MATEMATICKO-FYZIKÁLNÍ
FAKULTA
Univerzita Karlova

DIPLOMOVÁ PRÁCE

Ricardo Bolemant

Monitorování a řízení chování hráčů v reálném čase v Unity pomocí full-stack webové aplikace

Katedra softwaru a výuky informatiky

Vedoucí diplomové práce: Mgr. Lukáš Hejtmánek, Ph.D.

Studijní program: Informatika – Vizuální výpočty a vývoj počítačových her

Studijní obor: IVVP

Praha 2025

Týmto by som chcel poďakovať vedúcemu mojej práce, pánovi Mgr. Lukášovi Hejtmánekovi, Ph.D. za užitočné rady pri vývoji diplomovej práce. Ďalej by som chcel poďakovať mojim rodičom a priateľke za ich podporu pri štúdiu.

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V Praze dne 5.1.2025 podpis

Název práce: Monitorování a řízení chování hráčů v reálném čase v Unity pomocí full-stack webové aplikace

Autor: Ricardo Bolemant

Katedra / Ústav: Katedra softwaru a výuky informatiky

Vedoucí diplomové práce: Mgr. Lukáš Hejtmánek, Ph.D., Katedra softwaru a výuky informatiky

Abstrakt: Cílem práce je vyvinout full-stack webovou aplikaci, pomocí které bude možné sledovat aktivitu hráče během experimentu v reálném čase a různými příkazy zasahovat do průběhu experimentu. Aplikace komunikuje s experimentem vytvořeným v herním enginu Unity prostřednictvím balíčku. Balíček lze snadno implementovat a nastavit. Díky aplikaci už například nebude v případě experimentů ve virtuální realitě nutné sundávat hráči dolů headset pro nastavování experimentu a hráč také nebude muset komentovat základní věci, jako například to, kde se v experimentu právě nachází.

Klíčová slova: unity, webová aplikace, kognitivní výzkum

Title: Real time monitoring and control of player behavior in Unity using a full-stack web app

Author: Ricardo Bolemant

Department: Department of Software and Computer Science Education

Supervisor: Mgr. Lukáš Hejtmánek, Ph.D., Department of Software and Computer Science Education

Abstract: The aim of the work is to develop a full-stack web application, which will allow you to monitor the player's activity during the experiment in real time and to intervene in the course of the experiment with various commands. The application communicates with the experiment created in the Unity game engine via a package. The package can be easily implemented and configured. For example, in the case of experiments in virtual reality, the application will no longer require the player to take off the headset to set up the experiment, and the player will also not have to comment on basic things, such as where he is in the experiment.

Keywords: unity, web app, cognitive experiments

Názov práce: Monitorovanie a riadenie chovania hráčov v reálnom čase v Unity pomocou full-stack webovej aplikácie

Autor: Ricardo Bolemant

Katedra / Ústav: Katedra softwaru a výuky informatiky

Vedúcí diplomovej práce: Mgr. Lukáš Hejtmánek, Ph.D., Katedra softwaru a výuky informatiky

Abstrakt: Cieľom práce je vyvinúť full-stack webovú aplikáciu, pomocou ktorej bude možné sledovať aktivitu hráča počas experimentu v reálnom čase a rôznymi príkazmi zasahovať do priebehu experimentu. Aplikácia komunikuje s experimentom vytvoreným v hernom engine Unity prostredníctvom balíčku. Balíček je možné jednoducho implementovať a nastaviť. Vďaka aplikácii už napríklad nebude v prípade experimentov vo virtuálnej realite nutné dávať hráčovi dole headset na nastavovanie experimentu a hráč taktiež nebude musieť komentovať základné veci, ako napríklad to, kde sa v experimente práve nachádza.

Kľúčové slová: unity, webová aplikácia, kognitívny výskum

Obsah

1. Úvod

Monitorovanie a riadenie správania hráčov počas experimentov je dôležitou súčasťou kognitívnych výskumov, najmä v oblastiach ako virtuálna realita (VR) či simulácie v herných engine. Tieto experimenty často vyžadujú presné a okamžité sledovanie správania hráča, čo môže byť náročné, ak sa tento proces vykonáva manuálne. Rovnako je potrebné včas a efektívne zasahovať do priebehu experimentu, aby sa zabezpečila jeho správnosť a zhodnosť s predpokladanými podmienkami. V tomto kontexte sa ukazuje, že tradičné metódy sledovania a riadenia experimentov môžu byť neefektívne, čo si vyžaduje vývoj nových, inovatívnych prístupov.

Virtuálna realita sa v poslednom desaťročí stala jedným z kľúčových nástrojov v oblasti behaviorálnych a psychologických experimentov, a to vďaka schopnosti vytvárať kontrolované, opakovateľné a zároveň vysoko realistické prostredia [1]. Výskum ukazuje, že VR má obrovský potenciál napríklad pri trénovaní kognitívnych schopností, nácviku sociálnych situácií, rehabilitácii či hodnotení správania v stresových alebo komplexných situáciách [2] [3].

Cieľom tejto práce je vyvinúť full-stack webovú aplikáciu, ktorá umožní monitorovanie a riadenie správania hráčov v reálnom čase počas experimentov realizovaných v hernom engine Unity. Aplikácia bude umožňovať prístup k údajom o správaní hráča v priebehu experimentu a zároveň umožní výskumníkovi zasahovať do experimentu prostredníctvom intuitívneho rozhrania. To znamená, že nie je potrebné používať fyzické zásahy do experimentu, ako je napríklad odstraňovanie VR headsetu, čo uľahčuje priebeh výskumu a zvyšuje efektivitu experimentov.

1.1 Popis a zameranie softwarového diela

Riešenie pozostáva z webovej aplikácie a Unity balíčku, ktorý by malo byť možné jednoducho implementovať do nových aj už existujúcich projektov. Unity balíček slúži na komunikáciu Unity projektu s webovou aplikáciou. Má na starosti prijímať príkazy a odosielať dáta experimentu. Webová aplikácia si uchováva záznamy momentálne pripojených a taktiež aj minulých experimentov. Ponúka prehľad prijatých dát a ovládací panel na riadenie experimentu v reálnom čase. Dôležitou súčasťou je aj zobrazovanie aktuálnej polohy a rotácie hráča alebo iných

objektov na mape experimentu. Konkrétne experimenty spolu s ich ovládacími prvkami a očakávanými prijímanými dátami je možné nadefinovať v konfiguračnom súbore webovej aplikácie. Aplikácia umožňuje pripojenie viacerých experimentov na rôznych zariadeniach naraz. Riešenie je zamerané na experimenty vo virtuálnej realite, no je možné ho použiť aj na bežné experimenty alebo hlavne pre vzdialené online experimenty, ktoré bude možné ovládať na diaľku.

1.2 Motivácia k vzniku softwarového diela

Výskum vo virtuálnej realite (VR) sa v posledných rokoch výrazne rozšíril, no jeho realizácia je stále spojená s viacerými výzvami. Virtuálna realita poskytuje jedinečné možnosti na skúmanie ľudského správania v kontrolovanom prostredí, no zároveň si vyžaduje špecifické nástroje a postupy, ktoré umožňujú efektívne sledovanie a riadenie priebehu experimentov. Jednou z hlavných prekážok pri realizácii experimentov vo VR je spôsob, akým výskumník komunikuje a zasahuje do priebehu experimentu. Keďže účastník nosí VR headset, je plne ponorený do virtuálneho sveta, zatiaľ čo výskumník vidí situáciu iba sprostredkovane cez monitor. Táto odlišná perspektíva vedie k viacerým problémom, ako sú obmedzená kontrola nad priebehom experimentu, zložité resetovanie a nastavovanie alebo nevyhnutnosť fyzických zásahov, ktoré môžu narúšať autenticitu experimentu.

Manuálne riadenie experimentov zaberá veľa času, čo predlžuje celý proces výskumu. Tento čas navyše často zahŕňa aj opakovanie experimentov kvôli technickým či ľudským chybám spôsobeným neefektívnymi nástrojmi. Automatizácia určitých procesov a možnosť ovládať experimenty na diaľku by významne znížila časovú náročnosť, umožnila vykonávať viac experimentov v kratšom čase a minimalizovala pravdepodobnosť chýb. Pre účastníkov experimentu môže byť opakované zloženie a nasadenie VR headsetu nepríjemné a rušivé. Tento proces môže navyše spôsobiť dezorientáciu, najmä ak je experiment zameraný na vnímanie priestoru či pohyb. Systém, ktorý umožní výskumníkovi zasahovať do priebehu experimentu bez potreby interakcie účastníka, by tento problém eliminoval.

Zároveň sa ukazuje, že zníženie technických prekážok môže prispieť k väčšiemu prijatiu VR technológií aj medzi menej technicky zdatnými výskumníkmi [4]. Ak sú nástroje intuitívne a umožňujú rýchle zásahy do experimentu, výskumníci

môžu viac času venovať samotnému zberu dát a analýze namiesto technického riešenia problémov.

2. Analýza požiadaviek

V tejto kapitole najskôr definujeme potrebné pojmy a potom zanalyzujeme požadované vlastnosti aplikácie a Unity balíčku. Podľa toho prevedieme analýzu návrhu aplikácie a Unity balíčku a výber technológií na ich vývoj.

2.1 Požadované vlastnosti aplikácie

Aplikácia musí spĺňať niekoľko kľúčových požiadaviek, ktoré budú umožňovať jej efektívnu interakciu s experimentami a správu dát. Tieto požiadavky zahŕňajú nasledujúce aspekty:

1. **Podpora viacerých experimentov naraz**
    - Aplikácia musí umožniť pripojenie viacerých experimentov súčasne. Táto požiadavka zabezpečuje, že výskumníci môžu pracovať na rôznych experimentoch paralelne, čo je nevyhnutné pri testovaní viacerých scenárov naraz.

2. **Zobrazenie zoznamu aktívnych a neaktívnych experimentov**
    - Aplikácia bude rozdelená do dvoch sekcií: aktívne a neaktívne experimenty. Týmto sa zabezpečí, že používateľ bude mať okamžitý prehľad o tom, ktoré experimenty sú práve pripojené, a ktoré sú momentálne neaktívne. Táto kategorizácia je dôležitá z pohľadu organizácie a efektívneho riadenia viacerých experimentov.

3. **Interakcia s experimentom cez ovládací panel**
    - Ovládací panel v aplikácii bude umožňovať interakciu s experimentom v reálnom čase. To znamená, že používateľ bude môcť posielať príkazy do experimentu a manipulovať s ním podľa potreby. Tento mechanizmus je kľúčový, pretože umožňuje dynamické prispôsobenie experimentu v závislosti od aktuálnych výsledkov a výskumných potrieb.

4. **Zobrazenie detailu experimentu**
    - Pri zobrazení detailu experimentu by mala aplikácia zobraziť infopanel so základnými údajmi (ako názov experimentu, stav, dátum a čas začatia), ovládací panel s rôznymi ovládacími prvkami pre manipuláciu s experimentom, a prijaté dáta v reálnom čase. Tento prehľad umožňuje výskumníkom rýchlo a jednoducho monitorovať priebeh experimentu, čo je dôležité pre okamžité zásahy alebo analýzu.

5. **Ukladanie dát experimentu do súboru**
    - Aplikácia bude umožňovať uloženie dát získaných počas priebehu experimentu do súboru vo formáte JSON[1]. Tieto dáta budú obsahovať všetky prijaté informácie z Unity aplikácie počas aktívneho pripojenia. Cieľom je zabezpečiť možnosť archivácie a neskoršieho prezerania priebehu experimentov. Uložené súbory bude možné stiahnuť alebo odstrániť priamo v detaile neaktívneho experimentu.

6. **Konfigurácia aplikácie pomocou konfiguračného súboru**
    - Aplikácia bude konfigurovateľná prostredníctvom konfiguračného súboru, v ktorom sa definujú parametre pre konkrétne experimenty. Tento súbor bude obsahovať informácie ako:
        - zoznam kompatibilných experimentov,
        - definície tlačidiel ovládacieho panela vrátane príkazov a parametrov,
        - očakávané typy dát a veľkosti histórie,
        - odkazy na obrázky máp pre jednotlivé úrovne experimentov,
        - podmienky viditeľnosti jednotlivých ovládacích prvkov.
    - Súbor nebude nutné upravovať výlučne mimo aplikácie. Jeho obsah bude možné priamo prepisovať cez webové. Používateľ tak bude mať prístup k celému súboru v jeho pôvodnej štruktúre a bude ho môcť upraviť ako obyčajný text. Toto riešenie zachováva plnú flexibilitu konfigurácie bez potreby tvorby špeciálnych používateľských prvkov.

[1] JSON - JSON (JavaScript Object Notation) je jednoduchý formát na výmenu údajov. [5]

7. **Vizuálny štýl aplikácie**
    - Aplikácia musí byť dizajnovaná tak, aby jej vzhľad a užívateľské rozhranie boli intuitívne a konzistentné. Tento vizuálny štýl bude inšpirovaný webovou stránkou https://cyberspacelab.cz, čím sa zabezpečí estetika a prívetivosť pre používateľa.

2.2 Požadované vlastnosti Unity balíčku

Unity balíček, ktorý sa integruje s webovou aplikáciou, musí spĺňať niekoľko kľúčových požiadaviek pre správnu funkcionalitu experimentu. Tento balíček bude zodpovedný za prijímanie príkazov a odosielanie dát o experimentálnom priebehu. Požiadavky na tento balíček sú nasledovné:

1. **Jednoduchá implementácia do nových aj existujúcich projektov**
    - Unity balíček musí byť navrhnutý tak, aby bol čo najjednoduchšie implementovateľný do rôznych projektov, či už ide o nový alebo už existujúci projekt. Tým sa zabezpečí flexibilita pri použití balíčka v rôznych prostrediach, čím sa výrazne zjednoduší jeho použitie pre vývojárov. Tento aspekt je veľmi dôležitý, pretože vývojári nebudú chcieť investovať príliš veľa času do integrácie balíčka, keďže by to mohlo ovplyvniť celkový vývoj experimentu.

2. **Automatické nadviazanie spojenia s webovou aplikáciou**
    - Po spustení experimentu v Unity sa balíček automaticky pokúsi pripojiť k webovej aplikácii bez potreby manuálneho zásahu. Tým sa zjednoduší proces integrácie a minimalizuje sa riziko chýb spôsobených nesprávnym nastavením zo strany vývojára. Po úspešnom pripojení sa začne výmena dát medzi aplikáciami.

3. **Obnova spojenia po prerušení**
    - V prípade dočasného výpadku spojenia medzi Unity projektom a webovou aplikáciou musí balíček automaticky rozpoznať stratu pripojenia a priebežne sa pokúšať o jeho obnovenie. Tento

mechanizmus musí fungovať na pozadí a bez potreby reštartovania experimentu, aby sa zabezpečil plynulý priebeh a minimalizovali sa výpadky v komunikácii.

4. **Monitorovanie a zobrazenie stavu pripojenia**
    - Balíček musí poskytovať informácie o aktuálnom stave pripojenia k webovej aplikácii. Tento stav pripojenia by mal byť jednoducho dostupný v Unity projekte a mal by umožniť vývojárom vizualizovať stav pripojenia (či už je pripojenie aktívne, alebo nie). Tento mechanizmus umožní včasnú detekciu problémov s pripojením a ich rýchlu opravu.

5. **Obojsmerná komunikácia medzi Unity a webovou aplikáciou s podporou Unity editora**
    - Balíček musí zabezpečiť spoľahlivú obojsmernú komunikáciu medzi Unity experimentom a webovou aplikáciou. To znamená, že musí byť schopný:
        - prijímať príkazy alebo udalosti (eventy) z webovej aplikácie a reagovať na ne v reálnom čase,
        - odosielať dáta o priebehu experimentu späť na web (napríklad pozíciu hráča, aktuálnu úroveň, skóre a pod.).
    - Pre zjednodušenie integrácie a konfigurácie v prostredí Unity musí balíček poskytovať komponenty s možnosťou nastavovania priamo cez Unity Inspector. Tým sa vývojárovi umožní jednoduchým spôsobom priradiť spracovanie jednotlivých eventov, definovať odosielané dáta a sprístupniť základné nastavenia (napr. URL pripojenia, ID experimentu) bez potreby zásahu do zdrojového kódu.

3. Analýza existujúcich riešení

Predtým, ako sme začali implementovať naše riešenie, sme si chceli naštudovať, či už neexistujú nejaké podobné prístupy, a ak áno, ako daný problém riešia. Bohužiaľ sme nenašli žiadne verejne dostupné riešenie, ktoré by umožňovalo riadiť a sledovať experimenty prostredníctvom webového rozhrania. Existujú rôzne frameworky na implementáciu experimentov v prostredí Unity. Taktiež je k dispozícii viacero riešení, kde Unity aplikácie komunikujú s dedikovaným serverom a odosielajú naň rôzne analytické dáta. My však vytvárame riešenie, v ktorom sa z Unity aplikácie posielajú dáta týkajúce sa priebehu experimentu (nie konkrétne analytické dáta určené na jeho vyhodnotenie) do webovej aplikácie, kde si ich môžeme v rôznych formách vizualizovať. Zároveň môžeme experiment v Unity ovládať cez webovú aplikáciu zasielaním rôznych udalostí.

Napriek tomu si v tejto kapitole v krátkosti rozoberieme aspoň niektoré riešenia, ktoré sa zaoberajú podobnou problematikou.

3.1 UXF – Unity Experiment Framework

Unity Experiment Framework [6] (alebo v skratke UXF) je open-source framework navrhnutý špeciálne na uľahčenie tvorby behaviorálnych experimentov v prostredí Unity. Tento nástroj umožňuje vývojárom a výskumníkom organizovať experimenty do tzv. blokov a trialov, čím napodobňuje bežnú štruktúru experimentálnych dizajnov využívanú v psychológii či neurovede. Poskytuje aj robustný systém na zaznamenávanie dát, ktoré sú následne použité na analytické vyhodnocovanie výsledkov experimentu ako napríklad reakčné časy, odpovede používateľa, presnosť alebo pozície objektov v čase.

UXF je navrhnutý primárne pre off-line zber a vyhodnocovanie dát. To znamená, že všetky údaje sú zhromaždené počas behu experimentu a analyzujú sa až po jeho ukončení. UXF neponúka natívnu podporu pre živú komunikáciu s webovou aplikáciou počas prebiehajúceho experimentu.

