Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky

Využitie systému OptiTrack pre snímanie polohy používateľa v hernom jadre Unreal

Bakalárska práca

2018 **Dominik Tóth**

Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky

Využitie systému OptiTrack pre snímanie polohy používateľa v hernom jadre Unreal

Bakalárska práca

Študijný program:	Informatika
Študijný odbor:	9.2.1. Informatika
Školiace pracovisko:	Katedra počítačov a informatiky (KPI)
Školiteľ:	Ing. Štefan Korečko, PhD.
Konzultant:	

Košice 2018 **Dominik Tóth**

Názov práce:	Využitie systému OptiTrack pre snímanie polohy používateľa v hernom jadre Unreal
Pracovisko:	Katedra počítačov a informatiky, Technická univerzita v Košiciach
Autor:	Dominik Tóth
Školiteľ:	Ing. Štefan Korečko, PhD.
Konzultant:	
Dátum:	25. 5. 2018
Kľúčové slová:	Unreal Engine 4, CAVE, OptiTrack, Projekcia

Abstrakt: Bakalárska práca sa zaoberá využitím systému OptiTrack v Unreal Engine 4 a vytvorením projekcie mimo os pre CAVE systém. Prvá časť analýzy sa zaoberá dostupnými CAVE systémami a podkladmi pre projekciu mimo os. Druhá časť opisuje základy Unreal Engine-u a možnosti ako posielať dáta o pozícií z OptiTrack-u do Unreal Engine-u. Na základe analýzy sa navrhuje a implementuje riešenie, ktoré dáta o pozícií využíva pre projekciu mimo os. Projekcia mimo os je vytvorená ako zásuvný modul pre Unreal Engine a je použiteľná aj bez OptiTrack-u. Na záver sa vyhodnocuje a testuje riešenie s použitím OptiTrack-u.

Thesis title:	Utilization of OptiTrack for User Tracking In Unreal Game Engine
Department:	Department of Computers and Informatics, Technical University of Košice
Author:	Dominik Tóth
Supervisor:	Ing. Štefan Korečko, PhD.
Tutor:	
Date:	25. 5. 2018
Keywords:	Unreal Engine 4, CAVE, OptiTrack, Projection

Abstract: This bachelor thesis is about utilization of OptiTrack in Unreal Engine 4 and creation of off-axis projection for CAVE system. First part of analysis deals with available CAVE systems and fundamentals of off-axis projection. Second part describes basics of Unreal Engine and options for sending position data from OptiTrack to Unreal Engine. Based on analysis, a solution using position data for off-axis projection is designed and implemented. Off-axis projection is created as plugin for Unreal Engine and can be used without OptiTrack. In the end, the solution using OptiTrack is tested and evaluated.

TECHNICKÁ UNIVERZITA V KOŠICIACH

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Katedra počítačov a informatiky

ZADANIE BAKALÁRSKEJ PRÁCE

Študijný odbor: **Informatika**

Študijný program: **Informatika**

Názov práce:

Využitie systému OptiTrack pre snímanie polohy používateľa v hernom jadre Unreal

Utilization of OptiTrack for User Tracking In Unreal Game Engine

Študent: **Dominik Tóth**

Školiteľ: **Ing. Štefan Korečko, PhD.**

Školiace pracovisko: **Katedra počítačov a informatiky**

Konzultant práce:

Pracovisko konzultanta:

Pokyny na vypracovanie bakalárskej práce:

1. Oboznámiť sa so softvérovým a hardvérovým vybavením virtuálno-realitnej jaskyne LIRKIS CAVE.
2. Analyzovať možnosti využitia systému OptiTrack pre snímanie polohy používateľa v hernom jadre Unreal so zreteľom na využitie v LIRKIS CAVE.
3. Na základe analýzy navrhnúť a implementovať prototypové riešenie podpory OptiTrack v Unreal pre LIRKIS CAVE. V riešení je potrebné zohľadniť nutnosť prispôsobenia výstupov jednotlivých obrazoviek polohe používateľa.
4. Funkčnosť a použiteľnosť implementovaného riešenia overiť v LIRKIS CAVE.
5. Pri návrhu a implementácii spolupracovať s riešiteľmi prác súvisiacimi s prispôsobením herného jadra Unreal pre LIRKIS CAVE.
6. Vypracovať dokumentáciu podľa pokynov vedúceho práce.

Jazyk, v ktorom sa práca vypracuje: slovenský

Termín pre odovzdanie práce: 25.05.2018

Dátum zadania bakalárskej práce: 31.10.2017

doc. Ing. Jaroslav Porubän, PhD.
vedúci garantujúceho pracoviska

TECHNICKÁ UNIVERZITA V KOŠICIACH FAKULTA ELEKTROTECHNIKY A INFORMATIKY -13-

prof. Ing. Liberios Vokorokos, PhD.
dekan fakulty

Čestné vyhlásenie

Vyhlasujem, že som záverečnú prácu vypracoval(a) samostatne s použitím uvedenej odbornej literatúry.

Košice, 25.5.2018 .....

Vlastnoručný podpis

Poďakovanie

Chcel by som sa poďakovať vedúcemu mojej bakalárskej práce, Ing. Štefanovi Korečkovi, PhD. za odbornú pomoc pri písaní záverečnej práce.

Obsah

Zoznam obrázkov

Úvod

Virtuálno-realitná jaskyňa, ktorá sa nachádza v laboratóriu LIRKIS prechádza zmenou softvéru, ktorý používa pre vykresľovanie scén. Systém SuperEngine, ktorý používala jaskyňa doteraz je graficky zastaralý, a preto potrebuje zmenu. Ako vhodné riešenie pre vykresľovanie scén pre systém bolo v rámci diplomovej práce minulý rok zvolené herné jadro Unreal Engine 4. Avšak stále ostalo niekoľko problémov, ktoré pre plnú funkčnosť virtuálno-realitnej jaskyne musia byť vyriešené [1].

Jaskyňa doteraz používala systém na zachytenie pohybu OptiTrack a jeho softvér Motive. V pláne je používať tento systém aj naďalej, avšak kvôli zmene softvéru na vykresľovanie scén je teraz nutné vyriešiť problém prenosu údajov, získaných sledovaním pohybu. Tieto údaje je potrebné preniesť zo softvéru Motive do herného jadra Unreal Engine 4 v reálnom čase, aby sa s nimi dalo ďalej pracovať. Tento prenos zo systému Motive do jadra Unreal nie je priamo podporovaný, preto sa v tejto práci snažíme nájsť spôsob, akým by sme boli schopní takýto prenos vykonať.

Upravenie projekcie v Unreal Engine tak, aby sa obraz na monitore zobrazoval vzhľadom na pozíciu osoby nachádzajúcej sa v jaskyni je ďalší problém, ktorý sa rieši v tejto práci. Na to, aby bol obraz v jaskyni reálny a vierohodný, je potrebné, aby sa menilo zobrazenie, keď sa zmení pozícia osoby, ktorá sa nachádza v jaskyni, v reálnom svete. Na to, aby sme zistili ako treba zmeniť zobrazenie, aby obraz vyzeral ako realita, nám poslúžia informácie získané systémom na sledovanie pohybu osoby. Zobrazenie vzhľadom na polohu je v niektorých zásuvných moduloch už vyriešené, avšak vzhľadom na špecifické potreby našej jaskyne nie je toto riešenie použiteľné, pretože je limitované počtom obrazoviek. Naša jaskyňa má veľký počet monitorov, z ktorých každý zodpovedá jednej hernej kamere. Scénu je nutné zobrazovať na každom monitore presne tak, aby zodpovedala polohe osoby. Preto sa v tejto práci pokúsime vyriešiť tento problém.

Vyriešenie týchto problémov by malo viesť k zvýšeniu reálnosti a zlepšeniu zážitku vo virtuálno-realitnej jaskyni.

1 Formulácia úlohy

Cieľom tejto bakalárskej práce bolo využitie systému OptiTrack v spolupráci s herným jadrom Unreal vo virtuálno-realitnej jaskyni v laboratóriu LIRKIS.

Prvý bod zadania pozostával z oboznámenia sa s vybavením virtuálno-realitnej jaskyne LIRKIS CAVE. Táto jaskyňa pozostáva zo siedmich vzájomne prepojených počítačov, ktoré poháňajú 20 obrazoviek umiestnených v polkruhu a sledovacieho systému OptiTrack.

V druhom bode zadania sme mali analyzovať možnosti využitia systému Optitrack pre snímanie polohy používateľa v hernom jadre Unreal.

V treťom a štvrtom bode zadania sme mali navrhnúť a implementovať riešenie podpory OptiTrack v Unreal pre LIRKIS CAVE, taktiež sme mali zaistiť dynamické nastavenie uhlov medzi kamerami podľa pozície hráča. Na základe analýzy sa zistilo, že najideálnejšie bude využiť OptiTrack - NatNet Streaming Client, ktorý je zásuvný modul pre Unreal Engine 4. Riešenie sme potom mali otestovať v LIRKIS CAVE.

2 Projekcia mimo os v jaskyniach virtuálnej reality

Virtuálnu realitu zadefinoval Howard Rheingold ako zážitok, v ktorom je človek obklopený trojrozmerným, počítačom generovaným prostredím a je schopný sa hýbať v tomto virtuálnom svete, vidieť ho z viacerých uhlov, meniť ho [2].

Ako inú definíciu si môžeme ukázať definíciu od Earnshawa z knihy Virtual reality systems. Virtuálnu realitu charakterizuje ako ilúziu, v ktorej sa zúčastňujeme v umelom prostredí. Virtuálna realita je odkázaná na trojdimenzionálne, stereoskopické zobrazenie, sledovanie hlavy alebo rúk a binaurálnom zvuku. Virtuálna realita je pohlcujúci, viac-zmyslový zážitok [3].

Tieto definície sa trochu od seba líšia, avšak v oboch definíciách sa spomína trojrozmerné prostredie a schopnosť zúčastniť sa virtuálneho sveta aktívne pomocou rôznych prostriedkov.

Virtuálna realita sa dá implementovať rôznymi spôsobmi. V dnešnej dobe sú najpoužívanejšie komerčne dostupné displeje, ktoré sú upevnené na hlave, ako napríklad Oculus Rift alebo HTC Vive. Ďalej existujú rôzne trenažéry, ktoré slúžia ako učebná pomôcka, napríklad automobilový alebo letecký trenažér. Cave automatic virtual environment (CAVE) je typ virtuálno-realitného systému, ktorému sa venujeme v tejto práci. V takomto systéme je človek obklopený obrazom z viacerých strán a môže sa v ňom voľne pohybovať.

2.1 LIRKIS

Laboratórium LIRKIS na Technickej Univerzite v Košiciach je vybavené virtuálno-realitnou jaskyňou. Pozostáva zo siedmych vzájomne prepojených počítačov, ktoré umožňujú vykresľovať stereoskopický obraz spolu na 20 obrazoviek, ktoré sú umiest-

nené do polkruhu, 3 z týchto obrazoviek sú na hornej a 3 na dolnej stene jaskyne. Také usporiadanie umožňuje pohyb po jaskyni a človek vidí obraz aj periférne.

Jaskyňa obsahuje aj systém OptiTrack, ktorý sa skladá z 8 kamier a reflexných sledovacích značiek. Tie sú umiestnené na okuliaroch a človeka, ktorý ich má nasadené na hlave systém OptiTrack sleduje. Softvérové vybavenie jaskyne sa skladá aj z aplikácie Motive (verzia 1.9.0), ktorá prijme údaje z kamier a vypočítava polohu hráča vo svete.

Druhá časť softvérového vybavenia jaskyne sa skladá zo systému SuperEngine, založeného na OpenSG, ktorý slúži ako hlavné jadro pre vykresľovanie scén pre jaskyňu. SuperEngine je už pár rokov zastaralý, a preto je snaha o jeho nahradenie Unreal Engine-om 4, čomu sa čiastočne venuje aj táto práca.

2.2 Implementácie CAVE systémov

Vo svete existuje niekoľko CAVE systémov. Pozrieme sa na niektoré z nich, a na to, aké sledovacie systémy používajú a ako riešia projekciu vzhľadom na pozíciu používateľa.

