Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky

Konfigurovateľné mimo-osové zobrazenie pre virtuálno-realitnú jaskyňu

Bakalárska práca

2019 **Petra Romanová**

Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky

Konfigurovateľné mimo-osové zobrazenie pre virtuálno-realitnú jaskyňu

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 2019 **Petra Romanová**

Názov práce:	Konfigurovateľné mimo-osové zobrazenie pre virtuálno-realitnú jaskyňu
Pracovisko:	Katedra počítačov a informatiky, Technická univerzita v Košiciach
Autor:	Petra Romanová
Školiteľ:	Ing. Štefan Korečko, PhD
Konzultant:	
Dátum:	24. 5. 2019
Kľúčové slová:	CAVE systém, UnrealEngine, OptiTrack, Motive, VRPN

Abstrakt: Práca sa zaoberá opravením chyby v zobrazení mimo os v hernom jadre Unreal Engine pre CAVE systém v laboratóriu LIRKIS. V prvej časti sú analyzované možné pôvodné implementačné chyby a predstavené riešenia týchto chýb. Druhá časť sa zaoberá návrhmi a implementáciami rôznych experimentálnych riešení. Výsledné riešenie pozostáva z nahradenia pôvodnej implementácie novším riešením od Epic Games, ktoré v sebe zahŕňa aj zobrazenie mimo os. Pre toto riešenie bolo nutné vykonanie niekoľkých testovaní, popísaných v práci. Na záver sú zhrnuté a predstavené výsledky celej práce.

Thesis title:	Configurable Off-Axis Projection for Virtual Reality CAVE
Department:	Department of Computers and Informatics, Technical University of Košice
Author:	Petra Romanová
Supervisor:	Ing. Štefan Korečko, PhD
Tutor:	
Date:	24. 5. 2019
Keywords:	CAVE system, UnrealEngine, OptiTrack, Motive, VRPN

Abstract: Thesis deals with the correction of the error in off-axis projection in the Unreal Engine for CAVE system in the LIRKIS laboratory. The first part analyzes possible initial implementation errors and presents solutions to these errors. The second part deals with the design and implementation of various experimental solutions. The resulting solution consists of replacing the original implementation with a newer solution presented by Epic Games, which includes off-axis projection. Several tests, described in the work, were necessary for this solution. Finally, the results of the work are summarized and presented.

TECHNICKÁ UNIVERZITA V KOŠICIACH

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Katedra počítačov a informatiky

ZADANIE BAKALÁRSKEJ PRÁCE

Študijný odbor: **Informatika**
Študijný program: **Informatika**

Názov práce:

Konfigurovateľné mimo-osové zobrazenie pre virtuálno-realitnú jaskyňu

Configurable Off-Axis Projection for Virtual Reality CAVE

Študent: **Petra Romanová**
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 súčasný stav podpory snímania polohy používateľa systémom OptiTrack a mimo-osového zobrazenia v hernom jadre Unreal a to v LIRKIS CAVE aj v iných systémoch virtuálnej reality.
3. Navrhnúť a implementovať rozšírenie súčasného subsystému snímania používateľa systémom OptiTrack a mimo-osového zobrazenia pre herné jadro Unreal v LIRKIS CAVE tak, aby odstránilo existujúce nedostatky a bolo ľahko prispôsobiteľné pre rôzne konfigurácie displejov.
4. Pri návrhu a implementácii spolupracovať s riešiteľom súvisiacej záverečnej práce na integrácii herného jadra Unreal do LIRKIS CAVE.
5. Vypracovať dokumentáciu podľa pokynov vedúceho práce.

Jazyk, v ktorom sa práca vypracuje: slovenský
Termín pre odovzdanie práce: 24.05.2019
Dátum zadania bakalárskej práce: 31.10.2018

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

TECHNICKÁ UNIVERZITA V KOŠICIACH FAKULTA ELEKTROTECHNIKY A INFORMATIKY -11-

prof. Ing. Liberios Vokorokos, PhD.
dekan fakulty

Čestné vyhlásenie

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

Košice, 24.5.2019 .....

Vlastnoručný podpis

Poďakovanie

Na tomto mieste by som rada poďakovala svojmu vedúcemu práce, Ing. Štefanovi Korečkovi, PhD. za odbornú pomoc pri písaní záverečnej práce.

Rovnako by som sa rada poďakovala Bc. Dominikovi Tóthovi za pomoc pri testovaní v LIRKIS CAVE, mojej mame Viere Romanovej a starej mame Jiřine Romanovej za podporu počas štúdia.

V neposlednom rade by som sa rada poďakovala Ing. Ladislavovi Pomšárovi za psychickú aj emocionálnu oporu počas štúdia a za upozornenie na gramatické chyby v tejto práci.

Obsah

Zoznam obrázkov

Úvod

Vo virtuálno-realitnej jaskyni v laboratóriu LIRKIS sa v súčasnosti využíva herné jadro SuperEngine, ktoré bolo navrhnuté pre Technickú univerzitu v Košiciach. Jadro sa považuje za zastaralé, a preto bol v roku 2018 pridaný nový výkonnejší Unreal Engine 4. O túto implementáciu nového jadra sa úspešne postaral Matúš Gabriška v jeho diplomovej práci [1]. Pre nové jadro Unreal Engine 4 bol prispôsobený systém optického snímania pohybu OptiTrack, vďaka ktorému získavame údaje o pohybe používateľa. Údaje o pozícii hráča sa spracovávajú v softvéri Motive a následne posielajú do jadra Unreal Engine 4. Do tohto jadra bolo pridané zobrazovanie mimo os, ktoré nám má pomôcť viac sa vžiť do jednotlivých scén, keďže upravuje obraz vzhľadom na pozíciu osoby, na základe pozície hráča v hre. Túto implementáciu vyriešil Dominik Tóth v jeho bakalárskej práci [2].

Avšak vykresľovanie obrazu nie je dokonalé, jednotlivé scény vykreslené na obrazovkách na seba nenadväzujú a používateľ nemá v jaskyni pocit úplného splynutia s virtuálnou realitou.
Po analyzovaní chýb sme dospeli k názoru, že nebude potrebné len upravenie implementácie D. Tótha, ale aj pridanie novej verzie herného jadra. V časti návrhu a implemetácii riešenia sú opísané chyby, ktoré nebolo možné inak odstrániť. V novej verzii Unreal Engine bolo pridané riešenie na komunikáciu počítačov v CAVE systémoch s názvom nDisplay. Riešenie je na trhu od novembra 2018, ale keďže sa jedná o experimentálne riešenie, všetky chyby v tomto riešení boli opravené až v marci roku 2019. Riešenie obsahuje aj konfiguračný súbor na modifikáciu parametrov zobrazovania. Problém s miznutím objektov, o ktoré sa mal postarať T. Tóth v rámci jeho záverečnej práce, nebolo potrebné opravovať v novej verzii, pretože po implementácii nDisplay sa už nevyskytol.

1 Formulácia úlohy

Prvou úlohou tejto práce je oboznámiť sa so softvérovým a hardvérovým vybavením virtuálno-realitnej jaskyne LIRKIS CAVE, zobrazením mimo os a aktuálnou implementáciou pre LIRKIS CAVE. CAVE si bližšie predstavíme v analýze, spolu aj s niekoľkými inými CAVE systémami vo svete.

Druhým bodom je návrh a implementácia rozšírenia súčasného subsystému snímania používateľa systémom OptiTrack a mimo-osového zobrazenia pre herné jadro Unreal Engine v LIRKIS CAVE, opísané v kapitole 4. Táto implementácia by mala odstrániť existujúce nedostatky a byť ľahko prispôsobiteľná pre rôzne konfigurácie displejov. Implementácia je rozdelená do dvoch častí v ktorých je detailne popísaný celý proces riešenia. Najskôr bude potrebné implementovať prepojenie komunikácie počítačov v klastri, a potom pridanie OptiTrack systému spolu s mimo-osovým zobrazovaním a ich konfiguračnému súboru.

Tretím bodom by mala byť spolupráca s riešiteľom súvisiacej záverečnej práce. Súvisiaca záverečná práca sa zaoberala miznutím objektov, táto chyba sa po novej implementácii už nevyskytovala, preto nebolo nutné sa týmto problémom zaoberať.

Posledným bodom je vypracovanie dokumentácie k implementácii podľa pokynov vedúceho práce. Potrebnou dokumentáciou je používateľská príručka, ktorá sa nachádza v prílohách k práci.

2 Virtuálna realita a zobrazovanie obrazu

Virtuálnu realitu (VR, ang. Virtual Reality) je možné definovať ako počítačom vytvorený priestor zachytávajúci pohyb používateľa v reálnom čase. Vďaka senzorom sa prostredie VR mení a umožňuje používateľovi precítiť vygenerované prostredie inšpirované reálnym alebo nereálnym svetom. Podľa Roberta V. Kenyona [3] je VR definovaná ako široko-plošná, počítačom vygenerovaná prezentácia, ktorá využíva multisenzory na snímanie používateľa .

Mimo VR je možné stretnúť sa aj s rozšírenou realitou (ang. Augmented Reality), ktorá nie je až tak dokonalá, najmä kvôli technickým požiadavkám a cene, alebo aj so zmiešanou realitou (ang. Mixed Reality), ktorej pohlaď na reálny svet je obohatený o počítačom generované objekty. Systém VR môžu obsahovať nasledujúce podsystémy [4]:

1. *Vizualizačný podsystém*, vnímanie obrazu používateľa, využíva princíp sledovania statického monitora (napr. sledovanie monitora) alebo princíp sledovania mobilnej zobrazovacej jednotky (napr. sledovanie cez displeje upevnené na hlave)

2. *Akustický podsystém*, zvuková reprezentácia vo VR.

3. *Kinematicky a statokinematický podsystém*, sledovanie alebo simulácia pohybu používateľa

4. *Akustický podsystém*, vytváranie pocitu dotyku a určitého odporu

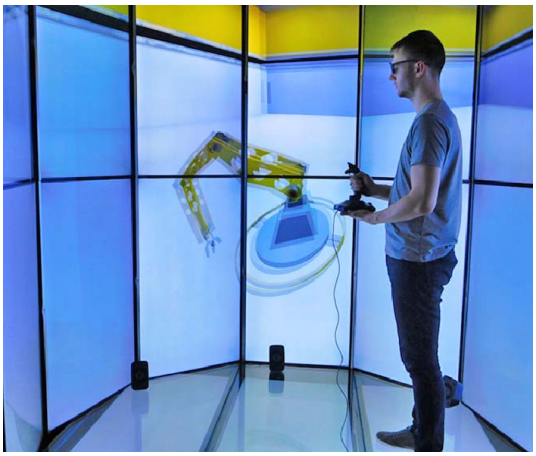
5. *Podsystémy pre ostatne vnemy*, simulácia menej podstatných vnemov vo VR

V roku 1992 okrem viac známych typov VR, ktoré používali displeje umiestnené na hlave či ramene, Univerzita v Chicagu, Illinois [3] prvý krát demonštro-

vala CAVE (Cave Automatic Virtual Environment). CAVE je definovaná ako implementácia systému virtuálnej reality pozostávajúceho z fyzickej inštalácie pre obrazový alebo videoprojekčný softvér [5]. Fyzická inštalácia sa zvyčajne skladá z uzavretej miestnosti, v ktorej sú videoprojektory nasmerované na dve alebo viac jej stien alebo monitory vytvárajúce polkruh, čo vytvára dojem, že sa používatelia nachádzajú vo vnútri virtuálneho prostredia. Softvérová súčasť CAVE je zodpovedná za riadenie premietania obrazov a videa, vykresľovanie 2D a 3D modelov, animácií a celkovú kontrolu interakcií medzi používateľmi inštalácie a virtuálnymi svetmi.

2.1 LIRKIS CAVE

Laboratórium LIRKIS [6], Technickej Univerzity v Košiciach, je vybavené virtuálnou realitou typu CAVE. Najväčšou výhodou LIRKIS CAVE je jej skladnosť a prenosnosť. Oceľový rám je možné rozobrať a prepraviť na iné miesto, vďaka čomu sa stala unikátom na Slovensku. Ďalšími významnými vlastnosťami sú vysoká kvalita obrazu, pokrytie celého zorného poľa používateľa a vykreslenie obrazu, ktorý reaguje na pozíciu v reálnom čase.



Obr. 2.1: CAVE systém v laboratóriu virtuálnej reality LIRKIS [6]

Navonok CAVE pozostáva z oceľového rámu o veľkosti 2,5m x 2,5m x 3m a dvadsiatich 55"LCD obrazoviek umiestnených do polkruhu. Každá obrazovka ponúka stereoskopický Full HD výstup. Dvanásť obrazoviek je umiestnených do polkruhu, šesť na hornej strane a šesť na dolnej strane CAVE(obr. 2.1). LIRIS CAVE

systém pozostáva zo siedmych počítačov, jeden hlavný (master) a šesť pomocných (slave). Master spracováva dáta a posiela informáciu slave-om a poskytuje informácie o aktuálnej 3D scéne. Jeden slave vykresľuje scénu na tri až štyri obrazovky. Pre poskytnutie čo najlepšieho grafického výstupu obsahuje každý slave NVIDIA Quadro grafickú kartu. Vykresľovanie jednotlivých scén je synchronizované skrze hlavný počítač, pomocné počítače medzi sebou nekomunikujú. Používateľ môže komunikovať s jednotlivými pomocnými počítačmi skrze hlavný. Na zaznamenanie polohy používateľa sa využíva systém snímania pohybu OptiTrack. Tento systém sa skladá z ôsmych kamier "OptiTrack Flex 13", sedem kamier je umiestnených na horných bočných stranách monitorov a jedna sa nachádza za používateľom, pripevnená o oceľovú konštrukciu. Okrem nich, obsahuje systém aj niekoľko sledovacích značiek umiestnených na okuliaroch. Hlavný používateľ s okuliarmi a používatelia najbližšie pri hlavnému používateľovi majú najrealistickejší zážitok v LIRKIS CAVE. Polohu hlavného používateľa počíta aplikácia Motive. Na pohyb v jednotlivých scénach sa využíva joystick, klávesnica, gamepad či počítačová myš.

Na vykresľovanie scén sa používalo jadro SuperEngine, ale z dôvodu zastaranosti bolo toto jadro nahradené novším hlavným jadrom Unreal Engine 4 (UE4). Vďaka diplomovej práci Matúša Gabrišku [1] sa podarilo úspešne implementovať UE4 do LIRKIS CAVE. V práci [2] sa Dominik Tóth venoval správnemu prenosu údajov z Optitacku do UE4 cez softvér Motive a implementácii projekčnej matice pre projekciu mimo os (off-axis perspective projection), pre dokonalejšie splynutie používateľa s VR.

2.1.1 OptiTrack

V systéme optického snímania pohybu (OptiTrack) sa na cieľovom zväzku snímania nainštalujú viaceré synchronizované kamery a z každej kamery sa zaznamenávajú 2D snímky [7] . Následne sa vypočítajú 2D pozície a údaje o prekrývajúcich sa polohách sa porovnávajú s výpočtom 3D pozícií prostredníctvom triangulácie. Systém na zachytávanie pohybu získava údaje prostredníctvom detekcie vyžarovaného alebo odrazeného svetla. Preto sa odporúča minimalizovať okolité osvetlenie, aby sa znížili rušivé vplyvy, ktoré zahŕňajú slnečné svetlo a vonkajšie zdroje osvetlenia alebo odrazu.

Pre väčšinu aplikácií sa spracovávajú 3D dáta v reálnom čase za celkovú systémovú snímkovaciu frekvenciu. Pre všetky aplikácie je možné zaznamenané dáta

spracovať pre všetky zachytené snímky, bez ohľadu na frekvenciu snímok.