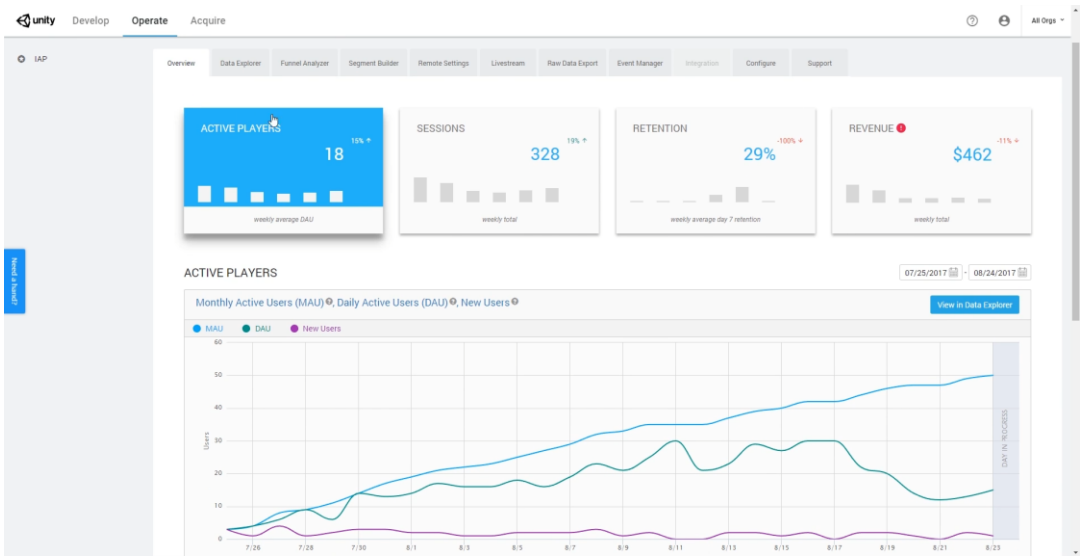
V tomto smere sa naše riešenie zásadne líši. Zameriavame sa na to, aby Unity aplikácia v reálnom čase odosielala do webovej aplikácie informácie o priebehu experimentu, nie o analytických parametroch účastníka. Medzi tieto údaje patrí

napríklad aktuálna pozícia hráča, názov scény (úrovne), jeho skóre, stav experimentu a podobne. Tieto dáta slúžia predovšetkým na monitorovanie a riadenie experimentu na diaľku a nie na samotné vyhodnotenie výsledkov.

Zároveň naša webová aplikácia umožňuje aj spätnú komunikáciu. Experiment v Unity je možné ovládať zasielaním udalostí z webového rozhrania, čím vytvárame obojsmerný kanál medzi výskumníkom a experimentálnym prostredím. UXF tak môže slúžiť ako užitočný nástroj pre inšpiráciu pri organizácii experimentu, no náš prístup posúva túto funkcionalitu smerom k dynamickej a interaktívnej webovej správe experimentov v reálnom čase.

3.2 Unity Analytics

Unity Analytics [7] je nástroj priamo integrovaný do vývojového prostredia Unity, ktorý umožňuje vývojárom zbierať dáta o správaní používateľov počas používania aplikácie alebo hry. Ide najmä o zber kvantitatívnych údajov, ako sú počty spustení, dĺžka hrania, progres používateľa, opakujúce sa chyby alebo výkonnosť aplikácie na rôznych zariadeniach. Tieto dáta sa ukladajú na servery Unity a sú prístupné prostredníctvom prehľadného webového rozhrania, ktoré umožňuje tvorbu grafov, štatistických prehľadov a jednoduchých vizualizácií.



Obrázok 1: Unity Analytics dashboard

Hlavným cieľom Unity Analytics je podpora vývojárov pri optimalizácii hier a aplikácií na základe reálneho správania používateľov. Tento nástroj je teda

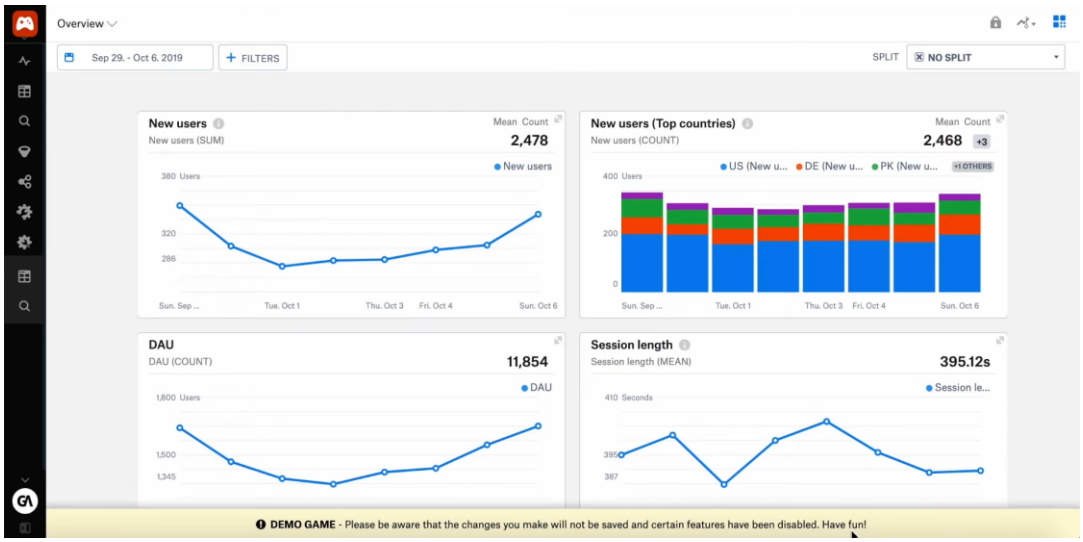
orientovaný najmä na produktovú analytiku a nie na výskumné účely. V prostredí experimentov, aké sú cieľom nášho riešenia, je jeho využitie značne obmedzené. Neumožňuje totiž živé sledovanie dát, ani priame prepojenie s vlastným webovým rozhraním, ktoré by mohlo experiment riadiť alebo dynamicky reagovať na priebeh.

Unity Analytics tiež nepodporuje odosielanie vlastných dátových štruktúr v reálnom čase na vlastný server, čo je pre náš projekt kľúčové. V našom prípade chceme mať kontrolu nad tým, aké konkrétne údaje sa zbierajú, kedy sa odosielajú a ako sa vizualizujú. Navyše, naše riešenie umožňuje okamžitú spätnú väzbu a možnosť interakcie medzi výskumníkom a experimentom počas jeho priebehu, čo je mimo rámec možností Unity Analytics.

Preto sa Unity Analytics javí ako vhodný nástroj pre vývoj a ladenie herných produktov, avšak pre potreby behaviorálnych alebo interaktívnych experimentov v akademickom kontexte nie je dostatočne flexibilný.

3.3 GameAnalytics pre Unity

GameAnalytics [8] je externý analytický nástroj, ktorý je možné integrovať do Unity aplikácií pomocou dostupného SDK. Ide o službu zameranú najmä na vývojárov hier, ktorí chcú získať prehľad o správaní hráčov, optimalizovať herné mechaniky a zvyšovať mieru udržania používateľov. SDK pre Unity je dobre zdokumentované a umožňuje vývojárom jednoducho sledovať rôzne metriky bez nutnosti vlastnej implementácie systému na zber a spracovanie dát.



Obrázok 2: GameAnalytics dashboard

Pomocou GameAnalytics možno sledovať základné udalosti, ako je spustenie hry, začiatok a koniec úrovní, dosiahnuté skóre, chyby, alebo ekonomické transakcie v hre. Vývojár môže definovať aj vlastné udalosti a sledovať ich výskyt, avšak tieto údaje sú spracovávané na strane služby GameAnalytics a nie sú priamo dostupné na spracovanie v reálnom čase vo vlastnej webovej aplikácii.

Rovnako ako v prípade Unity Analytics, aj GameAnalytics sa sústreďuje na produktovú analytiku, ktorá má slúžiť na zlepšenie herného zážitku. Tento nástroj poskytuje štatistické prehľady, vizualizácie a rôzne typy filtrov, ktoré sú určené skôr pre produktových manažérov a dizajnérov než pre výskumníkov pracujúcich s behaviorálnymi dátami v experimentoch.

Pre potreby nášho projektu predstavuje GameAnalytics podobné obmedzenia ako Unity Analytics. Neumožňuje živú vizualizáciu dát o priebehu experimentu, ani spätnú komunikáciu s aplikáciou. Okrem toho, údaje sú ukladané a spravované výhradne v rámci platformy GameAnalytics, čo znižuje flexibilitu a kontrolu nad dátovým tokom.

Z týchto dôvodov je GameAnalytics vhodným nástrojom pre zber herných metrík v rámci komerčných titulov, no neponúka funkcionalitu potrebnú pre interaktívne a diaľkovo ovládané experimenty, aké sú cieľom nášho riešenia.

3.4 Mixpanel pre Unity

Mixpanel [9] je pokročilá analytická platforma, ktorá umožňuje sledovanie a analýzu používateľského správania v reálnom čase. Jeho hlavnou výhodou oproti tradičným analytickým nástrojom je zameranie na udalosťami riadenú analytiku, kde každá interakcia používateľa (napríklad kliknutie, prechod medzi obrazovkami alebo dokončenie úlohy) predstavuje samostatnú udalosť, ktorú je možné zaznamenať, filtrovať a analyzovať.

Pre prostredie Unity je dostupné oficiálne SDK, ktoré umožňuje vývojárom integrovať Mixpanel do svojich hier a aplikácií. Po inicializácii môžu vývojári manuálne definovať a odosielať vlastné udalosti, spolu s ľubovoľnými atribútmi (napr. meno úrovne, skóre, čas). Zaznamenané dáta sú následne sprístupnené vo webovom rozhraní Mixpanelu, kde je možné ich analyzovať prostredníctvom grafov, segmentácie používateľov a vytvárania rôznych metrík.

Mixpanel sa vyznačuje vysokou mierou prispôsobiteľnosti a možnosťou pracovať s údajmi takmer v reálnom čase. Napriek tomu nie je jeho použitie ideálne pre prípad experimentálneho prostredia, ako je to v našom riešení. Hoci vývojár môže odosielať akékoľvek vlastné dáta, spätné ovládanie experimentu zo strany výskumníka nie je podporované.

Mixpanel teda ponúka výkonné nástroje na sledovanie správania používateľov, segmentáciu a personalizáciu obsahu, no jeho primárne využitie je v komerčných aplikáciách, marketingových kampaniach a produktoch zameraných na zvyšovanie zapojenia používateľov. V kontexte behaviorálnych experimentov sa ukazuje ako príliš robustné a zároveň málo flexibilné riešenie, ktoré neumožňuje riadenie experimentu počas jeho priebehu.

Naše riešenie kladie dôraz na priamu komunikáciu medzi Unity a webovou aplikáciou, a tiež na schopnosť okamžite reagovať na zmeny v priebehu experimentu, čo Mixpanel štandardne nepodporuje.

4. Návrh riešenia

V tejto kapitole sa podrobne zaoberáme návrhom aplikácie a výberom technológií, ktoré boli zvolené s cieľom zabezpečiť efektívny vývoj a jednoduchú rozšíriteľnosť systému.

4.1 Architektúra a návrh systému

Webová aplikácia musí komunikovať s viacerými hernými reláciami naraz, preto bude vyvinutá modelom klient-server, kde backendová nezávislá časť bude serverom a frontendová časť aplikácie s hernými reláciami Unity budú klientami. Server s frontendom bude bežať na cloude, aby bol dostupný odkiaľkolvek. Taktiež musí byť podporované pristupovať k aplikácii z viacerých zariadení a prehliadačov naraz.

4.1.1 Pripojenie viacerých relácií

Aby bolo možné súčasne spravovať viacero relácií a zabezpečiť prístup k systému z viacerých zariadení, aplikácia musí byť dostupná cez internet. Z tohto dôvodu bude nasadená v cloudovom prostredí.

Klientmi systému sú webové prehliadače a Unity aplikácie. Webové rozhranie umožňuje používateľom prezerať a ovládať aktívne relácie, zatiaľ čo Unity klienti odosielajú a prijímajú dáta o stave herného prostredia. Tieto dve časti sú prepojené cez centrálny server, ktorý sprostredkúva komunikáciu, spracúva požiadavky a distribuuje príkazy.

4.1.2 Kontajnerizácia a nasadenie

Pre zabezpečenie jednoduchého nasadenia, škálovateľnosti a konzistentnosti prostredí, bola aplikácia plne dockerizovaná pomocou Dockeru [10]. Použitím Docker Compose [11] sú definované jednotlivé kontajnery – samostatne pre backend a frontend. Docker Compose zároveň zabezpečuje vytvorenie internej siete medzi kontajnermi, v ktorej môžu jednotlivé komponenty aplikácie efektívne komunikovať.

Vzhľadom na to, že frontendová časť nedokáže priamo adresovať backend cez internú adresu, je medzi kontajnery zaradený reverzný proxy server. Pre túto úlohu bol zvolený nginx [12], ktorý preposiela HTTP požiadavky z používateľského prehliadača na príslušnú službu v pozadí. Vďaka tomuto riešeniu nie je potrebné priamo špecifikovať sieťové adresy v kóde klienta.

4.1.3 Užívateľské rozhranie

Používateľské rozhranie aplikácie je navrhnuté pre počítače s veľkým rozlíšením a maximálnym oknom. Aplikácia nebude optimalizovaná pre mobilné zariadenia, keďže sa očakáva jej používanie najmä počas experimentov v laboratórnom alebo výskumnom prostredí, kde je k dispozícii počítač.

Rozhranie bude pozostávať zo štyroch základných častí: zoznam aktívnych relácií, zoznam neaktívnych relácií, detail aktívnej relácie a detail neaktívnej relácie. Detail relácie zobrazuje prehľadný ovládací panel, mapu prostredia, informácie o stave relácie a výpis prijatých údajov.

4.1.4 Správa a uchovávanie dát

Dáta sú spravované s dôrazom na efektívnosť a rýchlu dostupnosť. Informácie o aktívnych reláciách sú uchovávané v pamäti RAM, čím sa zabezpečuje nízka latencia a okamžitá reakcia na zmeny. Po ukončení relácie sú príslušné údaje serializované a uložené na disk, čo umožňuje ich opätovné načítanie po reštarte aplikácie.

Pôvodne zvažované databázové riešenie bolo zamietnuté, keďže potreby systému nevyžadujú robustné databázové operácie ani zložité dotazy. Pracuje sa výhradne s aktuálnym stavom experimentov a poslednými zaznamenanými údajmi.

4.2 Komunikačný protokol a WebSocket technológia

Na komunikáciu medzi serverom a klientmi bola zvolená technológia WebSockets [13]. Oproti tradičnému REST API [14], ktoré bolo krátko testované v úvodnej fáze vývoja, WebSockety umožňujú trvalé obojsmerné spojenie, ktoré je nevyhnutné pre riadenie a monitorovanie relácií v reálnom čase. REST API by si

vyžadovalo opakované aktívne požiadavky zo strany klienta, čo by nebolo vhodné vzhľadom na typické nasadenie klientov – za NAT[2] alebo bez verejnej adresy.

WebSocket spojenie je po počiatočnom nadviazaní (handshake) udržiavané v trvalom stave, čo umožňuje okamžité doručovanie príkazov aj spätných hlásení bez potreby opätovného nadväzovania spojenia.

4.3 Unity balíček a jeho integrácia

Pre zjednodušenie integrácie Unity projektov so serverom aplikácie bude vytvorený jednotný balíček. Tento balíček bude obsahovať skripty, ktoré zabezpečia naviazanie a konfiguráciu spojenia, ako aj pomocné funkcie pre odosielanie a prijímanie dát.

Súčasťou balíčka bude aj komponenta zodpovedná za pravidelné odosielanie dát o pozíciách a rotáciách objektov v scéne. Tieto údaje budú následne vizualizované v používateľskom rozhraní webovej aplikácie.

V rámci balíčka bude taktiež dostupné rozhranie pre prijímanie udalostí (eventov) z webovej aplikácie a odosielanie odpovedí alebo stavových správ naspäť. V záujme pohodlnej konfigurácie bude možné všetky dôležité parametre nastavovať aj priamo cez Unity Editor, prostredníctvom inšpektora.

4.4 Použité technológie

Na serverovej strane aplikácie bol zvolený Python framework Flask. Tento framework je vhodný pre rýchly vývoj menších až stredne veľkých webových aplikácií a ponúka veľkú mieru flexibility. Na strane klienta bola použitá knižnica Vue.js, ktorá umožňuje tvorbu komponentového a reaktívneho používateľského rozhrania.

Obe technológie boli vybrané pre svoju jednoduchosť, nízku náročnosť na učenie a možnosť ľahkého rozšírenia. Navyše sú veľmi dobre podporované v rámci kontajnerových technológií, čo ešte viac uľahčuje ich nasadenie.

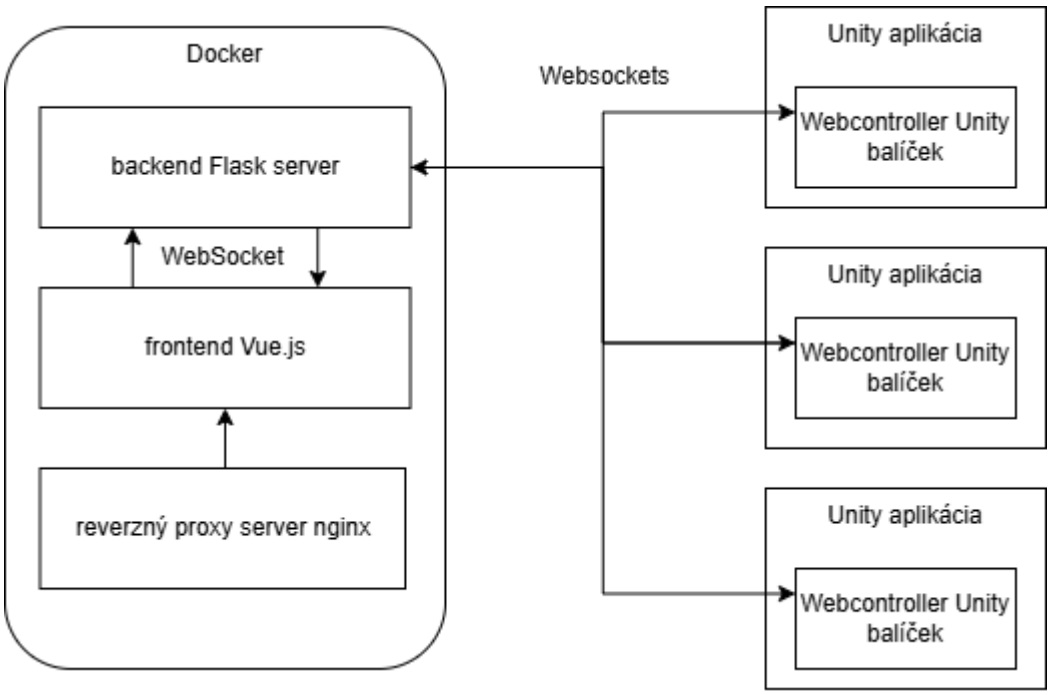
[2] NAT - Network address translation (preklad sieťových adries) je technológia, ktorá umožňuje viacerým zariadeniam v lokálnej sieti zdieľať jednu verejnú IP adresu na prístup do internetu.