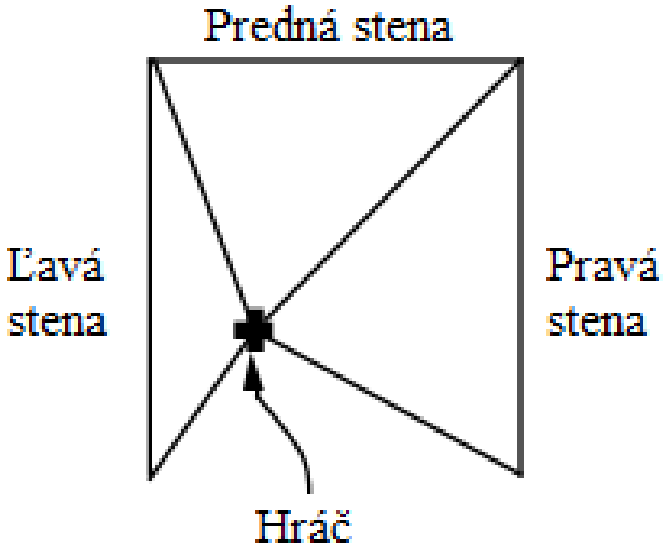
2.2.1 CAVE SIGGRAPH 93

Ako jeden z prvých CAVE systémov vznikol systém na Illinoiskej Univerzite v Chicagu. Tento systém bol veľkosti 10x10x10m a pomocou projektorov sa obraz premietal na tri steny a podlahu. Používateľ mal na sebe stereoskopické okuliare a systém sledoval polohu hlavy a ruky hráča pomocou Polhemus alebo Ascension elektromagnetických senzorov. Kvôli tomu musela byť konštrukcia systému zostavená z nemagnetickej ocele. Avšak napriek tomu sa mohol vyskytovať šum, pretože vodivý kov sa nachádza napríklad v zrkadlách.

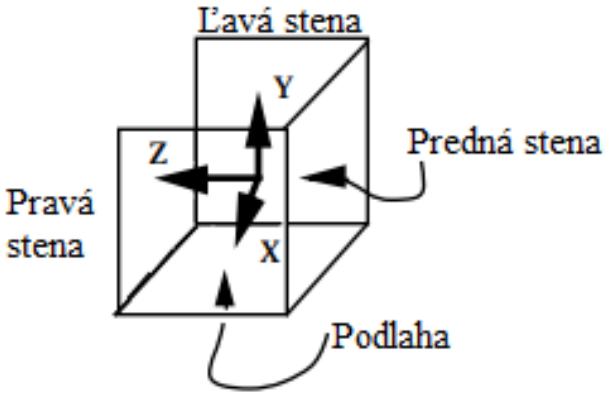
Stereoskopické zobrazenie vo virtuálnej jaskyni je prispôsobené na pohľad zo stredu jaskyne. Ak je poloha hlavy hráča v jaskyni mimo jej stredu, vzniká deformácia obrazu.

V CAVE systéme využili na vytvorenie správneho stereoskopického zobrazenia perspektívnu projekciu mimo os (off-axis perspective projection). Lokácia projekčnej roviny korešponduje s lokáciou stien (obr. 2.1), takže ako sa hráč pohybuje po prostredí jaskyne, stereo projekcia mimo os je vypočítavaná vzhľadom na jeho polohu. Všetky steny zdieľajú jeden súradnicový systém, ktorý je zobrazený na

obr. 2.2. Začiatok súradnicového systému je v strede jaskyne a údaje získané sledovaním polohy sú transformované aby zodpovedali tomuto systému.



Obr. 2.1: Projekcia mimo os [4].

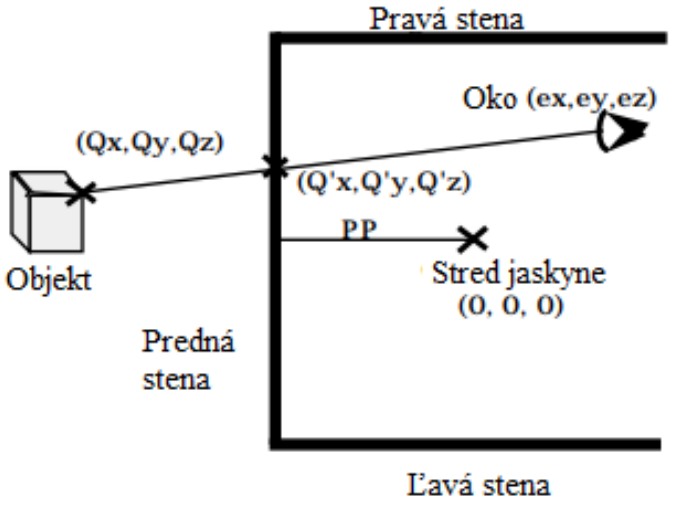


Obr. 2.2: Súradnicový systém jaskyne [4].

Na obr. 2.3 je zobrazený pohľad zhora na CAVE. Q' je projekcia Q a PP je vzdialenosť prednej steny od stredu jaskyne.

Projekcia Q', bodu Q (Qx, Qy, Qz) sa tým pádom dá vypočítať nasledovne:

- $Q'x = Qx + \frac{(PP - Qz)(ex - Qx)}{ez - Qz}$
- $Q'y = Qy + \frac{(PP - Qz)(ey - Qy)}{ez - Qz}$



Obr. 2.3: Pohľad zhora na jaskyňu [4].

Projekčná matica je potom:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -\frac{ex}{ez-PP} & -\frac{ey}{ez-PP} & 1 & -\frac{1}{ez-PP} \\ \frac{exPP}{ez-PP} & \frac{eyPP}{ez-PP} & 0 & \frac{ex}{ez-PP} \end{bmatrix}$$

Vzhľadom na to, že steny tejto jaskyne majú medzi sebou uhol $90°$, pozíciu hráča vieme vyjadriť vzhľadom k jednotlivým stenám:

- Ľavá stena: (ez, ey, ex)
- Pravá stena: (-ez, ey, ex)
- Podlaha: (ex, ez, -ey)

Na stereoskopické zobrazenie sa takto využili dve projekcie mimo os, na každé oko jedna [4].

2.2.2 StarCAVE

StarCAVE systém bol vytvorený na Kalifornskej univerzite v San Diegu. Skladá sa z piatich stien, na ktorých je zobrazovaná scéna. Na sledovanie pohybu hlavy a ruky sa použil system ART Tracking, ktorý pozostáva zo štyroch infračervených

kamier umiestnených na vrchu stien. ART systém má presnosť na jeden milimeter na frekvencii 60Hz.

Pre čo najlepšiu perspektívu a zobrazenie na osi(on-axis) využili vhodne zvolený 15° uhol medzi LCD monitormi, ktoré sú nad sebou. Toto riešenie však eliminuje zobrazenie mimo os iba čiastočne. Príliš veľka zmena pozície zapríčiní nesprávne zobrazenie, takže stále je potrebné meniť ho podľa polohy používateľa [5].

2.2.3 VRCluster

VRCluster je softvér, ktorý bol navrhnutý pre CAVE systémy a obsahuje konfiguráciu pre ľubovolný počet obrazoviek a ich usporiadanie pomocou vlastnej aplikácie s grafickým rozhraním a konzolu pre vzdialený prístup. O používateľský vstup sa stará VRPN. Zobrazenie je plne stereoskopické a podporuje aj zobrazenie mimo os.

Softvér bol vyvíjaný pod voľnou licenciou, avšak zdrojové súbory už na oficiálnej stránke nie sú dostupné [6].

2.3 RobCoG

Cieľom projektu RobCoG je vybaviť robotov logickým zmýšľaním a základnými vedomosťami o fyzike pomocou hier s účelom. Počas hier sa údaje o hraní zapisujú do logov a neskôr sa využívajú na výučbu robotov.

Hry pre tento projekt sú vytvorené v Unreal Engine 4 a sú zamerané na virtuálnu realitu. Využíva sa OptiTrack pre snímanie celého tela používateľa. Pre integráciu OptiTrack-u do Unreal Engine sa využil oficiálný plugin pre Optitrack od NaturalPoint, Inc. - OptiTrack – NatNet Streaming Client. Boli vytvorené triedy pre ovládanie rôznych častí, ktoré OptiTrack sníma, ako napríklad `OptitrackHand` pre ruku a `OptitrackSkeleton` pre kostru používateľa.

Viac informácii o projekte RobCoG je dostupných na oficiálnej stránke projektu [7].

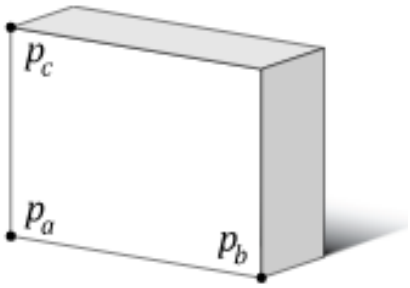
2.4 Projekcia mimo os

Perspektívna projekcia sa využíva na tvorbu obrazov, ktoré majú vyzerať prirodzene. Závisí od pozície očí vzhľadom na projekčnú rovinu [8]. Pri stereoskopii sa pre čo najlepšie výsledky využíva on-axis (na osi) projekcia, pri ktorej je pozícia očí v strede vzhľadom na projekčnú rovinu. Pre každé oko sa tak vytvorí samostatné zobrazenie. Stred perspektívnej projekcie je bežne nastavený ako stred obrazovky. Problém nastáva, keď oko pozorovateľa je mimo stredu projekčnej roviny a toto zobrazenie sa nazýva projekcia mimo os (off-axis). V CAVE systémoch je toto bežná vec, pretože človek má voľnosť pohybovať sa v priestore.

2.4.1 Všeobecná perspektívna projekcia

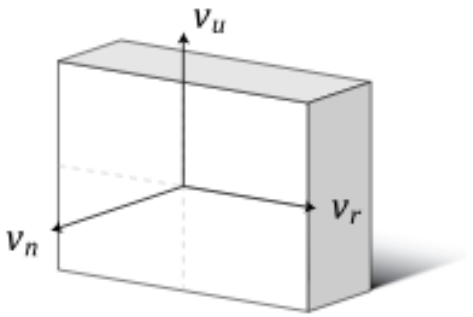
Robert Kooima vo svojom článku o všeobecnej perspektívnej projekcii [9] vysvetľuje teoretické podklady, implementáciu v jazyku C za pomoci OpenGL a popisuje možné využitia off-axis projekcie.

Ako prvý krok si definujeme 3 vektory p_a, p_b a p_c, ktoré označujú dolný ľavý, dolný pravý a horný ľavý roh obrazovky (obrázok 2.4). Z týchto vektorov si potom vypočítame ortonormálnu bázu, vektory v_r, v_u a v_n, označujúce vektor smerujúci vpravo, hore a vektor kolmý na obrazovku (obrázok 2.5). Nasledujúca časť

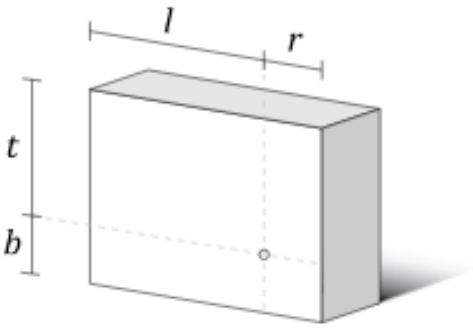


Obr. 2.4: Označenie rohov obrazovky [9].

spočíva v zistení parametrov do funkcie `glFrustum`, do výpočtu je potrebné zistiť parametre l,r,b,t,n a f. Parametre n a f označujú blízku a vzdialenú orezávaciu rovinu. Význam zvyšných parametrov je ukázaný na obrázku 2.6. Z týchto para-



Obr. 2.5: Ortonormálna báza obrazovky [9].



Obr. 2.6: Graficke znázornenie parametrov [9].

metrov funkcia `glFrustum` vypočítava nasledujúcu projekčnú maticu.

$$P = \begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Projekčnú rovinu je potrebné otočiť tak, aby bola zarovnaná s rovinou XY, ktorú definuje súradnicový system. Na zarovnanie sa využije matica M^T.

$$M^T = \begin{bmatrix} v_{rx} & v_{ux} & v_{nx} & 0 \\ v_{ry} & v_{uy} & v_{ny} & 0 \\ v_{rz} & v_{uz} & v_{nz} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Nakoniec potrebujeme, aby vrchol zrezaného ihlana (frustum), ktorý definuje projekciu, bol na rovnakej pozícii ako oko. Predchádzajúci krok, v ktorom sa zarovnávala projekčná rovina s rovinou XY, otočil priestor používateľa vzad. Preto potrebujeme posunúť pozíciu očí naspäť na vrchol. V OpenGL k tomu slúži funkcia

`glTranslatef`, ktorá aplikuje nasledujúcu maticu.

$$T = \begin{bmatrix} 1 & 0 & 0 & -p_{ex} \\ 0 & 1 & 0 & -p_{ey} \\ 0 & 0 & 1 & -p_{ez} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Výslednú maticu potom dostaneme zložením troch matíc, ktoré sme si popísali.

$$P' = PM^T T$$

Toto riešenie je veľmi zaujímavé, pretože poskytuje správne zobrazenie pre ľubovoľne orientované obrazovky pomocou definície troch rohov obrazovky a pozície oka. Riešenie bolo taktiež implementované v Unity, ako skript pre kameru [10] a členovia Unreal Engine fóra ho využili pri ich implementáciách zobrazenia mimo os [11]. VRCluster, ktorý sa spomínal vyššie, taktiež používa toto riešenie na vypočítanie správnej matice pre zobrazenie scén.