2.1.2 Motive

Motive je softvérová platforma určená na ovládanie systémov snímania pohybu pre rôzne aplikácie určené na sledovanie [8]. Motive nielenže umožňuje používateľovi kalibrovať a konfigurovať systém, ale poskytuje aj rozhrania na zachytávanie a spracovanie údajov 3D. Motive získava 3D informácie prostredníctvom rekonštrukcie, čo je proces zostavovania viacerých 2D obrázkov značiek na získanie 3D súradníc. S využitím 3D súradníc zo sledovaných značiek Motive dokáže získať údaje o 6 stupňoch voľnosti (3D pozícia a orientácia) pre viaceré rigidné telesá (ang. rigid bodies), skelety a umožniť sledovanie zložitých pohybov v 3D priestore. Správa súborov Motive je sústredená na súbor *Take* (TAK). Súbor TAK je záznam snímania jediným pohybom, ktorý obsahuje všetky informácie potrebné na opätovné vytvorenie celého záberu zo súboru, vrátane kalibrácie fotoaparátu, údajov 2D kamery, rekonštruovaných a označených 3D údajov, úprav údajov, vyriešených údajov o uhloch kĺbov, modelov sledovania a akékoľvek ďalšie údaje o zariadení (zvuk, silová doska atď.). Súbor Motive *Take* je úplne samostatný záznam snímania pohybu a môže byť otvorený inou verziou Motive na inom systéme. Softvérové konfigurácie sa ukladajú do súborov Motive (*.motive). V profile aplikácie Motive sa ukladajú a uchovávajú všetky konfigurácie súvisiace s aplikáciou, zoznamy aktív a načítané priečinky relácií. Profily môžeme exportovať a importovať tak, aby sa pri každom spustení motívu ľahko udržali rovnaké konfigurácie softvéru.

2.2 CAVE vo svete

Prvý CAVE system bol skonštruovaný v 90-tých rokoch a dnes vo svete funguje mnoho týchto systémov VR. Aj keď nie všetky pripomínajú na pohľad LIRKIS CAVE, funkcionalitou sa veľmi nelíšia.

2.2.1 Prvá virtuálno-realitná jaskyňa

Ako už bolo vyššie spomenuté, prvá CAVE bola odprezentovaná roku 1992 Carolinou Cruz-Neira, Danielom J. Sand, Thomasom A. DeFanti [9] na konfe-

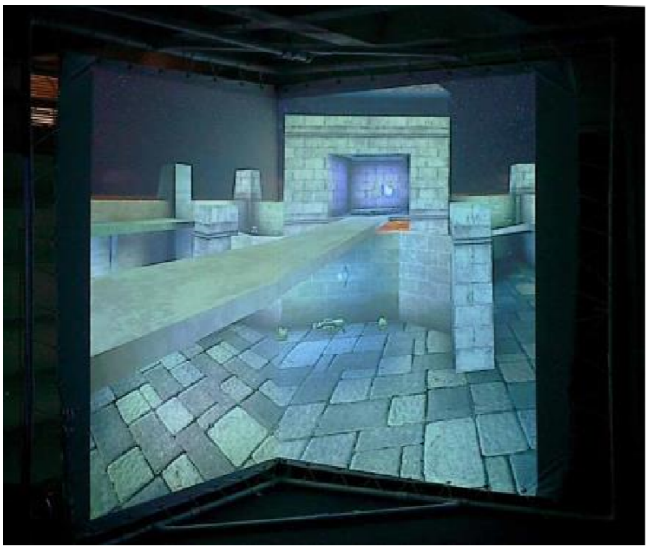
rencii SIGGRAPH (obr. 2.2). Konštrukcia tohto VR systému je vyrobená z nemagnetickej ocele, jej veľkosť je 10 x 10 x 10m a sníma hlavu a ruky pomocou elektromagnetických snímačov Polhemus alebo Ascension. Projektory vytvárajú celofarebnú plochu v rozlíšení 1280 x 512 pri obnovovacej frekvencii 120Hz. Počítačom ovládaný zvuk poskytuje sonifikačné vlastnosti viacerým reproduktorom, pre realistickejší zážitok. Stereografické LCD okuliare sa používajú na oddelenie alternatívnych polí oči.

Obr. 2.2: CAVE prezentovaná v roku 1992 na konferencii SIGGRAPH [9]

Pôvodne bola jaskyňa prispôsobená len na prípad, že používateľ sa pozeral do stredu obrazovky (on-axis perspective projection), avšak neskôr bola implementovaná pre mimo osové zobrazovanie, ktoré umožnilo používateľovi prirodzenejší pohyb v priestore. Zobrazenie mimo os umožňuje používateľovi pohybovať sa v jaskyni, keďže obraz vytvorený v CAVE sa mení vzhľadom na jeho polohu.

2.2.2 V-CAVE

Existujú aj obmedzené verzie systémov CAVE s menšou stenou alebo dokonca prenosné. Jednou z nich je aj V-CAVE [10], tvorená dvoma stenami, ktorých vzájomná poloha pripomína písmeno V. Dva digitálne projektory ukazujú na roh miestnosti, čím sa vyhýbajú požiadavkám vyhradených obrazoviek (obr. 2.3). Tento systém je tiež založený na používaní herného jadra, ktorý ponúka kvalitnú grafiku na počítačoch.



Obr. 2.3: V-CAVE projekcia na dvoch plátnach [10]

2.2.3 NexCAVE

NexCAVE spoločnosti Calit2 [11] má 10 panelov (3 x 3 pole s pridaným displejom na obrazovke, nachádza sa na spodnej strane stredného stĺpca) a je prispôsobená tak, aby umožňovala ľahkú prepravu a mohla byť použitá na demonštráciu výskumov na konferenciách (obr. 2.4). Keďže zariadenie funguje dobre aj pri osvetlení okolia a je voľne stojaca, inštalácia je veľmi zjednodušená. Samozrejme, na akékoľvek komplexné zobrazenie, ako napríklad v múzeu či neohraničenom verejnom priestore, by bola potrebná ďalšia zabudovaná ochrana pred manipuláciou.

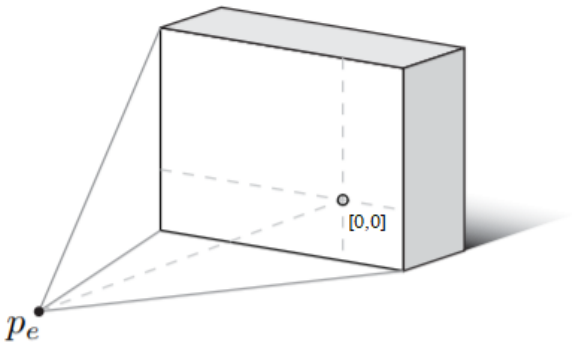
Obr. 2.4: NexCAVE zobrazujúca stereo 360 ° skenovanie [11]

Dlaždice musia byť usporiadané tak, aby umožňovali v okolí osi(90 ° kolmo

na obrazovky) zobrazovať čo najviac. Displej JVC Xpol má široký horizontálny mimo-osový stereoskopický uhol pohľadu približne 140 °, ale vertikálny uhol pohľadu iba 20 °.

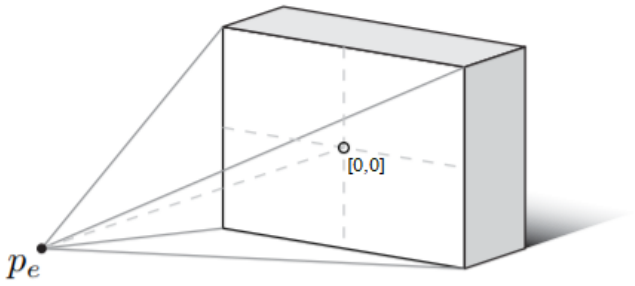
3 Zobrazenie mimo os

Zobrazenie mimo osi alebo off-axis perspective projection je zobrazenie pre ľudské oko najprirodzenejšie. Zobrazenie mimo os (obr. 3.1) je porovnateľne s pohľadom z okna v reálnom živote. Ak sa pozeráme na objekt z boku, dokážeme vidieť objekty, ktoré sa nachádzajú za objektom, tak, že sú umiestnené v strede nášho zorného poľa, pričom pri pohľade zo stredu tieto objekty nevidíme. Stred projekcie je v tomto prípade miesto na ktoré sa pozeráme. Ak používateľ zmení pozíciu očí voči projekčnej rovine, referenčný bod sa zmení tak aby odpovedal jeho pozícii v sústave.



Obr. 3.1: Zobrazenie mimo os, p_e je pozícia očí voči projekčnej rovine, referenčný bod [0,0] sa nachádza mimo stredu obrazovky [12]

Na rozdiel od off-axis zobrazenia, pri on-axis zobrazení (obr. 3.2) vidíme stále ten istý obraz v strede nášho zorného poľa. Tento objekt vidíme aj v prípade, že sa pohneme z miesta či zmeníme uhol oči, keďže je stred projekcie nastavený stále ako stred obrazovky.



Obr. 3.2: Zobrazenie na os, p_e je pozícia očí voči projekčnej rovine, referenčný bod [0,0] sa nachádza v strede obrazovky [12]

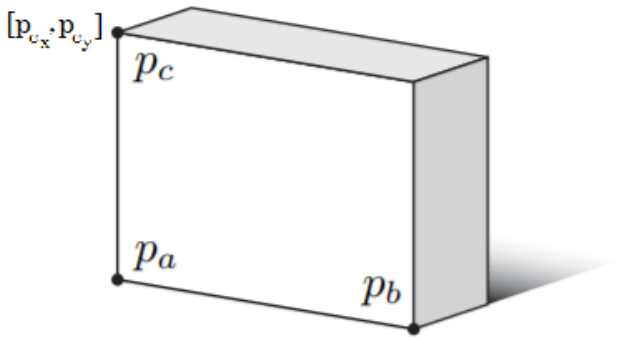
V CAVE systémoch je zobrazenie mimo os mimoriadne dôležité pre dosiahnutie realistickosti obrazu v priestore a voľného pohybu používateľa v priestore.

3.1 Výpočet projekčnej matice pre zobrazenie mimo os

V riešení od Roberta Kooima [12], v ktorom popisuje teóriu výpočtu zobrazenia pre virtuálnu realitu a implementáciu v jazyku C za použitia OpenGL je pre výpočet matice mimo os dôležité najskôr určiť si perspektívnu projekciu. Tá sa určuje samostatne pre každý pár obrazovky a očí, preto musíme zobrať do úvahy jednu obrazovku sledovanú pravé jedným okom.

3.1.1 Matica perspektívnej projekcie

Najdôležitejšie body, ktoré potrebujeme poznať sú hrany obrazovky. Z týchto bodov vieme vypočítať veľkosť obrazovky, pomer strán ale aj orientáciu v priestore. Na obrázku 3.3 vidíme označenie hrán pričom p_c je ľavý horný roh, p_a je ľavý dolný roh a p_b je pravý dolný roh.

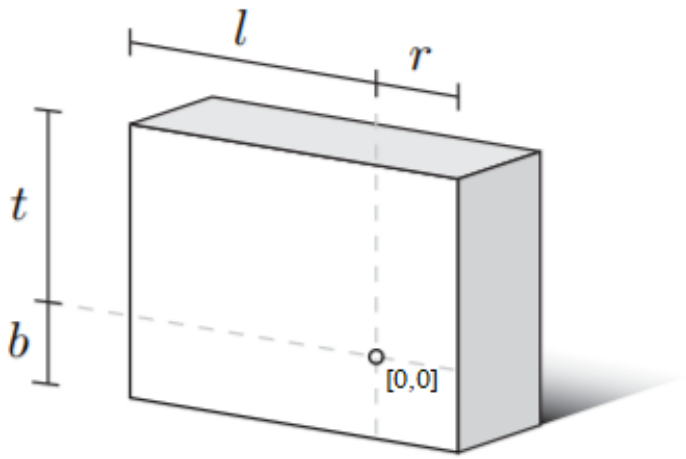


Obr. 3.3: Označenie hrán na projekčnej rovine [12]

Vďaka týmto bodom si vyrátame ortonormálnu bázu obrazovky, podpriestor tohto vektorového priestoru, vytvorenú pomocou vektorov v_r, vektor smerujúci vpravo a v_u , vektor smerujúci nahor. Menovateľ je v tomto prípade dĺžka vektora. Normálový vektor v_n musí byť záporný skalárny súčin vektorov v_r a v_u, kvôli orientácii jadra Unreal, ktoré je vľavo orientované (vpravo orientovaných systémoch je tento vektor kladný) [13] :

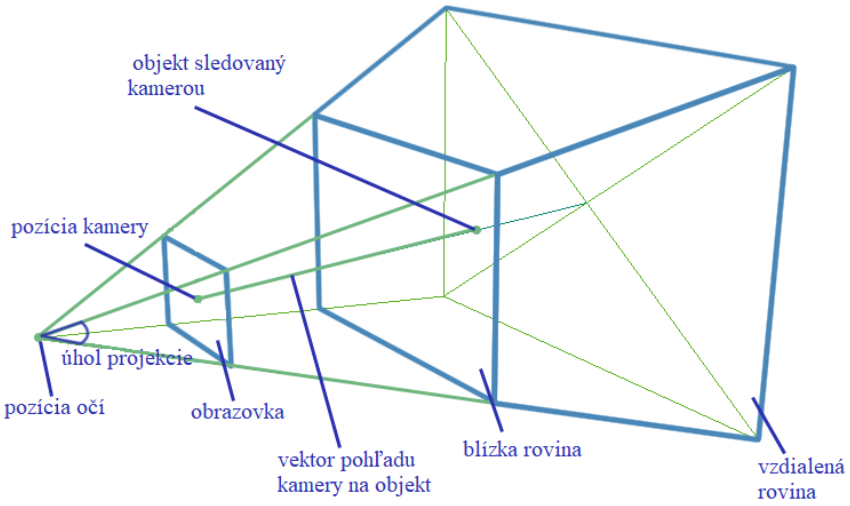
$$v_r = \frac{p_b - p_a}{\|p_b - p_a\|} \quad v_u = \frac{p_c - p_a}{\|p_c - p_a\|} \quad v_n = -\frac{v_r \times v_u}{\|v_r \times v_u\|} \tag{3.1}$$

Keď máme vektory všeobecnej perspektívnej projekcie, môžeme prejsť na výpočet projekčnej matice pre zobrazenie mimo os. Na obrázku 3.4 vidíme vzdialenosti hrán obrazovky od referenčného bodu, tieto vzdialenosti pomenujeme *b*, *t*, *l*, *r*. Každý z týchto parametrov môže byť kladné aj záporne číslo. V príklade na obrázku 3.4 je vzdialenosť medzi *l* a *b* záporná, keďže v mimo-osovom zobrazení sa smer nahor a doľava od referenčného bodu berie ako záporný smer (v on-axis projekcii stále platí $|r| = |l|$ a $|t| = |b|$).



Obr. 3.4: Znázornenie vzdialeností voči referenčného bodu [12]

Na výpočet vzdialenosti hrán budeme potrebovať aj vektory pozície očí od rohov obrazovky v_a, v_b, v_c. Tieto vektory získame ako rozdiely bodov jednotlivých hrán p_a, p_b, p_c a bodu pozície očí p_e. Pre maticu perspektívnej projekcie P sú potrebné aj parametre f, n, a, d, pričom f a n sú vzdialenosti od blízkej a vzdialenej roviny. Blízka rovina a vzdialená rovina sú ohraničenia pre prierez kužeľa smeru pozorovania (obr. 3.5). Ak je objekt bližšie ku kamere, ale zároveň sa nachádza pred blízkou rovinou alebo za vzdialenou rovinou, nebude zobrazený. Niekedy je vzdialená rovina umiestnená nekonečne ďaleko od kamery, takže všetky objekty vo vnútri zrezaného kužeľa sú vykreslené bez ohľadu na vzdialenosť od kamery. Tieto objekty sa budú líšiť veľkosťou, ktorá je nepriamo úmerná vzdialenosti od kamery.



Obr. 3.5: Poloha blízkej a vzdialenej roviny voči pozícii očí

Parameter d je najkratšou vzdialenosťou od oka k rovine obrazovky. Definujeme ho ako zápor skalárneho súčinu najkratšieho vektora pozície očí od rohov obrazovky, v tomto prípade vektor v_a, a normálového vektora v_n. Tento výraz je záporný z dôvodu opačného smeru vektorov. Parameter teda vyrátame pomocou vzťahu:

$$d = -(v_n \cdot v_a) \tag{3.2}$$

Keďže už poznáme všetky hodnoty, môžeme ich dosadiť do vzorca pre výpočet vzdialenosti referenčných bodov od hrán obrazovky:

$$l = (v_r \cdot v_a)n/d \quad r = (v_r \cdot v_b)n/d \quad b = (v_u \cdot v_a)n/d \quad t = (v_u \cdot v_c)n/d \tag{3.3}$$

Vzniknuté hodnoty ďalej dosadíme do matice perspektívnej projekcie P:

$$P = \begin{bmatrix} \frac{2n}{r-t} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix} \tag{3.4}$$

3.1.2 Výpočet transformačnej matice

Po vytvorení matice perspektívnej projekcie P, dokážeme vytvoriť len obraz v rovine XY. Pre vytvorenie plnej projekcie mimo os je potrebné obrátiť obrazovku z

roviny *XY* a správne ju umiestniť vzhľadom na používateľa.