Komunikácia v reálnom čase je riešená pomocou knižnice Socket.IO [15], ktorá je dostupná ako pre Flask (Flask-SocketIO [16]), tak aj pre Vue.js (socket.io-client [17]) a Unity (SocketIOUnity [18]). Táto knižnica zabezpečuje efektívnu výmenu správ a jednoduchú prácu s udalosťami medzi všetkými troma zložkami systému.

5. Implementácia

5.1 Architektúra aplikácie

Aplikácia je postavená na architektúre klient-server, kde frontend je implementovaný pomocou frameworku Vue.js a backend využíva Python framework Flask. Komunikácia medzi frontendom a backendom prebieha prostredníctvom WebSocketov, čo umožňuje obojsmernú komunikáciu v reálnom čase. Reverzný proxy server nginx je využitý na dotazovanie serveru skrz frontend v internetovom prehliadači. Herné relácie Unity predstavujú klientov, ktorý so serverom komunikujú pomocou implementovaného balíčku. Webová aplikácia je dockerizovaná. Pre komunikáciu skrz websockety sa používa knižnica Socket.IO. Pri pripojení klienta k serveru sa mu automaticky pridelí unikátne socket.id, na základe ktorého server odosiela požiadavky konkrétnemu klientovi. Aplikácia podporuje viacero Vue.js frontendových klientov zároveň.



Obrázok 3: Architektúra riešenia

5.2 Flask server

Server je implementovaný v súbore app.py. Pozostáva z inicializácie serveru, Socket.IO event handlerov a pomocných metód. CORS (Cross-Origin Resource Sharing) je povolený pre všetky zdroje, čo umožňuje klientom z rôznych domén pripojiť sa k serveru. Server beží na porte číslo 4000. Pracuje s konfiguráciou aplikácie, kde sú definované dáta možných pripojených experimentov herných relácií Unity.

5.2.1 Správa relácií

Relácie sú uložené v RAM pamäti servera v slovníku **sessions**, kde každá relácia má unikátny identifikátor **device_id**. Štruktúra dát jednej relácie je nasledovná:

- [**device_id**] – Identifikačné číslo zariadenia, na ktorom herná relácia beží.
    - **session_name** – Meno experimentu.
    - **start_time** – Čas kedy sa relácia pripojila.
    - **last_ping** – Čas kedy relácia naposledy odoslala dáta.
    - **data** – Dáta relácie.
    - **sid** – Identifikačné číslo relácie.
    - **is_connected** – Označenie, či je relácia práve pripojená.

V RAM pamäti servera sa uchovávajú aktívne relácie a najnovšie neaktívne relácie. Všetky neaktívne relácie sa taktiež ukladajú na disk na serveri.

5.2.2 Dáta relácie

Dáta relácie sú reprezentované slovníkom, pričom kľúčom je názov typu dát a hodnotou je zoznam prijatých dát. Napríklad „position“, čo obsahuje pozície hráčov a iných objektov odoslaných reláciou alebo „context“, kde sú definované aktuálne kontexty pre tlačidlá v grafickom rozhraní webovej aplikácie, podľa ktorých sa rozhoduje, či sa práve budú alebo nebudú zobrazovať. Tieto dva kľúče sú aplikáciou rezervované a nemôžu sa používať inak, zatiaľ čo ostatné kľúče sú definované špecificky pre každý typ hernej relácie v konfiguračnom súbore.

Príklad definície prijímaných dát v konfiguračnom súbore:

```
"receivers": [
  {
    "currentScene": {
      "maxHistory": 5
    },
    "coinsCollected": {
      "maxHistory": 10
    },
    "location": {
      "maxHistory": 10
    },
    "position": {
      "maxHistory": 15
    }
  }
],
```

5.2.3 Pripojenie klienta frontendu

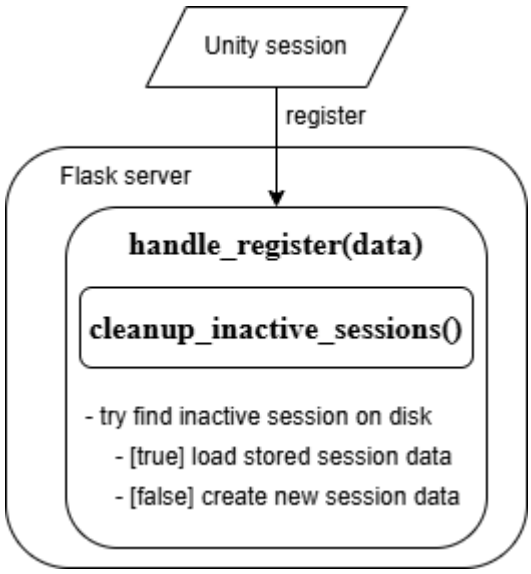
Po pripojení klienta, a to buď frontendu Vue.js alebo Unity relácie, sa klient zaregistruje. V prípade frontendu daná inštancia Vue.js odošle udalosť „register_vue“, ktorú server spracuje v metóde **handle_register_vue()**, kde identifikačné číslo danej relácie uloží do globálneho zoznamu **vue_sessions**. Tento zoznam sa neskôr používa na odosielanie aktuálnych dát na všetkých klientov. Následne sa metódou **emit_sessions_update()** na všetkých klientov odošle aktuálny zoznam aktívnych aj neaktívnych relácií.

5.2.4 Pripojenie klienta Unity

Pripojenie klienta hernej relácie Unity zabezpečuje metóda **handle_register(data)**, ktorá sa zavolá v reakcii na prijatie udalosti „register“ odoslanej z Unity. Parametrom metódy je reťazec vo formáte JSON, v ktorom je definované identifikačné číslo zariadenia, na ktorom herná relácia beží, a meno hernej relácie:

```
{
  "maxHistory_id": "31ac3153s1",
  "session_name": "Experiment 1"
}
```

Metóda najskôr zavolá, čo v prípade nedostatku miesta na serveri vymaže staré neaktívne relácie z pamäte RAM. Potom sa daná relácia podľa identifikačného čísla zariadenia hľadá na disku medzi predošlími už teraz neaktívnymi reláciami a ak sa nájde tak sa jej dáta načítajú a relácia sa uloží do zoznamu **sessions**. V opačnom prípade, ak sa jedná o novú reláciu, sa vytvorí nový záznam a vyplnia sa údaje ako čas začiatku relácie a podobne. Keď je záznam relácie založený, tak sa podľa konfiguračného súboru založia kľúče typov údajov, ktoré daná relácia odosiela, do slovníka **data** v zázname relácie. Podľa týchto kľúčov môže potom frontend zobrazovať všetky typy údajov, aj keď ešte žiadne nerpišli, čo sa tiež môže hodiť. Nakoniec sa odošle udalosť „unity_connected“ na všetky inštancie frontendu, čo v prípade, ak si niekto práve zobrazuje detail neaktívnej relácie, prepne stránku na zobrazenie detailu aktívnej relácie. Potom všetkom sa ešte odošle aktualizácia zoznamu relácií.



Obrázok 4: Priebeh pripojenia Unity klienta na server

5.2.5 Odosielanie príkazov na Unity

Server odosiela príkazy do Unity relácie prostredníctvom metódy **handle_send_command(data)**. Tá očakáva meno eventu, ktorý sa má v unity spracovať, a prípadne aj parametre, ktorými sa má daný príkaz vykonať. Vo vstupe metódy je taktiež identifikačné číslo relácie, na ktorú sa tento príkaz nakoniec odošle. Parametre sú vždy reťazce (string).

5.2.6 Prijímanie dát z Unity

Zo strany Unity zas server prijíma dáta skrz udalosť „update_data“, ktorá vyvolá metódu **handle_update_data(data)** a dáta spracuje. Tu sa taktiež najskôr, podobne ako pri registrácií relácie, prípadne uvoľní miesto tým, že sa z pamäte odstránia najstaršie neaktívne relácie. Potom sa prijaté dáta, ktoré majú tvar kľúča a hodnoty, uložia k dátam danej relácie podľa identifikačného číslo daného zariadenia a aktualizuje sa čas posledného prijatia dát. V konfiguračnom súbore sa pre každý typ dát taktiež definuje, koľko posledných záznamov sa má uchovávať. Pri presiahnutí tohto počtu sa staršie záznamy odstránia. Nakoniec sa odošle aktualizácia dát do každej pripojenej inštancie Vue.js.

5.2.7 Načítanie zoznamov relácií

Metódy **handle_get_active_sessions()** a **handle_get_inactive_sessions()** odosielajú zoznam aktívnych a neaktívnych relácií zo serveru na frontend Vue klienta. Pomocné metódy **get_active_sessions()** a **get_inactive_sessions()** tieto zoznamy vyťahujú na serveri zo spoločného slovníka popísaného na začiatku kapitoly. V prípade neaktívnych relácií sa dáta ťahajú taktiež z disku a nielen z pamäte RAM, pretože v nej nemusia byť práve všetky dostupné. Server disponuje aj metódami pre odosielanie konkrétnej aktívnej či neaktívnej relácie. Tie sa používajú pri náhľade konkrétneho detailu hernej relácie vo frontende.

5.2.8 Mazanie relácie

Po prijatí udalosti „delete_session“ sa zavolá metóda **handle_delete_session(data)**, ktorá reláciu odstráni z pamäte RAM a taktiež aj z disku. Vstupným parametrom je identifikačné číslo zariadenia, pre ktoré sa má relácia zmazať.

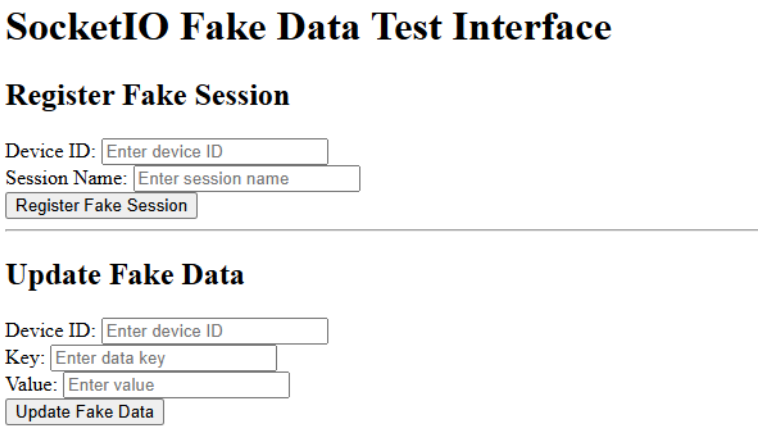
Metóda **cleanup_inactive_sessions()** slúži na uvoľňovanie miesta v pamäti RAM ak to je potrebné. Zistí sa koľko miesta momentálne server využíva a minimálne koľko voľného miesta musí byť vždy k dispozícií, čo je definované v konfiguračnom súbore. Podľa toho sa odstráni potrebný počet najstarších neaktívnych relácií. Tieto relácie sa ale odstránia iba z pamäte RAM a nie z disku. Relácie sú na disku uložené vo formáte JSON na ceste './inactive_sessions'.

5.2.9 Overovanie stavu pripojenia

Server disponuje aj jednoduchou metódou **ping()**, ktorá sa volá pre udalosť „ping“ a všetko čo robí je, že danej relácii naspäť odošle udalosť „pong“. Toto používa balíček Unity na zisťovanie aktuálneho pripojenia k serveru, ktorý sa na server pravidelne v malých intervaloch dotazuje.

5.2.10 Testovanie správy relácie

Pri dotaze na cestu „/test“ server užívateľa presmeruje na stránku „test_interface.html“, na ktorej je možné založiť testovaciu reláciu a dáta k nej. To však pre potreby testu funguje iba v lokálnom prostredí a nie na ostrom serveri.



Obrázok 5: Testovacie rozhranie

5.3 Vue.js frontend

Frontend aplikácie je implementovaný javascriptovým frameworkom Vue.js. V súbore *vite.config.js* sa nastavuje server Vite [19] pre vývoj a produkciu aplikácie. Inicializuje sa plugin Vue, ktorý umožňuje Vite spracovávať súbory *.vue*. a taktiež proxy server, ktorý presmerováva požiadavky na /socket.io na server Flask na porte 4000, čo umožňuje komunikáciu medzi frontendom a backendom. Alias @ je nastavený na adresár ./src pre jednoduchší import súborov.

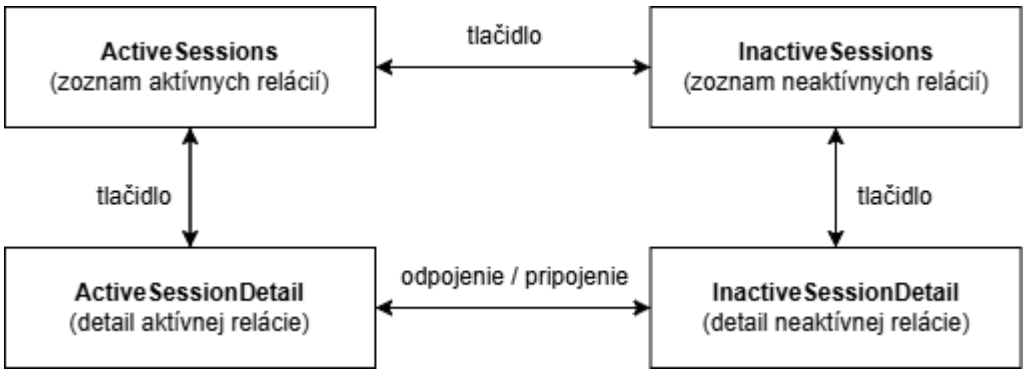
Základná štruktúra webovej aplikácie je definovaná v súbore *index.html*. Obsahuje element **div** s identifikátorom „app“, do ktorého Vue.js aplikácia vloží svoj obsah, a odkaz na hlavný JavaScript súbor na ceste */src/main.js*.

V hlavnom JavaScript súbore main.js sa najskôr importuje Vue.js, hlavná komponenta *App.vue*, router a Socket.IO klient. Potom sa vytvorí a nastaví klient Socket.IO, ktorý sa pripojí k serveru pri načítaní aplikácie a odošle správu „register_vue“, keď sa pripojenie úspešne nadviaže. Pri zatvorení stránky sa socket od serveru odpojí. Nakoniec sa inicializuje samotná Vue aplikácia, Vytvorí sa jej inštancia a poskytne socket ako globálnu premennú, aby sa naprieč používaním aplikácie používal len jeden.

5.3.1 Router

Router je definovaný v súbore *router/index,js* a umožňuje navigáciu medzi rôznymi stránkami aplikácie. Definované trasy sú:

- */activesessions*: Stránka so zoznamom aktívnych relácií, používa komponentu **ActiveSessions**.
- */activesessiondetail/:deviceId*: Detail aktívnej relácie, používa komponentu **ActiveSessionDetail** a odovzdáva **deviceId** ako parameter.
- */inactivesessions*: Stránka s neaktívnymi reláciami, používa komponentu **InactiveSessions**.
- */inactivesessiondetail/:deviceId*: Detail neaktívnej relácie, používa komponentu **InactiveSessionDetail** s parametrom **deviceId**.
- */configedit*: Stránka pre editáciu konfiguračného súboru.



Obrázok 6: Navigácia medzi stránkami

5.3.2 Zoznam aktívnych relácií

Komponenta **ActiveSessions** predstavuje súčasť používateľského rozhrania, ktorá slúži na zobrazenie zoznamu aktívnych relácií s okamžitým zobrazením zmien bez potreby obnovovať stránku. Zameriava sa na dynamické načítanie a zobrazenie dát v reálnom čase pomocou integrácie so serverom cez Socket.IO. Je navrhnutá tak, aby poskytovala používateľovi prehľad o aktuálnych reláciách spolu s možnosťou prechodu na detaily jednotlivých relácií a navigáciou k zoznamu neaktívnych relácií. Pri inicializácii komponenta odošle požiadavku „get_active_sessions“ na server cez socket, aby získala zoznam aktívnych relácií a registruje poslucháča, ktorý aktualizuje zoznam relácií pri príchode udalosti „active_sessions_update“. Pred odstránením zas komponenta odstraňuje registrovaného poslucháča, aby sa predišlo únikom pamäte.

5.3.3 Zoznam neaktívnych relácií

Komponenta **InactiveSessions** je zameraná na zobrazenie zoznamu neaktívnych relácií v používateľskom rozhraní aplikácie. Podobne ako komponenta **ActiveSessions**, aj táto využíva dynamickú komunikáciu so serverom prostredníctvom Socket.IO, čo umožňuje efektívne spracovanie dát v reálnom čase. Taktiež pri inicializácií požiada server o zoznam tentoraz neaktívnych relácií požiadavkou „get_inactive_sessions“ a registruje poslucháča pre udalosť „inactive_sessions_update“, čím sa bude zoznam relácií aktualizovať v reálnom čase bez nutnosti obnovovať stránku.

5.3.4 Detail aktívnej relácie

Zobrazenie detailu aktívnej relácie zabezpečuje komponenta **ActiveSessionDetail**. Pozostáva zo štyroch hlavných častí a to z panelu základných informácií, mapy, ovládacieho panelu a zoznamu prijatých dát. Pri načítaní stránky si komponenta od serveru vyžiada všetky dáta danej relácie podľa identifikačného čísla zariadenia, na ktorom je relácia spustená a uloží ich do premennej **sessionData**. Pri načítaní sa taktiež zaregistruje poslucháč pre udalosť „session_data_key_update“, ktorá pri prijatí nových dát z relácie serverom v reálnom čase aktualizuje dáta na stránke. Zaregistruje sa tiež aj poslucháč pre udalosť „unity_disconnected“, ktorý pri

odpojení relácie užívateľovi automaticky zobrazí detail rovnakej ale už neaktívnej relácie. Komponenta pracuje aj s konfiguračným súborom aplikácie, z ktorého si podľa mena experimentu, čiže hernej relácie, načíta zoznam podporovaných príkazov, typy prijímaných dát a obrázky máp. Z príkazov filtrovaných podľa aktuálneho kontextu sa vytvoria a vykreslia dané tlačidlá do ovládacieho panelu. Podľa aktuálneho indexu úrovne a posledných pozícií hráčov a objektov sa vykreslí mapa pomocou samostatnej komponenty **LatestPositions**. Aktuálny kontext príkazov a aktuálny index úrovne sa vytiahnu zo slovníka dát relácie a v reálnom čase sa aktualizujú, čo prekresľuje ovládací panel a mapu. Po kliknutí na tlačidlo sa serveru odošle meno príkazu aj s prípadným parametrom získaným z textového poľa vedľa tlačidla. Prijaté dáta sa v zozname taktiež aktualizujú v reálnom čase podľa toho ako na server prichádzajú. Celú hernú reláciu s dátami si je možné uložiť do súboru vo formáte JSON.

