku sa dostal k jeho blízkosti. Zelená farba znamená, že sa spustila udalosť “kliknutia”. Pomocou tlačidiel je možné manipulovať s celým grafom, kontrolovať layoutovanie, zmeniť jednotlivé faktory grafu alebo zmeniť vizuálne mapovanie.



Obrázok č.5 - Legenda HoverUI

Vysvetlenie legendy:

4. **Manipulation** - obsahuje tlačidlá, ktoré sa týkajú manipulácie s grafom (napr. umiestnenie grafu na určité miesto alebo škálovanie grafu).
5. **Layout** - obsahuje tlačidlá týkajúce sa layoutovania grafu (napr. štart/stop layoutu, pauza layoutu, zmena layouterov, zmena builderov layoutu, zmena layout funkcií a zmena premenných a funkcií).
6. **Factors** - nastavenie faktorov (attractive a repulsive faktory).
7. **Visual Mapping** - zmena vizuálneho mapovania pre jednotlivé Lua objekty - funkcie, priečinky, rozhrania, globálne funkcie a moduly. Pre každý typ objektu je na výber zo šiestich tvarov.
8. **Minimum Node Distance -** nastavenie minimálnej vzdialenosti medzi uzlami.
9. **Back** - tlačidlo pre návrat do rodičovského menu.
10. ◄ - tlačidlo pre návrat do rodičovského menu. Toto tlačidlo sa taktiež dá stlačiť zovretím ľavej dlane do päste.
11. ·□**)** - tlačídlo pre skrytie/zobrazenie menu.

4.6 Známe problémy

Problém: lpeg.c:793: checkrule: Assertion op[start - 1].i.code == IChoice && dest(op, start - 1) == target(op, i + 1)' failed.
Riešenie: Je použitá nesprávna verzia *luacomments* submodulu. Je potrebné nastaviť *luacomments* submodule na TAG 0.2.1 alebo na commit hash 047b26e035558ce458b9f9d14ef064df3f0e2db7 (cez git checkout)