Vytvoríme maticu pre lokálne súradnice M^T. Pomocou jednotkovej matice a základných vektorov v_r, v_u, v_n uložených do riadkov matice M^T, transformujeme maticu M^T do roviny obrazovky.

$$M^T = \begin{bmatrix} v_{rx} & v_{ry} & v_{rz} & 0 \\ v_{ux} & v_{uy} & v_{uz} & 0 \\ v_{nx} & v_{ny} & v_{nz} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.5}$$

Posledná matica, ktorú potrebujeme je transformačná matica *T*. Je to jednotková matica v ktorej sú v poslednom stĺpci záporné súradnice pozície očí p_e.

$$T = \begin{bmatrix} 1 & 0 & 0 & -p_{ex} \\ 0 & 1 & 0 & -p_{ey} \\ 0 & 0 & 1 & -p_{ez} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.6}$$

Keďže sme definovali všetky potrebné matice, môžeme zhotoviť konečnú maticu *P'* pre projekciu mimo os pomocou vzorca:

$$P' = PM^T T \tag{3.7}$$

3.2 Momentálna implementácia v LIRKIS CAVE

V LIRKIS CAVE bola implementácia herného jadra Unreal Engine, ako aj mimo-osového zobrazenia dokončená minulý rok. Riešenie zobrazenia mimo os sa ale neukázalo ako ideálne, vykresľovanie nie je presné. Projekcia mimo os bola zhotovená na základe riešenia od Roberta Kooima [12], doplnená o orientáciu pre Unreal Engine, ktoré je vľavo orientované.

3.2.1 Možné chybné implementácie v LIRKIS CAVE

Z pohľadu používateľa stredná obrazovka (server) je správne vykresľovaná. Na základe toho môžeme zhotoviť hypotézu, že chyba je v momentálnej implementácii mimo-osového zobrazenia do UE4 a nie vo vstupných parametroch.

Možnú chybu sme identifikovali vo funkcii *FrustumMatrix*(), ktorá je definovaná v súbore *TestLocalPlayer.h*. V tejto knižnici sa vytvára matica *P*. Matica je v

typu *FMatrix*, pre tento typ prvý index predstavuje riadok a druhý index stĺpec matice, implementácia je tu ale naopak, prvý index označuje stĺpec a druhy index riadok. Túto hypotézu je nutné otestovať vo virtuálno-realitnej jaskyni laboratória LIRKIS.
Problémy sa vyskytujú v konečnom násobení matíc. Násobenie dvoch matíc nie je komutatívna operácia $(A \cdot B \neq B \cdot A)$. Vo funkcii *GenerateOffAxisMatrix*() v knižnici *TestLocalPlayer.h* pri násobení súčinu matíc PM^T s T je opačné poradie matíc, namiesto $PM^T \cdot T$ je implementácia $T \cdot PM^T$. Keďže na poradí matíc pri súčine záleží aj toto môže byť problém vo vykresľovaní obrazov.
zobrazenia do UE4 a nie vo vstupných parametroch.

Momentálne spúšťanie Unreal klienta pre monitor spočíva v pridelení jedného klienta pre jeden monitor v klastri. V odkaze klienta pre daný monitor sa nastavia špecifické parametre pre daný monitor, ktorými sú súradnice *x*, *y* pre ľavý dolný bod monitora a pre pravý horný roh.

3.2.2 Možná optimalizácia LIRKIS CAVE

V rámci optimalizácie tohto systému CAVE by bolo ideálnejšie nastaviť jedného klienta pre jeden klaster monitorov, čím by sme sa vyhli problémom s opakovaným vykresľovaným hráča v hre pre každú obrazovku, keďže sa do úvahy neberie či je klient vytvorený ako pohľad kamery, alebo ako aktuálny výsek v hre. Ako možné riešenie by sme navrhovali zanedbať minimálny uhol natočenia medzi obrazovkami alebo v hernom jadre implementovať, aby po jednej šírke obrazovky natočilo zobrazenie scény o uhol medzi aktuálnou obrazovkou a nasledujúcou. Ďalšou možnosťou by bolo vytvoriť v klientovi zobrazovanie podľa výseku klienta na strednej obrazovke, resp. hlavnom pre daný klaster. Toto nie je optimálne pre LIRKIS CAVE, keďže má veľké množstvo obrazoviek rôzne natočených v priestore a tým pádom by sme boli nútený prepojiť možnosť pridávania parametrov v odkaze a Unreal Editor, aby sme mohli priamo cez odkaz upravovať natočenie a presný výsek klienta.

Táto optimalizácia zapínania celej CAVE spoločne s konfiguračným súborom by vedela uľahčiť manipuláciu s CAVE.

3.2.3 Konfiguračný súbor pre LIRKIS CAVE

Jedným z výsledkov tejto práce bude aj konfiguračný súbor, ktorý bude slúžiť na jednoduché nastavenie jednotlivých parametrov pre klientov aplikácie. Tento klient by mal mať dopredu uložené súradnice pravého horného rohu a ľavého dolného rohu každej obrazovky (p_b a p_c) a takisto pomenované podľa klastrov v ktorých sa nachádzajú. Súbor *.ini* v priečinku *Config* dokáže meniť nastavenia hry v teoreticky reálnom čase, ale nám prakticky postačuje zmena nastavené pri zapnutí, podľa preddefinovaného konfiguračného súboru *.xml* [14]. Pre tento súbor sa musí implementovať funkcionalita do *.ini* súboru, ktorá dokáže čítať a prepisovať *.xml* súbor s používateľom zvolenou konfiguráciou aj pri spustení zabaleného projektu [15].

Funkcie v *.ini* súbore môžeme volať aj cez skriptovací systém Blueprints [16]. Tento vizuálny kompletný herný skriptovací systém je postavený na koncepte používania rozhrania založeného na uzloch na vytvorenie elementov hry v rámci aplikácie Unreal Editor.

Blueprint-y môžu riadiť udalosti, ovládať interné skriptované správanie pre postavy v hre a dokonca byť použité na ovládanie zložitých animácií vo vysoko realistických systémoch herných znakov. Blueprint-y priamo nadväzujú na *.ini* súbory keďže vedia čítať a prepájať funkcionality. Tým sa developerom umožňuje upravovať funkcionalitu konfigurovateľných súborov a ich nadväznosť aj bez lepšej znalosti jazyka C++, v ktorom je Unreal jadro implementované, či rozsiahlej znalosti knižníc a tried, ktoré využíva.

3.2.4 Doplnenie konfigurácie pomocou CaveConsole

Tento konfiguračný súbor už bol vytvorený a upravený M. Gabriškom v jeho diplomovej práci [1]. Vytvoril taktiež používateľské prostredie na jednoduchšiu modifikáciu parametrov CaveConsole. Táto aplikácia mení nastavenia klienta na jednotlivých počítačoch v klastri, v reálnom čase, pomocou prepisovania prednastavených nastavení zadaných v UE4 Editore. Komunikuje so všetkými počítačmi z klastra a je založená na protokole WebSockets a TCP. Vďaka TCP zisťuje či soket získal dáta a zároveň sa kontroluje stav soketu. CaveConsole si sama vyžiada momentálne nastavenia v CaveTUKE skrze CaveService a vytvorí sa JSON objekt, konkrétne clients.*json*, ktorý obsahuje jednotlivé *id* reprezentujúce jednotlivé počí-

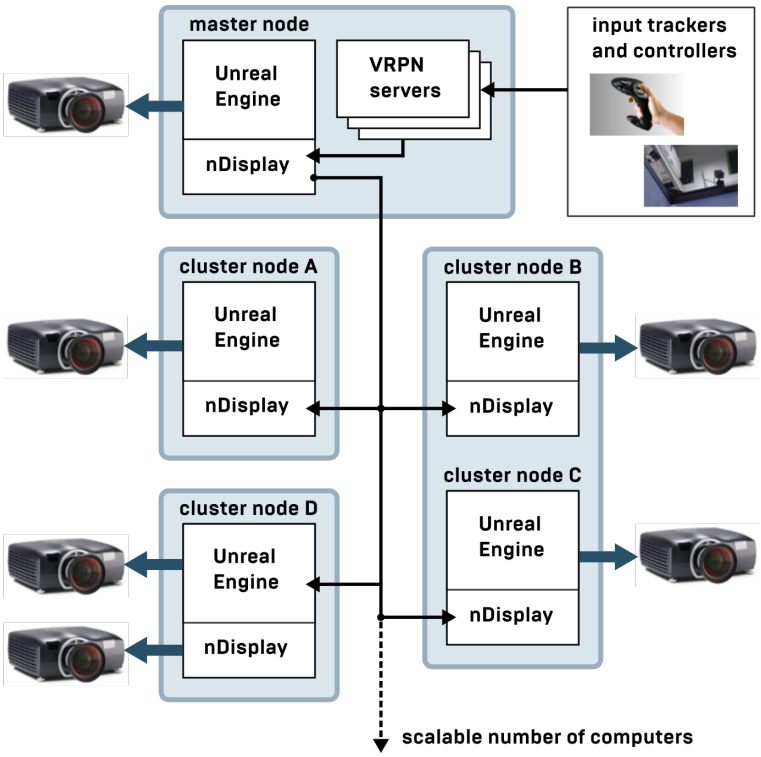
tače v klastroch. V tomto súbore sa nachádzajú informácie pre server, pre klientov a súradnice X a Y (pre spustenie okna klienta). Tieto dáta sú poslane pre CaveService, ktorý deserializuje JSON a spustí inštanciu so správnymi parametrami. CaveService je spustená od spustenia operačného systému jednotlivých počítačov a snaží sa naviazať spojenie s CaveConsole. Parametre pravého horného rohu a ľavého dolného rohu každej obrazovky (p_b a p_c) nie sú zahrnuté v tejto implementácii.

Momentálne sa parametre pre mimo-osové zobrazenie udávajú ako argumenty pri spustení klienta pre aktuálnu obrazovku. V týchto parametroch sú najdôležitejšie pravé *-llx* (ľavý dolný roh, súradnica x), *-lly* (ľavý dolný roh, súradnica y), *-rux* (pravý horný roh, súradnica x), *-ruy* (pravý horný roh, súradnica y). Najideálnejšie hodnoty týchto parametrov pre usporiadanie LIRKIS CAVE monitorov uviedol Dominik Tóth v jeho bakalárskej práci [2]. Tieto parametre sa ďalej spracujú v *UOffAxisLocalPlayer.cpp*, kam sa dostavajú z hlavičkového súboru pre *OffAxisLocalPlayer*. Načítavanie do zdrojového kódu funguje cez Blueprint-y. Parametre pre spracovanie projekčnej matice sa nachádzajú v Blueprint-e s názvom *OffAxis*, ktoré implementoval Dominik Tóth.

3.3 Experimentálna implementácia CAVE podľa Epic games

S novou verziou jadra 4.21 prišla na trh aj nová experimentálna implementácia softvéru pre CAVE systémy. Táto implementácia, vytvorená Epic Games, je určená aj pre nami použitý systém Lirkis CAVE. Hoci je tento softvér dostupný niekoľko rokov, nebol určený pre verejné publikum, hlavne z dôvodov chybovosti pri použití OS Windows.

Hlavné problémy boli komunikácia ako aj zle vykresľovanie, menenie farieb alebo prepisovanie projektu. Táto nová implementácia sa zakladá na predpoklade, že CAVE je už nastavená a existuje master-slave komunikácia. Nastavenie pre CAVE sa nastavuje už pri vzniku projektu a to pod projektom typu nDisplay [17]. K tomuto projektu nDisplay sa po vytvorení *.exe* súboru vytvoria aj spustiteľné súbory slúžiace na komunikáciu master/slave, konkrétne launcher pre master a listener pre slave počítač, ako aj niekoľko ukážkových súborov slúžiacich na konfiguráciu.



Obr. 3.6: Architektúra nDisplay navrhnutá Epic Games [17]

Ako si môžeme všimnúť na obr. 3.6, master zapne spustiteľný súbor *.exe* skrze launcher aplikáciu a podľa zapnutých listenerov s IP adresami na jednotlivých slavoch spustí súbor. Tento je prispôsobený podľa konfigurácie pre jednotlivé monitory. Vstupné zariadenia ako napríklad snímače pohybu, klávesnice alebo iné kontrolery sa v tejto verzii dajú pripojiť jedine skrze VRPN server.

3.3.1 VRPN server

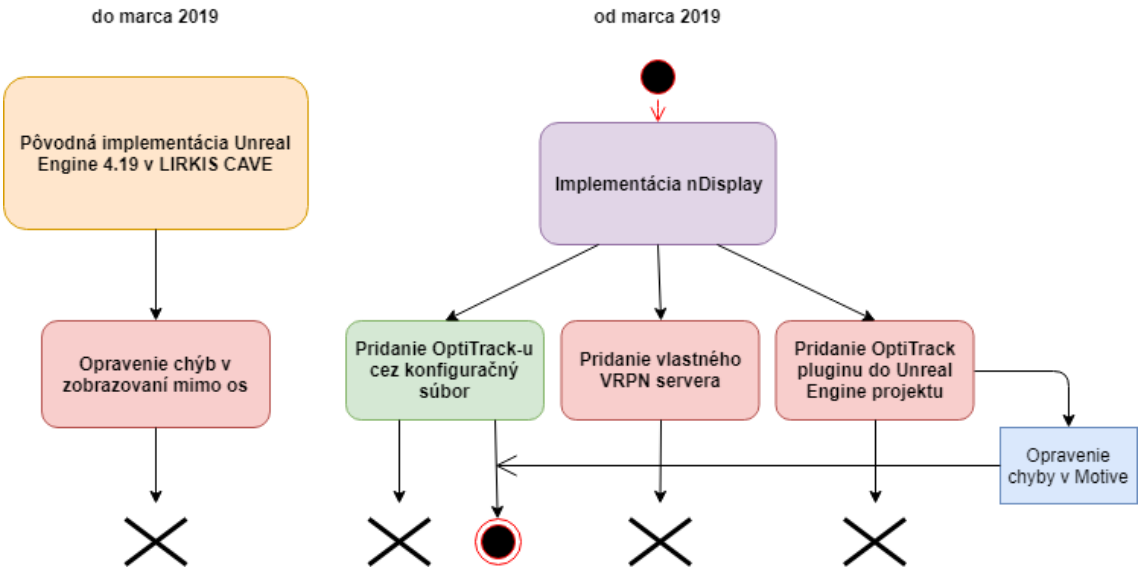
Periférna sieť virtuálnej reality alebo VRPN (v angličtine Virtual-Reality peripheral network) je niekoľko desiatok tried a knižníc poskytujúcich jednotné rozhranie pre vstupné zariadenia, ako sú napríklad pohybové sledovače alebo ovládače joysticku. VRPN pozostáva z programovacích rozhraní pre klientsku aplikáciu, ako aj hardvérových ovládačov a serverovej aplikácie, ktorá komunikuje s hardvérovými zariadeniami. Klientske rozhrania sú napísané v jazyku C++, ale boli zabalené v jazykoch Python a Java.

Vo verzii 4.22 Unreal jadra bude možné používať aj takzvané klaster udalosti (v ang. cluster events) slúžiace na prepojenie pluginov priamo v projekte do ce-

lého klastra. Vďaka tomuto by bolo možné sa teoreticky vyhnúť používaniu VRPN serveru, keďže nie všetky hardvérové zariadenia podporujú tento spôsob komunikácie.

4 Návrh a implementácia riešenia

Z analýzy uvedenej v časti 3.2 je jasné, že máme dve možnosti vyhotovenia riešenia. Jednou z možností implementovania riešenia je pridanie implementácie pre modifikáciu parametrov zobrazovania mimo os do CaveConsole, druhou možnosťou je využitie riešenia pre CAVE systémy od Epic Games.