5.3.5 Detail neaktívnej relácie

Zobrazenie detailu neaktívnej relácie je podobné ako zobrazovanie detailu aktívnej relácie a zabezpečuje ho komponenta **InactiveSessionDetail**. Líši sa v tom, že neobsahuje ovládací panel ale obsahuje tlačidlo na odstránenie relácie z pamäte a zároveň aj z disku. Pri znovupripojení hernej relácie sa stránka opäť automaticky prepne na detail danej aktívnej relácie.

5.3.6 Mapa úrovne

Poslednou komponentou je komponenta **LatestPositions**, ktorá je obsiahnutá v komponentách **ActiveSessionDetail** a **InactiveSessionDetail**. Táto komponenta slúži na zobrazovanie niekoľkých posledných pozícií a rotácií hráčov a iných objektov na mape aktuálnej úrovne. Pozície sú spájané čiarou, čo tvorí nedávnu cestu pohybu hráča.

Vstupnými parametrami komponenty sú:

- **positions** – Pole pozícií a rotácií hráčov a iných objektov s indexom úrovne a hráča alebo objektu. Sú tu zahrnuté aj farby, akými sa majú pozície vykresľovať.

- **mapUrl** – Odkaz na obrázok mapy.
- **realWidth** – Skutočná šírka mapy v Unity jednotkách.
- **realHeight** – Skutočná výška mapy v Unity jednotkách.
- **maxWidth** – Maximálna šírka mapy v pixeloch, ktorá sa môže užívateľovi zobraziť.
- **maxHeight** – Maximálna výška mapy v pixeloch, ktorá sa môže užívateľovi zobraziť.
- **offsetX** – Rozdiel v súradnici x v jednotkách Unity pre kalibráciu pozície.
- **offsetY** – Rozdiel v súradnici y v jednotkách Unity pre kalibráciu pozície.
- **offsetRot** – Rozdiel v rotácii v stupňoch pre kalibráciu rotácie.

Komponenta obsahuje iba jeden element **canvas**, ktorý vykresľuje mapu a pozície, zabalený v elemente **div**. Pozície sa vytriedia podľa indexu hráča alebo objektu, ku ktorému patria. Keď sa komponenta načíta, vykreslí mapu a na ňu nakreslí pozície objektov. Pozície sú vykreslené ako body a šípky. Čiary medzi bodmi ukazujú pohyb objektu. Farba a veľkosť bodov a čiar sa prispôsobujú podľa veku pozície, novšie pozície sú väčšie a staršie menšie a vyblednuté. Smer šípky zobrazuje orientáciu objektu. Tento komponent je interaktívny, čo znamená, že reaguje na zmenu pozícií a aktualizuje vykresľovanie v reálnom čase a to pomocou vlastnosti komponenty **watch**, ktorá sleduje zmeny v zozname pozícií a zmenu odkazu na obrázok mapy. Rozmery mapy sa vypočítajú v závislosti na škálovací faktor, ktorý závisí od maximálnych rozmerov mapy a reálnych rozmerov mapy. Pre správne vykreslenie pozícií sa ukladajú aj premenné **scaleX** a **scaleY**, podľa ktorých sa pozície prepočítavajú relatívne ku konečným rozmerom mapy.

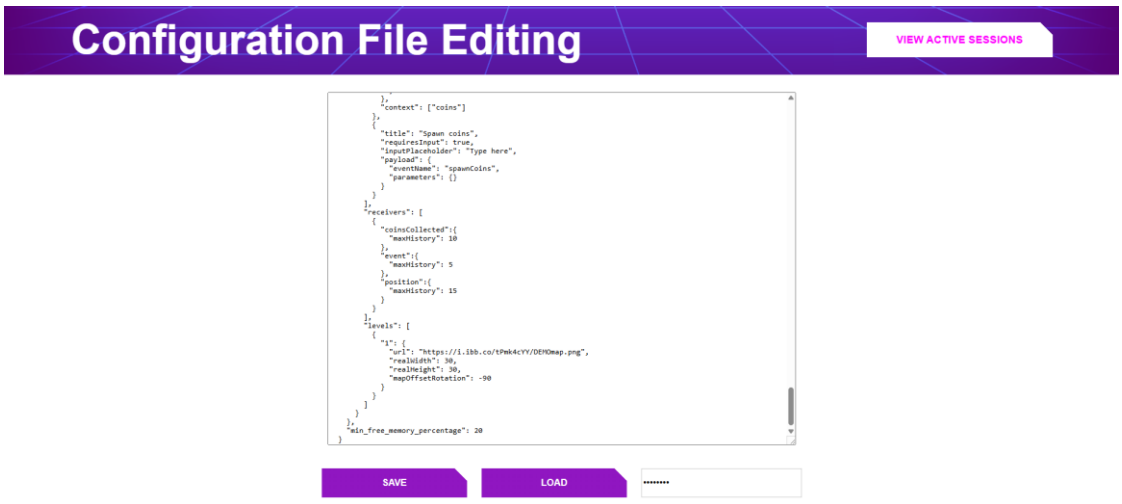
5.3.7 Editácia konfigurácie

Editáciu konfiguračného súboru zabezpečuje samostatná komponenta **ConfigEdit**. Táto stránka umožňuje užívateľovi načítať a upraviť aktuálnu konfiguráciu systému a následne ju uložiť späť na server. Na stránke sa nachádza textové pole typu textarea, do ktorého sa načítava obsah konfiguračného súboru vo formáte JSON. Užívateľ môže tento obsah upraviť a následne uložiť zmeny.

Užívateľské rozhranie obsahuje aj dve tlačidlá – SAVE a LOAD. Tlačidlo LOAD odošle požiadavku na server na načítanie aktuálneho konfiguračného súboru

spolu s heslom. Po jeho prijatí sa obsah súboru zobrazí v textovom poli. Tlačidlo SAVE odošle na server aktuálny obsah textového poľa spolu s heslom. Heslo je definované v súbore *docker-compose.yml* ako premenná prostredia (environment variable). Pred samotným odoslaním prebieha validačný proces, ktorý zabezpečuje, že JSON súbor má správnu štruktúru a obsahuje všetky potrebné polia. Validačný mechanizmus kontroluje:

- či je súbor platný JSON formát,
- či obsahuje pole *applications* typu objekt,
- pre každú aplikáciu:
    - či pole *controlButtons* je pole a každý prvok obsahuje platné pole *title*, *payload.eventName*,
    - či pole *receivers* je pole,
    - či pole *levels* je pole a každá úroveň obsahuje platné polia *url*, *realWidth* a *realHeight* s korektnými číselnými hodnotami,
- či pole *min_free_memory_percentage* je číselná hodnota v rozsahu 0 až 100.



Obrázok 7: Stránka editácie konfigurácie aplikácie

5.4 Reverzný proxy server nginx

Docker síce pri builde vytvorí vlastnú internú sieť medzi kontajnermi ale v našom prípade nie je táto sieť rozpoznávaná externým internetovým prehliadačom, ktorý by cez frontend chcel komunikovať s backend serverom. Keďže sme nikde v kóde aplikácie nechceli definovať konkrétnu adresu, na ktorej aplikácia beží, tak

sme dotazovanie sa na server vyriešili reverzným proxy serverom nginx. Ten je definovaný v súbore *proxy/default.conf*.

Počúva na porte 80. Keď sa požiadavky posielajú na koreňovú URL (/), nginx ich presmeruje na frontend aplikáciu, ktorá beží v kontajneri vue_frontend na porte 5173. Tento proxy pass znamená, že všetky požiadavky budú presmerované na http://vue_frontend:5173. Požiadavky na URL, ktorá začína na /server, budú presmerované na backendovú Flask aplikáciu, ktorá beží v kontajneri flask_app na porte 4000.

5.5 Docker

Celý systém je nasadený a riadený pomocou Docker Compose, kde každá časť beží v samostatnom kontajneri a medzi sebou komunikuje cez definované porty. Súbor je definovaný nasledovne:

```
version: "3.9"

services:
  flask_app:
    container_name: flask_app
    image: dockerhub-flask_live_app:1.0.0
    build: .
    ports:
      - "4000:4000"
    environment:
      - PASSWORD=aloha123
    volumes:
      - ./vue-frontend/src:/vue-frontend/src

  vue_frontend:
    container_name: vue_frontend
    image: node:latest
    working_dir: /app
    volumes:
      - ./vue-frontend:/app
    ports:
      - "5173:5173"
    depends_on:
      - flask_app
    command: ["/bin/sh", "-c", "npm install && npm run dev"]

  proxy:
```

```
    build: ./proxy
    ports: ['8000:80']
```

5.5.1 Backend

Flask aplikácia, ktorá slúži ako backend, beží na porte 4000. Je postavená na Docker obraze *dockerhub-flask_live_app:1.0.0*, ktorý sa vytvára podľa Dockerfile umiestneného v aktuálnom adresári. Na inštaláciu potrebných závislostí je použitý súbor *requirements.txt*, ktorý obsahuje knižnice ako flask, flask-cors, flask-socketio a psutil. Definuje sa tu aj heslo potrebné pre zmenu konfigurácie skrz frontend.

5.5.2 Frontend

Frontendová aplikácia je postavená na Vue.js a beží v kontajneri, ktorý využíva najnovší Docker obraz pre Node.js. Aplikácia sa spúšťa na porte 5173 a pracuje s kódom v adresári *./vue-frontend*, ktorý je pripojený do kontajnera. Po spustení sa nainštalujú všetky potrebné závislosti pomocou príkazu *npm install* a aplikácia sa spustí príkazom *npm run dev*. Tento frontend je závislý na backendovej Flask aplikácii, takže sa spustí až po jej inicializácii.

5.5.3 Reverzný proxy server

Reverzný proxy server nginx je nastavený tak, aby nasmeroval požiadavky prichádzajúce na port 8000 na správne služby. Nginx spracováva požiadavky a podľa cesty */server* smeruje požiadavky na Flask backend a všetky ostatné požiadavky na frontendovú aplikáciu. Tento proxy server beží na porte 80, ktorý je mapovaný na port 8000 hostiteľského systému, a používa konfiguračný súbor *default.conf* na správne nastavenie.

5.6 Testovací formulár

Na adrese „http://localhost:4000/test_interface.html“ pri lokálnom vývoji je dostupný jednoduchý testovací formulár. Možno ním overiť komunikáciu frontendu s backendom alebo správnosť zobrazovania na strane frontendu.

Formulár obsahuje vstupné textové polia pre identifikačné číslo zariadenia a meno experimentu hernej relácie, z ktorých sa po kliknutí na tlačidlo vytvorí testovacia herná relácia. Taktiež je možné vytvoriť testovacie prijaté dáta vyplnením identifikačného čísla zariadenia, pre ktoré sa majú dáta odoslať, názvu (kľúču) dát a ich hodnoty.

Na registráciu testovacej hernej relácie slúži metóda **registerFakeSession()**, ktorá si vezme dáta z formulára a odošle ich na server. V prípade testovacích prijatých dát pre hernú reláciu to spracuje a odošle metóda **updateFakeData()**.

5.7 Konfiguračný súbor

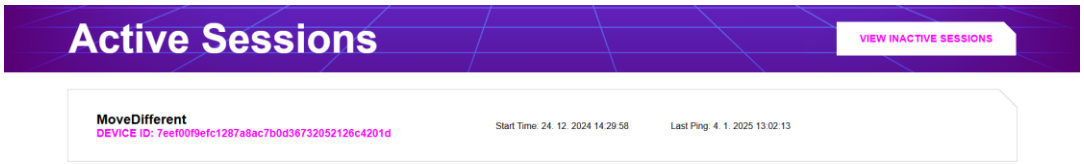
Aplikáciu je nutné nakonfigurovať pre rôzne herné relácie v konfiguračnom súbore *src/config.json*. V ňom sú definované herné relácie jednotlivých experimentov a taktiež ešte aj parameter na určenie minimálneho voľného miesta v pamäti, po ktorého presiahnutí sa pamäť uvoľní. Konfiguračný súbor je možné upravovať aj cez webové rozhranie na adrese */configedit*. Ak chýba konfigurácia niektorého z experimentov, tak sa vo webovej aplikácii nezobrazí. Štruktúra konfiguračného súboru je nasledovná:

- *"applications"*: Zoznam experimentov.
    - [meno experimentu]
        - *"controlButtons"*: Zoznam tlačidiel s príkazmi.
            - *"title"*: Názov/text tlačidla.
            - *"requiresInput"*: Či je možné zadať vlastnú hodnotu parametru v užívateľskom rozhraní. Buď „true“ alebo „false“.
            - *"inputPlaceholder"*: Zástupka v prípade vlastnej hodnoty parametru.
            - *"payload"*
                - *"eventName"*: Meno príkazu.
                - *"parameters"*: Zoznam statických parametrov príkazu.

        - *"context"*: Pole názvov kontextov, v ktorých sa bude dané tlačidlo zobrazovať. Ak nie je vyplnené tak sa zobrazí vždy.
    - *"receivers"*: Zoznam prijímaných dát.
        - [názov typu dát]
            - *"maxHistory"*: Maximálny počet najnovších uložených dát daného typu. Predvolená hodnota je 10.
    - *"levels"*: Zoznam úrovní.
        - [index úrovne]
            - *"url"*: Odkaz na obrázok mapy úrovne.
            - *"realWidth"*: Šírka mapy v Unity.
            - *"realHeight"*: Výška mapy v Unity.
            - *"mapOffsetX"*: Rozdiel v súradnici x.
            - *"mapOffsetY"*: Rozdiel v súradnici y.
            - *"mapOffsetRotation"*: Rozdiel v rotácii.
- *"min_free_memory_percentage"*: Minimálny percentuálny počet voľnej pamäte.

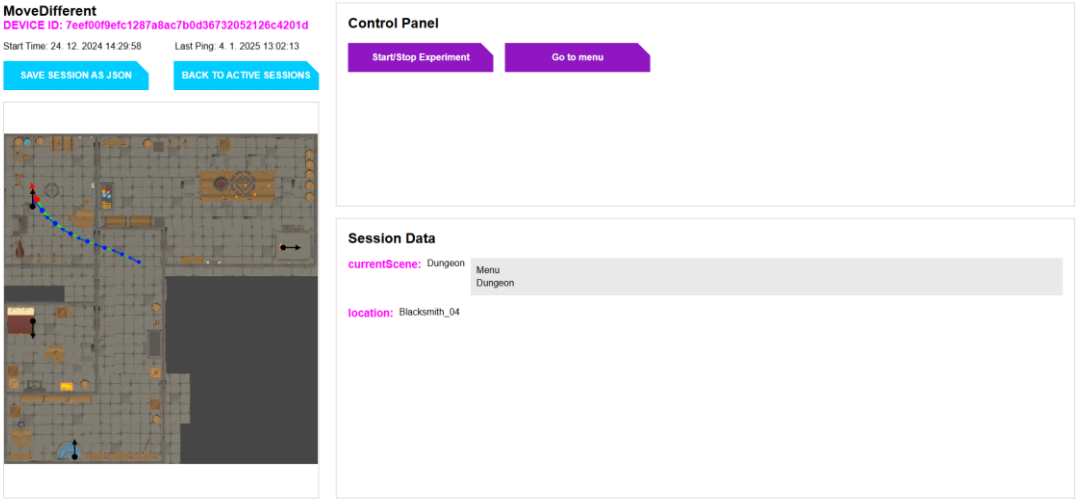
5.8 Kaskádové štýly

Štýl a vizuál aplikácie bol navrhnutý tak, aby sa čo najviac podobal stránke inštitútu https://cyberspacelab.cz/. Štýlovanie je napísané v jazyku CSS a nachádza sa všetko na jednom mieste a to v súbore *src/assets/main.css*. Boli tu použité rôzne efekty pri prechode myši cez elementy, napríklad tlačidlá, a podobne. Užívateľské grafické rozhranie sa snaží byť intuitívne, prehľadné a čo najviac jednoduché. Webová aplikácia je prispôsobená iba pre prácu v celom okne na osobnom počítači. Štýlovanie a rozloženie nie je prispôsobené na zobrazovanie na mobilnom telefóne, keďže chýba rozumné využitie.



Obrázok 8: Vzhľad stránky zoznamu aktívnych relácií

Stránky zoznamov aktívnych a neaktívnych relácií sú štýlované podobne. Ako hlavička sa zobrazuje banner s názvom stránky, s obrázkom zo stánky cyberspacelab.cz a s tlačidlom pre navigáciu na druhý zoznam. Pod ňou je zoznam herných relácii zobrazovaných ako veľké tlačidlá s názvom experimentu a časom začiatku. Detail aktívnej relácie je rozdelený na štyri časti. V ľavej hornej časti sa nachádzajú hlavné informácie o relácii. V ľavej dolnej časti sa zobrazuje mapa úrovne, ak je nejaká k dispozícii. V pravej hornej časti je ovládací panel a pod ním zoznam prijímaných dát. Detail neaktívnej relácie je podobný, až na to, že tam chýba ovládací panel.



Obrázok 9: Vzhľad stránky detailu aktívnej relácie

5.9 Unity balíček

Balíček skriptov pre Unity slúži na komunikáciu s webovou aplikáciou a je potrebné ho implementovať do projektu experimentu. Balíček pozostáva z piatich skriptov napísaných v jazyku C#. V nasledujúcich podkapitolách si každý skript podrobnejšie rozoberieme.

5.9.1 WebControllerManager

Skript *WebControllerManager.cs* obsahuje triedu **WebControllerManager**, ktorá dedí od Unity triedy **MonoBehaviour**. **MonoBehaviour** poskytuje životný cyklus Unity skriptov, ktorý je bežne používaný pri vytváraní hier a aplikácií v Unity. Tento skript slúži na pripojenie k serveru cez knižnicu Socket.IO.

Trieda používa singleton [20] vzor, čo znamená, že bude existovať len jedna inštancia tejto triedy v aplikácii. Skript je potrebné pridať k prázdnemu objektu v hlavnej scéne (menu), ktorý bude počas celej hry aktívny a nebude sa znovu vytvárať pri prechode do ďalších scén.