2.4.2 Projekcia mimo os v Unreal Engine 4

Na vytvorenie projekcie mimo os je potrebné zmeniť projekčnú maticu tak, aby zahŕňala polohu používateľa.

Matúš Gabriška vo svojej diplomovej práci [12] experimentoval s touto projekciou, avšak nepodarilo sa mu opraviť chyby, ako miznutie objektov zo scény, takže nakoniec bolo rozhodnuté, že sa využije symetrické zobrazenie.

Vďaka jeho experimentom však máme predstavu, ako by táto implementácia mohla vyzerať. Využila sa metóda `CalcSceneView` z triedy `ULocalPlayer`. Matice, ktoré bolo potrebné zmeniť, boli v čase písania privátne, preto bolo nutné modifikovať jadro Engine-u. To sa však vo verzii 4.19 dá obísť pomocou využitia getter-ov (metódy pre sprístupnenie privátnych premenných), ktoré nám vrátia referencie na premenné obsahujúce matice.

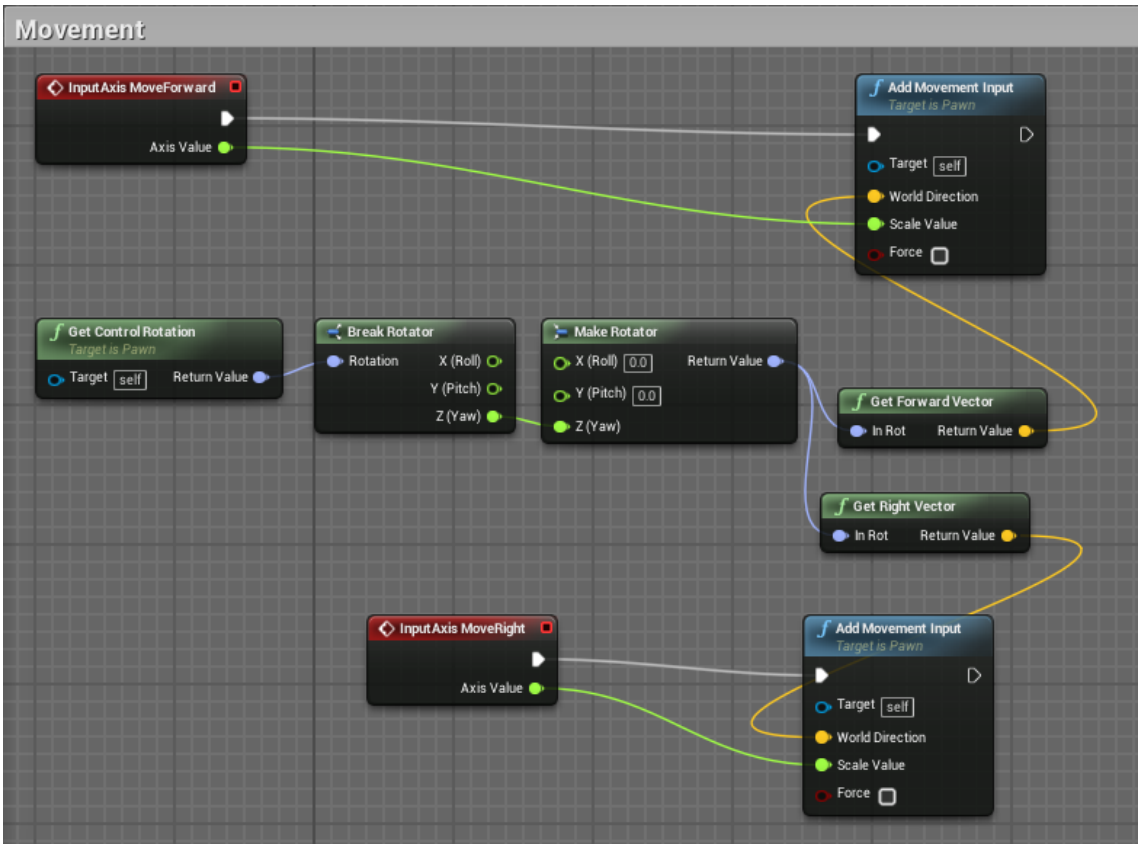
3 Možnosti využitia systému Optitrack v Unreal Engine 4

Unreal Engine 4 je dôležitý nástroj, ktorý budeme v tejto práci využívať, preto si o ňom povieme niečo bližšie.

Unreal Engine 4 je súprava nástrojov od spoločnosti Epic Games pre tvorbu vysokokvalitných aplikácií, filmových zážitkov a hier. Je napísaný v C++ a všetky zdrojové kódy sú od roku 2015 voľne dostupné zdarma. Ponúka nástroje na úpravu mriežok(mesh) a animácií, ktoré zahŕňajú, stavové automaty, prelínanie animácií, inverznú kinematiku a animáciou poháňanú fyziku. V oblasti renderovania ponúka dynamické osvetlenie, tvorbu materiálov, ako aj vytváranie častíc a tieňov.

Natívny jazyk pre programovanie logiky je C++, avšak Unreal Engine 4 ponúka aj ich vlastné blueprinty, pomocou ktorých sa dá naprogramovať chod hry aj bez znalosti jazyka C++. Blueprinty sú grafické skripty, ktoré slúžia na rýchle prototypovanie a vytvorenie správania a interakcií objektov.

Na obr. 3.1 vidíme ukážku blueprintu, ktorého úlohou je prijať vstup z klávesnice a na základe neho upraviť pozíciu postavy. Blueprinty sú zložené z uzlov a čiar, ktoré ukazujú tok dát a času. Tok času je znázornený bielymi čiarami a tok dát je znázornený farebnými čiarami. Udalosti sú označené červenou farbou. Hlavné funkcie sú označené modrou farbou a pomocné funkcie sú označené zelenou farbou. Tento skript sa spustí, keď dôjde buď k udalosti `MoveForward` alebo `MoveRight`, ktoré sú v ľavom hornom a dolnom rohu a vracajú hodnotu o polohe na osi. Túto polohu podáva ako vstupný parameter funkcii `AddMovementInput`, ktorá má nasledovať v toku času. Táto funkcia sa však ešte nemôže vykonať, pretože nemá ďalší vstupný parameter. Preto sa najprv volajú funkcie `GetControlverbRotation!`, `BreakRotation`, `MakeRotation` a `GetForwardVector`, ktoré nám určia smerový vektor vo svete a vrátia ho ako vstupný parameter pre funkciu `Add-`



Obr. 3.1: Blueprint pre ovládanie postavy.

`MovementInput`. Funkcia `AddMovementInput` sa následne vykoná a pridá postave pohybový vstup, ktorý sa vykonal.

3.1 Prehľad tried v Unreal Engine 4

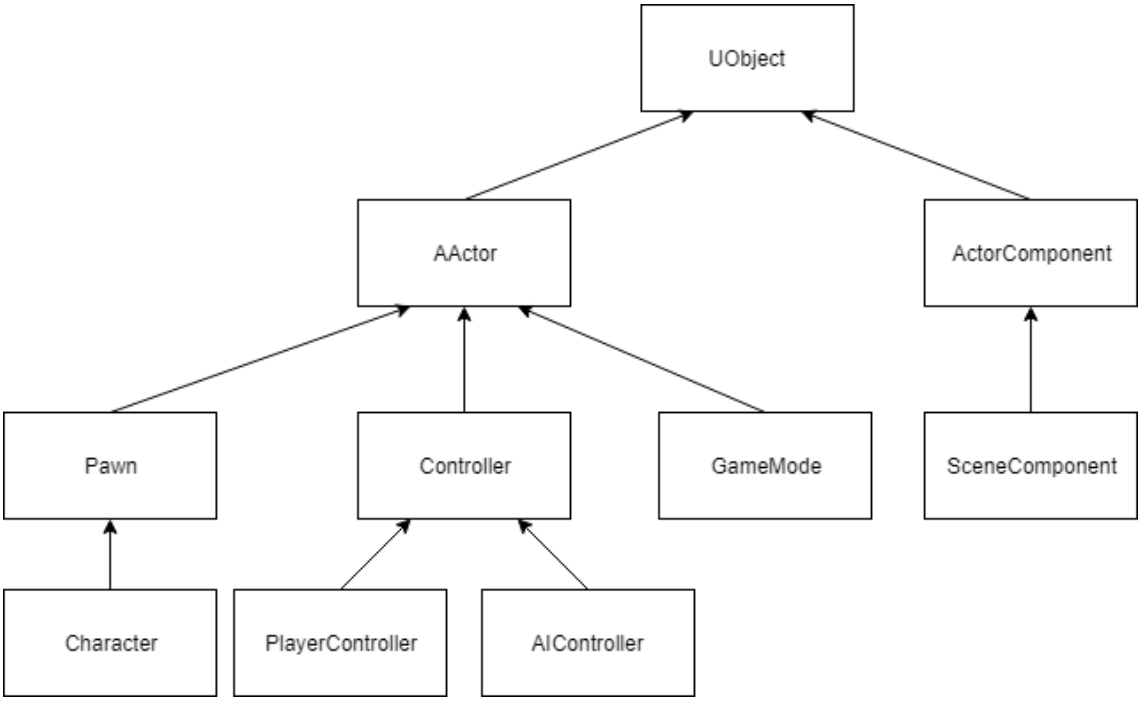
Unreal Engine 4 nám poskytuje množstvo tried na vytváranie a prispôsobenie programov.

`UObject` je základná trieda pre všetky objekty v Unreal Engine. Poskytuje nasledovnú funkcionalitu:

- Garbage collection
- Reflexia
- Serializácia
- Automatická aktualizácia zmeny štandardných vlastnosti

- Automatická integrácia do editora
- Automatická inicializácia vlastností
- Dostupnosť informácií o typoch počas behu programu.
- Replikácia po sieti

Hierarchiu základných objektov, a vzťahy dedenia môžeme vidieť na obrázku 3.2.



Obr. 3.2: Hierarchia základných objektov v Unreal Engine 4.

`AActor` je jedna zo základných tried, ktorá predstavuje akýkoľvek objekt, ktorý sa nachádza v hernom leveli. `AActor` je generická trieda, ktorá podporuje 3D transformáciu, ako napríklad posunutie, rotáciu a zmenu veľkosti. Triedy, ktoré dedia od `AActor` nazývame Actori, Actorov môžeme vytvárať a pridávať do scény alebo ničiť pomocou herného kódu v C++ alebo Blueprintoch. Unreal Engine 4 ponúka mnoho pripravených Actorov, ktorí slúžia na rôzne prípady použitia, napríklad:

- Pawn je postava, ktorá môže byť ovládaná pomocou Controllera. Špeciálny prípad je Character, ktorý predstavuje postavu podobnú človeku, a obsahuje základné funkcionality ako pohyb a detekciu kolízie.

- Controller obsahuje funkcionalitu pre ovládanie Actor-a, či už toho ovládaného hráčom, kde slúži ako rozhranie, alebo ovládaného počítačom, kde simuluje správanie.
- GameMode obsahuje pravidlá hry a podmienky na výhru.

Actori slúžia tiež ako kontajnery, ktoré v sebe držia objekty nazývané komponent.

Komponenty sú objekty, ktoré sú navrhnuté, aby boli použiteľné ako súčasť Actorov. Pomocou komponentov vieme poskladať funkcionalitu odlišných Actorov, na základe toho, ako chceme, aby sa správal. Taktiež nám to umožňuje navrhovať funkcionalitu, ktorá môže byť znova použiteľná.

Základné typy komponentov sú:

- `UActorComponent` je základná trieda pre komponenty, ktoré definujú znova použiteľné správanie. Môžu popisovať napríklad ovládanie, správanie sa Actor-a, ale aj kontrolu kolízií a renderovanie svetiel a tieňov.
- `USceneComponent` je rozšírenie triedy `UActorComponent`, ktoré obsahuje transformačnú štruktúru: pozícia, rotácia a mierka. Môžu sa na seba pripájať, čím vytvárajú hierarchiu komponentov.
- `UPrimitiveComponent` je `USceneComponent`, ktorý ma grafickú podobu. Napríklad mriežka alebo časticový systém.