Obr. 4.1: Časový diagram implementácie riešenia

V diagrame (obr. 4.1) si môžeme všimnúť cestu, ktorá viedla k implementácii nového riešenia. Diagram opisuje v akom poradí prebiehala implementácia a následne testovanie návrhov, ako aj ich výsledný stav. Červené entity označujú riešenia, ktoré neboli úspešné, znak **X** označuje neúspešný výsledok testovania. Plný čierny krúžok označuje začiatok implementácie a čierny krúžok s červeným prstencom označuje úspešný výsledok po testovaní implementácie.

Experimentálne sme zistili, že vyhotovenie riešenia nebude až tak ľahké ako sa mohlo zdať. Vykonané experimenty ako aj ich návrhy a výsledky sú rozpísané

v nasledujúcich častiach tejto kapitoly. Podľa časového horizontu vidíme, že prvým možným riešením bolo pokračovanie v momentálnej implementácii. Riešenie nebolo použiteľné aj napriek opraveniu pôvodného riešenia, táto implementácia a výsledky sú podrobne opísané v časti 4.1. Druhým riešením bolo pridanie nDisplay, tým sme nahradili implementáciu M. Gabrišku [1], viac o tomto riešení nájdeme v časti 4.2. Vďaka tomu sme pripravili vhodné podmienky na pridanie zariadenia na sledovania pohybu, v našom prípade OptiTrack (časť 4.3). V nasledujúcich častiach sú podrobne opísané všetky experimentálne riešenia aj koncové riešenie.

4.1 Pokračovanie v momentálnej implementácii

Opravenie zdrojových kódov D. Tótha [2] pre pridanie mimo-osového zobrazovania bolo prvým riešením. Tieto chyby sú opísané v časti 3.2.1.

4.1.1 Návrh opravenia momentálnej implementácie

Pridaním súradníc *-llx, -lly, -rux, -ruy* do JSON súboru a následným načítaním do *settings.ini* pomocou aplikácie CaveConsole a CaveService by bolo možným riešením v prípade zjednodušenia nastavovania parametrov pre LIRKIS CAVE. Tým pádom by bolo možné odstrániť parametre v Blueprint-e pre mimo-osové zobrazenie, pretože už nebude potreba naďalej modifikovať argumenty pri sputení *CaveTuke.exe* . Vytvorením novej záložky v CaveConsole s názvom *Off-axis* a jej prepojením s inštanciami nachádzajúcimi sa v *UOffAxisLocalPlayer.cpp* by bolo možné aj manuálne modifikovať súradnice pre zobrazenie mimo os v reálnom čase. Ukladanie týchto nastavení by mohlo byť realizované pomocou načítania posledných súborov v *settings.ini* alebo v prípade zálohovania viacerých nastavení pre LIRKIS CAVE, načítaním textového súboru, ktorý bude reprezentovať zálohu pre nastavenia. Tento textový súbor sa môže načítať do *clients.json* a tým sa spustiť pri prvom spustení LIRKIS CAVE. Pridaním implementácie pre zmenu parametrov pre *UOffAxisLocalPlayer.cpp* by sa zjednodušilo používateľské prostredie LIRKIS CAVE.

4.1.2 Výsledné problémy s momentálnou implementáciou

Tento návrh riešenia, ale nevyrieši problém so zobrazovaním mimo os. Zobrazovanie mimo os je totiž problém verzie 4.19. Táto verzia nedokáže prepísať *viewportMatrix*. Táto matica je hlavnou maticou zobrazovania pohľadu používateľa v hre. Implementácia Dominika Tótha [2] funguje len za predpokladu, že stred obrazovky sú súradnice [0, 0]. Experimentálne sme zistili, že matica sa prepíše podľa jeho implementácie, ale keďže pri vykresľovaní sú vo vlastnostiach spustiteľného súboru podané nové súradnice pre rohy obrazovky, tak sa nahradí matica pre mimo-osové zobrazenie základnou maticou. Tento problém je opravený v najnovšej verzii jadra 4.21, ale zvýšenie verzie jadra v tomto prípade nie je možné z dôvodu implementácie M. Gabrišku [1]. Keďže riešenie M. Gabrišku išlo veľmi hlboko do kódu jadra a prepísal nejednu z hlavných metód herného jadra, zvýšenie verzie môže spôsobiť stratu týchto zmien a zahodiť celú pôvodnú implementáciu.

Aj keď riešenie M. Gabrišku riešilo komunikáciu LIRKIS CAVE veľmi dobre, zahodenie tejto implementácie, ako aj implementácie Tótha a využitie možných riešení priamo od Epic Games spolu so zvýšením verzie jadra dokáže odstrániť nedostatky jednoduchšie a efektívnejšie. Toto riešenie má rovnakú architektúru ako momentálna implementácia LIRKIS CAVE. Pre použitie nDisplay je potrebné realizovať niekoľko zmien.

4.2 Využitie nDisplay

Táto implementácia je najvhodnejšia, keďže dokáže nahradiť momentálnu TCP/IP komunikáciu v klastri a rieši problémy, ktoré nebolo možné opraviť vo verzii 4.19, napríklad generovanie iba jedného hráča, nastavovanie rohov obrazovky a zorného poľa, neustále zapínanie projektu na každom slave počítači osobitne a nastavovanie mimo-osového zobrazenia aj pre CAVE systém. Keďže riešenie nDisplay nutne nepotrebuje žiadne zasahovanie do hlavných metód jadra, zvýšenie verzie v budúcnosti kvôli väčšej využiteľnosti LIRKIS CAVE bude bezproblémové.

4.2.1 Návrh nastavenia pre implementáciu nDisplay

Pre použitie nDisplay je potrebné zmeniť základnú konfiguráciu nDisplay na konfiguráciu vyhovujúcu LIRKIS CAVE. Unreal Engine má vlastné argumenty, ktoré

reprezentujú jednotlivé monitory. Obrazovka sa v nDisplay označuje ako *cluster_node*, pre každý *cluster_node* nastavíme vlastné id, IP adresu a viewport. Viewport v tomto prípade znamená, kde presne na obrazovke sa má zobraziť projekt. Keďže v LIRKIS CAVE jeden klaster predstavuje jednu obrazovku zloženú z troch alebo štyroch monitorov, bude možné presne vypočítať, kde sa má projekt zobraziť.

4.2.2 Implementácia nDisplay

Epic Games prišlo s experimentálnou verziou nDisplay pre Windows až vo verzii 4.21, ktorá bola uvedená na trh až v novembri 2018. Táto verzia mala stále veľa chýb, a preto ju nebolo odporúčané používať mimo testovacích účelov. My sme sa o nDisplay dozvedeli až v marci 2019, v tej dobe bolo nDisplay taktiež označované ako experimentálne, ale množstvo chýb bolo opravených a v komunite ľudí pracujúcich s Unreal Engine sa táto verzia považovala za stabilnú. Uvedenie nDisplay na trh bolo plánované spolu s vydaním verzie 4.22. Kvôli včasnému dokončeniu práce sme sa rozhodli pracovať s verziou 4.21, v ktorej boli odstránené všetky chyby ovplyvňujúce nDisplay.

4.2.2.1 Nahradenie starej implementácie

Po prechádzajúcich tvrdeniach je viac ako jasné, že bude potrebná kompletná výmena herného jadra UE4 v LIRKIS CAVE. Ako prvé je nutnosť stiahnuť zdrojové súbory objasňujúce herné jadro a nainštalovať Microsoft Visual Studio s prekladačmi potrebnými k UE4. Pomocou *GenerateProjectFiles.bat* si vygenerujeme súbor spustiteľný v Microsoft Visual Studio, pretože toto prostredie je odporúčané samotným Epic Games. Vygenerované súbory otvoríme ako projekt v IDE a spustíme nástroj Debugger. Po skompilovaní projektu sa nám spustí hlavná aplikácia UE4 určená na vytvorenie alebo editáciu projektu. Vytvoríme si projekt a vyberieme si nDisplay ako základné nastavenie. Zabalenie projektu (ang. package project) automaticky vytvorí tri spustiteľné súbory *LIRKISProject.exe* , *Launcher.exe* , *Listener.exe* a viacero prikladov konfiguračných súborov, ktoré tvoria nami modifikovateľné nastavenie jaskyne. Tieto súbory tvoria najzákladanejšiu zložku na vybudovanie CAVE systému a sú kompatibilné s akýmkoľvek zabaleným projektom vytvoreným cez UE4 nDisplay. Vďaka tomu nebude nutné v budúcnosti meniť

nastavenia nDisplay pre rôzne nové projekty.

4.2.2.2 Konfigurovateľný súbor

Keďže žiaden z príkladov konfiguračného súboru nespĺňa požiadavky LIRKIS CAVE je potrebné upraviť súbor na mieru pre LIRKIS CAVE.

Ako prvé je nutné definovať usporiadanie obrazoviek pomocou argumentu *cluster_node*. V tomto prípade berieme do úvahy, že jeden počítač má obrazovku zloženú z viacerých monitorov. Preto nastavovanie parametra *WinX* či *WinY* sa zameriava na šírku a výšku jednotlivých obrazoviek. Podľa časti nastavenia sme nakonfigurovali ako master spodný stredný monitor, podobne ako tomu bolo v pôvodnej implementácii. IP adresy sú používané podľa pôvodnej architektúry LIRKIS CAVE, *id* jednotlivých monitorov sa nastavuje z ľava doprava podľa polohy obrazovky. Vďaka týmto parametrom bude nDisplay vedieť, kde má zobraziť spustený projekt. Vďaka parametru *viewport* dokážeme presne definovať šírku a výšku spusteného projektu, v našom prípade 1080x1920. Takéto nastavenie môžeme zapísať do konfiguračného súboru napríklad takto:

```
[cluster_node] id=node_3_up addr=192.168.10.40 screen=screen_3_up viewport=vp_1080
[cluster_node] id=node_3_dn addr=192.168.10.30 screen=screen_3_dn viewport=vp_1080
[cluster_node] id=node_4_up addr=192.168.10.40 screen=screen_4_up viewport=vp_1080 Winx=1080 WinY=0
[cluster_node] id=node_4_dn addr=192.168.10.30 screen=screen_4_dn viewport=vp_1080 port_cs=7071
port_ss=7072 master=true  Winx=1080 WinY=0
[cluster_node] id=node_5_up addr=192.168.10.40 screen=screen_5_up viewport=vp_1080 Winx=2160 WinY=0
[cluster_node] id=node_5_dn addr=192.168.10.30 screen=screen_5_dn viewport=vp_1080 Winx=2160 WinY=0
```

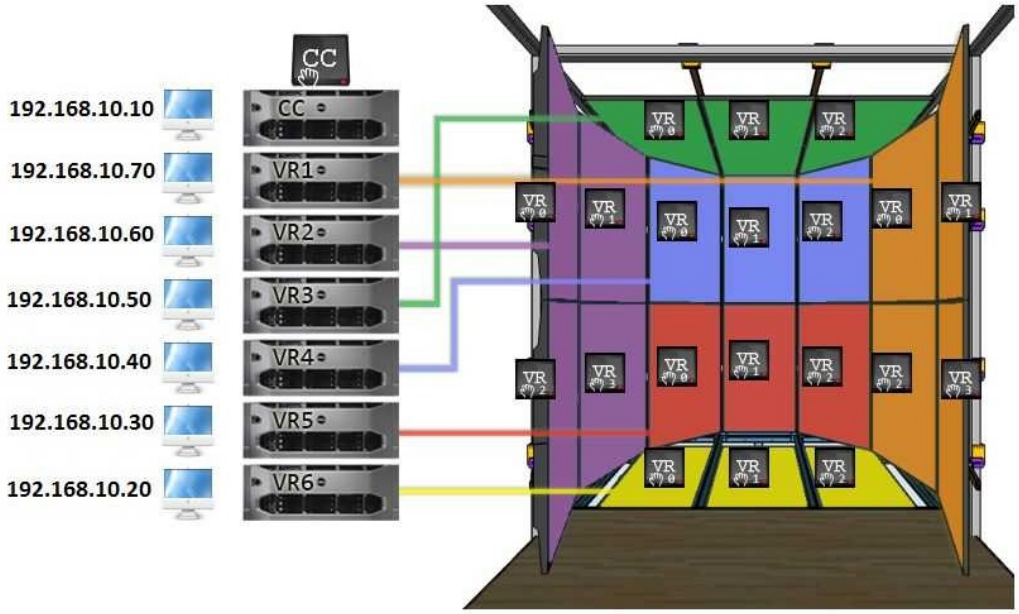
Argument *screen_node* slúži na presné vyhotovenie transformačnej matice vytvárajúcej pohyb obrazu s užívateľom. Parameter *loc* je relatívna lokácia k nadradenému komponentu (ang. parent component). Umiestnenie je relatívne do koreňového adresára projektu, ak nie je špecifikovaný žiadny rodič. Centrálnym bodom je stred obrazovky, hodnoty sú v uvádzané v metroch. V našom prípade, keďže *display_4_dn* je referencia na master, ktorý má pozíciu oproti užívateľovi, tak *display_4_up* je referencia na obrazovku nachádzajúcu sa nad mastrom. Nastavenie *loc* v prípade *display_4_up* je teda upravené o veľkosť obrazovky v súradnici *Z*, v našom prípade 1.23 metra, dĺžku predstavujúcu pozdĺžnu vzdialenosť od mastra. *Rot* je relatívna rotácia vzhľadom na materský komponent. Rovnako ako pri *loc*, centrálnym bodom je stred obrazovky a hodnoty sú v stupňoch. Pre *column_4* sme nastavili parent ako *angle_4*, pomocou neho referujeme na pôvodnú pozíciu

očí pre UE4, neskôr nahradíme tento argument súradnicami z OptiTrack-u. Pre *screen_node* zapíšeme nastavenie nasledovne:

```
[scene_node] id=angle_4      loc="X=1,Y=0,Z=0" rot="P=0,Y=0,R=0" parent=eye_level
[scene_node] id=column_4     loc="X=0,Y=0,Z=0" rot="P=0,Y=0,R=0"   parent=angle_4
[scene_node] id=display_4_up loc="X=0,Y=0,Z=1.23"  rot="P=0,Y=0,R=0" parent=column_4
[scene_node] id=display_4_dn loc="X=0,Y=0,Z=0" rot="P=0,Y=0,R=0"   parent=column_4
```

4.2.2.3 Prepojenie LIRKIS CAVE a nDisplay

Pre prepojenie LIRKIS CAVE a nDisplay musíme vytvoriť prostredie na spúšťanie projektu. V tomto kroku je potrebné poznať architektúru LIRKIS CAVE. Keďže hlavný počítač *Console_Cave* bol už predtým použitý na hlavné prepojenie SuperEngine s klastrom, môžeme predpokladať, že bude mať povolenia na modifikáciu všetkých súborov počítačov v klastri. Exprimentálne sme zistili, že tento predpoklad sa nepotvrdil, pretože povolenia sú nastavené len pre SuperEngine projekty.



Obr. 4.2: Momentálne IP adresy a názvy počítačov v klastri pre LIRKIS CAVE

Vytvoríme priečinok na pracovnej ploche pre ktorý je potrebné nastaviť povolenia na zdieľanie a modifikáciu súborov v sieti. Na nastavenie zdieľania je vhodné vytvoriť domácu skupinu v sieti. Domáca skupina je skupina počítačov v domácej sieti, ktoré môžu zdieľať súbory. Používanie domácej skupiny uľahčuje zdieľanie.

Umožňuje zdieľať obrázky, hudbu, videá a dokumenty s ostatnými používateľmi v domácej skupine. Túto skupinu môžeme chrániť heslom, ktoré môžeme kedykoľvek zmeniť. Po vytvorení skupiny pridáme všetky počítače v klastri, potrebné IP adresy a názvy nájdeme na obrázku 4.2.

Do priečinka k vlastnostiam pridáme novovytvorenú domácu skupinu. Tento postup sme zvolili hlavne z dôvodu obmedzenia prepojenia so SuperEngine, keďže v prechádzajúcej implementácii nebolo potrebné brať do úvahy modifikáciu súborov v sieti. Do priečinka pridáme všetky potrebné súbory na spustenie nDisplay pre jednotlivé počítače. Vďaka tomu bude schopné spúšťať nDisplay len cez mastra.