❖ **Atribúty triedy**

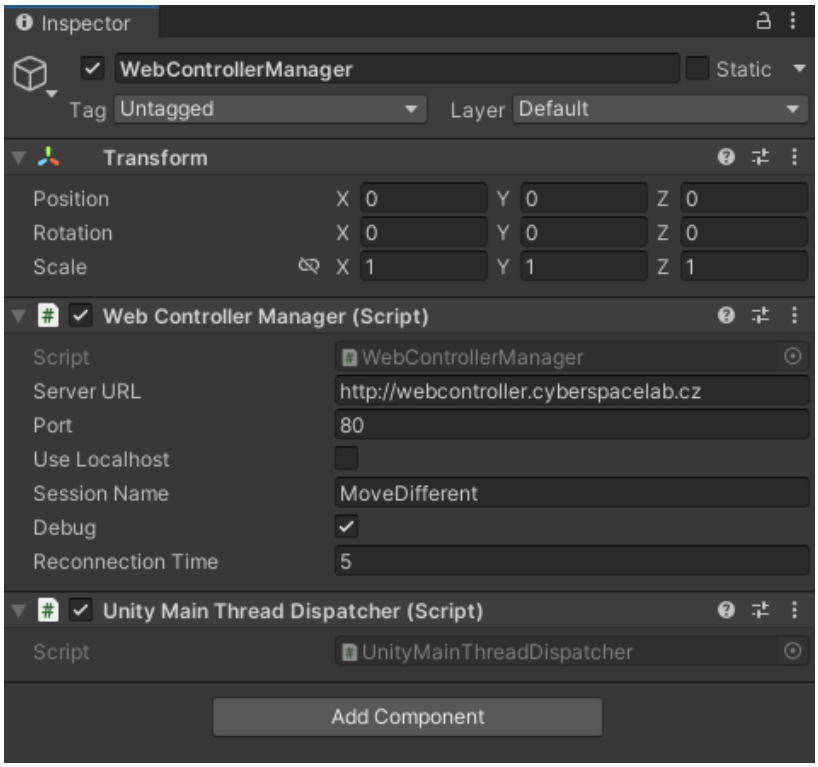
- **[DefaultExecutionOrder(-20)]** – Určuje, že tento skript sa načíta 20 ms pred inými skriptmi v scéne, čo zabezpečuje, že pripojenie k serveru prebehne ako prvé.
- **[RequireComponent(typeof(UnityMainThreadDispatcher))]** – Tento atribút zabezpečuje, že skript **UnityMainThreadDispatcher** bude vždy pripojený k objektu v hernej scéne spolu so skriptom **WebControllerManager**. Tento skript je potrebný pre vykonávanie kódu v hlavnom vlákne Unity a zabezpečuje, že všetky operácie, ktoré potrebujú interagovať s hernými objektmi alebo komponentmi Unity, budú vykonané bezpečne a správne. Atribút **[RequireComponent]** automaticky pridá skript **UnityMainThreadDispatcher** k hernému objektu, ku ktorému je pripojený skript **WebControllerManager**, ak už tento skript nie je prítomný. Toto zabezpečenie je dôležité najmä v prípadoch, keď sa používajú asynchrónne operácie alebo udalosti, ako napríklad komunikácia so serverom cez sockety. Ak by chýbal **UnityMainThreadDispatcher**, hrozilo by riziko, že kód bežiaci vo vedľajšom vlákne sa pokúsi modifikovať herné objekty, čo by mohlo viesť k nepredvídateľným chybám. Použitie **[RequireComponent]** teda eliminuje možnosť, že by vývojár alebo tím zabudol pripojiť potrebný komponent, čím sa zvyšuje spoľahlivosť a konzistentnosť správania aplikácie. **[DefaultExecutionOrder(-20)]** – Určuje, že tento skript sa

načíta 20 ms pred inými skriptmi v scéne, čo zabezpečuje, že pripojenie k serveru prebehne ako prvé.

❖ Konfigurovateľné polia v Unity

Trieda obsahuje niekoľko polí, ktoré sú konfigurovateľné priamo v inšpektore v Unity editore. Tieto polia sú:

- **string serverURL** – Verejná adresa serveru.
- **string port**: Port serveru.
- **bool useLocalHost** – Či sa má namiesto adresy používať localhost (pri testovaní).
- **string sessionName** – Meno experimentu, ak ostane nevyplnené tak sa vezme z nastavenia Unity projektu. Toto meno sa musí zhodovať s menom v konfiguračnom súbore webovej aplikácie.
- **bool debug** – Či sa majú vypisovať hlášky v konzole.
- **float reconnectionTime** – Čas v sekundách, po ktorom sa v prípade, že herná relácia nie je pripojené k serveru, pokúša znova pripojiť.



Obrázok 10: Objekt s pripojeným skriptom WebControllerManager zobrazený v inšpektore Unity a príklad vyplnenia jeho parametrov

❖ **Statické udalosti**

Trieda obsahuje aj štyri premenné, ktoré sú deklarácie statických udalostí vo forme **Action** delegátov v C#. Slúžia na to, aby umožnili rôznym častiam aplikácie reagovať na konkrétne zmeny stavu pripojenia. Tu je ich opis:

- **OnConnected**: Táto udalosť sa vyvolá vtedy, keď sa klient úspešne pripojí k serveru. Je užitočná na vykonanie akcií, ktoré závisia od aktívneho pripojenia, napríklad zobrazenie notifikácie používateľovi.
- **OnDisconnected**: Táto udalosť sa vyvolá, keď sa klient odpojí od servera. To môže byť spôsobené chybou siete, ukončením servera alebo manuálnym odpojením.
- **OnConnecting**: Táto udalosť sa spustí, keď sa klient začína pripájať k serveru. Môže sa použiť na aktualizáciu používateľského rozhrania, napríklad zobrazenie animácie načítania alebo hlásenia, že prebieha pokus o pripojenie.
- **OnReconnecting**: Táto udalosť sa vyvolá, keď klient po strate pripojenia začne opätovne nadväzovať spojenie so serverom. Môže byť užitočná pre informovanie používateľa o stave opätovného pripojenia, napríklad prostredníctvom upozornenia alebo vizuálnej indikácie.

Každá z týchto premenných je **Action**, čo znamená, že ide o delegát, ktorý nevyžaduje žiadne vstupné parametre a nevracia hodnotu. Iné časti aplikácie môžu tieto udalosti odberať priradením svojich vlastných metód, ktoré sa vykonajú, keď sa udalosť vyvolá.

❖ **Stav pripojenia**

Pre kontrolu stavu pripojenia je dostupná aj verejná vlastnosť (property) **ConnectionState** typu **eConnectionState**. Tento typ je verejnou enumeráciou implementovanou v triede **WebControllerManager**. Má štyri hodnoty, *Connected*, *Disconnected*, *Connecting* a *Reconnecting*, ktoré indikujú, v akom stave pripojenie k serveru práve je.

❖ **Inicializácia skriptu**

Pri spustení aplikácie sa v metóde **Awake()** zavolá metóda **Init()**, ktorá vykoná nasledovné:

1. Zistí identifikačné číslo zariadenia a meno projektu.
2. Vytvorí adresu servera a pripojí sa k nemu pomocou Socket.IO.
3. Nastaví poslucháčov udalostí ako **OnConnected**, **OnDisconnected**, **On("error")**, **On("pong")** a **On("registered")**.
4. Pokúsi sa pripojiť k serveru každých niekoľko sekúnd, kým sa pripojenie nepodarí, pričom používa Unity coroutines [21].

Po úspešnom pripojení sa spustí metóda **Register()**. Táto metóda zaregistruje hernú reláciu na serveri pričom odošle identifikačné číslo zariadenia a meno experimentu. Na konci životného cyklu triedy sa socket korektne odpojí a uvoľní sa pamäť.

❖ **Monitorovanie stavu pripojenia**

V Unity metóde Update volanej každý frame behu experimentu sa v intervale každé dve sekundy odošle udalosť „ping“ na server a každých šesť sekúnd sa očakáva odpoveď „pong“ so serveru. V prípade, že odpoveď v danom intervale nedostaneme predpokladáme, že sa spojenie prerušilo a pokúsime sa znova automaticky pripojiť. Pripojenie, odpojenie a znovu pripájanie k serveru si tým pádom riadi skript automaticky a nie je potrebný vonkajší zásah užívateľa.

❖ **Odosielanie dát na server**

Skrz triedu **WebControllerManager** sa taktiež posielajú všetky dáta na server a to statickou metódou **SendData(string key, string value)**, ktorú užívateľ implementujúci balíček do projektu používa na odosielanie potrebných dát. Ako parametre sa vyplnia kľúč, čo je názov typu odosielaných dát, a hodnota dát. Kľúče „position“ a „context“ sú obsadené interným používaním aplikácie a pri pokuse o ich použitie užívateľom Unity vyhodí chybovú hlášku s vysvetlením. Pre posielanie dát pozícií a aktuálneho kontextu slúžia interné metódy **SendPositionData(string value)** a **SetContext(string[] values)**. Pri odosielaní dát sa serveru odošle JSON obsahujúci kľúč, hodnotu a identifikačné číslo zariadenia.

5.9.2 UnityMainThreadDispatcher

Trieda **UnityMainThreadDispatcher** je pomocná komponenta, ktorá umožňuje vykonávať akcie v hlavnom vlákne Unity. Obsahuje statickú frontu **executionQueue**, kam sa pridávajú akcie typu **Action** na spracovanie. V metóde **Update()**, ktorá sa volá každý frame, sa všetky akcie vo fronte vykonávajú. Fronta je chránená pomocou uzamknutia *lock*, aby sa zabezpečila bezpečnosť vlákien pri pridávaní alebo odoberaní akcií. Statická metóda **Enqueue(Action action)** umožňuje pridať akciu do fronty, pričom taktiež používa uzamknutie *lock*, aby sa zabránilo kolíziám pri prístupe z rôznych vlákien. Táto trieda je užitočná v prípadoch, keď je potrebné vykonať kód, ktorý ovplyvňuje Unity komponenty, ale bol vyvolaný z iného vlákna.

5.9.3 WebControllerCommandParameters

Trieda **WebControllerCommandParameters** je implementáciou **IReadOnlyDictionary<string, object>**, ktorá poskytuje nemenný spôsob prístupu k párom kľúč-hodnota uloženým vo vnútornej súkromnej premennej **_dictionary**. Táto premenná je typu **Dictionary<string, object>** a je inicializovaná v konštruktore triedy, ktorý prijíma vstupný slovník a vytvára jeho kópiu, čím zabezpečuje, že originálny slovník nie je priamo ovplyvnený operáciami na tejto triede. Toto obalenie slovníka sa používa pri prijímaní príkazov s parametrami so serveru. Užívateľ implementujúci tento balíček do Unity projektu tým pádom môže k týmto parametrom pristupovať jednoduchšie a prehľadnejšie.

5.9.4 GenericEventListener

Trieda **GenericEventListener** poskytuje flexibilný spôsob dynamickej reakcie na udalosti prijaté zo servera v rámci Unity projektu. Tento mechanizmus umožňuje efektívnu implementáciu modelu udalostí, ktorý je často využívaný pri asynchrónnych operáciách, ako je napríklad komunikácia so serverom cez sockety. Hlavným účelom tejto triedy je umožniť jednoducho nakonfigurovať a spravovať

udalosti, ktoré prichádzajú zo servera, a priradiť im odpovede vo forme **UnityEvent** akcií.

❖ EventBag

Trieda obsahuje vnorenú triedu **EventBag**, ktorá je označená ako **Serializable**. Toto označenie umožňuje nastavenie inštancií tejto triedy priamo v Unity inšpektore, čo zjednodušuje konfiguráciu udalostí. Trieda **EventBag** obsahuje dve hlavné polia:

- **eventName** – Reťazec, ktorý reprezentuje názov udalosti, na ktorú trieda bude reagovať. Tento názov je definovaný zo strany servera a musí sa zhodovať s názvom udalosti, ktorú sa klient pokúša spracovať.
- **response** – Typ **UnityEvent<WebControllerCommandParameters>**, ktorý definuje akciu, ktorá sa má vykonať pri prijatí danej udalosti. Tento event je spustený pri príchode udalosti a vykonáva akcie ako aktualizácie používateľského rozhrania alebo vykonanie príkazov na klientovej strane.

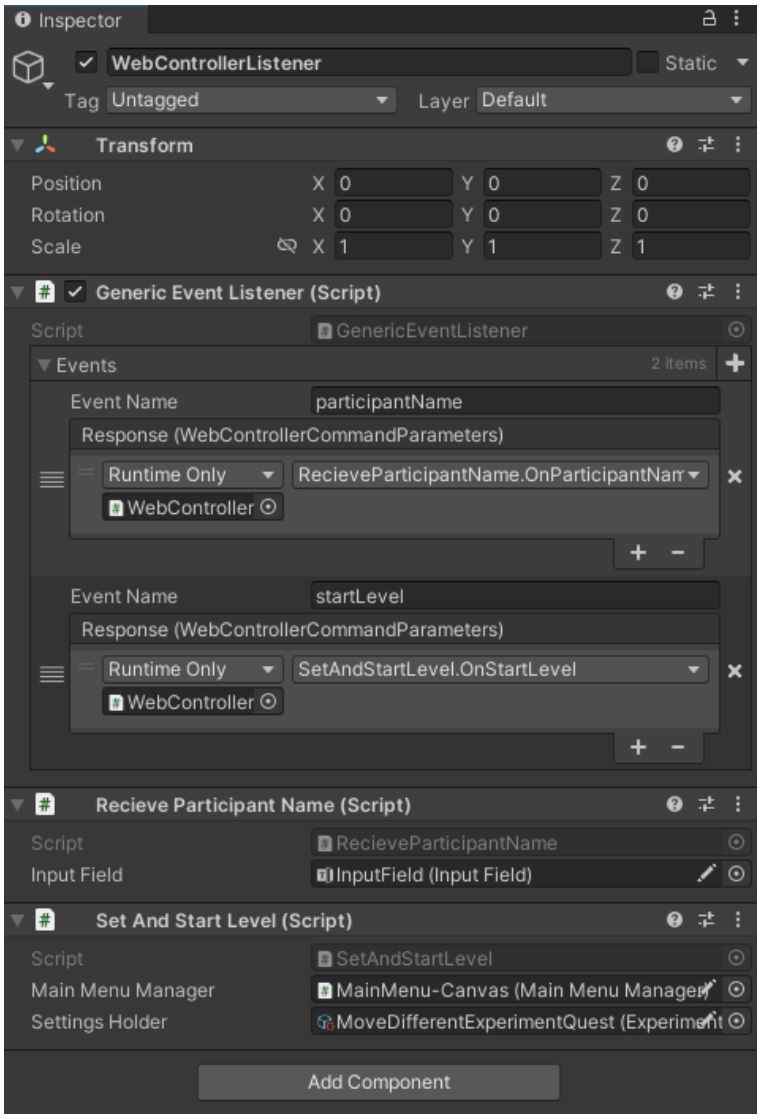
❖ Metódy OnEnable() a OnDisable()

Trieda **GenericEventListener** obsahuje dve kľúčové metódy na správu poslucháčov (listeners) udalostí:

- **OnEnable()** – Táto metóda je volaná pri aktivácii skriptu alebo GameObjectu v Unity. Pre každú definovanú udalosť sa v tejto metóde pridáva poslucháč na socket, ktorý čaká na príchod udalosti zo servera. Po prijatí udalosti sa extrahujú parametre vo forme **Dictionary<string, object>**, ktoré sú následne obalené do objektu **WebControllerCommandParameters**. Následne je táto akcia pridaná do **UnityMainThreadDispatcher**, aby bola vykonaná v hlavnom vlákne Unity.
- **OnDisable()** – Táto metóda je volaná pri deaktivácii skriptu alebo GameObjectu v Unity. Jej úlohou je odstrániť poslucháča zo socketu, čím sa zabezpečí, že po deaktivácii objektu nebudú prijímané žiadne ďalšie udalosti zo servera. Tým sa zabráni neželaným reakciám na udalosti, ktoré by mohli byť vyvolané po deaktivácii objektu.

❖ Konfigurácia v Unity Inšpektore

Trieda **GenericEventListener** je navrhnutá tak, aby bola jednoducho konfigurovateľná v Unity inšpektore. Užívateľ môže nastaviť názvy udalostí a priradiť im konkrétne odpovede, ktoré môžu zahŕňať rôzne akcie, ako napríklad aktualizácie používateľského rozhrania alebo vykonávanie príkazov. Tento mechanizmus umožňuje flexibilnú a efektívnu správu udalostí prichádzajúcich zo servera bez potreby priameho zásahu do kódu, čo výrazne zjednodušuje implementáciu. Metódy implementované užívateľom, ktoré sa zavolajú v reakcii na niektorú z prijatých udalostí sú buď bez parametru alebo ako parameter očakávajú **WebControllerCommandParameters**.



Obrázok 11: Príklad prijímania dát skriptom GenericEventListener kde akcie pre udalosti sú metódy v dvoch pripojených skriptoch

5.9.5 PositionSender

Trieda **PositionSender** periodicky odosiela údaje o pozíciách a rotáciách transformácií objektov v scéne na server cez **WebControllerManager**. Skript sleduje a odosiela informácie o viacerých objektoch, ktoré sú definované v poli **transformTrackings**. Každý objekt v tomto poli obsahuje transformáciu, ktorú sleduje, spolu s rôznymi farbami, ktoré určujú vizuálne vlastnosti týchto objektov. Tieto objekty sa jednoducho referencujú cez inšpektor spolu s nastavením daných farieb.

❖ **Premenné**

- **currentLevelID** – Identifikátor aktuálnej úrovne.
- **intervalInSeconds** – Interval v sekundách, ktorý určuje, ako často sa odosielajú údaje na server.
- **transformTrackings** – Pole objektov typu **TransformTrackingBag**, ktoré obsahuje referencie na transformácie sledovaných objektov a ich vizuálne vlastnosti.

❖ **Štruktúry**

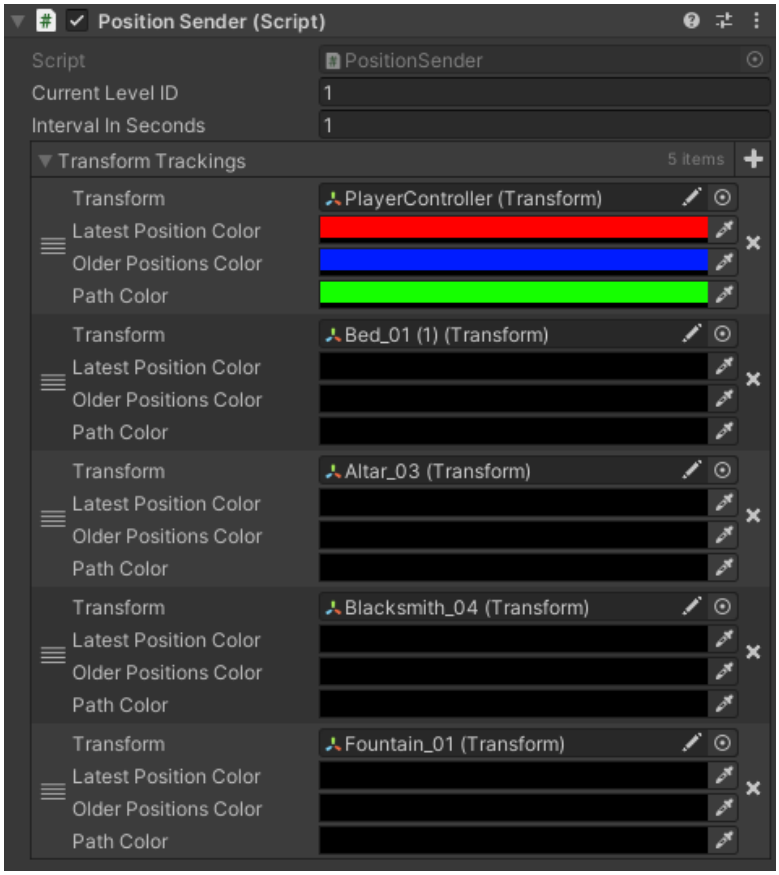
- **PositionBag** – uchováva informácie o pozícii a rotácii objektu, vrátane jeho **x** a **z** pozície, **y** rotácie a troch rôznych farieb: pre najnovšiu pozíciu, staršie pozície a cestu. Okrem toho sa v tejto štruktúre nachádza aj identifikátor úrovne a index objektu. Tieto údaje sa následne serializujú do formátu, ktorý je možné odoslať na server.
- **TransformTrackingBag** –obsahuje transformáciu sledovaného objektu (pozícia a rotácia) a tri farby:
  - **latestPositionColor** – Farba pre najnovšiu pozíciu objektu.
  - **olderPositionsColor** – Farba pre staršie pozície objektu.
  - **pathColor** – Farba, ktorá zobrazuje cestu objektu.
- **SerializableColor** – slúži na serializáciu farieb z Unity systému (typ **Color**) do formátu, ktorý môže byť odoslaný cez sieť. Farby sú uložené ako hodnoty **r, g, b, a**, pričom každá z týchto hodnôt je normalizovaná na

rozsah 0 až 255. Táto štruktúra umožňuje bezpečné prenášanie farieb ako číselné hodnoty a ich opätovné rekonštruovanie na strane servera.