3.2 Virtual-Reality Peripheral Network

Virtual-Reality Peripheral Network (VRPN) je sada tried v rámci knižnice a sada serverov, ktoré implementujú sieťovo transparentné rozhranie nezávisle na zariadení medzi aplikáciami a fyzickými zariadeniami (sledovacie zariadenia, tlačidlá atď.) používané vo virtuálno-realitných systémoch.

VRPN poskytuje:

- Prístup k rôznym virtuálno-realitným periférnym zariadeniam prostredníctvom spoločného, rozšíriteľného rozhrania.
- Sieťovo transparentné rozhranie pre zariadenia.
- Časové pečiatky pre všetky správy do a zo zariadení.

- Synchronizácia medzi klientmi a servermi na rôznych zariadeniach.
- Súčasné pripojenie k zariadeniu z viacerých zdrojov.
- Automatické pripojenie ku vzdialeným serverom po zlyhaní siete.
- Ukladanie a prehrávanie interaktívnych relácií.

VRPN poskytuje aj abstrakciu, pri ktorej pôsobia všetky zariadenia tej istej základnej triedy rovnako. Napríklad všetky sledovacie zariadenia (tracker) vyzerajú tak, že sú typu vrpn_Tracker. To znamená, že každé sledovacie zariadenie produkuje rovnaký typ správ. Taktiež je ale možné, aby aplikácia, ktorá vyžaduje prístup k špecializovaným funkciám určitého sledovacieho zariadenia (napríklad informovanie určitého typu sledovača o tom, ako často generovať správy), mohla odvodiť triedu, ktorá komunikuje s týmto typom sledovača. Ak by táto špecializovaná trieda bola použitá so sledovacím zariadením, ktoré by nevedelo, ako nastaviť jeho aktualizáciu, špecializované príkazy by boli ignorované.

Najbežnejšie typy zariadení sú uvedene nižšie, pričom nové typy zariadení môžu byť vytvorené.

- *Sledovacie zariadenia* posielajú správy o pozícií, orientácii, rýchlosti a/alebo zrýchlení.
- *Tlačidlá* posielajú správy o stlačení a uvoľnení pre jedno alebo viac tlačidiel.
- *Analógové zariadenia* hlásia správy jednej alebo viacerých analógových hodnôt.
- *Dial* posiela správy o inkrementálnej rotácii.
- *ForceDevice* umožňuje špecifikovať povrchy a silové polia v trojrozmernom priestore.

Nové typy zariadení môžu byť vytvorené mimo iné aj použitím týchto zariadení [13].

3.3 OptiTrack – NatNet Streaming Client

OptiTrack poskytuje sledovanie objektov a sledovanie pohybu celého tela s mimoriadnou presnosťou a veľmi krátkou odozvou. Zásuvný modul (plugin) pre

Unreal Engine 4 od NaturalPoint Inc. umožňuje prenos dát o tuhých telesách (rigid bodies), získaných sledovaním, z aplikácie Motive do Unreal Engine 4 [14]. V tejto časti si podrobne opíšeme štruktúru plugin-ov a implementáciu OptiTrack NatNet plugin-u v Unreal Engine 4.

3.3.1 Štruktúra pluginov

Každý plugin, ktorý chceme použiť, musí byť umiestnený v projekte alebo v samotnom Engine v adresari `Plugins`. Unreal Engine hľadá v adresari `Plugins` súbory s príponov .uplugin, inak nazývané aj plugin descriptors.

Plugin descriptor je súbor v JSON formáte a obsahuje základné informácie o plugin-e. `FileVersion` je jediné povinné pole a označuje verziu plugin descriptor súboru. Medzi najdôležitejšie voliteľné polia patria `Version`, `VersionName`, `FriendlyName`, `Description`, `Category`, `CreatedBy`, `Installed` a `Modules`. Pluginy, ktoré obsahujú kód, musia mať aspoň jeden záznam v `Modules`, ktorý sa skladá z povinných polí `Name` a `Type` a nepovinných `LoadingPhase`, ktoré nám určí fázu načítania plugin-u(`PreDefault`, `Default` alebo `PostConfigInit`), `WhitelistPlatforms` a `BlacklistPlatforms`, ktoré špecifikujú platformy, na ktorých má byť plugin skompilovaný a na ktorých má byť kompilácia zakázaná.

Adresár s plugin-om obsahuje adresár `Config`, v ktorom sú uložene konfiguračné súbory. Ak plugin obsahuje kód, tak by mal byť v adresári `Source`, knižnice tretej strany by mali byť v adresári `ThirdParty` a v `Binaries` sa nachádza skompilovaný kód. Herný obsah, ako napríklad blueprinty a modely, sa nachádza v adresári `Content` [15].

Za zmienku stojí, že niektoré pluginy nemusia obsahovať kód. Takéto pluginy sa nazývaju Content pluginy a mali by mať obsah v adresári `Content`. Ďalšie typy pluginov sú samostatné (Standalone) pluginy, ktoré rozširuju funkcionalitu Unreal Enginu a mimoherné (non-game) pluginy, ktoré ponúkaju rozšírenie funkcionality a zároveň ponúkajú rozhranie, ktoré umožňuje využívať jeho funkcionalitu v iných moduloch [16].

3.3.2 Implementácia plugin-u

OptiTrack - NatNet Streaming Client vo verzii 1.8.3 obsahuje priečinok `ThirdParty`, v ktorom sa nachádza NatNetSDK knižnica pre vývoj klient-server sieťovej ko-

munikácie pre vysielanie dát o zachytení pohybu. Priečinok `Source` pozostáva zo zdrojových a hlavičkových súborov. `OptitrackNatnet.cpp` obsahuje funkcie, ktoré sa vykonajú pri zapnutí a vypnutí plugin-u a funkcie `CreateClient` a `DestroyClient`, ktoré vytvárajú a ničia triedu `NatNetClient` z *NatNetSDK* knižnice, ktorá slúži na komunikáciu s NatNet serverom (Motive). V súbore `OptitrackClientOrigin.cpp` je hlavná funkcionalita tohoto plugin-u. Actor obsahuje funkciu `PreInitializeComponents`, ktorá sa volá pri vytváraní Actorov. V nej sa zavolá funkcia `InitializeClient`, ktorá sa pripojí na server (Parametre pre pripojenie sú uložené v Actor-ovi) a nastaví callback, ktorý spracuje prijaté údaje a uloží ich. Pre ukončenie prijímania údajov sa zavolá funkcia `ShutdownClient`.

Funkcie `InitializeClient` a `ShutdownClient` sú kľúčové pre dynamické nastavovanie IP adries a od verzie 1.8.3 sa zmenil ich prístup z privátneho na verejný, čo umožní použitie tohoto plugin-u bez ďalšej úpravy zdrojových súborov.

Pre uľahčenie používania je mnoho funkcií vystavených na používanie z blueprintov a nad premennými tejto triedy je použité makro `UPROPERTY`, ktoré umožňuje ich nastavenie v Unreal Editore.

Plugin ďalej obsahuje `OptitrackDebugHUD`, ktorý zobrazuje prijaté dáta a slúži na debugovanie, `OptitrackRigidBodyActor` a `OptitrackRigidBodyComponent`, ktoré slúžia pre pohyb Actorov po scéne a môžu sa využiť pre animáciu kostier modelov.

V našom prípade môžeme plugin použiť pre získavanie dát z Motive-u a neskôr ako vstupný parameter pre výpočet projekčnej matice, na základe ktorej sa vytvorí asymetrické zobrazenie mimo os (off-axis projection).

4 Záver analýzy

Z analýzy CAVE systémov sme zistili, že pre ideálne zobrazenie scény bude potrebné implementovať projekciu mimo os. Opísali sme si matematické podklady tejto projekcie, ktoré môžeme využiť pre zmenu projekčnej matice. Ďalej sme si opísali implementáciu zmeny projekčnej matice v Unreal Engine 4 a v krátkosti sme si predstavili Unreal Engine 4 a jeho základnú funkcionalitu.

Na presun dát z Motivu do Unreal Engine 4 sme si uviedli dve možnosti: Virtual-Reality Peripheral Network (VRPN) a plugin OptiTrack – NatNet Streaming Client. Vzhľadom na to, že OptiTrack plugin je oficiálny plugin pre Unreal Engine 4 od výrobcu OptiTracku a obsahuje rozsiahlu dokumentáciu, bude ideálnejšie využiť túto možnosť. Plugin nám taktiež ponúka automatické mapovanie prijatých dát na štruktúru, ktorú môžeme priamo využívať v Unreal Engine.

5 Návrh a implementácia dynamického nastavenia kamier

5.1 Architektúra

Výsledky analýzy nám načrtli základnú architektúru projektu pre dynamickú zmenu projekcie podľa pozície používateľa.

Ako prvý z komponentov využijeme OptiTrack Unreal Engine 4 Plugin, o ktorom sme si povedali v analýze a umožňuje nám komunikáciu s NatNet serverom, čo je v našom prípade aplikácia Motive, v Unreal Engine 4 (UE4). Vďaka tomuto plugin-u budeme dostávať informácie o pozícií očí hráča v reálnom svete, ktoré využijeme na výpočet projekčnej matice. Aby sme nemuseli meniť parametre pre pripojenie v Unreal Editore, využijeme grafické rozhranie. Toto grafické rozhranie je vytvorené ako Widget.

Druhý komponent je zásuvný modul pre vytvorenie zobrazenia mimo os. Zásuvný modul obsahuje najmä triedu `UOffAxisLocalPlayer`. Táto trieda je rozšírenie triedy `ULocalPlayer`, ktorá v Unreal Engine 4 predstavuje jedného hráča. Triedu `UOffAxisLocalPlayer` môžeme nastaviť ako Local Player triedu pre Unreal Engine projekt a Unreal Engine sa postará o automatické načítanie tejto triedy pri spustení hry. V tejto triede bude najdôležitejšia funkcia `CalcSceneView`, ktorá sa vykonáva v každom cykle hry a umožní nám prístup k `FSceneView`. `FSceneView`, obsahuje projekčné matice pre zobrazenie scény, ktoré môžeme modifikovať tak, aby sme dosiahli efekt off-axis zobrazenia scény.

Spojenie týchto dvoch komponentov by malo spôsobiť dynamickú zmenu scény na základe pozície používateľa v jaskyni.

Pre jednoduché využitie v scéne vytvoríme triedu `OffAxisActor`, ktorá bude rozširovať triedu Actor a jej inštancia môže byť umiestnená v scéne. Táto trieda

bude využívať funkcie zadefinované v triede `UOffAxisLocalPlayer` aby jej nastavila maticu zobrazenia.

5.2 Využitie NatNet Streaming Client

OptiTrack Unreal Engine 4 Plugin si môžeme voľne stiahnuť zo stránky OptiTracku [17]. Vybrali sme si verziu 1.8.3, ktorá je v dobe písania najnovšou verziou pre UE4. Využijeme Unreal Engine vo verzii 4.19. Vytvoríme si experimentálny prototyp, na ktorom skúsime, ako bude fungovať tento plugin vo virtuálnej jaskyni.

Najprv si vytvoríme projekt. UE4 ponúka mnoho pripravených projektov, pomocou ktorých môžeme rýchlo začať vytvárať hru. My sme si vybrali ako typ projektu C++ a z neho šablónu pre kameru z prvej osoby. V projekte typu C++ môžeme písať logiku programu pomocou jazyka C++, aj pomocou grafických blueprintov, preto je lepšie radšej si zvoliť tento typ projektu. Blueprint projekt sa dá dodatočne skonvertovať na C++ projekt, ale podľa vlastných skúsenosti je to mnohokrát náročnejšie a blueprinty niekedy miznú.

Akonáhle sa nám vytvorí projekt, mali by sme sa uistiť, že je plugin aktívny v menu Edit- >Plugins. Na obr. 5.1 vidíme, ako vyzerá položka pre tento plugin a krátky popis k nemu a zároveň sa môžeme uistiť, že je spustený zaškrtnutím políčka *Enabled*. Pokiaľ bol plugin spustený prvýkrát, je potrebné reštartovať Unreal Editor.

Obr. 5.1: Aktívny plugin v Unreal Engine 4.

Po pridaní zásuvného modulu do projektu je umožnený prenos dát o tuhých telesách a objekt `OptitrackClientOrigin` môže byť vložený do scény.