4.3 Pridanie OptiTrack systému

Posielanie súradníc pozície pomocou OptiTrack-u do projektu je možné tromi spôsobmi. Prvým je využitie konfiguračného súboru pre nDisplay pridaním IP adresy a názvu zariadenia na snímanie pohybu. Nanešťastie nDisplay podporuje VRPN server len s verziou 7.33 a vyššie preto nie je možné použiť Motive, pretože tento softvér momentálne používa VRPN server s verziou 7.20. Možná aktualizácia Motive softvéru môže narušiť chod SuperEngine, ktorý sa stále využíva. Druhý spôsob je využitie vlastného VRPN serveru. Pre tento spôsob je potrebné vytvoriť vlastný VRPN server, implementovať posielanie súradníc z OptiTrack-u a pripojiť tento server na mastra. Tento server by mal byť spustený iba na mastrovi. Tretím spôsobom je použitie OptiTrack pluginu pre Unreal Engine a neskoršie pridanie súradníc pomocou Blueprint-ov, tento spôsob je riskantný, keďže môže narušiť chod nDisplay.

4.3.1 Mimo-osové zobrazenie v nDisplay

Pridanie sledovacieho zariadenia nám vytvorí vhodné podmienky na následnú implementáciu mimoosového zobrazenia. Toto zobrazenie bude pomocou konfiguračného súboru pre nDisplay automaticky aplikované na všetky obrazovky.
V konfiguračnom súbore môžeme definovať hierarchiu uzlov scén (pod argumentom *scene_node*), z ktorých každá predstavuje transformáciu v 3D priestore. Konfiguračný súbor si vyžaduje pozíciu a rotáciu v 3D priestore, ako je napríklad kamera alebo premietacie plátno. Môžeme použiť jednu z týchto konfigurácií

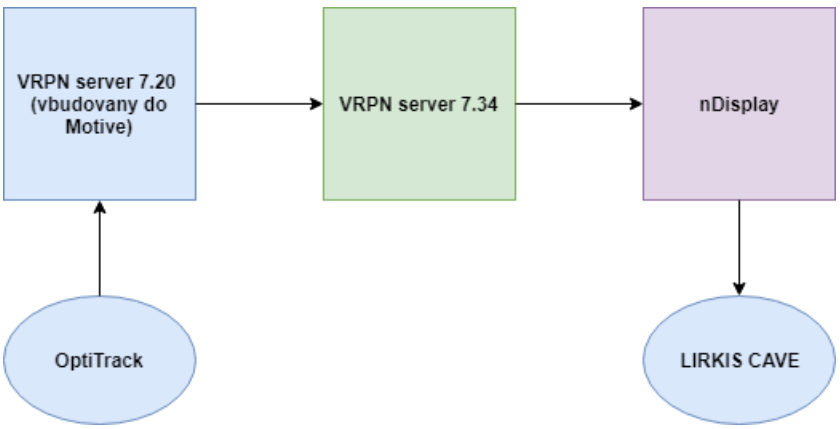
scén a nastaviť ju ako rodičovskú scénu. Toto nám pomôže definovať priestorové vzťahy medzi všetkými rôznymi komponentmi vizualizačného systému. Priestorové vzťahy a ich prepojenie na sledovanie pozície pomocou OptiTrack by malo automaticky implementovať mimo-osové zobrazenie, keďže súradnice sa budú automaticky posielať. Pozícia očí už nebude na stred obrazovky, ale bude závisieť od natočenia a uhlu pohľadu hráča, to znamená, že projekčná matica sa sama prepočíta a prispôsobí.

Experimentálne sme zistili, že pri momentálnej konfigurácii Motive pre SuperEngine alebo pôvodnej konfigurácii prispôsobenej riešeniu D. Tótha [2] nie je možné prepojiť nDisplay a Motive softvér. Predpokladáme, že chyba je zrejme na strane podpory nDisplay pre zastaralú verziu VRPN serveru v Motive aplikácii.

4.3.2 Pridanie vlastného VRPN rozhrania na odosielanie súradníc z OptiTrack-u

VRPN server je dostupný vo voľne šíriteľnej verzii. Po vytvorení dvoch VRPN serverov by sme mali byť schopný prijímať súradnice z Motive VRPN serveru do vlastného VRPN serveru s vyššou verziou a súradnice následne odoslať do nDisplay. Týmto sa vyhneme možným problémom s kompatibilitou verzií VRPN a nDisplay.

4.3.2.1 Návrh vlastného VRPN rozhrania



Obr. 4.3: Diagram návrhu riešenia posielania súradníc z OptiTrack-u do nDisplay cez VRPN server

Na diagrame (obr. 4.3) je opísaná komunikácia medzi servami a nDisplay. Modrou farbou sú vyznačené entity, ktoré boli implementované už pre začatím tejto práce. Fialovo znázornený nDisplay bol implementovaný už v prechádzajúcej časti, takže je potrebné už len implementovať VRPN server a spracovanie/odosielanie súradníc. VRPN server má stiahnuteľné zdrojové kódy pre server aj pre klienta na oficiálnej stránke. Najskôr je potrebné otestovať kompatibilitu verzií a potom prejsť k implementácii spracovania a odosielania dát.

4.3.2.2 Implementácia komunikácie medzi servarmi s rôznou verziou

Po stiahnutí zdrojového kódu najnovšej verzie VRPN je potrebné vygenerovať riešenie v ktorom sú projekty určené na vybudovanie VRPN servera a klienta. Toto vygenerovanie sa odporúča robiť pomocou C-Make, zdrojové súbory pre VRPN obsahujú vlastný C-Make list, v ktorom sa nachádzajú argumenty na generovanie súborov. Softvér C-Make sa využíva na riadenie procesu kompilácie softvéru pomocou jednoduchých konfiguračných súborov nezávislých na platforme a generovanie natívnych makefile-ov. Vygenerované riešenie obsahuje v sebe zdrojový kód pre VRPN server a klienta. Pre otestovanie prepojenia verzií sme sa pripojili na sieť v ktorej sa nachádza aj komunikácia OptiTrack zariadenia a Motive softvéru, vygenerovali spustiteľný súbor pre klienta a vytvorili odkaz. V argumentoch na spustenie odkazu sme pridali IP adresu na ktorú sa chceme pripojiť, v našom prípade 192.168.10.10 a názov zariadenia *Rigid Body 1*. V Motive sme nastavili odosielanie údajov pre *Rigid Body* na **True**. Toto odosielanie nám zabezpečí odosielanie momentálnej vzdialenosti senzorov od ich začiatočnej pozície v reálnom čase. Potrebné je aj nastavenie pre posielanie údajov cez VRPN na porte 3883. Tento port je základný pre softvéry využívajúce VPRN spojenie.

Po otestovaní prepojenia môžeme vidieť, že verzie VRPN neovplyvňujú posielanie údajov. Pre začatie odosielania do nového VRPN serveru s vyššou verziou je potrebné pridať funkcionalitu do funkcie *mainloop*(). Táto funkcia sa volá pri každom spustení vypisovania súradníc z Motive a nadväzuje spojenie na komunikáciu s IP adresou a zariadením, ktoré sme pridali do argumentov pri spustení odkazu klienta a odosielateľom. Do tejto funkcie sme pridali premenné, do ktorých sme uložili momentálne súradnice a poslali ich na druhý, nami vygenerovaný server s vyššou verziou.

4.3.2.3 Výsledky impletácie a testovania príjmania súradníc pomocou VRPN servera

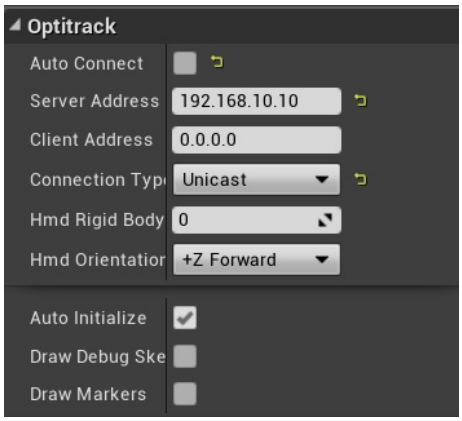
Po implementovaní funkcionality na odosielanie sme predpokladali, že druhý server bude prijímať aktuálne súradnice. Táto hypotéza sa nám však nepotvrdila. Po dlhšom analyzovaní problému, za pomoci VRPN komunity, sme dospeli k záveru, že nie je možné odosielať a prijímať dáta dvoch VRPN serverov v rovnakej sieti, na rovnakej IP adrese. Tieto VRPN servery medzi sebou vedia komunikovať len na jednom porte 3883, no tento port je potrebný na posielanie súradníc do nDisplay.

4.3.3 Pridanie OptiTrack pluginu pre Unreal Engine

Počas implementácie komunikácie medzi novovytvoreným VRPN serverom a Motive, spoločnosť OptiTrack prišla na trh s pluginom pre nové jadro Unreal Engine 4.21. Tento plugin by nemal vytvárať problémy s pridaním ako tomu bolo pri starších pluginoch v novších verziách Unreal Engine.

4.3.3.1 Testovanie OptiTrack pluginu

Prvým krokom je stiahnutie pluginu a pridanie do priečinku Plugin v UE4 projekte. Po načítaní projektu je potrebné pridanie nového *OptiTrack Clienta Origin* aktéra do scény. Tento aktér umožňuje komunikáciu s Motive VRPN serverom, nastavenia komunikácie so serverom sa nachádzajú v detailoch aktéra (obr. 4.4). Staršia verzia tohto pluginu nepoužívala ešte *auto-connect*. Táto funkcionalita uľahčuje pripojenie na OptiTrack, keďže automaticky vyhľadá spojenie a začne prijímať súradnice. Na otestovanie tohto pluginu si vytvoríme Blueprint a skrze funkciu *print* si vypíšeme súradnice z OptiTrack klienta. Súradnice môžeme vidieť v spustenej hre na obrazovke.



Obr. 4.4: Detaily nastavenia OptiTrack Plugin pre Unreal Engine 4.21

Pre modifikáciu lokácie a rotácie kamier všetkých scén v klastri pridáme aktéra *Display Cluster Pawn* do scény. Tento aktér patrí medzi kamery meniace pohľad užívateľa pre všetky obrazovky v klastri a nachádza sa len vo verzii Unreal Engine 4.21 a vyššej. Pre túto kameru vytvoríme Blueprint pri ktorom na každé vykreslenie scény vložíme udalosť pomocou funkcie *Event Tick*. Otestujeme si OptiTrack klienta v novom Blueprint-e, tým že v Blueprint-e zavoláme jeho inštanciu použitím *Find Default Client Origin* a cez *print* funkciu si opäť vypíšeme súradnice.

4.3.3.2 Výsledky testovania OptiTrack pluginu

Pri odosielaní súradníc do *Display Cluster Pawn* Blueprint-u sa nám nepodarilo vypísať súradnice, aj keď sme si overili stále pripojenie na Motive. Keďže prijímanie súradníc išlo bez problémov v Blueprint-e, ktorý bol implementovaný mimo nDisplay. Môžeme sa domnievať, že problém je práve v príjmaní súradníc ak sa jedná o nDisplay projekt, a len v Blueprint-och, ktoré sú prepojené na nDisplay. Tento problém je potrebné analyzovať v Motive softvéri a v odosielaní súradníc, prípadne modifikovať odosielanie tak aby vyhovovalo nDisplay a vedel ich spracovať. Ak sa podarí táto modifikácia, predpokladajme, že bude možné pridať OptiTrack do konfiguračného súboru bez pridania pluginu či skrze vlastný VRPN server.

4.3.4 Koncové riešenie pridania OptiTrack-u a mimo-osového zobrazovania

Po analýze možných pochybení v nastavení Motive softvéru, sme dospeli k záveru že na nesprávnom mieste boli nastavované detaily pre odosielanie súradníc *Rigid Bodies*. Pri nastavení mena pre zariadenie v hlavných nastaveniach pre odosielanie dát sme zistili, že nastavenie mena sa modifikovalo na zlom mieste. Pomohla k tomu zle zdokumentovaná cesta a mylné označenie *Detail of Rigid Body* v hlavných nastaveniach. Po prečítaní oficiálnej dokumentácie pre Motive [8] tiež nebolo úplne jasné kde mohla nastať chyba. Našou domnienkou je, že kým OptiTrack plugin a VRPN server dokážu rozoznať biele znaky, nDisplay nedokáže. V Motive nastaveniach pre *Assets*, je možné pravým kliknutím nastaviť nové meno pre *Rigid Body*. Po premenovaní z *Rigid Body 1* na *Optitrack*, môžeme opätovne otestovať najskôr riešenie s konfiguračným súborom, a neskôr aj s riešením obsahujúcim plugin.

Možnosť ako otestovať natívne riešenie konfiguračného súboru s novým pomenovaním zariadenia bez bielych znakov je napríklad pripísaním argumentu *input*, pri ktorom parameter *addr* je *meno_zariadenia@IP_adresa* a podľa VPRN klienta dokážeme určiť smer osí pre parametre *front, right* a *up*. Argument *input* pre OptiTrack bude vyzerať nasledovne:

```
[input] id=Optitrack type=tracker addr=Optitrack@192.168.10.10 loc="X=0,Y=0,Z=0"
rot="P=0,Y=0,R=0" front=Z right=X up=Y
```

Potom pomocou *id* vieme pridať toto sledovacie zariadenie do argumentu *scene_node* ako parent argument, od ktorého bude dediť *eye_level* na vykreslenie scény a automatické nastavenie matice pre mimo-osové zobrazenie :

```
[scene_node] id=cave_tracker loc="X=0,Y=0,Z=0"  rot="P=0,Y=0,R=0" tracker_id=Optitrack
[scene_node] id=eye_level    loc="X=0,Y=0,Z=0" rot="P=0,Y=0,R=0" parent=cave_tracker
```

5 Vyhodnotenie

Vďaka novému riešeniu pre CAVE systémy od Unreal Engine-u sme boli schopný opraviť posledné nedostatky pôvodnej implementácie. Aj keď odstránenie celého pôvodného riešenia vyzeralo na začiatku ako veľmi odvážny krok, dokázali sme implementovať nové riešenie do LIRKIS CAVE. Implementáciu sme rozdelili do dvoch celkov.

Prvým celkom bolo nahradenie pôvodnej implementácie Unreal Engine pre LIRKIS CAVE od M. Gabrišku [1]. V pôvodnej implementácii bolo potrebné nastaviť parametre zobrazovania v argumentoch na spustenie súboru pre jednotlivé monitory osobitne. Vo verzii nDisplays tento problém rieši práve konfiguračný súbor, ktorý informáciu o parametroch sám prepočíta skrze veľkosť a natočenie monitoru voči mastrovi. Môžeme si všimnúť, že riešenie nDisplays sa podobá riešeniu od M. Gabrišku. Je veľmi pravdepodobne, že aj vďaka tejto podobnosti v komunikácii počítačov v klastri bolo nasadenie riešenia nDisplays do LIRKIS CAVE bezproblémové. Nemali sme žiadne problémy s povoleniami alebo TCP/IP komunikáciou medzi počítačmi, keďže v jeho práci všetky tieto problémy už boli vyriešené a my sme použili rovnaké IP adresy na master počítači. Veľkou výhodou nDisplays v porovnaní s predchádzajúcim riešením je aj spôsob spúšťania aplikácie. Grafické prostedie zjednodušuje prácu pri výbere spustitelného projektu a zadávaní parametrov určených na správne zobrazenie.