❖ Sledovanie a odosielanie pozícií

V metóde **Start()** sa nastaví opakované volanie metódy **SendPositions()** podľa definovaného intervalu, ktorý je určený v **intervalInSeconds**. Táto metóda pre každý objekt v **transformTrackings** vytvorí inštanciu **PositionBag**, ktorá obsahuje pozíciu a rotáciu objektu, ako aj farby pre najnovšiu pozíciu, staršie pozície a cestu. Tieto údaje sa potom serializujú do JSON formátu pomocou **JsonConvert.SerializeObject()** a odosielajú na server pomocou **WebControllerManager.SendPositionData()**.

Skript stačí pripojiť na objekt v scéne a cez inšpektor nastaviť referencie sledovaných objektov, ich farby a index úrovne.



Obrázok 12: Skript PositionSender v inšpektore sa príkladom nastavenia parametrov a s referenciou na objekt hráča a ďalšie objekty v scéne

6. Funkcionalita systému

Táto kapitola popisuje jednotlivé časti používateľského rozhrania webovej aplikácie, ako aj funkcionalitu, ktorú ponúka pri práci s hernými reláciami. Prehľadne sú tu rozdelené možnosti zobrazenia aktívnych aj neaktívnych relácií, ich detailov a možností interakcie s nimi. Cieľom bolo používateľovi poskytnúť jednoduchý, ale zároveň dostatočne bohatý nástroj na monitorovanie a správu experimentov.

6.1 Prehľad aktívnych herných relácií

Zoznam práve pripojených herných relácií je dostupný na ceste "*.../activesessions*". Každá položka obsahuje meno experimentu, identifikačné číslo zariadenia, na ktorom experiment beží, čas začiatku pripojenia a čas poslednej zmeny údajov. Po kliknutí na niektorú z položiek si užívateľ môže zobraziť detail danej hernej relácie. Zo zoznamu aktívnych herných relácií sa je možné prekliknúť na zoznam neaktívnych herných relácií.

6.2 Prehľad neaktívnych herných relácií

Zoznam neaktívnych herných relácií je dostupný na ceste ".../inactivesessions". Zobrazujú sa tu všetky minulé pripojenia. Každá položka obsahuje meno experimentu, identifikačné číslo zariadenia, na ktorom experiment bežal, čas začiatku pripojenia a čas poslednej zmeny údajov. Po kliknutí na niektorú z položiek si užívateľ môže zobraziť detail danej hernej relácie. Zo zoznamu neaktívnych herných relácií sa je možné prekliknúť na zoznam aktívnych herných relácií.

6.3 Detail aktívnej relácie

Detail aktívnej hernej relácie je dostupný skrz preklik zo zoznamu aktívnych herných relácií. Detail obsahuje meno experimentu, identifikačné číslo zariadenia, na ktorom je experiment spustený, čas začiatku experimentu a čas poslednej zmeny údajov. Pod základnými údajmi v ľavej časti obrazovky sa zobrazuje mapa ak je

práve dostupná. V pravej časti sa nachádza ovládací panel a pod ním zoznam prijatých údajov. Hernú reláciu je možné uložiť vo formáte JSON.

6.4 Detail neaktívnej relácie

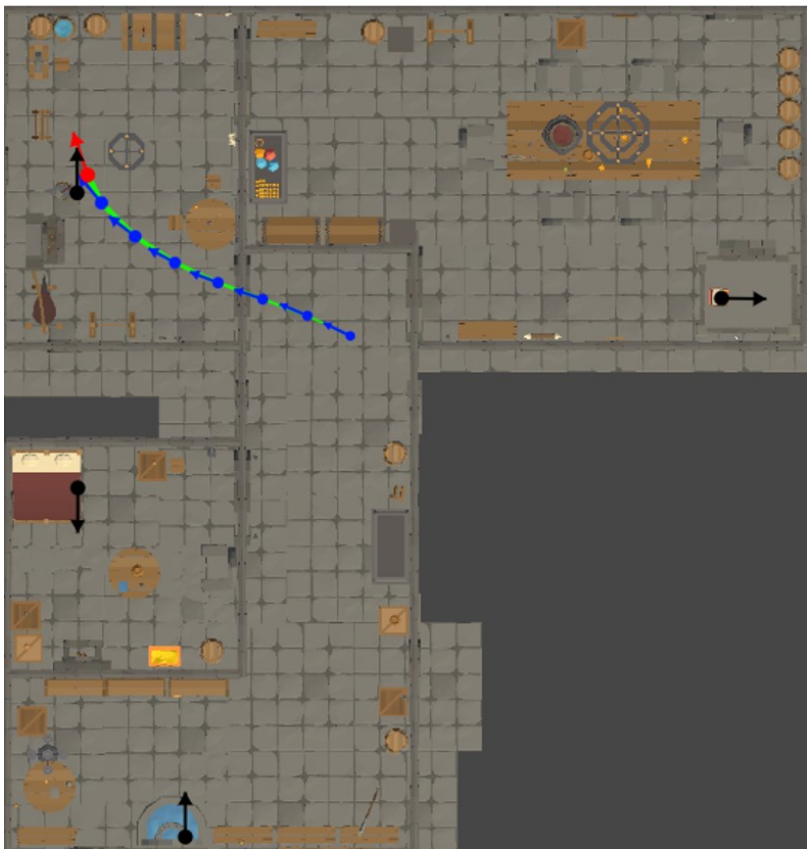
Detail neaktívnej hernej relácie je dostupný skrz preklik zo zoznamu neaktívnych herných relácií. Detail obsahuje meno experimentu, identifikačné číslo zariadenia, na ktorom bol experiment spustený, čas začiatku experimentu a čas poslednej zmeny údajov. V pravej časti obrazovky sa nachádza zoznam prijatých údajov počas behu experimentu. Hernú reláciu je možné zmazať a aj uložiť vo formáte JSON.

6.5 Mapa aktuálnej úrovne

V prípade, že je dostupná mapa aktuálnej úrovne v experimente, tak sa zobrazí v ľavej časti obrazovky v detaile hernej relácie aj s aktuálnou pozíciou a rotáciou všetkých sledovaných hráčov a predmetov. Na mape sa taktiež zobrazujú predošlé pozície a cesta, ktorá ich spája.

Aktuálne pozície sú od tých predošlých rozlíšené veľkosťou šípky, ktorá značí pozíciu s rotáciou, a farbou. Pozície každého sledovaného hráča a objektu môžu byť taktiež rozlíšené odlišnými farbami. Pozície sa aktualizujú v reálnom čase podľa preddefinovaného intervalu.

Obrázok mapy sa taktiež mení v reálnom čase podľa aktuálnej úrovne. Úroveň je označená pomocou indexu, čiže zmenou indexu je možné aj napríklad zmeniť mapu pri zmene poschodia, na ktorom sa hráč práve nachádza a podobne, nie je to obmedzené iba na zmenu úrovne.



Obrázok 13: Príklad mapy v aplikácii. Červená šípka zobrazuje aktuálnu pozíciu hráča a modré šípky predošlé pozície hráča. Zelené čiary zobrazujú cestu pohybu hráča. Čierne body so šípkami zobrazujú pozície ďalších objektov.

6.6 Ovládací panel, kontext a odosielanie príkazov

Ovládací panel sa nachádza v detaile aktívnej hernej relácie a obsahuje všetky tlačidlá pre ovládanie experimentu, ktoré sú dostupné v aktuálnom kontexte. Herná relácia môže odosielať kontext podľa toho v akom stave sa práve nachádza. Podľa neho sa v ovládacom panely zobrazia iba tie tlačidlá, ktoré pre daný kontext dávajú zmysel. Tlačidlá môžu byť definované pre viacero kontextov zároveň a taktiež môže byť validných viacero kontextov naraz. Tlačidlá bez definovaného kontextu sa budú zobrazovať vždy.

Tlačidlá odošlú príkaz do hernej relácie buď s parametrami alebo bez. Parametre môžu byť statické, predom nadefinované v konfiguračnom súbore, alebo dynamické. Pre tlačidlo s dynamickým parametrom je vedľa neho dostupné textové pole, do ktorého sa parameter napíše, čo znamená, že sa parametre vždy odosielajú ako reťazce.

Príkazmi je možné napríklad v reálnom čase spúšťať alebo zastavovať experiment, presunúť hráča na konkrétnu úroveň alebo resetovať jeho pozíciu ak sa stratí.

6.7 Prijímanie dát

Prijaté dáta z hernej relácie sa zobrazujú v detaile aktívnej aj neaktívnej hernej relácie. Sú roztriedené podľa svojho typu. Typy prijímaných dát sú definované v konfiguračnom súbore, kde sa taktiež definuje maximálny počet uložených predošlých záznamov daného typu. Dáta sú prijímané a zobrazované ako reťazce. Vedľa každého názvu typu údajov sa zobrazuje posledná prijatá hodnota, ak existuje. Po kliknutí na ňu sa zobrazí pole predošlých prijatých hodnôt a po opätovnom kliknutí sa toto pole znova skryje.

Herná relácia tak môže odosielať viaceré užitočné dáta v reálnom čase ako napríklad meno úrovne, v ktorej sa hráč nachádza, informácie o aktuálnom stave experimentu alebo dosiahnuté ciele hráča v experimente a podobne.

6.8 Ukladanie herných relácií

Pri odpojení sa hernej relácie sa z aktívnej relácie stáva neaktívna relácia, ktorá sa automaticky uloží na disk na serveri. Tieto herné relácie sa tak prechovávajú na disku aj po vypnutí alebo reštartovaní aplikácie. Následne sú vždy znova načítané ak je to potrebné.

6.9 Zmazanie záznamu hernej relácie

Záznam neaktívnej hernej relácie je možné vymazať kliknutím na dané tlačidlo v detaile danej neaktívnej relácie. Takto zmazaný záznam už nie je možné obnoviť

6.10 Automatické čistenie pamäte

Pri prekročení minimálneho voľného miesta v pamäti definovaného v konfiguračnom súbore sa z pamäte automaticky odstránia najstaršie neaktívne herné relácie. Na disku však ostávajú aj naďalej uložené.

6.11 Uloženie hernej relácie vo formáte JSON

Dáta hernej relácie je možné si lokálne stiahnuť z detailu danej relácie, či už aktívnej alebo neaktívnej, kliknutím na dané tlačidlo. Dáta sú serializované a uložené vo formáte JSON. V tomto súbore sú obsiahnuté všetky dáta vrátane posledných pozícií kontextov pre tlačidlá.

6.12 Konfigurácia aplikácie

Webovú aplikáciu je možné konfigurovať skrz konfiguračný súbor. V ňom je potrebné nadefinovať všetky typy herných relácií, ktoré sa k aplikácií budú pripájať. Definujú sa tu tlačidlá s príkazmi na odoslanie, prijímané dáta, mapy úrovní a minimálny percentuálny počet voľnej pamäte aplikácie. Konfigurácia je vo formáte JSON.

Definícia tlačidiel obsahuje ich názov, názov príkazu, parametre, príznak, či sú parametre dynamické, zástupný text textového políčka pre vstup parametru a kontext, v ktorom sa má tlačidlo zobraziť.

Definícia prijímaných dát obsahuje názov typu dát a maximálny počet histórie daných dát, ktorého predvolená hodnota je desať. Po prijatí väčšieho množstva sa najstaršie záznamy odstránia.

Definícia máp obsahuje adresu odkazu obrázku mapy, reálnu výšku a šírku mapy, odchýlky v pozícii a rotácii a index úrovne, pre ktorú sa má daná mapa zobrazovať.

Okrem priamej úpravy súboru je možné konfiguráciu pohodlne upravovať aj cez webové rozhranie aplikácie. V tomto prípade je obsah súboru zobrazený v textovom poli (textboxe), kde ho môže používateľ priamo editovať ako text. Po potvrdení zmien sa aktualizovaný obsah uloží a aplikuje. Tento prístup ponecháva flexibilitu formátu JSON, pričom zároveň zjednodušuje prístup ku konfigurácii bez potreby externého editora alebo priameho prístupu k súborom na serveri.

6.13 Sledovanie stavu pripojenia

Unity balíček poskytuje udalosti stavu aktuálneho pripojenia, ktoré je možné odpočúvať a následne na ne reagovať. Dostupné udalosti sú pripojenie k serveru, odpojenie, prvotné pripájanie a znovu pripájanie po strate pripojenia. Taktiež je dostupná verejná enumerácia aktuálneho stavu pripojenia. Užívateľ tak môže implementovať notifikácie v grafickom užívateľskom prostredí alebo adekvátne reakcie v experimente.

6.14 Podpora viacerých klientov

Webová aplikácia podporuje pripojenie a ovládanie viacerých herných relácií zároveň. Taktiež sú podporované viaceré inštancie frontendu aplikácie zároveň. V aplikácií môžu byť definované viaceré odlišné typy experimentov alebo iných herných relácií.

6.15 Testovanie aplikácie

Na adrese „http://localhost:4000/test_interface.html“ pri lokálnom vývoji je dostupný jednoduchý testovací formulár. Skrz tento formulár je možné vytvoriť falošnú hernú reláciu lebo falošné prijaté dáta. Možno tak jednoducho overiť komunikáciu frontendu so serverom a zároveň aj vzhľad frontendovej aplikácie a zobrazovanie údajov v nej.

7. Použitie v experimente

Pre demonštrovanie funkčnosti webovej aplikácie s Unity balíčkom bol balíček implementovaný do už existujúceho experimentu v Unity s názvom Move Different. [22]. Riešenie sa nachádza v prílohe A.1.

7.1 Popis experimentu

Experiment sa zaoberá pohybom hráča vo virtuálnej realite. Skúma koordináciu hráča vo virtuálnom prostredí pri rôznych spôsoboch pohybu a premiestňovania sa. Pre hráča sú pripravené viaceré úrovne, kde sa vždy ocitne v inom prostredí, napríklad v rodinnom dome, byte alebo v stredovekej pivnici. Hráč má za úlohu nájsť niekoľko predmetov a dotknúť sa ich. Ak ich všetky nájde, tak bude premiestnený na náhodnú pozíciu v danom objekte, pričom celé prostredie a všetky predmety zmiznú až na jeden styčný bod. Teraz musí hráč čo najpresnejšie ukázať na smer, v ktorom sa nachádzajú predmety, ktoré predtým hľadal, pričom sa už nesmie pohybovať. Experiment zbiera viaceré dáta ako napríklad presnosť a rýchlosť hráča.

Obrázok 14: Ukážka behu experimentu MoveDifferent. Hráč sa nachádza v miestnosti a hľadá predmety s fialovým obrysom.

7.2 Prepojenie s webovou aplikáciou

Pre demonštráciu pridáme do webovej aplikácie niekoľko príkazov, ktorými budeme riadiť hlavný chod experimentu. Samotný experiment zas bude odosielať polohu a rotáciu hráča a taktiež polohu objektov, ktoré hráč musí nájsť. Pri zmene scény alebo úrovne sa odošle meno aktuálnej scény. Ak sa hráč dotkne niektorého z predmetov, tak sa do aplikácie odošle meno daného predmetu. V aplikácií pripravíme mapu pre každú úroveň. Z aplikácie budeme taktiež môcť vyplniť meno hráča v experimente, vybrať a spustiť konkrétnu úroveň alebo spustiť a ukončiť daný experiment.

7.3 Konfigurácia webovej aplikácie

Na strane webovej aplikácie je potrebné iba doplniť konfiguráciu pre tento experiment v konfiguračnom súbore. Pridáme tlačidlá na ovládanie experimentu a typy prijímaných dát, ktoré nám bude experiment odosielať. Čo sa týka tlačidiel, konkrétne pridáme jedno aj s možnosťou vyplnenia dynamického parametra pre nastavenie mena hráča, a potom po dvoch tlačidlách pre každú úroveň na jej spustenie. Dve tlačidlá preto, lebo každé tlačidlo spustí úroveň s iným spôsobom pohybu hráča. Tieto tlačidlá budú mať kontext pre hlavné menu. Každé tlačidlo bude mať statický parameter indexu danej úrovne. Potom pridáme jedno tlačidlo pre spustenie a zastavenie experimentu a jedno tlačidlo pre návrat do hlavného menu. Tieto tlačidlá budú mať kontext úrovne. Prijímané dáta definujeme pre meno aktuálnej scény, meno práve nájdeného objektu a pre pozície hráča a objektov.

Nakoniec nastavíme mapy pre každú z úrovní. Obrázky máp úrovní získame tak, že v editore Unity v každej úrovni odstránime objekt strechy daného prostredia, kameru situujeme kolmo nad scénu, pričom bude nastavené izometrické zobrazovanie, a zhotovíme snímok scény. V konfiguračnom súbore potom ešte doladíme odchýlky v pozíciách a rotáciách, ak sa nejaké objavia.

Obrázok 15: Hlavné menu experimentu MoveDifferent

7.4 Implementácia Unity balíčka

Na strane Unity projektu doplníme do scény hlavného menu prázdny objekt, ktorý bude mať na sebe pripojený skript **WebControllerManager**. V ňom nastavíme pripojenie na server. Pridané skripty budeme umiestňovať do priečinka „*/Assets/Movedifferent/Scripts*“.