`OptitrackClientOrigin` je trieda, ktorá pripája projekt k serverovej aplikácii pomocou protokolu NatNet a importuje údaje o sledovaní. `OptitrackClientOrigin`

predstavuje zjednotený začiatok scény v Motive, zabezpečuje komunikáciu medzi Motivom a UE4 a poskytuje funkcie na získanie polohy hráča, ktoré sú získané pomocou OptiTracku. Tento objekt môže byť umiestnený kdekoľvek do scény a jeho veľkosť môže byť tiež zmenená. Po prepojení projektu so serverovou aplikáciou môžu byť dáta získané sledovaním použité na implementáciu displeja na hlavu (Head-mounted display) alebo na animáciu objektov v aplikáciach virtuálnej reality.

Adresu servera môžeme nastaviť priamo pri robení projektu v premenných `ServerAddress`. Do premennej `ClientAddress` nastavíme IP adresu klienta, teda počítača, na ktorom spúšťame tento program. Tieto premenné sa nachádzajú v objekte `OptitrackClientOrigin`. Ďalej si nastavíme premennú `RigidBodyID`, tak aby zodpovedala telesu, ktoré posielame z aplikácie Motive. Pokiaľ je na strane Motive-u nastavená rovnaká adresa ako v UE4, `OptitrackClientOrigin` po spustení hry inicializuje spojenie a Motive bude posielať dáta do projektu v UE4.

5.3 Konfigurácia Optitrack pluginu

Pri nastavovaní OptiTrack pluginu sa vyskytlo niekoľko problémov, kvôli ktorým sa na prvý pohľad zdalo, že plugin nefunguje správne.

Prvým z nich bolo nesprávne nastavenie parametra `RigidBodyID`, ktorý musí byť nastavený rovnako ako je nastavený v aplikácii Motive. `RigidBodyID` je potrebné nastaviť v `OptitrackClientOrigin`, a taktiež je potrebné, aby bol nastavený vo všetkých Optitrack Rigid Body komponentoch, ktorými chceme hýbať.

Ďalší dôvod, prečo pohyb postavy pôvodne nefungoval, bol spôsobený tým, že Optitrack Rigid Body komponent bol pripojený na Actor-a `FirstPersonCharacter`. Tento Actor je zo šablóny pre FPS projekt v UE4 a má definované ovládanie myšou a klávesnicou. Toto správanie bránilo plugin-u aby Actor-om hýbal. Po umiestnení Optitrack Rigid Body komponentu na iného Actor-a, ktorý nemal žiadne správanie fungoval pohyb na základe dát z OptiTrack-u podľa očakávaní.

Dôležité je aj nastavenie správnych IP adries v objekte `OptitrackClientOrigin`.

- `ServerAddress` zodpovedá IP adrese počítača, na ktorom beží NatNet server, v našom prípade aplikácia Motive.
- `ClientAddress` zodpovedá IP adrese počítača, na ktorom beží klientský pro-

jekt.

Pri konfigurácií IP adries rozlišujeme niekoľko prípadov.

- Klientský aj serverový počítač je na jednej lokálnej sieti. V tomto prípade zvolíme typ pripojenia (Connection type) Multicast. `ServerAddress` nastavíme na adresu počítača, na ktorom beží serverová aplikácia Motive. `ClientAddress` nastavíme na adresu klientského počítača.
- Klientský počítač sa pripája pomocou wi-fi routera na serverový počítač. V tomto prípade je nutné zvoliť typ pripojenia Unicast. `ServerAddress` nastavíme na adresu počítača, na ktorom beží serverová aplikácia Motive. `ClientAddress` v tomto prípade môžeme nastaviť na 0.0.0.0, pretože pri Unicastovom pripojení nie je potrebná.
- Pokiaľ sa klientský projekt spúšťa na počítači, na ktorom je zapnutý Motive, tak obe IP adresy môžeme nastaviť ako loopback adresu 127.0.0.1.

5.4 Zmena pozície na základe Optitracku

OptiTrack posiela údaje pre UE4 v štruktúre `FOptitrackRigidBodyState`, ktorá má v sebe nasledujúce položky:

- `FVector Position` predstavuje vektor, ktorý označuje pozíciu vo svete.
- `FQuat Orientation` predstavuje kvaternión, ktorý označuje orientáciu vo svete.

OptiTrack plugin má v sebe funkcionalitu pre zmenu polohy objektu v scéne. Na využitie funkcie stačí pridať postave, ktorou chceme hýbať, komponent Optitrack Rigid Body a v tomto komponente nastaviť ID tak, aby sa zhodovalo s Rigid Body ID v Motive.

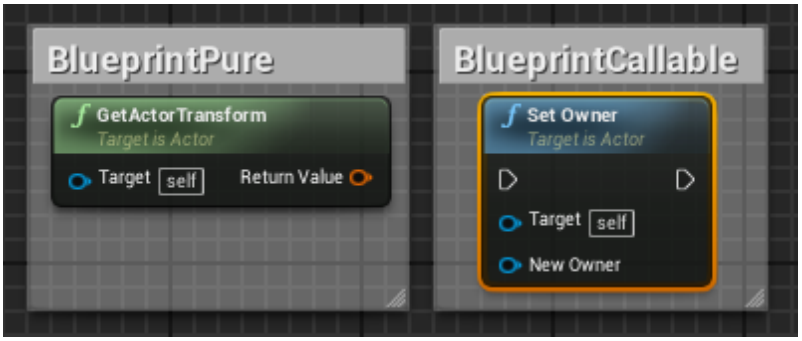
Pre potreby úpravy zobrazenia v UE4 je potrebné, aby sme mali priamo prístup k údajom, ktoré prišli sledovaním z OptiTrack-u. Preto si vytvoríme Blueprint, ktorý nám získa tieto údaje.

Pre zmenu pozície objektu v scéne je potrebné vedieť dva údaje: pozíciu (premenná typu `FVector`) a rotáciu (premenná typu `FRotator`). Keďže údaje o rotácii nám prídu ako kvaternión, je potrebné, aby sme spravili konverziu na `FRotator`.

Unreal Engine nepodporuje prácu s triedou `FQuat` priamo z blueprintov. Vytvorili sme si jednoduchú funkciu v C++, ktorej úlohou je nám túto konverziu umožniť. Aby sme mohli túto funkciu použiť z blueprintu, pridáme do hlavičky makro, ktoré môžeme vidieť na obr.5.2. `BlueprintPure` označuje, že táto funkcia nebude mať body pre zreťazenie vykonávania. Toto makro sa zvyčajne využíva pre prístup k premenným. Pre funkcie, ktoré musia byť vykonané v určitom poradí a menia stav hry, je lepšie využiť `BlueprintCallable`. Na obrázku 5.3 sú znázornené obidva typy blueprintov. Teraz môžeme nastaviť lokáciu a rotáciu objektu v scéne.

```
UFUNCTION(BlueprintPure, Category = "Conversion")
static FRotator QuatToRot(FQuat quaternion);
```

Obr. 5.2: Makro pre prístup k funkcii z blueprintu.



Obr. 5.3: BlueprintPure a BlueprintCallable.

Pozíciu používateľa môžeme získať pomocou funkcie `GetLatestRigidBodyState`, ktorá vracia boolean o tom, či prišli nové dáta, avšak táto funkcia k prijatým dátam pripočíta offset podľa toho, kde sa nachádza objekt `OptitrackClientOrigin`. Ak chceme dostavať nezmenené dáta, je lepšie využiť funkciu `GetLatestRigidBody-StateUntransformed`.

Pomocou funkcií `SetActorLocation` a `SetActorRotation`, ktoré ako parameter berú Actor-a, ktorým majú pohnúť a lokáciu, respektíve rotáciu Actor-a v hernom svete, zmeníme polohu objektu.

5.5 Komponent pre získavanie údajov z OptiTracku

Pre použitie vo virtuálnej jaskyni potrebujeme navrhnúť časť programu, ktorá bude zabezpečovať získavanie údajov z OptiTracku a bude zároveň kompatibilná s riešením replikácie herných obrazoviek, na ktorom sa momentálne pracuje v diplomovej práci [12]. Preto bude najideálnejšie implementovať túto časť pomocou rozšírenia triedy `UActor-Component`.

Vytvorili sme si triedu `UOptitrackDataComponent`, ktorá obsahuje 3 premenné:

- `int32 TrackingId` je identifikátor, ktorý korešponduje s Rigid Body ID v Motive;
- `class AOptitrackClientOrigin* TrackingOrigin` je smerník na objekt `OptitrackClientOrigin` v scéne, ktorý sa stará o komunikáciu;
- `FOptitrackRigidBodyState rbState` je posledná získaná pozícia z OptiTracku.

`TrackingOrigin` sa nastavuje vo funkcií `BeginPlay`, ktorá sa volá počas inicializačného cyklu každého Actor-a. Nakoľko je funkcia `FindDefaultClientOrigin` náročná na vykonávania, uložíme si návratovú hodnotu do premennej.

Vo funkcii `TickComponent` zabezpečíme, že v každom cykle hry sa `rbState` nastaví na najnovšie dáta, ktoré boli prijaté. Využíva funkciu `GetLatestRigidBody-StateUntransformed`, ktorá je zadefinovaná v OptiTrack plugine.

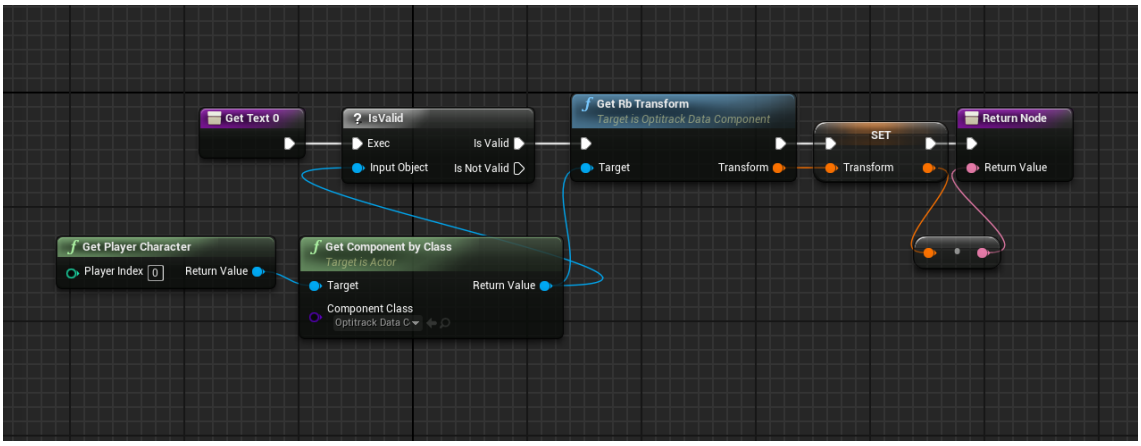
Komponent obsahuje funkciu `GetRbTransform`, ktorá má jeden vstupno výstupný parameter, ktorý nastaví podľa premennej `rbState`.

```
1 void UOptitrackDataComponent::GetRbTransform(FTransform&
     transform)
2 {
3     transform.SetRotation(rbState.Orientation);
4     transform.SetLocation(rbState.Position);
5 }
```

Funkcia je sprístupnená pre použitie z blueprintov, pomocou makra v hlavičkovom súbore.

Teraz je na použite nutné pridať komponent do ľubovoľného Actor-a v scéne. Komponent môžeme pridať priamo v editore. Stačí vybrať Actor-a zo scény, ktorému chceme pripojiť komponent a kliknúť na Add Component a vybrať si z listu, ktorý komponent chceme pripojiť. Komponenty sa môžu pridávať do Actorov aj pomocou C++ kódu alebo Blueprintov.

Vytvorili sme si nový Widget, pre zobrazovanie údajov na obrazovku. Blueprint tohoto widgetu je na obrázku 5.4. Blueprint využíva funkciu `GetRbTransform`



Obr. 5.4: Blueprint pre ovládanie widgetu.

a na základe nej nastavuje text tak, aby zodpovedal údajom. Funkcia, ktorá je zadefinovaná na blueprinte, je zviazaná (z angl. binded) s textovým poľom, takže ak sa zmení poloha, zmení sa aj text.

5.5.1 Rozhranie pre zmenu parametrov OptiTracku

Parametre OptiTrack-u, ako sú adresa servera, adresa klienta a typ pripojenia je potrebné vedieť zmeniť mimo Unreal Editora. Môžeme si vytvoriť nový Widget, ktorý bude obsahovať kolónky pre tieto parametre.