Druhým celkom bola implementácia sledovacieho zariadenia OptiTrack-u, čím sme zabezpečili mimo-osové zobrazenie. Pridaním OptiTrack-u do konfiguračného súboru pre nDisplays by sme mali zabezpečiť automatické nastavenie mimo-osového zobrazovania. Prvým problémom bola práve implementácia OptiTrack-u do nDisplays, keďže súradnice sme nevedeli automaticky spracovať v master počítači. Pri odosielaní súradníc sme využili VRPN server. Účelom VRPN je poskytnúť jednotné rozhranie pre vstupné zariadenia, ako sú pohybové sledovače

alebo joystickové ovládače. Po nastavení vlastného VRPN klienta sme videli odosielanie súradníc, vďaka tomuto sme usúdili, že nDisplays nevie prijať spracovane súradnice. V dokumentácii pre nDisplays je uvádzaná podpora pre verzie VRPN 7.33 a vyššie, my sme využívali pôvodnú verziu 7.20 zahrnutú v aplikácii Motive. Motive je softvér od spoločnosti NaturalPoint, ktorá vytvára aj Optitrack systémy, určená na zobrazovanie a odosielanie súradníc z OptiTrack-u. Preto našim účelom bolo vytvoriť vlastné rozhranie určené na odosielanie súradníc za pomoci pôvodného VRPN s verziou 7.20 a nového VRPN s verziou 7.34. Toto rozhranie bohužiaľ nebolo použité z dôvodu využívania rovnakého portu určeného na komunikáciu nDisplays a oboch VRPN serverov. Druhou možnosťou bolo pridanie Optitrack pluginu do projektu v Unreal Engine. Pridanie pluginu prebehlo bezproblémovo ale prijímanie súradníc bolo možné vidieť len v nastaveniach neurčených na komunikáciu v klastri. Keďže my sme potrebovali komunikáciu počítačov v klastri a nie mimo, tak sme sa zamerali na správne odosielanie súradníc v pôvodnej verzii VRPN 7.20. Po analýze nastavení v Motive sme boli schopný identifikovať chybu a prečo vznikla. Nastavenia na sledovanie pevných telies (ang. ridig bodies), bolo nastavované na zlom mieste. V sekcii hlavné nastavenia je pod sekcia s názvom Rigid Bodies. Nastavenia boli modifikované tam, ale nijak neovplyvňovali reálne výstupy. Nie sme si istí, prečo doteraz tieto nastavenia neovplyvňovali odosielanie súradníc. Po opravení súradníc, a to hlavne názvu parametra prístroja na sledovanie pohybu používateľa, sme zistili chybu, ktorá nebola v dokumentácii Motive ani nDisplays uvedená. V názve pre nDisplays sa nesmú objavovať biele znaky. Aj keď sme túto možnosť vylúčili na začiatku implementácie, ukázalo sa, že sme túto možnosť testovali na zlom mieste. Po správnom nastavení odosielania súradníc, sme ešte raz otestovali komunikáciu s nDisplays cez konfiguračný súbor. Aj napriek nižšej verzii VRPN bolo nDisplays schopné spracovať súradnice a automaticky pridalo zobrazenie mimo os. Vďaka tomu bolo naše riešenie kompletné a úspešne implementované.

6 Záver

Cieľom tejto práce bolo opravenie chýb pôvodnej implementácie mimo-osového zobrazovania od D. Tótha [2] a pridanie konfiguračného súboru slúžiacemu jednoduchšiu na modifikáciu dát. Aj kde sme na začiatku predpokladali, že v pôvodnej implementácii sú iba minoritné problémy, analýza ukázala presný opak. Vo verzii jadra Unreal Engine ktorú použil M. Gabriška nebolo možne meniť matice zobrazenia keďže jadro ich pri zobrazovaní samo prepísalo. Vďaka tomu nebolo možné vidieť mimo-osové zobrazenie implementované D. Tóthom.

Pridanie Optitrack systému bolo zložitejšie, vďaka začiatočným problémom, ktoré sme pripisovali nekompatibilite verzii. Počas implementácie sme objavili chybu v nastavení odosielania súradníc v Motive aplikácii, slúžiacej na sledovanie a odosielanie dát zo sledovacieho zariadenia. Po opravení tejto chyby sme boli schopní úspešne pridať sledovacie zariadenie do konfiguračného súboru pre nDisplay. Vďaka tomuto pridaniu sa automaticky zmenila matica zobrazovania v nDisplay a mimo-osové zobrazenie bolo úspešne implementované.

Mimo-osové zobrazenie bolo úspešne pridané spolu s konfiguračným súborom, slúžiacim na modifikáciu parametrov akými sú rotácia alebo rozmery obrazoviek v LIRKIS CAVE. Keďže toto riešenie bolo vytvorené na základe riešenia od Epic Games, nemali by v budúcnosti nastáť žiadne problémy s rozšírením tohto riešenia. Možné rozšírenia vidíme v implementácii grafického rozhrania pre konfiguračný súbor, podobne ako má SuperEngine, pre jednoduchšiu modifikáciu parametrov a grafické zobrazenie všetkých spustených monitorov. Epic Games len nedávno začalo podporovať CAVE systémy na trhu, ak tento trend bude v budúcnosti pokračovať možností na rozšírenie pre LIRKIS CAVE bude pribúdať.

Literatúra

1. GABRIŠKA, Matúš. *Prispôsobenie herného jadra Unreal pre virtuálno-realitnú jaskyňu*. 2018. Diplomová práca. Technická Univerzita v Košiciach.
2. TÓTH, Dominik. *Využitie systému OptiTrack pre snímanie polohy používateľa v hernom jadre Unreal*. Technická Univerzita v Košiciach, 2018. Bakalárska práca.
3. KENYON, Robert V. The CAVE (TM) automatic virtual environment: characteristics and applications. 1995.
4. SOBOTA, Branislav; HROZEK, František. Virtuálna realita a jej technológie. In: Technická Univerzita v Košiciach, 2013, kap. 1, s. 158.
5. JUAREZ, Alex; SCHONENBERG, Willem; BARTNECK, Christoph. Implementing a low-cost CAVE system using the CryEngine2. *Entertainment Computing*. 2010, roč. 1, č. 3-4, s. 157–164.
6. HUDÁK, Marián; KOREČKO, Štefan; SOBOTA, Branislav. On architecture and performance of LIRKIS CAVE system. In: *2017 8th IEEE International Conference on Cognitive Infocommunications (CogInfoCom)*. 2017, s. 000295–000300.
7. NATURALPOINT. Optitrack. *Natural Point, Inc.,*[*Online; cit. 20-5-2019*]. *Dostupné:* `http://www.naturalpoint.com/optitrack`. 2011.
8. NATURALPOINT. Motive Documentation. *Natural Point, Inc.,*[*Online; cit. 20-5-2019*]. *Dostupné:* `https://optitrack.com/products/motive`. 2011.
9. CRUZ-NEIRA, Carolina; SANDIN, Daniel J; DEFANTI, Thomas A. 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*. 1993, s. 135–142.

10. JACOBSON, Jeffrey. Configuring multiscreen displays with existing computer equipment. In: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*. 2002, zv. 46, s. 761–765. Č. 7.

11. DEFANTI, Thomas A et al. The future of the CAVE. *Central European Journal of Engineering*. 2011, roč. 1, č. 1, s. 16–37.

12. KOOIMA, Robert. Generalized perspective projection. *J. Sch. Electron. Eng. Comput. Sci*. 2009.

13. WIKIBOOKS. Cg Programming/Unity/Projection for Virtual Reality. [*Online; cit. 20-5-2019*] *Dostupné:* `https://en.wikibooks.org/wiki/Cg_Programming/Unity/Projection_for_Virtual_Reality`. 2011.

14. EPICGAMES. Build Configuration/ Unreal Engine Documentation. [*Online; cit. 20-5-2019*] *Dostupné:* `https://docs.unrealengine.com/en-us/Programming/BuildTools/UnrealBuildTool/BuildConfiguration`. 2019.

15. UNREALWIKI. Config Files, Read and Write to Config Files/ Unreal Engine Wiki. [*Online; cit. 20-5-2019*] *Dostupné:* `https://wiki.unrealengine.com/Config_Files,_Read_%26_Write_to_Config_Files`. 2019.

16. SEWELL, Brenden. *Blueprints Visual Scripting for Unreal Engine*. Packt Publishing Ltd, 2015.

17. EPICGAMES. Rendering to Multiple Displays with nDisplay. [*Online; cit. 20-5-2019*] *Dostupné:* `https://docs.unrealengine.com/en-us/Engine/Rendering/nDisplay`. 2019.

Zoznam skratiek

CAVE Cave automatic virtual environment.

IDE Integrated Development Environment.

JSON Javascript Object Notation.

TCP/IP Transmission Control Protocol/Internet Protocol.

UE4 Unreal Engine 4.

VR Virtual reality.

VRPN Virtual-Reality Peripheral Network.

Zoznam príloh

Príloha A Používateľská príručka

Príloha B DVD médium - záverečná práca v elektronickej podobe

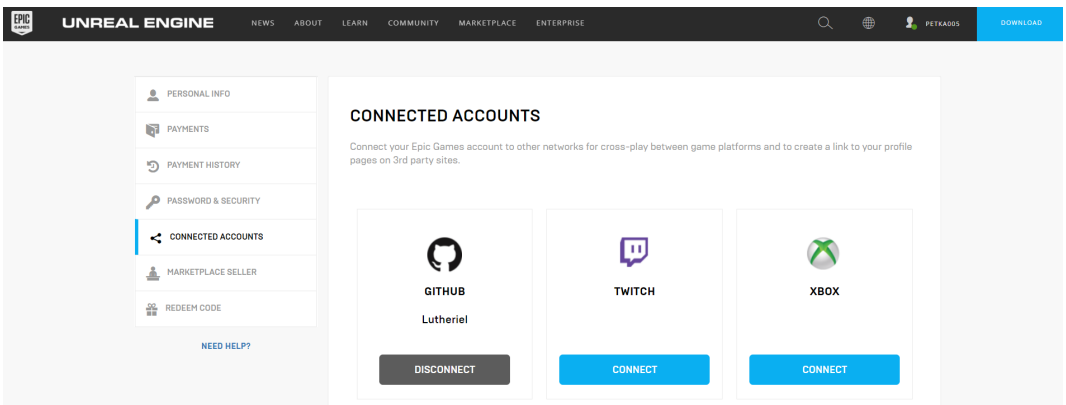
A Používateľská príručka

Úvod

V tejto prílohe je vysvetlený postup inštalácie a používania nDisplay spolu s OptiTrack systémom v LIRKIS CAVE.

Stiahnutie projektu Unreal Engine z oficiálnej stránky

Prvým krokom si stiahnutie projektu pre Unreal Engine. Tu je potrebné sa zaregistrovať na stránke Epic Games (`https://www.unrealengine.com`) aj na stránke GitHub (`https://github.com/`). Následne si nastaviť práva na stránke Epic Games pre pridanie projektu na github-e. Práva pre Epic Games github sa nastavania v sekcii Personal, v pravom menu v časti Connected Accounts (obr.A.1).



Obr. A.1: Prepojenie účtov GitHub a Epic Games na oficiálnej stránke Epic Games

Druhý krok je vytvorenie si prostredia na spustenie aplikácie. Z github stránky si stiahneme projekt do nami vytvoreného priečinka v počítači, tu môžeme použiť

príkaz `git clone <URL_GitHub_stránky_UnrealEngine>`, ak sme už predtým so systémom Git pracovali, alebo stiahnuť projekt pomocou tlačidla download. Ak je projekt komprimovaný, je potrebné ho rozbaliť.

Spustenie Unreal projektu a vytvorenie nDisplay projektu

Spustenie prostredia na vytváranie projektu, cez zdrojový kód :

1. Pre spustenie Unreal Projektu, prípadnú modifikáciu zdrojového kódu je potrebné stiahnuť a nainštalovať integrované vývojové prostredie *Visual Studio 2017* (*spolu s balíkom určením developerom pre C++*)

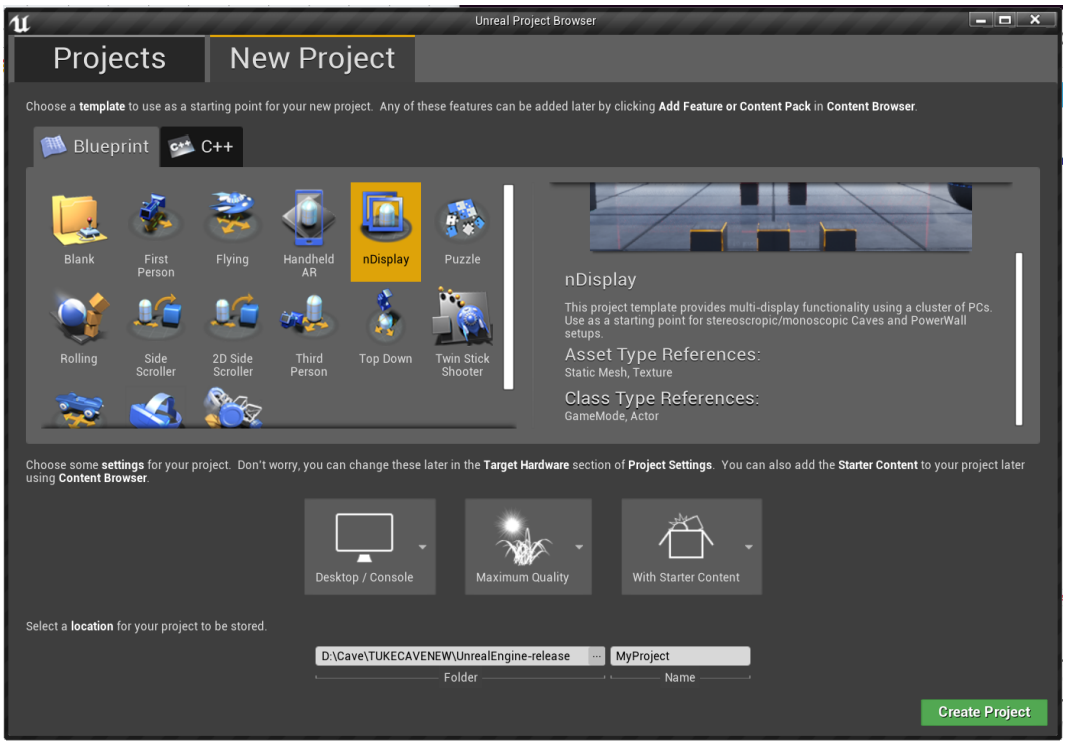
2. Otvoríme súbor so zdrojovým kódom pre Unreal Engine a spustíme **Setup.bat**, na vygenerovanie súborov na spustenie aplikácie. Tento proces môže trvať dlho

3. Po dokončení nastavovania, spustíme **GenerateProjectFiles.bat**. na vytvorenie projektových súborov pre jadro. Dokončenie by malo trvať menej ako minútu

4. Spustíme si riešenie *UE4.sln* a stlačíme **Build**, pre kompiláciu projektu

5. Po skompilovaní bez chýb, možeme prejst k spusteniu Unreal Editora v nástroji Debugger, stlačením **F5** alebo *Debug→Start new Instance* v hornej lište Visual Studia

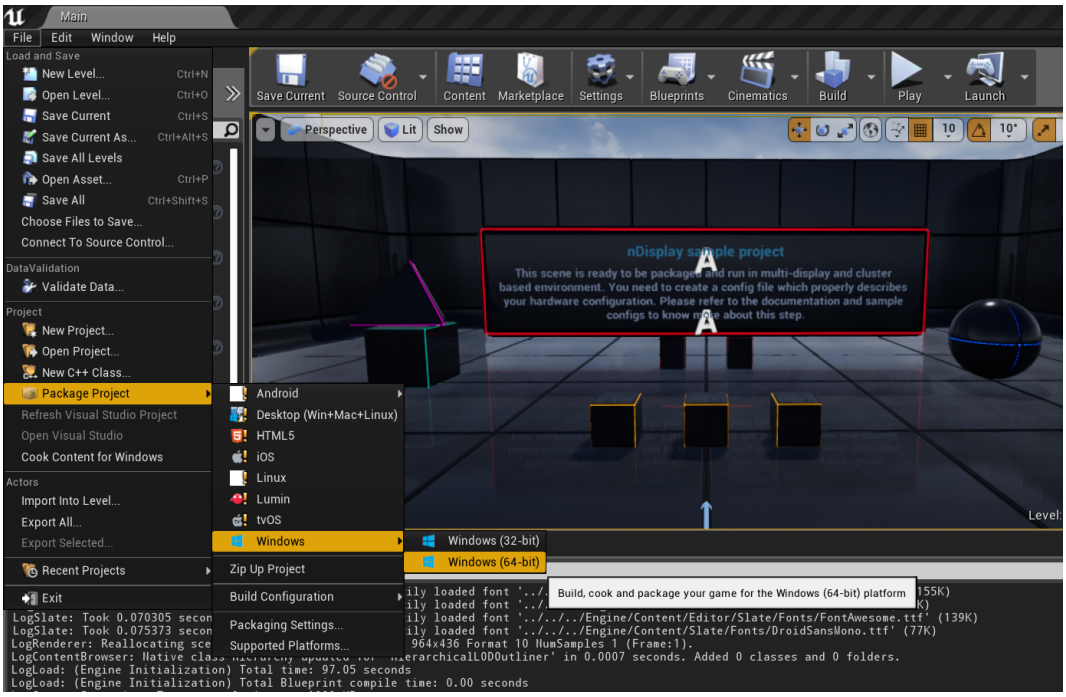
Po načítaní prostredia pre Unreal Engine, vytvoríme nový projekt (obr. A.2), zvolením sekcie **New Projekt**, **Blueprint**, **nDisplay**. V dolnom menu necháme pôvodné nastavenia a pridáme cestu, kam chceme aby sa nám projekt uložil spolu s názvom projektu.