Do každej scény pridáme jednoduchý skript **SendCurrentSceneName**, ktorý pri spustení danej scény na server odošle jej meno. Do už existujúceho skriptu **MoveDifferentLearning** pridáme do metódy **LocationFound(TargetLocation location)**, ktorá sa zavolá zakaždým ak hráč nájde daný predmet, riadok **WebControllerManager.SendData("location", location.name)**, ktorý odošle meno predmetu na server. Skript **GoToMenu** vložíme do scény každej úrovne ako súčasť nového objektu v hernej scéne, na ktorom bude aj skript **GenericEventListener**. Ten pri zaznamenaní príkazu zo serveru skrz skript **GoToMenu** prepne scénu a hráča premiestni do hlavného menu. Podobne bude fungovať aj skript **StartStopExperiment**, ktorý po príchode príkazu experiment buď spustí alebo zastaví.

Skripty **ConnectionText**, **SetAndStartLevel** a **RecieveParticipantName** budú v hlavnom menu. **ConnectionText** bude sledovať stav pripojenia pomocou

WebContollerManager skriptu a aktualizovať nápis informujúci o pripojení v užívateľskom grafickom rozhraní. **SetAndStartLevel** po príchode príkazu spustí konkrétnu úroveň podľa prijatého indexu a **RecieveParticipantName** nastaví meno hráča, ktoré prijme od serveru.

Nakoniec bol do každej scény pridaný aj skript **PositionSender**, v ktorom boli nastavené referencie na objekt hráča a hľadané objekty spolu s ich farbami. Tento skript bude posielať ich polohy a rotácie na server.

7.5 Výsledok

Na základe integrácie webovej aplikácie a Unity balíčka do experimentu *Move Different* bolo možné overiť, že riešenie napĺňa všetky požiadavky stanovené v analytickej časti práce. Aplikáciu experimentu sme nainštalovali na dve zariadenia Oculus Quest 2 a taktiež sme ju testovali aj na počítači s Windows 11. V nasledujúcom prehľade sumarizujeme jednotlivé požiadavky a ich konkrétne naplnenie v kontexte tohto experimentu:

❖ **Požiadavky na webovú aplikáciu:**

1. **Podpora viacerých experimentov naraz**
    - Riešenie umožňuje súčasné pripojenie viacerých inštancií experimentu. V praxi sme testovali beh experimentu súčasne na zariadení Oculus Quest 2 a na počítači s Windows, pričom obe relácie bolo možné riadiť z webovej aplikácie.

2. **Zobrazenie zoznamu aktívnych a neaktívnych experimentov**
    - Webová aplikácia rozlišuje medzi aktívnymi a neaktívnymi experimentmi. Po ukončení experimentu sa jeho záznam presunie medzi neaktívne a je k dispozícii na prezeranie alebo stiahnutie dát.

3. **Interakcia s experimentom cez ovládací panel**
    - V aplikácii boli nakonfigurované príkazy pre spúšťanie experimentu, prepínanie úrovní, nastavovanie mena hráča a návrat do hlavného

menu. Tieto príkazy bolo možné posielať v reálnom čase a okamžite sa prejavili v experimente.

4. **Zobrazenie detailu experimentu**
    - V detaile experimentu sa zobrazujú informácie o aktuálnom stave, prijaté dáta (napr. meno scény, pozície objektov, meno nájdeného objektu) a ovládací panel s dostupnými príkazmi. Mapy jednotlivých úrovní s pozíciami hráča a objektov boli úspešne zobrazované v reálnom čase.

5. **Ukladanie dát experimentu do súboru**
    - Všetky prijaté dáta sa počas behu experimentu bolo možné po ukončení experimentu stiahnuť vo formáte JSON alebo vymazať.

6. **Konfigurácia aplikácie pomocou konfiguračného súboru**
    - Všetky aspekty interakcie s experimentom boli nakonfigurované v konfiguračnom súbore – od definície príkazov, cez prijímané dátové typy až po mapy jednotlivých úrovní. Súbor bolo možné upravovať aj priamo cez webové rozhranie.

7. **Vizuálny štýl aplikácie**
    - Aplikácia bola navrhnutá v súlade s dizajnom webu https://cyberspacelab.cz a poskytuje intuitívne, vizuálne konzistentné používateľské rozhranie.

❖ **Požiadavky na Unity balíček:**

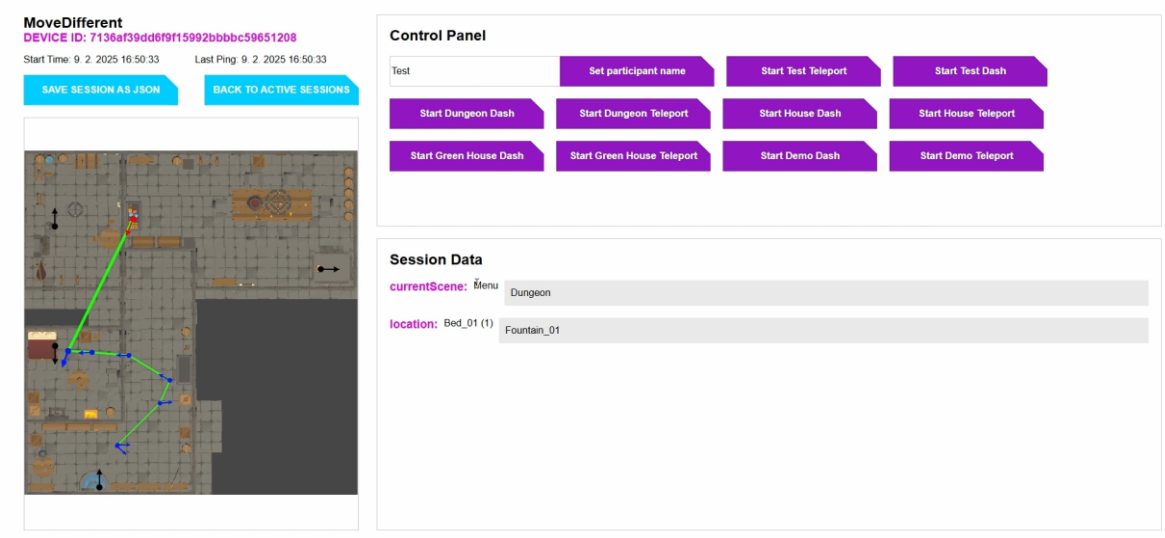
1. **Jednoduchá implementácia do nových aj existujúcich projektov**
    - Balíček bol úspešne implementovaný do existujúceho experimentu bez potreby rozsiahlych zásahov do jeho pôvodnej architektúry. Väčšina implementácie prebehla v editore pomocou predpripravených skriptov skrz inšpektor.

2. **Automatické nadviazanie spojenia s webovou aplikáciou**
    - Po spustení experimentu došlo automaticky k pripojeniu na server bez potreby manuálneho zásahu.

3. **Obnova spojenia po prerušení**
    - V prípade krátkodobého výpadku pripojenia sa balíček automaticky pokúsil o obnovenie spojenia. Toto správanie sa potvrdilo počas testovania pri výpadku internetu na oboch stranách spojenia.

4. **Monitorovanie a zobrazenie stavu pripojenia**
    - V hlavnom menu experimentu sa nachádzal indikátor stavu pripojenia, ktorý informoval používateľa o aktuálnom stave komunikácie s webovou aplikáciou.

5. **Obojsmerná komunikácia medzi Unity a webovou aplikáciou s podporou Unity editora**
    - Experiment prijímal príkazy zo serveru (napr. zmena úrovne, nastavenie mena, štart a stop experimentu) a odosielal späť údaje o hráčovi, scénach a nájdených objektoch.



Obrázok 16: Detail pripojenej relácie experimentu MoveDifferent vo webovej aplikácii

7.6 Externé využitie v praxi

Navrhnuté riešenie bolo úspešne využité aj iným výskumníkom v rámci samostatného experimentu. Študent Pavel Srp z Fakulty humanitních studií Karlovej univerzity, odbor teoreticko-výskumná psychológia, implementoval webovú aplikáciu aj Unity balíček do vlastného experimentálneho výskumu. Išlo o pilotný test v reálnych podmienkach.

Webové rozhranie mu umožnilo efektívne zadávať údaje o účastníkoch priamo z ovládacieho panela, bez potreby ich manuálneho zadávania v prostredí Unity. Zároveň mal počas experimentu prístup k informáciám o priebehu testovania v reálnom čase, vrátane polohy a správania účastníka vo VR prostredí, čo výrazne uľahčilo jeho kontrolu a dohľad nad experimentom.

Podľa spätnej väzby od výskumníka bola implementácia riešenia rýchla a bezproblémová vďaka prehľadnej a vecnej dokumentácii. Riešenie zároveň eliminovalo problémy, ktoré sa vyskytli pri predchádzajúcom výskume bez použitia tohto systému. Webová aplikácia bola vyhodnotená ako flexibilná a vhodná aj pre širšie využitie, napríklad počas popularizačných akcií, ktoré si vyžadujú jednoduché a spoľahlivé riadenie VR obsahu.

8. Záver

V riešení boli implementované všetky hlavné požiadavky definované na začiatku práce. Webová aplikácia aj Unity balíček boli úspešne otestované v rôznych prostrediach a na viacerých zariadeniach súčasne, pričom bolo overené ich spoľahlivé fungovanie. Riešenie bolo integrované aj do reálneho experimentu, kde sa potvrdila jeho praktická použiteľnosť. Spätná väzba potvrdila, že riešenie prispelo k vyššej efektivite a lepšiemu prehľadu o správaní účastníkov v reálnom čase.

Aplikácia poskytuje prehľadné a efektívne rozhranie na monitorovanie a riadenie experimentov v reálnom čase, a to najmä vo virtuálnej realite, pri diaľkových online experimentoch, ako aj pri tradičných experimentoch. História herných relácií umožňuje jednoduchý prístup k údajom z predchádzajúcich experimentov. Unity balíček je navrhnutý tak, aby bolo jeho nasadenie do existujúcich projektov jednoduché a zásah do pôvodného kódu minimálny.

Výsledkom práce je ucelený a prakticky využiteľný nástroj, ktorý výrazne uľahčuje a zrýchľuje realizáciu behaviorálnych experimentov a prispieva k zefektívneniu výskumného procesu.

Na základe týchto výsledkov možno riešenie považovať za funkčný a prakticky využiteľný nástroj pre podporu realizácie experimentov vo virtuálnom prostredí, s možnosťou ďalšieho rozšírenia a adaptácie aj mimo akademického výskumu.

8.1 Možnosti využitia

Aj keď pôvodným impulzom pre vývoj Unity balíčka bola potreba vytvoriť spoľahlivú infraštruktúru pre experimenty vo virtuálnej realite (VR), jeho potenciál siaha ďaleko za hranice tohto konkrétneho kontextu. Flexibilita, modularita a dôraz na univerzálnosť z neho robia nástroj, ktorý možno uplatniť v širokej škále scenárov – či už výskumných, testovacích, alebo vývojárskych.

8.1.1 VR experimenty

V prostredí virtuálnej reality balíček umožňuje v reálnom čase zaznamenávať správanie používateľa, jeho pohyb v priestore a interakcie s objektmi. Kombinácia

presného zaznamenávania pozícií a rýchlej spätnej väzby od servera robí z tohto nástroja výbornú platformu pre psychologické, behaviorálne alebo UX experimenty realizované v imerzívnom prostredí.

8.1.2 Online experimenty

Vďaka sieťovej komunikácii cez Socket.IO a možnosti vzdialeného monitoringu sa balíček výborne hodí aj na realizáciu experimentov na diaľku. Subjekty môžu spúšťať Unity aplikácie zo svojich počítačov alebo VR headsetov, zatiaľ čo výskumník sleduje priebeh experimentu v reálnom čase cez webové rozhranie.

8.1.3 Lokálne experimenty

Balíček možno bez problémov použiť aj pri klasických experimentoch realizovaných priamo v laboratórnych podmienkach – bez potreby VR zariadení či vzdialenej komunikácie. Umožňuje zaznamenávať priebeh experimentu, logovať správanie používateľa alebo priebežné dáta z prostredia a zároveň poskytovať vizuálnu spätnú väzbu priamo v rámci aplikácie.

Takéto využitie je vhodné napríklad pri behaviorálnych alebo kognitívnych testoch, edukatívnych aktivitách či výcvikových simuláciách, kde je dôležité precízne sledovať reakcie používateľa a zároveň ich zaznamenávať pre ďalšiu analýzu.

8.1.4 Herné testovanie

Okrem výskumného využitia môže byť balíček cenným nástrojom aj v hernom priemysle. Môže slúžiť na:

- zber údajov o správaní hráča počas testovania prototypu,
- analýzu interakcie s herným prostredím,
- detekciu technických problémov pri hraní,
- získanie kvalitatívnych aj kvantitatívnych dát pre dizajnové rozhodnutia.

Webové rozhranie môže slúžiť nielen ako nástroj na pasívne pozorovanie, ale aj na aktívne ovládanie – napríklad spúšťanie udalostí, zmenu prostredia, prepínanie stavov hry či simulovanie vonkajších podnetov.

8.2 Možnosti budúceho rozšírenia

Aplikácia ponúka mnohé možnosti na riadenie a sledovanie experimentov. Stále je tu ale priestor pre vylepšenie a pre pridanie novej funkcionality. Ako prvé ma napadá pridanie kontextu pre prijímané dáta rovnako ako už existuje pre tlačidlá v ovládacom paneli. Tým by sa pri veľkom množstve rôznych prijímaných dát ich prehľad zjednodušil.

Pridanie mapovania identifikačného čísla zariadenia k nejakej poznávacej značke, či už názvu alebo obrázku, do konfiguračného súboru, by zjednodušilo rozpoznávanie aktuálne aktívnych herných relácií. Zariadenie by na sebe mohlo mať nejaké označenie, napríklad nálepku, a to by slúžilo k rýchlejšej a lepšej identifikácií ako jeho identifikačné číslo.

Zoznam použitej literatúry

[1] Bohil, C. J., Alicea, B., & Biocca, F. A. (2011). Virtual reality in neuroscience research and therapy. *Nature Reviews Neuroscience*, 12(12), 752–762.

[2] Riva, G., & Wiederhold, B. K. (2015). The new dawn of virtual reality in health care: medical simulation and experiential interface. *Cyberpsychology, Behavior, and Social Networking*, 18(9), 559–561.

[3] Freeman, D., Reeve, S., Robinson, A., et al. (2017). Virtual reality in the assessment, understanding, and treatment of mental health disorders. *Psychological Medicine*, 47(14), 2393–2400.

[4] Radianti, J., Majchrzak, T. A., Fromm, J., & Wohlgenannt, I. (2020). A systematic review of immersive virtual reality applications for higher education: Design elements, lessons learned, and research agenda. *Computers & Education*, 147, 103778.

[5] JSON.org. (n.d.). *JavaScript Object Notation*. Dátum citovania: 15. 4. 2025, dostupné na: https://www.json.org/

[6] Brookes, J., Warburton, M., Alghadier, M., Mon-Williams, M., & Mushtaq, F. (2020). Studying human behavior with virtual reality: The Unity Experiment Framework. *Behavior Research Methods*, 52(2), 455–463. https://doi.org/10.3758/s13428-019-01242-0

[7] Unity Technologies. (n.d.). Unity Analytics Documentation. Dátum citovania: 15. 4. 2025, dostupné na: https://docs.unity.com/ugs/manual/analytics/manual/overview

[8] GameAnalytics. (n.d.). Unity SDK Documentation. Dátum citovania: 15. 4. 2025, dostupné na: https://docs.gameanalytics.com/integrations/sdk/unity/

[9] Mixpanel. (n.d.). Mixpanel SDKs: Unity. Dátum citovania: 15. 4. 2025, dostupné na: https://docs.mixpanel.com/docs/tracking-methods/sdks/unity

[10] Docker Inc. (n.d.). *What is Docker?* Dátum citovania: 15. 4. 2025, dostupné na: https://docs.docker.com/get-started/overview/

[11] Docker Inc. (n.d.). *Docker Compose Documentation*. Dátum citovania: 15. 4. 2025, dostupné na: https://docs.docker.com/compose/

[12] nginx.org. (n.d.). *nginx Documentation*. Dátum citovania: 15. 4. 2025, dostupné na: https://nginx.org/en/docs/

[13] Fette, I., & Melnikov, A. (2011). The WebSocket Protocol (RFC 6455). *Internet Engineering Task Force (IETF)*. Dátum citovania: 15. 4. 2025, dostupné na: https://datatracker.ietf.org/doc/html/rfc6455

[14] Fielding, R. T. (2000). *Architectural Styles and the Design of Network-based Software Architectures* (doktorandská dizertačná práca, University of California, Irvine). Dátum citovania: 15. 4. 2025, dostupné na: https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

[15] Socket.IO. (n.d.). *Socket.IO Documentation*. Dátum citovania: 15. 4. 2025, dostupné na: https://socket.io/docs/v4/

[16] Flask-SocketIO. (n.d.). *Flask-SocketIO Documentation*. Dátum citovania: 15. 4. 2025, dostupné na: https://flask-socketio.readthedocs.io/

[17] Socket.IO. (n.d.). *Client API documentation (v4.x)*. Dátum citovania: 15. 4. 2025, dostupné na: https://socket.io/docs/v4/client-api/

[18] Najim, I. (n.d.). *SocketIOUnity* [softvérové úložisko]. GitHub. Dátum citovania: 15. 4. 2025, dostupné na: https://github.com/itisnajim/SocketIOUnity

[19] Vite. (n.d.). *Vite Documentation*. Dátum citovania: 15. 4. 2025, dostupné na: https://vite.dev/

[20] Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.

[21] Unity. (n.d.). *Unity Coroutine Documentation*. Dátum citovania: 15. 4. 2025, dostupné na: https://docs.unity3d.com/6000.0/Documentation/ScriptReference/Coroutine.html

[22] Hejtmánek, L. (2025). *Unity VR experiment Move Different*. GitLab. Dátum citovania: 15. 4. 2025, dostupné na: https://gitlab.com/cyberspace-lab/movedifferent

Zoznam obrázkov

A. Prílohy

A.1 Elektronická príloha

Elektronická príloha obsahuje:

- Zdrojové kódy webovej aplikácie WebControllerManager a Unity balíčku v priečinku *source_code*.
- Programátorskú dokumentáciu.
- Aplikačný súbor experimentu MoveDifferent na ukážkovú funkcionalitu webovej aplikácie a textový súbor *Readme.txt* s návodom na inštaláciu a s adresou webovej aplikácie v priečinku *experiment*.

A.2 Užívateľská dokumentácia

Webová aplikácia WebController umožňuje riadenie a sledovanie experimentov vytvorených v Unity. Poskytuje prehľad o aktívnych a neaktívnych herných reláciách. Užívatelia môžu prezerať detailné informácie o jednotlivých reláciách, odosielať im rôzne príkazy, prijímať od nich dáta a zobrazovať pozície hráčov na mape. Komunikácia medzi aplikáciou a hernými reláciami je zabezpečená prostredníctvom Unity balíčka, ktorý je potrebné integrovať do Unity projektu. Webová aplikácia je podporovaná iba pre používanie na osobnom počítači v móde zobrazenia celého okna.