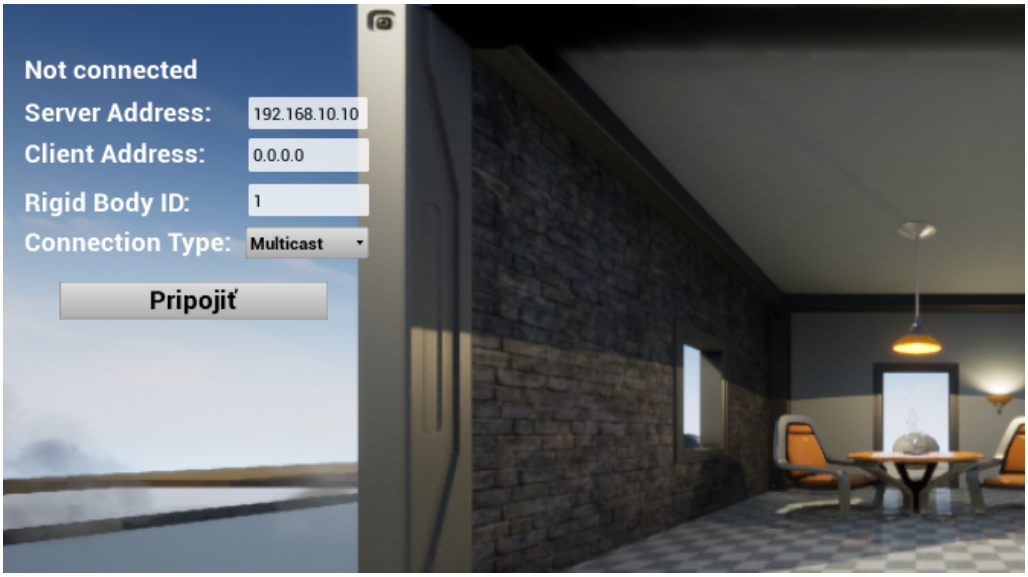
Pre nájdenie hlavného objektu `OptitrackClientOrigin` je potrebné mať referenciu na herný svet `World`, preto bola pridaná funkcia `GetWorldFromObject`, ktorá získa tento svet. Pri tejto funkcií sa využil v makre `UFUNCTION` príkaz `meta = (WorldContext = "WorldContextObject")`, ktorý do parametra funkcie vloží kontext sveta, z ktorého môžeme získať referenciu na inštanciu sveta.

Využili sme najmä funkcie `IsClientInitialized`, pre zistenie či je spojenie už nadviazané, `InitializeClient`, pre začatie spojenia, a `ShutdownClient`, pre

ukončenie spojenia.

Po stlačení klávesy P sa nastaví pauza v hre a zobrazí sa naše používateľské rozhranie. Na obrázku 5.5 vidíme, ako vyzerá. Na vrchu je nápis *Connected*, respektíve *Not connected*, ktorý značí, či sme pripojení na server. Po nastavení všetkých parametrov môžeme stlačiť tlačidlo s nápisom *Pripojiť*. Widget blueprint sa postará o to, že sa nastavia všetky parametre do inštancie `OptitrackClientOrigin` a pokúsi sa o inicializáciu klienta. Ak sa to podarí, nápis na vrchu sa zmení na *Connected*.

Pre pokračovanie v hre stačí opäť stlačiť klávesu P, čo vypne rozhranie a zruší pauzu v hre.



Obr. 5.5: Aktívny plugin v Unreal Engine 4.

5.6 Plugin pre zobrazenie mimo os

Je potrebné vytvoriť zobrazenie mimo os, aby sme nadobudli dojem reálnosti a zamedzili deformácii obrazu. Na zobrazenie mimo os je nutné zmeniť maticu zobrazenia. Toto v Unreal Engine 4 môžeme dosiahnuť pomocou implementácie triedy `ULocalPlayer`, ktorá obsahuje metódu `CalcSceneView`. Tá nám vráti referenciu na `View`, ktorý obsahuje matice potrebné pre projekciu.

Pre lepšiu kompatibilitu a neskoršie spojenie tohoto projektu s inými riešeniami pre virtuálnu jaskyňu si vytvoríme plugin pre off-axis zobrazenie. V Unreal Editore sa to dá spraviť jednoducho, stačí v hornom menu isť na položku

Edit- >Plugins- >New Plugin.

My sme si vybrali Blank Plugin šablónu, pretože najlepšie vyhovuje naším požiadavkám spojenia C++ súborov a blueprint obsahu.

V súbore `OffAxisPlugin.uplugin`, ktorý opisuje nastavenia pluginu, sme si nastavili typ plugin-u na `Runtime`, pretože sa využíva aj v spustiteľnej hre. Ďalej sme si nastavili fázu načítania na `PreDefault`, aby sa obsah využívaný v blueprintoch načítal dostatočne skoro a nevznikali tak chyby pri kompilácii.

5.6.1 Generácia projekčnej matice

Projekčnú maticu vypočítame v metóde `GenerateOffAxisMatrix`. Potrebujeme k tomu zadefinovať nasledujúce hodnoty:

- vzdialenosť od blízkej a vzdialenej roviny (near, far),
- ľavý a pravý spodný roh a ľavý horný roh (vektory pa, pb, pc),
- pozícia očí používateľa (vektor pe),
- osy smerujúce vpravo a hore (vektory vr, vu),
- normálový vektor vzhľadom na obrazovku (vektor vn).

Následne môžeme vypočítať maticu pre zobrazenie. Využijeme na to funkciu `FrustumMatrix`, ktorá nám na základe 6 parametrov vypočíta maticu. Táto funkcia je analógia k funkcii `glMatrix` z OpenGL. Prvý index označuje v tomto prípade riadok matice a druhý index značí stĺpec matice.

```
1  FMatrix Result;
2  Result.SetIdentity();
3  Result.M[0][0] = (2.0f * near) / (r - l);
4  Result.M[1][1] = (2.0f * near) / (t - b);
5  Result.M[2][0] = (r + l) / (r - l);
6  Result.M[2][1] = (t + b) / (t - b);
7  Result.M[2][2] = -(far + near) / (far - near);
8  Result.M[2][3] = -1.0f;
9  Result.M[3][2] = -(2.0f * far * near) / (far - near);
10 Result.M[3][3] = 0.0f;
11 return Result;
```

- r predstavuje vzdialenosť od pravej hrany obrazovky,
- l predstavuje vzdialenosť od ľavej hrany obrazovky,
- b predstavuje vzdialenosť od spodnej hrany obrazovky,
- t predstavuje vzdialenosť od hornej hrany obrazovky.
- near a far predstavujú vzdialenosť od blízkej a vzdialenej roviny orezania (clipping plane).

Pri generácii projekčnej matice využívame poznatky z analýzy o vytváraní všeobecnej perspektívnej projekcii. Matica je vytvorená pre ľavoruký súradnicový systém, ale Unreal Engine 4 používa pravoruký súradnicový systém, takže je nutné ju vynásobiť druhou maticou, ktorá otočí súradnicový systém.

5.6.2 Využitie triedy LocalPlayer

Trieda `ULocalPlayer` predstavuje jedného hráča v hre, a dá sa pomocou nej vytvoriť lokálna hra pre viacerých hráčov. My využijeme túto triedu hlavne pre získanie a nastavenie terajšej obrazovky (trieda `View`).

Vytvorili sme si triedu `OffAxisLocalPlayer`, ktorá dedí od triedy `LocalPlayer`. Na virtuálnu metódu `CalcSceneView` použijeme override, získame a upravíme `View` pomocou projekčnej matice. Pre možnosť nastavovania matice z blueprintov boli vytvorené funkcie `GenerateOffAxisMatrix` a `SetOffAxisMatrix`, ktoré vygenerujú a nastavia projekčnú maticu. Pomocou makra v hlavičkovom súbore sme sprístupnili tieto funkcie k využitiu z blueprintov. `OffAxisLocalPlayer` môžeme nastaviť ako Local Player triedu, ktorú Unreal Engine načíta pri štarte a bude používať počas behu programu. V menu prejdeme do Project Settings- >Engine->General Settings a v Default Settings nastavíme `OffAxisLocalPlayer` ako Local Player Class.

Trieda volá v každom cykle hry funkciu `CalcSceneView`, počas ktorej sa nastaví projekčná matica na vygenerovanú maticu. Okrem projekčnej matice je nutné nastaviť aj matice, ktoré s ňou súvisia, aby boli konzistentné a nevznikali deformácie. Medzi súvisiace matice patria: klasická a inverzná matica pohľadu, klasická a inverzná matica posunutého pohľadu, a matica, podľa ktorej sa vypočítavajú tiene.

5.7 Trieda OffAxisActor

Vytvorili sme objekt `OffAxisActor`, v ktorom môžeme generovať maticu a nastaviť ju na základe pozície oči a veľkosti obrazovky. Tento objekt je umiestnený v priečinku `Content` v plugin-e *OffAxisPlugin*, ktorý sme si vytvorili pre jednoduchšiu prenosnosť do iných projektov. Pre simulovanie pozície očí sme si vytvorili vstupné akcie. Tie je možné pridať v Edit- >Project Settings- >Input. Po vytvorení akcií a nastavení kláves, ktoré tieto akcie spúšťajú ich môžeme využiť v blueprintoch. Akcie menia premenné, ktoré definujú pozíciu používateľa v priestore.

Logika triedy `OffAxisActor` začína funkciou `BeginPlay`. V nej je potrebné nájsť referenciu na inštanciu `OptitrackClientOrigin`, ktorú si uložíme do premennej pre prípad využitia OptiTrack-u. Uloženie referencie na túto inštanciu je užitočné, nakoľko prehľadávanie scény je náročná a pomalá operácia a nemala by byť vykonávaná v každom cykle hry, pretože by mala značný vplyv na počet snímkov za sekundu.

Nasleduje funkcia, ktorá sa volá v každom cykle hry. V nej sa trieda rozhoduje podľa boolean parametra, či má používať OptiTrack, alebo simulované dáta, pre generáciu matice. Funkcia si v prípade použitia OptiTrack-u vyžiada informácie o poslednej pozícií a skonvertuje ich na údaje tak, aby zodpovedali súradnicovému systému Unreal Engine-u. Tá spočíva vo vzájomnej zámene hodnôt Y a Z v štruktúre vektora. Od tohoto momentu je chod programu pre simulovanú aj reálnu hodnotu totožný. Na základe simulovanej alebo reálnej pozície a šírky a výšky obrazovky sa vygeneruje projekčná matica a nastaví sa do inštancie `UOffAxisLocalPlayer`.

Pre otestovanie z jaskyne sme vytvorili akciu na prepínanie medzi zobrazovaním podľa simulovanej pozície a podľa pozície získanej OptiTrack-om. Akciu sme nastavili tak, aby reagovala na klávesu M.

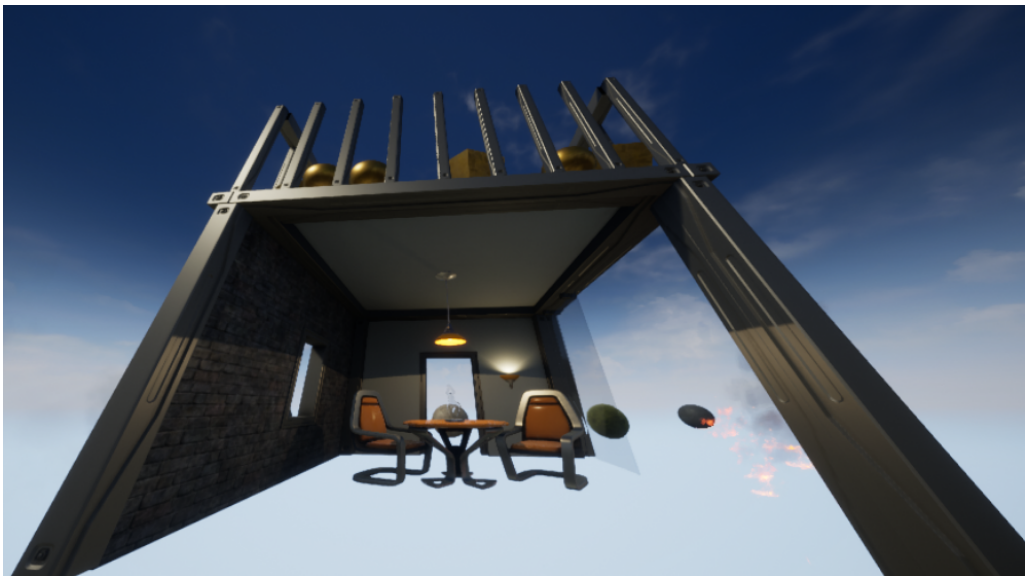
5.8 Miznutie objektov

Projekcia na prvý pohľad fungovala a obraz sa menil podľa pozície používateľa, avšak pri otáčaní kamerou sa zistilo, že objekty pri okrajoch obrazoviek miznú, ako vidíme na obrázku 5.6.