Obr. A.2: Vytvorenie nDisplay projektu

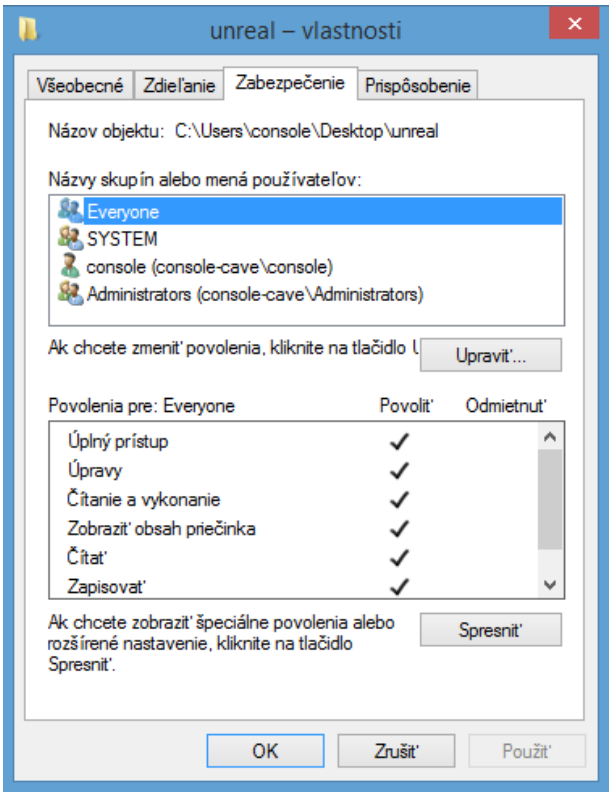
Pridanie projektu do LIRKIS CAVE

Prvým krok je zabalenie projektu (ang. package project). V Unreal Editore v hornej lište zvolíme *File*→*Package Project*→*Windows*→*Windows (64-bit)* (obr. A.3). Zabalenie projektu trvá dlho, okolo 40 až 90 minút, v závislosti od veľkosti projektu. Týmto vytvoríme spustiteľný súbor pre nDisplay aj pre projekt samotný.



Obr. A.3: Zabalenie projektu

Po úspešnom zabalení projektu si skontrolujeme či v zdrojovom súbore jadra *Engine→Binaries→DotNET* sa nachádza ***nDisplayLauncher.exe***, ***nDisplayListener.exe***. Tieto súbory spolu so zabaleným projektom pridáme do novovytvoreného priečinka „unreal“ na pracovnú plochu master počítača pre LIRKIS CAVE, CONSOLE-CAVE. Pre tento priečinok môžeme vytvoriť vlastnú domácu sieť alebo iným spôsobom pridať povolenia pre zdieľanie, modifikáciu a spustenie súborov v priečinku. Pravým kliknutím na priečinok, vybraním možnosti *Vlastnosti→Zabezpečenie* (obr. A.4) skontrolujeme, prípadne modifikujeme povolenia na zdieľanie.



Obr. A.4: Povolenia zdieľania pre priečinok v sieti

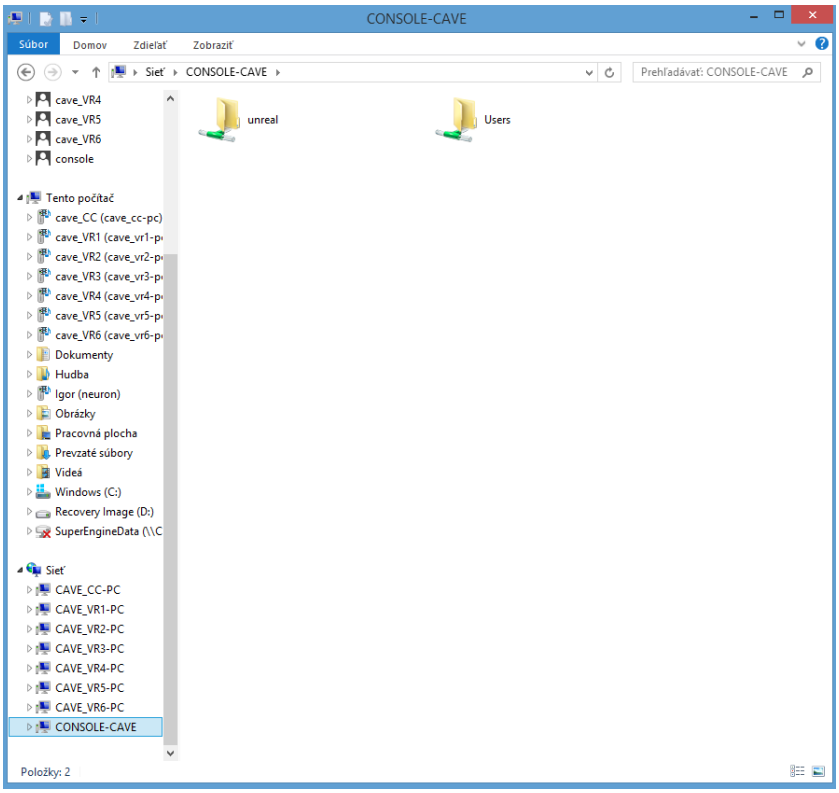
Vytvoríme si konfiguračný súbor na základe návrhových konfiguračných súborov. Časti konfiguračného súboru pre LIRKIS CAVE, jednotlivé argumenty a ich význam sú opísané v Návrhu a implementácii riešenia. V nasledujúcom, nami vytvorenom, konfiguračnom súbore pre LIRKIS CAVE, je zahrnuté aj snímanie pohybu používateľa a zobrazenie mimo. Tento konfiguračný súbor sa taktiež nachádza v riešení práce aj na DVD so zdrojovými súbormi. Sú v ňom implementované všetky obrazovky LIRKIS CAVE a IP adresy ich počítačov slúžiace na spúšťanie.

```
[cluster_node] id=node_1_dn addr=192.168.10.60 screen=screen_1_up viewport=vp_1080 Winx=0 WinY=-1920
[cluster_node] id=node_1_up addr=192.168.10.60 screen=screen_1_dn viewport=vp_1080
[cluster_node] id=node_2_up addr=192.168.10.60 screen=screen_2_up viewport=vp_1080 Winx=1080 WinY=-1920
[cluster_node] id=node_2_dn addr=192.168.10.60 screen=screen_2_dn viewport=vp_1080 Winx=1080 WinY=0
[cluster_node] id=node_3_up addr=192.168.10.40 screen=screen_3_up viewport=vp_1080
[cluster_node] id=node_3_dn addr=192.168.10.30 screen=screen_3_dn viewport=vp_1080
[cluster_node] id=node_4_up addr=192.168.10.40 screen=screen_4_up viewport=vp_1080 Winx=1080 WinY=0
[cluster_node] id=node_4_dn addr=192.168.10.30 screen=screen_4_dn viewport=vp_1080 port_cs=7071
port_ss=7072 master=true sound=true Winx=1080 WinY=0
[cluster_node] id=node_5_up addr=192.168.10.40 screen=screen_5_up viewport=vp_1080 Winx=2160 WinY=0
[cluster_node] id=node_5_dn addr=192.168.10.30 screen=screen_5_dn viewport=vp_1080 Winx=2160 WinY=0
[cluster_node] id=node_6_up addr=192.168.10.70 screen=screen_6_up viewport=vp_1080 Winx=0 WinY=-1920
[cluster_node] id=node_6_dn addr=192.168.10.70 screen=screen_6_dn viewport=vp_1080
```

```
[cluster_node] id=node_7_up addr=192.168.10.70 screen=screen_7_up viewport=vp_1080 Winx=1080 WinY=-1920
[cluster_node] id=node_7_dn addr=192.168.10.70 screen=screen_7_dn viewport=vp_1080 Winx=1080 WinY=0
[cluster_node] id=node_floor_left addr=192.168.10.20 screen=screen_floor_left viewport=vp_1080
[cluster_node] id=node_floor_mid addr=192.168.10.20 screen=screen_floor_mid
viewport=vp_1080 Winx=1080 WinY=0
[cluster_node] id=node_floor_right addr=192.168.10.20 screen=screen_floor_right
viewport=vp_1080 Winx=2160 WinY=0
[cluster_node] id=node_ceiling_left addr=192.168.10.50 screen=screen_ceiling_left viewport=vp_1080
[cluster_node] id=node_ceiling_mid addr=192.168.10.50 screen=screen_ceiling_mid
viewport=vp_1080 Winx=1080 WinY=0
[cluster_node] id=node_ceiling_right addr=192.168.10.50 screen=screen_ceiling_right
viewport=vp_1080 Winx=2160 WinY=0
[screen] id=screen_1_up loc="X=0,Y=0,Z=0" rot="P=0,Y=0,R=0" size="X=0.71,Y=1.23" parent=display_1_up
[screen] id=screen_1_dn loc="X=0,Y=0,Z=0" rot="P=0,Y=0,R=0" size="X=0.71,Y=1.23" parent=display_1_dn
[screen] id=screen_2_up loc="X=0,Y=0,Z=0" rot="P=0,Y=0,R=0" size="X=0.71,Y=1.23" parent=display_2_up
[screen] id=screen_2_dn loc="X=0,Y=0,Z=0" rot="P=0,Y=0,R=0" size="X=0.71,Y=1.23" parent=display_2_dn
[screen] id=screen_3_up loc="X=0,Y=0,Z=0" rot="P=0,Y=0,R=0" size="X=0.71,Y=1.23" parent=display_3_up
[screen] id=screen_3_dn loc="X=0,Y=0,Z=0" rot="P=0,Y=0,R=0" size="X=0.71,Y=1.23" parent=display_3_dn
[screen] id=screen_4_up loc="X=0,Y=0,Z=0" rot="P=0,Y=0,R=0" size="X=0.71,Y=1.23" parent=display_4_up
[screen] id=screen_4_dn loc="X=0,Y=0,Z=0" rot="P=0,Y=0,R=0" size="X=0.71,Y=1.23" parent=display_4_dn
[screen] id=screen_5_up loc="X=0,Y=0,Z=0" rot="P=0,Y=0,R=0" size="X=0.71,Y=1.23" parent=display_5_up
[screen] id=screen_5_dn loc="X=0,Y=0,Z=0" rot="P=0,Y=0,R=0" size="X=0.71,Y=1.23" parent=display_5_dn
[screen] id=screen_6_up loc="X=0,Y=0,Z=0" rot="P=0,Y=0,R=0" size="X=0.71,Y=1.23" parent=display_6_up
[screen] id=screen_6_dn loc="X=0,Y=0,Z=0" rot="P=0,Y=0,R=0" size="X=0.71,Y=1.23" parent=display_6_dn
[screen] id=screen_7_up loc="X=0,Y=0,Z=0" rot="P=0,Y=0,R=0" size="X=0.71,Y=1.23" parent=display_7_up
[screen] id=screen_7_dn loc="X=0,Y=0,Z=0" rot="P=0,Y=0,R=0" size="X=0.71,Y=1.23" parent=display_7_dn
[screen] id=screen_floor_left loc="X=0,Y=0,Z=0" rot="P=0,Y=0,R=0" size="X=0.71,Y=1.23"
parent=display_floor_left
[screen] id=screen_floor_mid loc="X=0,Y=0,Z=0" rot="P=0,Y=0,R=0" size="X=0.71,Y=1.23"
parent=display_floor_mid
[screen] id=screen_floor_right loc="X=0,Y=0,Z=0" rot="P=0,Y=0,R=0" size="X=0.71,Y=1.23"
parent=display_floor_right
[screen] id=screen_ceiling_left loc="X=0,Y=0,Z=0" rot="P=0,Y=0,R=0" size="X=0.71,Y=1.23"
parent=display_ceiling_left
[screen] id=screen_ceiling_mid loc="X=0,Y=0,Z=0" rot="P=0,Y=0,R=0" size="X=0.71,Y=1.23"
parent=display_ceiling_mid
[screen] id=screen_ceiling_right loc="X=0,Y=0,Z=0" rot="P=0,Y=0,R=0" size="X=0.71,Y=1.23"
parent=display_ceiling_right
[viewport] id=vp_1080 x=0 y=0 width=1080 height=1920 flip_h=False flip_v=False
[camera] id=camera_dynamic loc="X=0,Y=0,Z=0" parent=socket_cam tracker_id=Optitrack tracker_ch=0
[scene_node] id=cave_origin  loc="X=0,Y=0,Z=0"      rot="P=0,Y=0,R=0"
[scene_node] id=cave_tracker  loc="X=0,Y=0,Z=0"      rot="P=0,Y=0,R=0"
[scene_node] id=eye_level    loc="X=0,Y=0,Z=0"    rot="P=0,Y=0,R=0"   parent=cave_tracker
[scene_node] id=angle_1      loc="X=1,Y=0,Z=0" rot="P=0,Y=0,R=0" parent=eye_level
[scene_node] id=column_1     loc="X=0,Y=-2.13,Z=0" rot="P=0,Y=0,R=0"   parent=angle_1
[scene_node] id=display_1_up loc="X=0,Y=0,Z=1.23"  rot="P=0,Y=0,R=0"   parent=column_1
[scene_node] id=display_1_dn loc="X=0,Y=0,Z=0" rot="P=0,Y=0,R=0"   parent=column_1
[scene_node] id=angle_2      loc="X=1,Y=0,Z=0" rot="P=0,Y=0,R=0" parent=eye_level
[scene_node] id=column_2     loc="X=0,Y=-1.42,Z=0" rot="P=0,Y=0,R=0"   parent=angle_2
[scene_node] id=display_2_up loc="X=0,Y=0,Z=1.23"  rot="P=0,Y=0,R=0"   parent=column_2
[scene_node] id=display_2_dn loc="X=0,Y=0,Z=0" rot="P=0,Y=0,R=0"   parent=column_2
[scene_node] id=column_3     loc="X=1,Y=-0.71,Z=0"  rot="P=0,Y=0,R=0"   parent=eye_level
```