Bolo zjavné, že príčinou bude rozdiel medzi tým, čo sa zobrazuje v projekcii

mimo os a tým, čo vidí kamera. To, čo vidí kamera je pre nás zmenené pomocou projekčnej matice, avšak stále sa podľa pohľadu kamery vypočítava, ktoré objekty majú byť zobrazené. Objekty, ktoré by človek nevidel, nie je potrebné zobrazovať. Preto sa pre odľahčenie grafickej náročnosti tieto objekty nezobrazujú. Tento proces sa volá *Occlusion Culling*. Unreal Engine 4 má možnosť vypnúť *Occlusion Culling*, avšak po zmenách nastavení neprišlo k žiadnemu rozdielu oproti predošlým zobrazeniam. To nám naznačilo, že *Occlusion Culling* pravdepodobne nie je dôvod tohoto problému.

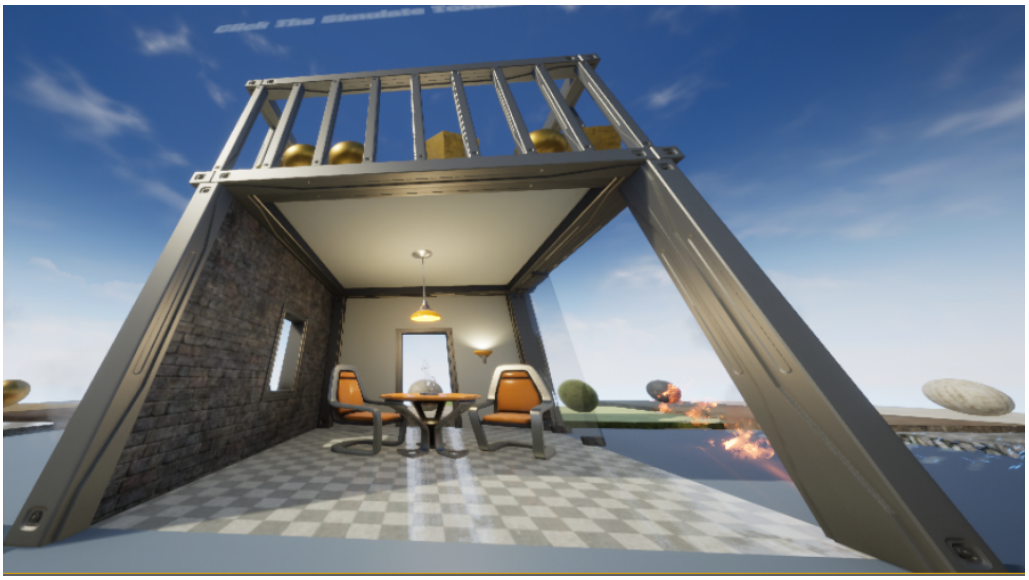
Ďalším potencionálnym dôvodom miznutia objektov mohlo byť nekonzistentné nastavenie matíc, ktoré riešia zobrazenie. Ani to sa po preskúmaní kódu, neukázalo byť dôvodom miznutia objektov.



Obr. 5.6: Miznutie objektov zo scény.

K riešeniu tohoto problému prispel článok o implementácii projekcií mimo os v Unity [10]. V ňom po nastavení projekčnej matice menili zorné pole (field of view) tak, aby bolo dostatočne veľké pre zachytenie objektov, ktoré vidíme pod zobrazením mimo os. Po nastavení zorného poľa kamery na vyššiu hodnotu ako v predošlých verziách, objekty už nemizli zo scény. Na obrázku 5.7 môžeme vidieť, ako vyzerá scéna po zmene veľkosti zorného poľa.

Pre jednoduchšie nastavenie veľkosti obrazovky sme si nakoniec do widgetu s nastaveniami pre OptiTrack pridali polia pre nastavenie šírky a výšky obrazovky. Widget sme nastavili tak, aby komunikoval s inštanciou `OffAxisActor` a údaje,



Obr. 5.7: Scéna po oprave miznutia objektov.

ktoré používateľ zadá uložil do príslušných premenných.

Ako mapa pre implementáciu a testovanie sa využíva `StarterMap`, ktorá je v Starter Content obsahu, a obsahuje ukážku textúr, objektov, častíc a zvukov základného obsahu pre Unreal Engine. Do mapy boli pridané inštancie `OptitrackClientOrigin` a `OffAxisActor`, ktoré dopĺňajú hru o funkcionalitu vytvorenú v tejto práci.

Výsledkom práce je ukážka zobrazenia mimo os, ktorá má možnosť využívať údaje o pozícii z OptiTrack-u. Projekt obsahuje aj plugin, ktorý je možné vložiť do ľubovoľného projektu v Unreal Engine 4, a tým dosiahnuť zobrazenie mimo os na akejkoľvek mape.

6 Vyhodnotenie

Integrácia OptiTrack-u do prostredia Unreal Engine 4 prebehla úspešne. O to sa z veľkej časti postaral plugin OptiTrack – NatNet Streaming Client, ktorý ponúkol skvelé API pre jednoduché vytváranie spojenia.

Projekt obsahuje používateľské rozhranie, pomocou ktorého môžeme nastavovať parametre priamo za behu programu. Pre jednoduchšie debugovanie sa na obrazovku vypisovali dáta prijaté z OptiTrack-u, avšak vo finálnej verzii toto nie je potrebné, pretože projekt bol otestovaný a prijímané dáta zodpovedali skutočnosti. Pri testovaní plugin-u sme prišli na niektoré situácie, v ktorých by plugin nemusel fungovať správne a postrehy z nich sú zapísané najmä v časti 5.3.

Zobrazenie mimo os sa podarilo vytvoriť pomocou využitia triedy `ULocalPlayer`. Spočiatku malo toto zobrazenie mnohé nedostatky, ktoré sa postupne odstránili.

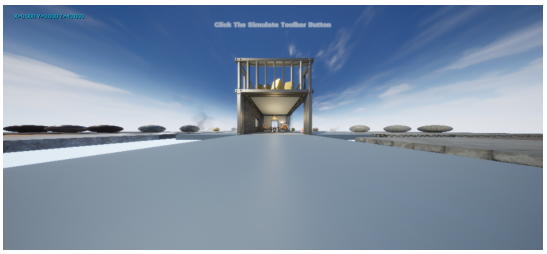
Najväčším nedostatkom pri zobrazovaní bolo miznutie objektov zo scény. To vznikalo, keď sa hráč otáčal, kvôli tomu, že to, čo videla kamera v Unreal Engine nekorešpondovalo s tým, čo malo byť viditeľné po upravení projekčnej matice. Problém sa podarilo vyriešiť upravením Field of View kamery, čím sme dosiahli to, že kamera mala viditeľné aj objekty, ktoré predtým mizli.

Pre účely testovania bez OptiTrack-u boli vytvorené akcie pre vstup, ktoré umožnili simulovať zmenu pozície hráča voči obrazovke.

Na obrázkoch vidíme, ako sa mení zobrazenie scény, keď sa zmení poloha hráča voči obrazovke, pričom X označuje súradnicu, ktorá korešponduje so šírkou obrazovky, Y korešponduje s výškou a Z je súradnica kolmá na obrazovku.

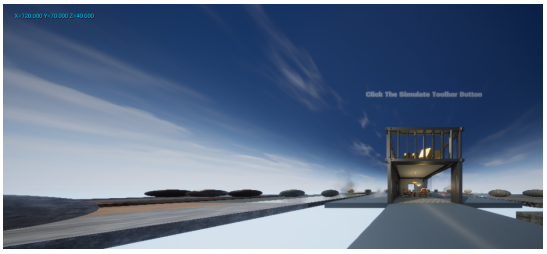
Na obrázkoch 6.1a a 6.1b sme najprv relatívne blízko k obrazovke a potom sa od nej vzdialime. Na obrázkoch 6.1c a 6.1d sme sa navyše posunuli mimo centra obrazovky a menili vzdialenosť od obrazovky.

Vo virtuálno-realitnej jaskyni sa riešenie otestovalo pre jeden monitor. Testo-

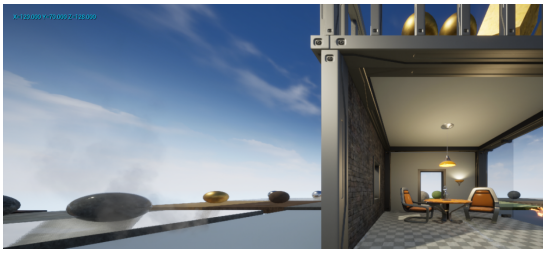


(a) X = 0, Y = 0, Z = 40

(b) X = 0, Y = 0, Z = 128



(c) X = 120, Y = 70, Z = 40



(d) X = 120, Y = 70, Z = 128

Obr. 6.1: Porovnanie zobrazenia pri zmene pozície očí.

vala sa najmä funkcia OptiTrack-u a to, či funguje zobrazovanie podľa pozície. Pri tomto testovaní sa prišlo na niekoľko poznatkov. Zistili sa hodnoty, aké udáva OptiTrack, čím sme si vytvorili predstavu o rozmeroch CAVE systému. To pomohlo pri mapovaní priestoru CAVE systému do priestoru v Unreal Engine.

V pláne je spojiť toto riešenie s riešením replikácie obrazoviek, vďaka čomu by bolo možné využívať zobrazenie mimo os v CAVE systéme na Technickej Univerzite v Košiciach.

7 Záver

Unreal Engine 4 je graficky vyspelé jadro a preto jeho využitie má veľký potenciál, ktorý sme zúžitkovali v tejto práci. Cieľom práce bolo využitie OptiTrack-u v hernom jadre Unreal Engine 4. Praktické využitie dát získavaných OptiTrack-om u nás predstavovalo zobrazenie mimo os, ktoré je pre CAVE systémy mimoriadne užitočné. Zobrazenie mimo os umožní používateľom voľný pohyb po virtuálnej jaskyni bez toho, aby bol obraz, ktorý vidia deformovaný.

OptiTrack sa dá jednoducho využiť za pomoci plugin-u od NaturalPoint, Inc. a zobrazenie scény sa mení v reálnom čase na základe pozície používateľa. Na zobrazenie mimo os sa využil výpočet projekčnej matice, ktorá sa nastavila na príslušnom mieste pomocou rozšírenia triedy, natívnej pre Unreal Engine 4, ktorá ma za úlohu spravovať jedného lokálneho hráča.

Táto práca zároveň slúžila, ako čiastkové riešenie vizualizačného softvéru pre CAVE systém v laboratóriu LIRKIS na Technickej Univerzite v Košiciach. Súčasne riešenie sa zatiaľ nepodarilo spojiť s ostatnými riešeniami pre virtuálno-realitnú jaskyňu. Dôvodom bolo najmä to, že na riešení replikácie obrazoviek sa pracovalo súčasne s touto prácou a naše riešenia bolo možne spojiť až po dokončení práce. Tieto riešenia je v pláne spojiť v budúcnosti a k spojeniu by malo napomôcť to, že riešenia v tejto práci boli vytvorené so zreteľom na kompatibilitu a znovapoužiteľnosť.

Literatúra

[1] Miloš Marcinčin. "Využitie herného jadra Unreal pre virtuálno-realitnú jaskyňu". Dipl. pr. Košice: Technická Univerzita v Košiciach, apr. 2017.

[2] Howard Rheingold. *Virtual Reality: Exploring the Brave New Technologies*. Simon & Schuster Adult Publishing Group, 1991. ISBN: 0671693638.

[3] Rae A Earnshaw. *Virtual reality systems*. Academic press, 2014.

[4] Carolina Cruz-Neira, Daniel J Sandin a Thomas A DeFanti. "Surround-screen projection-based virtual reality: the design and implementation of the CAVE". In: *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*. ACM. 1993, s. 135–142.

[5] Thomas A DeFanti et al. "The StarCAVE, a third-generation CAVE and virtual reality OptIPortal". In: *Future Generation Computer Systems* 25.2 (2009), s. 169–178.

[6] *VR Cluster– "Virtual Reality for multi screen and clustered solutions"*. [Online; cit. 17-05-2018]. URL: `http://vrcluster.io/`.

[7] Andrei Haidu. *Robot Commonsense games*. [Online]. URL: `http://robcog.org/`.

[8] Ken Power. *Perspective Projection*. URL: `http://glasnost.itcarlow.ie/~powerk/GeneralGraphicsNotes/projection/perspective_projection.html`.

[9] Robert Kooima. "Generalized perspective projection". In: *School of Elect. Eng. and Computer Science* (2008), s. 1–7.