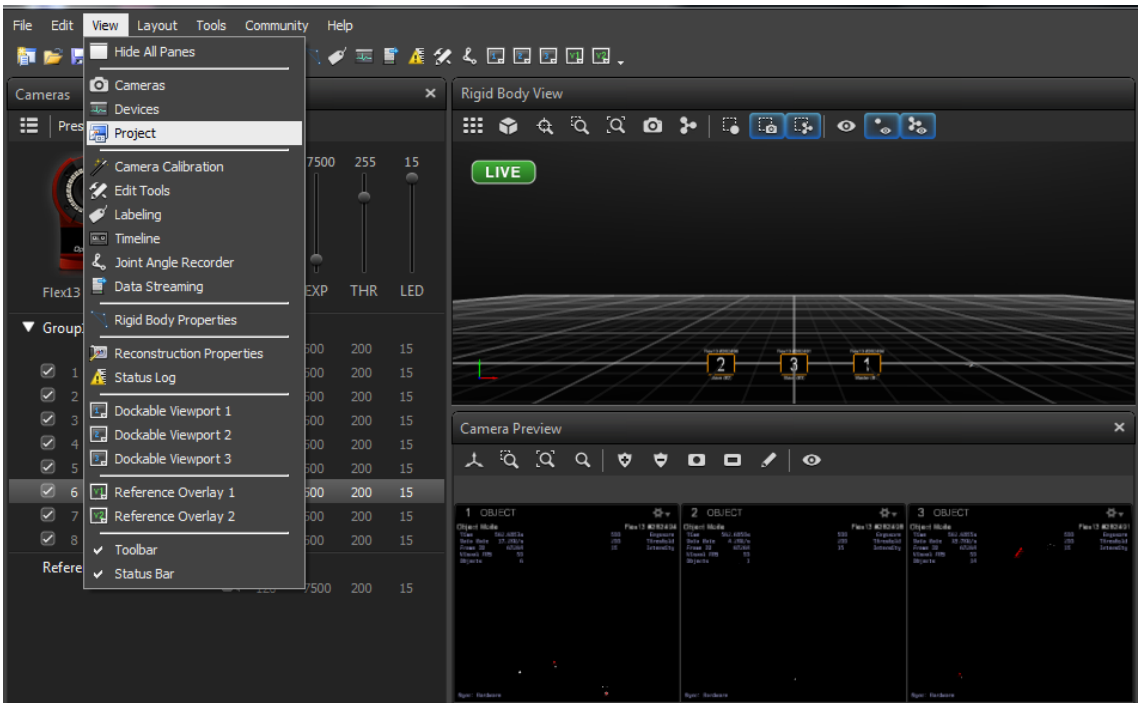
```
[scene_node] id=display_3_up loc="X=0,Y=0,Z=1.23"  rot="P=0,Y=0,R=0"   parent=column_3
[scene_node] id=display_3_dn loc="X=0,Y=0,Z=0" rot="P=0,Y=0,R=0"   parent=column_3
[scene_node] id=angle_4      loc="X=1,Y=0,Z=0" rot="P=0,Y=0,R=0" parent=eye_level
[scene_node] id=column_4     loc="X=0,Y=0,Z=0" rot="P=0,Y=0,R=0"   parent=angle_4
[scene_node] id=display_4_up loc="X=0,Y=0,Z=1.23"  rot="P=0,Y=0,R=0"   parent=column_4
[scene_node] id=display_4_dn loc="X=0,Y=0,Z=0" rot="P=0,Y=0,R=0"   parent=column_4
[scene_node] id=angle_5      loc="X=1,Y=0,Z=0" rot="P=0,Y=0,R=0" parent=eye_level
[scene_node] id=column_5     loc="X=0,Y=0.71,Z=0" rot="P=0,Y=0,R=0"   parent=angle_5
[scene_node] id=display_5_up loc="X=0,Y=0,Z=1.23"  rot="P=0,Y=0,R=0"   parent=column_5
[scene_node] id=display_5_dn loc="X=0,Y=0,Z=0" rot="P=0,Y=0,R=0"   parent=column_5
[scene_node] id=angle_6      loc="X=1,Y=0,Z=0" rot="P=0,Y=0,R=0" parent=eye_level
[scene_node] id=column_6     loc="X=0,Y=1.42,Z=0" rot="P=0,Y=0,R=0"   parent=angle_6
[scene_node] id=display_6_up loc="X=0,Y=0,Z=1.23"  rot="P=0,Y=0,R=0"   parent=column_6
[scene_node] id=display_6_dn loc="X=0,Y=0,Z=0" rot="P=0,Y=0,R=0"   parent=column_6
[scene_node] id=angle_7      loc="X=1,Y=0,Z=0" rot="P=0,Y=0,R=0" parent=eye_level
[scene_node] id=column_7     loc="X=0,Y=2.13,Z=0" rot="P=0,Y=0,R=0"   parent=angle_7
[scene_node] id=display_7_up loc="X=0,Y=0,Z=1.23"  rot="P=0,Y=0,R=0"   parent=column_7
[scene_node] id=display_7_dn loc="X=0,Y=0,Z=0" rot="P=0,Y=0,R=0"   parent=column_7
[scene_node] id=angle_floor      loc="X=1,Y=0,Z=0" rot="P=0,Y=0,R=0" parent=eye_level
[scene_node] id=column_floor     loc="X=0,Y=0,Z=0" rot="P=0,Y=0,R=0"   parent=angle_floor
[scene_node] id=display_floor_left loc="X=0,Y=-0.71,Z=0"  rot="P=-90,Y=0,R=0"   parent=column_floor
[scene_node] id=display_floor_mid loc="X=0,Y=0,Z=0" rot="P=-90,Y=0,R=0"   parent=column_floor
[scene_node] id=display_floor_right loc="X=0,Y=0.71,Z=0" rot="P=-90,Y=0,R=0"   parent=column_floor
[scene_node] id=angle_ceiling loc="X=1,Y=0,Z=0" rot="P=0,Y=0,R=0" parent=eye_level
[scene_node] id=column_ceiling loc="X=0,Y=0,Z=0" rot="P=0,Y=0,R=0"   parent=angle_ceiling
[scene_node] id=display_ceiling_left loc="X=0,Y=-0.71,Z=1.23"  rot="P=90,Y=0,R=0" parent=column_ceiling
[scene_node] id=display_ceiling_mid loc="X=0,Y=0,Z=1.23" rot="P=90,Y=0,R=0" parent=column_ceiling
[scene_node] id=display_ceiling_right loc="X=0,Y=0.71,Z=1.23" rot="P=90,Y=0,R=0" parent=column_ceiling
[scene_node] id=socket_cam   loc="X=0,Y=0,Z=0"      rot="P=0,Y=0,R=0"   parent=eye_level
[scene_node] id=wand         loc="X=0,Y=0,Z=1"      rot="P=0,Y=90,R=0"
[input] id=Optitrack type=tracker addr=Optitrack@192.168.10.10 loc="X=0,Y=0,Z=0"
rot="P=0,Y=0,R=0" front=Z right=X up=Y
[stereo] eye_swap=True eye_dist=0.064
[general] swap_sync_policy=1
[custom]
```

Na master počítači vytvoríme odkaz z ***nDisplayLauncher.exe,*** ktorý nájdeme v priečinku *unreal* v *Sieť→CONSOLE-CAVE* (obr. A.5). Tento odkaz pridáme na pracovnú plochu. Podobne na každom slave počítači, vytvoríme odkaz z ***nDisplayListener.exe,*** ktorý na nachádza na rovnakom mieste ako ***nDisplayLauncher.exe***. Tento odkaz na ***nDisplayListener.exe*** pridáme taktiež na pracovnú plochu každého počítača a spustíme.



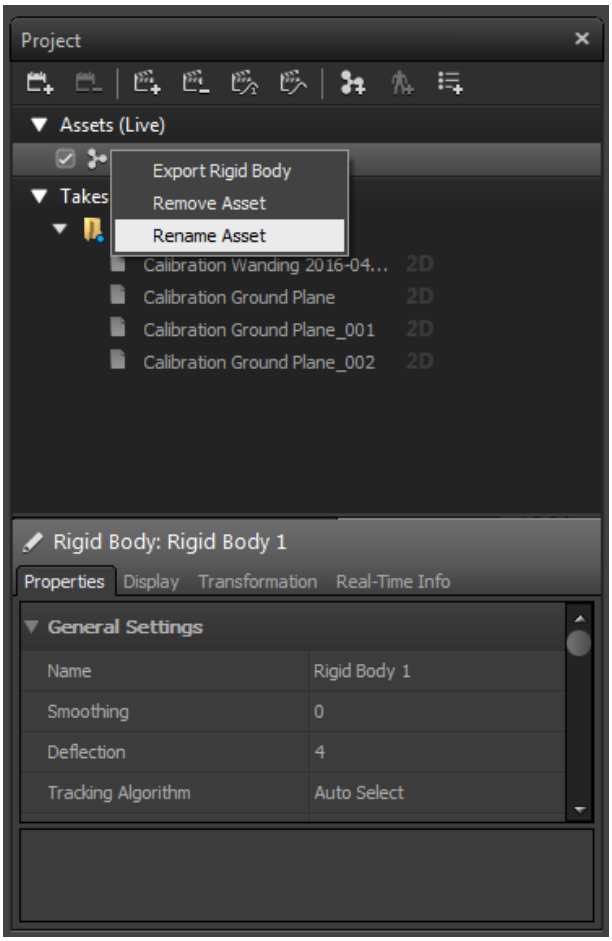
Obr. A.5: Ukážka zdieľaného súboru

Keďže prostredie máme pripravené, potrebné je ešte nastaviť vhodné odosielanie súradnic z Motive do nDisplay. Po zapnutí je potrebné prepísať základné nastavenie Motive. Stačí premenovanie odosielacieho zariadenia. Nový názov nesmie obsahovať biele ani špeciálne znaky. Toto premenovanie možeme nájsť vo *View→Project* (obr. A.6).



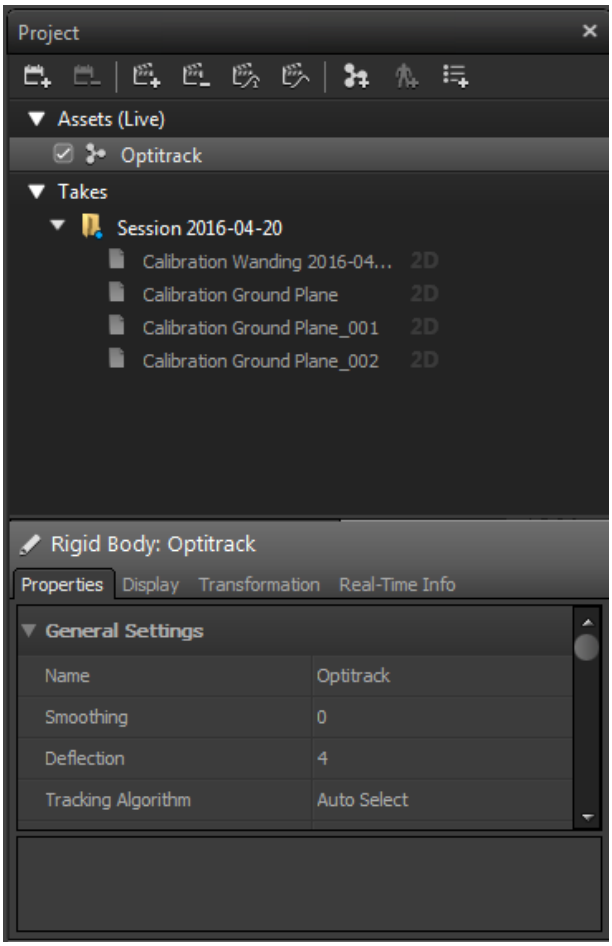
Obr. A.6: Modifikácia nastavení pre odosielanie súradníc

Pravým kliknutím na názov parametru určeného zdielanie súradníc z OptiTrack-u (obr . A.7).



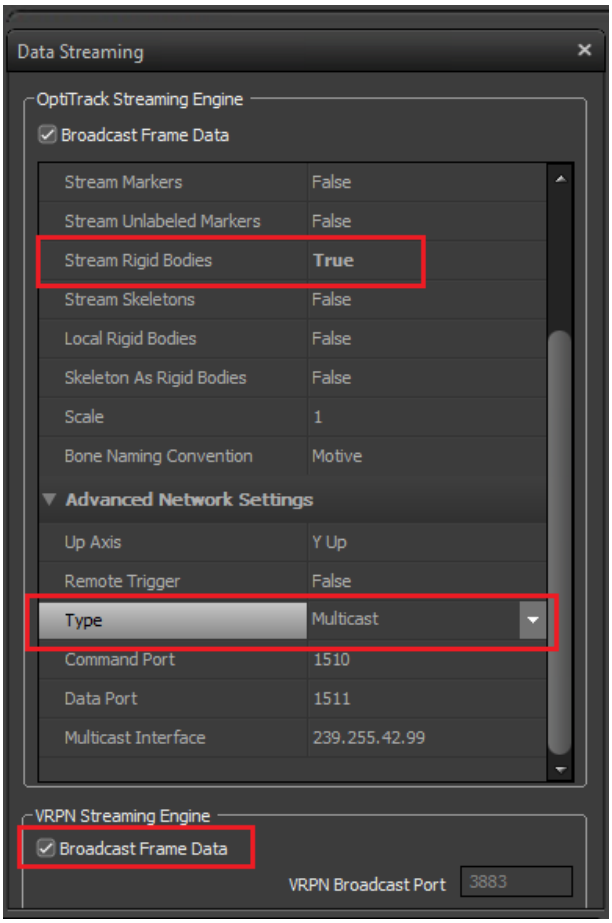
Obr. A.7: Premenovanie parametru na zdielanie kvôli nDisplay

Pre vyššie popísaný konfiguračný súbor, názov paramateru musí byť *Optitrack* (obr.A.8).



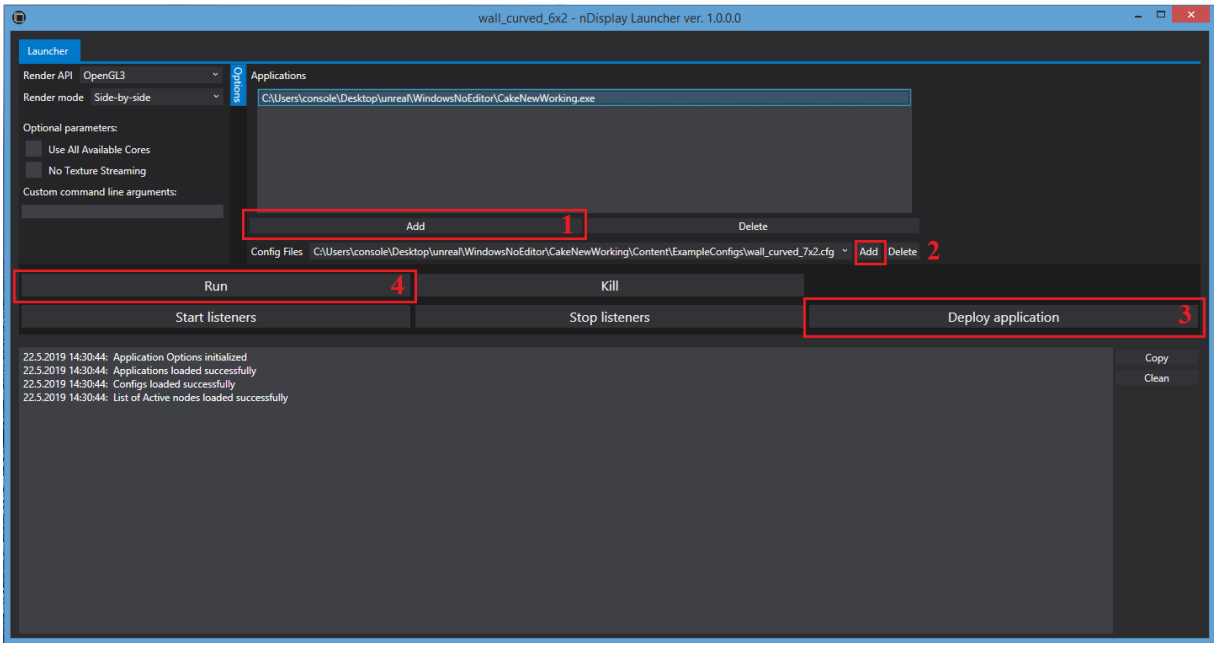
Obr. A.8: Potrebný názov parametru

Ďalsie potrebné nastavenie sa nachádza vo *View→Data Streaming*. Tu je potrebné nastaviť parameter *Stream Rigid Bodies* na **True**, v *Network setting*s nastaviť parameter *Type* na **Multicast** a zapnúť odosielanie dát cez VRPN (obr . A.9).



Obr. A.9: Potrebné nastavenia pre odosielanie dát cez VRPN v Motive

Keď máme Motive správne nastavený, môžeme prejsť späť na master počítač. Tu spustíme oba nami vytvorené odkazy na pracovnej ploche, ***nDisplayListener.exe*** aj ***nDisplayLauncher.exe***. Po spustení ***nDisplayLauncher.exe*** tlačilom **Add**, pridáme spusteľný projekt v Unreal Engine. Menším tlačidlom **Add**, pridáme nami vytvorený konfiguračný súbor, stlačením tlačidla **Deploy Application**, začne komunikácia so slave počítačmi a pridá sa do nich projekt. Tlačidlom **Run** zapneme project na všetkých počítačoch v klastri. Na obrázku č. A.10 vidíme očíslovaný postup pri zapínaní projektu v LIRKIS CAVE cez nDisplay.



Obr. A.10: Postup pri zapínaní projektu v LIRKIS CAVE v nDisplayLauncher.exe

Pre ukončenie stlačíme tlačidlo **Kill**.