[10] Wikibooks. *Cg Programming/Unity/Projection for Virtual Reality — Wikibooks, The Free Textbook Project*. [Online; cit. 17-05-2018]. 2017. URL: https://en.wikibooks.org/w/index.php?title=Cg_Programming/Unity/Projection_for_Virtual_Reality&oldid=3231767.

[11] *Howto modify the projection matrix*. [Online]. URL: https://answers.unrealengine.com/questions/65003/howto-modify-the-projection-matrix.html.

[12] Matúš Gabriška. "Prispôsobenie herného jadra Unreal pre virtuálno-realitnú jaskyňu". Dipl. pr. Košice: Technická Univerzita v Košiciach, 2018.

[13] Miguel Ribo, Axel Pinz a Anton L Fuhrmann. "A new optical tracking system for virtual and augmented reality applications". In: *Instrumentation and Measurement Technology Conference, 2001. IMTC 2001. Proceedings of the 18th IEEE*. Zv. 3. IEEE. 2001, s. 1932–1936.

[14] NaturalPoint Corporation. *OptiTrack Documentation Wiki*. URL: https://v20.wiki.optitrack.com/index.php?title=OptiTrack_Unreal_Engine_4_Plugin.

[15] Epic Games. *Plugins*. URL: https://docs.unrealengine.com/en-US/Programming/Plugins.

[16] Bob Gneu. *An Introduction to UE4 Plugins*. URL: https://wiki.unrealengine.com/An_Introduction_to_UE4_Plugins#Distribution.

[17] NaturalPoint Corporation. *OptiTrack Unreal Engine 4 Plugin*. URL: https://optitrack.com/downloads/plugins.html#unreal-plugin.

Zoznam skratiek

API Rozhranie pre programovanie aplikácií.

blueprint Skriptovací systém v Unreal Engine 4, na báze spájania uzlov.

CAVE Cave automatic virtual environment.

JSON JavaScript Object Notation.

VRPN Virtual-Reality Peripheral Network.

Zoznam príloh

Príloha A DVD médium – záverečná práca v elektronickej podobe,

Príloha B Používateľská príručka

Príloha C Systémová príručka

Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky

Príloha B: Používateľská príručka

2018 **Dominik Tóth**

Obsah

Zoznam obrázkov

1 Používateľská príručka

1 Úvod

V tejto príručke bude vysvetlené, ako spustiť a pracovať s programom OptitrackProject.

2 Spustenie

Projekt sa spúšťa pomocou súboru `OptitrackProject.exe`. Spustiteľná verzia je pripravená pre 64-bitovú verziu systému. Pre spustenie bez OptiTrack-u nie sú žiadne pre-rekvizity.

2.1 Spustenie s OptiTrack-om

Pre spustenie s OptiTrack-om je potrebné najprv zapnúť aplikáciu *Motive* a v záložke `Data Streaming` zmeniť hodnotu v `Stream Rigid Bodies` na `True`. V prípade, že klientský a serverový počítač nie sú na rovnakej podsieti je nutné zvoliť *Unicast* spojenie, inak volíme *Multicast*.

Je nutné udeliť výnimku vo firewall-e(systém sa pravdepodobne opýta, či chceme udeliť výnimku). Po spustení stlačíme klávesu P a nastavíme adresy nasledovne:

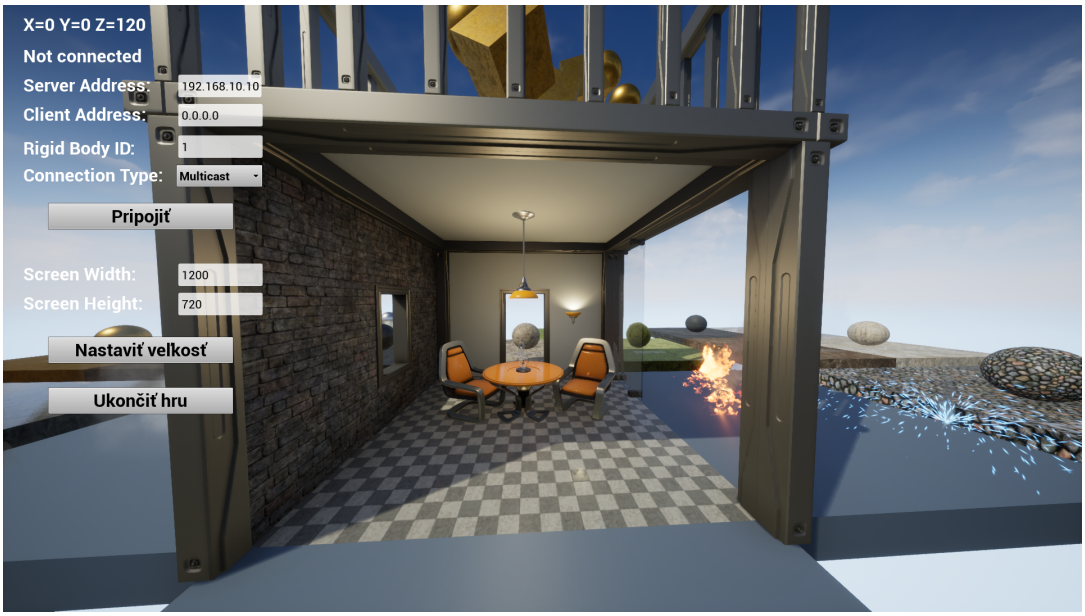
- *Server Address* na adresu servera, ktorá je napísaná v *Motive*;
- *Client Address* je adresa počítača, na ktorom beží projekt;
- *Rigid Body ID* je ID, ktoré máme v *Motive* pod *Streaming ID*(pravdepodobne 1);

- *Connection Type* je typ pripojenia: *Multicast*, v prípade rovnakej siete na klientskom aj serverovom počítači, alebo *Unicast*, v prípade rozdielnych sietí.

Po nastavení všetkých parametrov, stlačíme tlačidlo `Pripojiť` a nápis v druhom riadku by sa mal zmeniť na *Connected*. Môžeme stlačiť klávesu P a obraz by sa mal meniť podľa pozície používateľa.

3 Ovládanie

Pomocou klávesy M meníme vstup zo simulovaných hodnôt na hodnoty získane OptiTrack-om. Klávesa P preruší hru a zobrazí nám nastavenia pre OptiTrack, veľkosť obrazovky a tlačidlo pre ukončenie hry. Opätovne stlačenie klávesy P obnoví hru. Nastavenia môžeme vidieť na obrázku 1.1. Simulovanú pozíciu môžeme me-



Obr. 1.1: Blueprint pre ovládanie widgetu.

niť nasledovne:

- šípka hore a dole zvyšuje a znižuje hodnotu Y;
- šípka vľavo a vpravo zvyšuje a znižuje hodnotu X;
- klávesa page up a page down zvyšuje a znižuje hodnotu Z.

Klávesa R reštartuje pozíciu očí. Pohyb po mape sa uskutočňuje klávesami W,A,S,D a otáčanie je pomocou myši.

Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky

Príloha C: Systémová príručka

2018 **Dominik Tóth**

Obsah

1 Systémová príručka

1 Úvod

V tejto príručke sú popísané pre-rekvizity pre vývoj, ako pridať plugin do projektu a verejné triedy a premenné.

2 Pre-rekvizity pre vývoj

Ak chcete vyvíjať projekt v Unreal Engine 4 je potrebné mať nainštalované nasledovné položky: *Visual Studio 2017* a *Unreal Engine 4*.

Inštalácia prostredia *Visual Studio 2017* pre vývoj s Unreal Engine-om je popísaná na oficiálnej stránke Epic Games: `https://docs.unrealengine.com/en-us/Programming/Development/VisualStudioSetup`.

Unreal Engine 4 je potrebné mať vo verzii 4.19 alebo vyššie, kvôli kompatibilite assetov, ktoré sa v nižších verziách nebudú zobrazovať. Túto verziu si môžete nainštalovať pomocou Epic Launcher-u, ktorý môžete stiahnuť z `https://www.epicgames.com/unrealtournament/download` alebo môžete stiahnuť zdrojové kódy z *GitHub-u*(`https://github.com/EpicGames/UnrealEngine`) a Engine si skompilovať. Jednoduchšia je prvá možnosť.

3 Pridanie pluginu do projektu

Plugin *OffAxisPlugin* si do vlastného projektu pridáte skopírovaním priečinku *OffAxisPlugin* z priečinku */src/OptitrackProject/Plugins/* na priloženom médiu do priečinku *Plugins* vo vašom projekte. Ak priečinok *Plugins* nemáte, tak si ho vytvorte. K tomuto pluginu je potrebné skopírovať aj plugin OptitrackNatnet, z rovnakého

priečinku, alebo si ho stiahnuť z oficiálnej stránky OptiTrack-u: `http://optitrack.com/downloads/plugins.html`.

Po pridaní týchto dvoch pluginov do projektu, môžete pracovať s ich obsahom.

4 Opis tried a funkcií

Najprv si opíšeme triedu v hlavnom projekte v priečinku */src/OptitrackProject/Source/*.

Súbor `OptitrackDataComponent.cpp` je komponent, ktorý môžeme pripojiť k Actor-ovi. Obsahuje verejné funkcie:

- `GetRBTransform` vráti `FTransform`, na základe poslednej získanej pozície z OptiTrack-u.
- `GetOptitrackClient` vráti referenciu na `AOptitrackClientOrigin`.

Verejné premenné sú:

- `int32 TrackingID` - ID sledovaného Rigid Body
- `class AOptitrackClientOrigin* TrackingOrigin` - referencia na origin objekt v scéne.

Tieto funkcie sú sprístupnené na použitie z blueprintov a premenné sú nastavené ako vlastnosti, ktoré môžeme meniť z editora.

V priečinku *OptitrackProject/Content/* sa nachádzajú vytvorené Widgety:

- `OptitrackSettings` - grafické rozhranie pre nastavenia OptiTrack-u.
- `TransformWidget` - Widget pre výpis údajov z OptiTrack-u.

OffAxisPlugin obsahuje v zložke *Source* nasledujúce súbory:

HelpersLibrary.cpp blueprint knižnica, ktorá sprístupňuje do blueprintov kód. Obsahuje statické funkcie:

- `GetWorldFromObject` získa referenciu na svet.
- `QuatToRot` vykoná konverziu z `FQuat` na `FRotator`.

OffAxisLocalPlayer.cpp je trieda, ktorú môžeme nastaviť ako hlavnú Local Player triedu v Unreal Editore. Obsahuje aj nasledujúce statické funkcie, ktoré môžeme použiť v blueprintoch:

- `GenerateOffAxisMatrix` vygeneruje maticu na základe šírky a výšky monitora a pozície očí.
- `SetOffAxisMatrix` nastaví maticu do premennej v `OffAxisLocalPlayer`

OffAxisPlugin/Content/ obsahuje nasledujúce blueprinty:
PositionWidget.uasset, ktorý slúži na vypísanie terajšej pozície očí na obrazovku. Vypisuje simulovanú aj reálnu pozíciu.
OffAxisActor.uasset obsahuje nasledovné premenné:

- `FVector eyeRelativePos` - pozícia očí.
- `float width` - šírka obrazovky.
- `float height` - výška obrazovky.
- `OptitrackClientOrigin OptitrackClient` - referencia na inštanciu v scéne.
- `boolean useOpti` - ak `true` používa OptiTrack pre Off-Axis projekciu, inak používa simulovanú pozíciu.

5 Využitie Off-Axis projekcie

OffAxisActor.uasset obsahuje blueprint pre Off-Axis projekciu. Pre fungovanie tejto projekcie v scéne je potrebné pridať si inštanciu tejto triedy do scény. Ďalej je potrebné zadefinovať si akcie pre vstup v Project Settings- >Input.

Pod Action Mappings je potrebné pridať akcie:

- *Toggle* - zmení zobrazenie podľa simulovanej pozície na zobrazenie podľa pozície z OptiTrack-u
- *Pause* - Preruší hru a zobrazí nastavenia(`OptitrackSettings` widget).

Pod Axis Mappings je potrebné pridať akcie a zadať skratky pre os -1.0 a +1.0:

- *Up* - mení pozíciu Y osi.

- *Left* - mení pozíciu X osi.
- *Forward* - mení pozíciu Z osi.

Klávesové skratky môžete zadať podla vlastného uváženia.

Ak chcete využívať aj OptiTrack, tak je potrebné pridať do vašej scény inštanciu `OptitrackClientOrigin`, do ktorej môžete nastaviť IP adresy a parametre pripojenia. Parametre sa počas behu programu môžu meniť pomocou `OptitrackSettings` widget-u.