Prehľad Architektúry

Aplikácia je postavená na klient-server architektúre a skladá sa z nasledujúcich komponentov:

- **Frontend**: Implementovaný pomocou Vue.js, zobrazuje aktívne a neaktívne relácie, detaily relácií a poskytuje ovládacie prvky na interakciu s Unity reláciami.
- **Backend**: Flask aplikácia, ktorá spravuje relácie, zabezpečuje WebSocket komunikáciu s klientmi a frontendom.
- **Unity klient**: Herné relácie v Unity predstavujú klientov, ktorí komunikujú so serverom pomocou implementovaného balíčka.
- **Proxy server**: NGINX reverzný proxy server na správne smerovanie požiadaviek.
- **Dockerizované prostredie**: Všetky komponenty sú kontajnerizované pre jednoduché nasadenie a škálovateľnosť.

Predpoklady

Pred začatím práce sa uistite, že máte nainštalované nasledujúce:

- **Docker**
- **Docker Compose**
- **Git** (na klonovanie repozitára)

Inštalácia aplikácie

1. Klonovanie repozitára:
   - *git clone https://github.com/cyberspace-lab/cyberframe-webcontroller.git*
   - *cd cyberframe-webcontroller*

2. Vytvorenie a spustenie Docker kontajnerov:
   - *docker-compose up --build*
   - Tento príkaz vykoná:
     - Vytvorenie obrazov pre Flask backend a Vue frontend.
     - Spustenie Nginx reverzného proxy servera.
     - Sprístupnenie potrebných portov.

Konfigurácia aplikácie

Aplikáciu je potrebné nakonfigurovať pre každý experiment na stránke */configedit* alebo priamo v súbore „*./vue-frontend/src/config.json*“, ktorý má nasledovnú štruktúru:

- **applications**: Slovník aplikácií s ich konfiguráciami.
  - **Názov aplikácie** (napr. "MoveDifferent"):
    - **controlButtons**: Definuje tlačidlá, ktoré odosielajú príkazy Unity klientom.
    - **receivers**: Určuje kľúče dát a limity histórie, ktoré backend očakáva od Unity klientov.
    - **levels**: Obsahuje informácie o rôznych úrovniach/scénach vrátane obrázkov máp a reálnych rozmerov.
- **min_free_memory_percentage**: Minimálne percento voľnej pamäte, ktoré je potrebné pred začiatkom čistenia neaktívnych relácií.

Definícia tlačidla vyzerá nasledovne:

- **title**: Názov tlačidla.
- **requiresInput**: Definuje, či tlačidlo vyžaduje dynamický vstup parametru.
- **inputPlaceholder**: Zástupka vstupného parametru.
- **payload**:
  - **eventName**: Meno príkazu.

    - **parameters**: Parametre príkazu.
- **context**: Obsahuje zoznam mien kontextov, v ktorých sa bude tlačidlo zobrazovať. Ak sa pole nevyplní, tak sa bude tlačidlo zobrazovať vždy.

Definícia prijímaných dát:

- **Názov typu dát** (napr. „currentScene“):
    - **maxHistory**: Počet predošlých záznamov, ktoré budú uložené. Predvolená hodnota je 10.

V prípade, že chcete zobrazovať mapu úrovne a sledovať na nej pozície hráča, je potrebné ako prijímaný dát vydefinovať aj typ „position“. Tento typ je interný, ostatné typy sú vlastné.

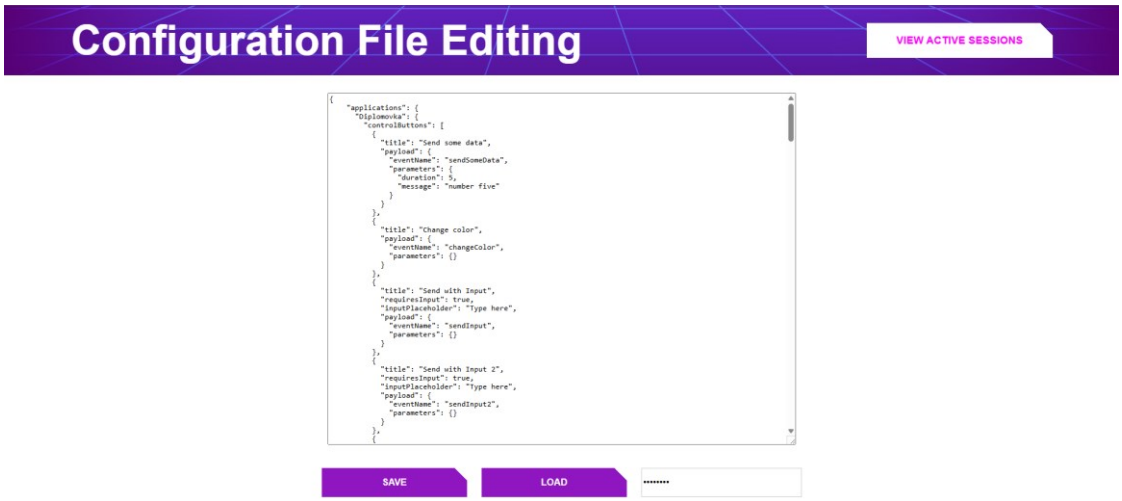
Definícia mapy úrovne:

- **Index úrovne** (napr. „1“):
    - **url**: Adresa odkazu obrázku mapy.
    - **realWidth**: Skutočná šírka mapy v jednotkách Unity.
    - **realHeight**: Skutočná výška mapy v jednotkách Unity.
    - **mapOffsetX**: Rozdiel pozície v súradnici x pre kalibráciu.
    - **mapOffsetY**: Rozdiel pozície v súradnici y pre kalibráciu.
    - **mapOffsetRotation**: Rozdiel rotácie v stupňoch pre kalibráciu.

Príklad definície konfigurácie:

```
{
    "applications": {
      "MoveDifferent": {
        "controlButtons": [
          {
            "title": "Set participant name",
            "requiresInput": true,
            "inputPlaceholder": "Participant",
            "payload": {
              "eventName": "participantName",
              "parameters": {}
            },
            "context": ["menu"]
          },
          {
```

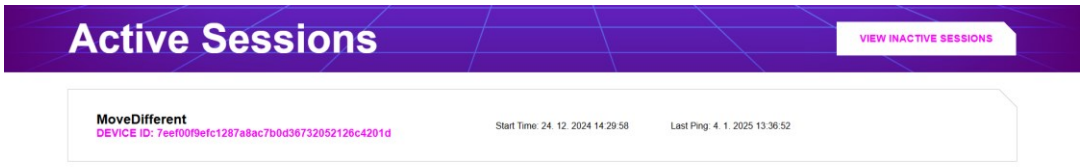
```
            "title": "Start Test Teleport",
            "payload": {
              "eventName": "startLevel",
              "parameters": { "settingsIndex": 0 }
            },
            "context": ["menu"]
          },
          {
            "title": "Start/Stop Experiment",
            "payload": {
              "eventName": "startStopExperiment",
              "parameters": {}
            },
            "context": ["level"]
          }
        ],
        "receivers": [
          {
            "currentScene":{ "maxHistory": 10 },
            "location":{ "maxHistory": 20 },
            "position":{ "maxHistory": 10 }
          }
        ],
        "levels": [
          {
            "1": {
              "url": "https://i.ibb.co/PCLSBCJ/map-dungeon.png",
              "realWidth": 25.5,
              "realHeight": 26.5,
              "mapOffsetX": -0.7,
              "mapOffsetY": 3.3,
              "mapOffsetRotation": -90
            }
          }
        ]
      }
    },
    "min_free_memory_percentage": 20
}
```



Obrázok 1: Stránka pre editáciu konfigurácie aplikácie

Zoznam aktívnych relácií

Na adrese „{ADRESA_APLIKÁCIE}/activesessions“ sa zobrazuje zoznam všetkých momentálne pripojených herných relácií. Zoznam obsahuje meno relácie, identifikačné číslo zariadenia, na ktorom je herná relácia spustená, čas pripojenia a čas poslednej aktualizácie údajov. Kliknutím na položku danej hernej relácie budete presmerovaný na stránku jej detailu. Kliknutím na tlačidlo v hornom pravom rohu „VIEW INACTIVE SESSIONS“ budete presmerovaný na zoznam neaktívnych herných relácií.



Obrázok 2: Stránka zoznamu aktívnych relácií

Zoznam neaktívnych relácií

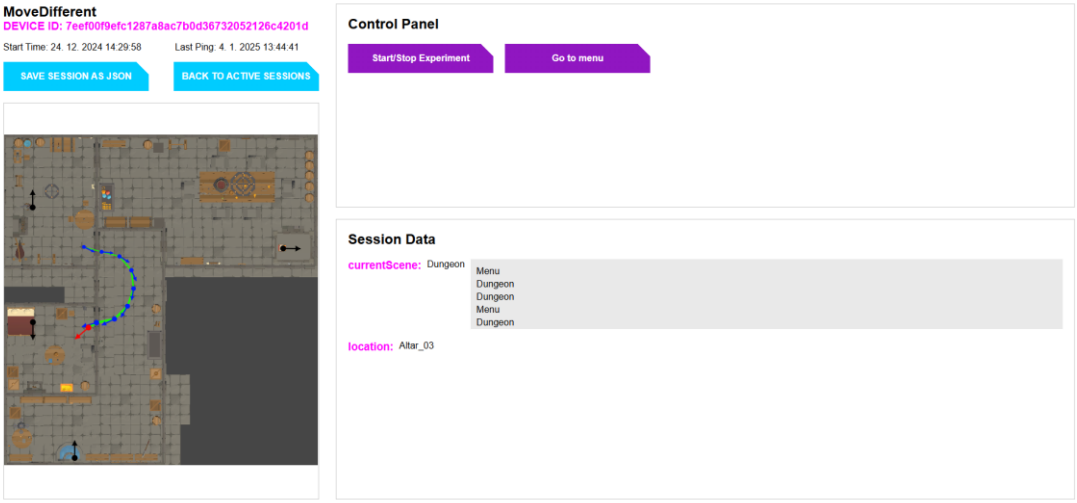
Na adrese „{ADRESA_APLIKÁCIE}/inactivesessions“ sa zobrazuje zoznam všetkých minulých práve nepripojených herných relácií. Zoznam obsahuje meno relácie, identifikačné číslo zariadenia, na ktorom bola herná relácia spustená, čas pripojenia a čas poslednej aktualizácie údajov. Kliknutím na položku danej hernej relácie budete presmerovaný na stránku jej detailu. Kliknutím na tlačidlo v hornom pravom rohu „VIEW ACTIVE SESSIONS“ budete presmerovaný na zoznam aktívnych herných relácií.

Obrázok 3: Stránka zoznamu neaktívnych relácií

Detail aktívnej relácie

Detail hernej relácie je dostupný preklikom zo zoznamu herných relácií. Stránka je rozdelená na štyri časti. V ľavom hornom rohu môžete vidieť názov experimentu, identifikačné číslo zariadenia, na ktorom je herná relácia spustená, čas pripojenia, čas poslednej aktualizácie údajov, tlačidlo pre návrat na zoznam aktívnych herných relácií a tlačidlo pre uloženie dát hernej relácie vo formáte JSON. V ľavej dolnej časti sa zobrazuje mapa aktuálnej úrovne aj s pozíciami a rotáciami sledovaných hráčov a úrovní. Mapa sa aktualizuje v reálnom čase. V pravej hornej časti sa nachádza ovládací panel a v ňom tlačidlá pre príkazy definované v konfiguračnom súbore. Po kliknutí na tlačidlo sa daný príkaz odošle do hernej relácie. Ak má dynamický parameter, tak sa vedľa tlačidla zobrazí textové pole, do ktorého je možné parameter napísať a následne kliknutím na tlačidlo odoslať. V pravej dolnej časti sa zobrazuje zoznam všetkých prijatých dát z hernej relácie okrem pozície (tá sa zobrazuje na mape alebo je možné si ju stiahnuť s ostatnými

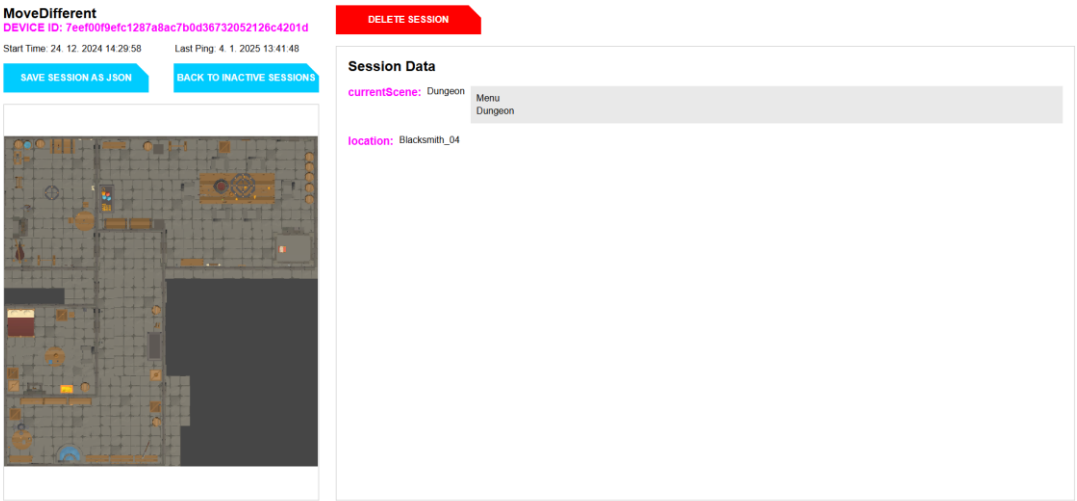
dátami relácie). Pre zobrazenie alebo skrytie predošlých prijatých dát je potrebné kliknúť na dané dáta. Prijaté dáta sa aktualizujú v reálnom čase.



Obrázok 4: Stránka detailu aktívnej relácie

Detail neaktívnej relácie

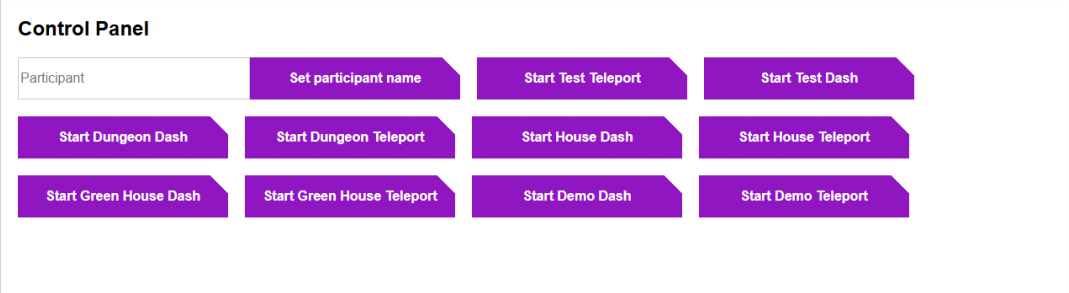
Detail neaktívnej hernej relácie je dostupný preklikom zo zoznamu neaktívnych herných relácií. Je rovnaký ako detail aktívnej hernej relácie až na dve odlišnosti. Nezobrazuje sa tu ovládací panel, keďže nepripojené reláciu nie je možné ovládať. Pribudlo tu tlačidlo pre odstránenie hernej relácie z pamäte a disku serveru. Odstránenie nie je možné vrátiť späť.



Obrázok 5: Stránka detailu neaktívnej relácie

Ovládací panel

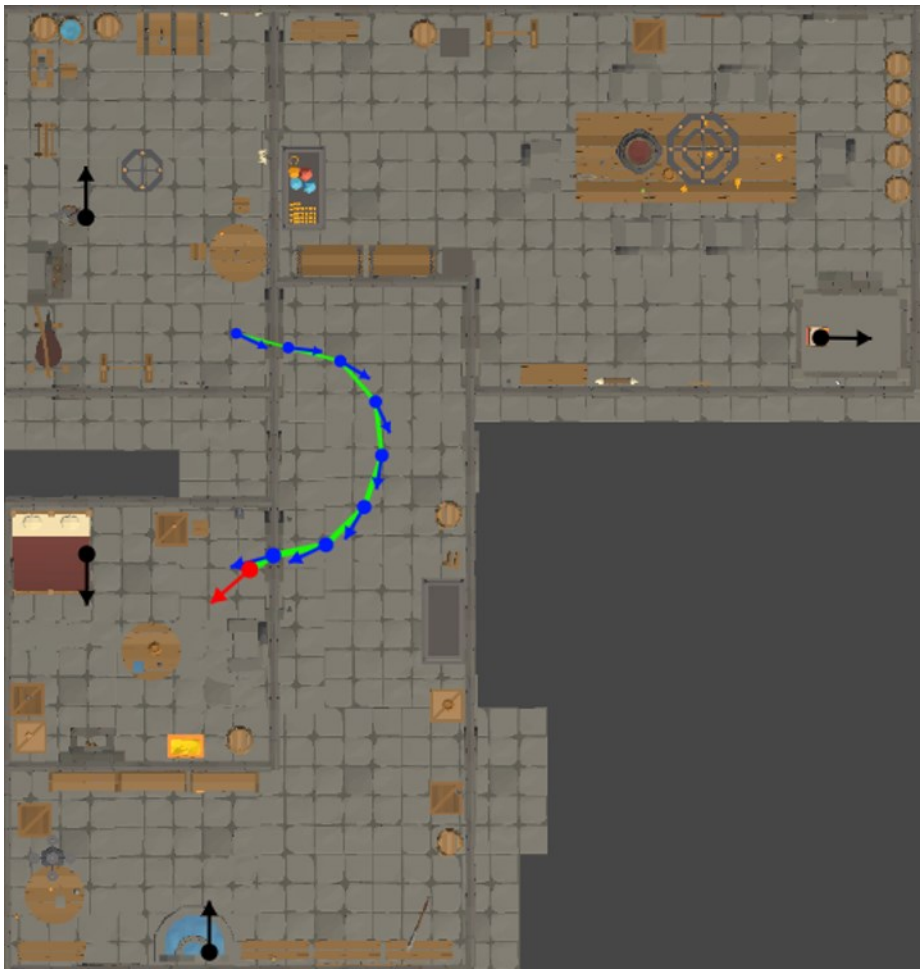
Ovládací panel sa nachádza v pravej hornej časti stránky detailu aktívnej hernej relácie. Zobrazujú sa tu všetky tlačidlá pre aktuálny kontext. V prípade, že tlačidlo očakáva vyplnenie parametru, sa vedľa neho zobrazí aj textové pole, kam je možné parameter zadať. Po kliknutí sa príkaz tlačidla, prípadne aj s parametrom, odošle na server. Parameter je buď dynamický z textového poľa alebo statický definovaný v konfiguračnom súbore. Príkaz môže byť ak bez parametru alebo v prípade statického parametru ich môže byť aj viacero.



Obrázok 6: Ovládací panel v detaile aktívnej relácie

Mapa a sledovanie pozícií

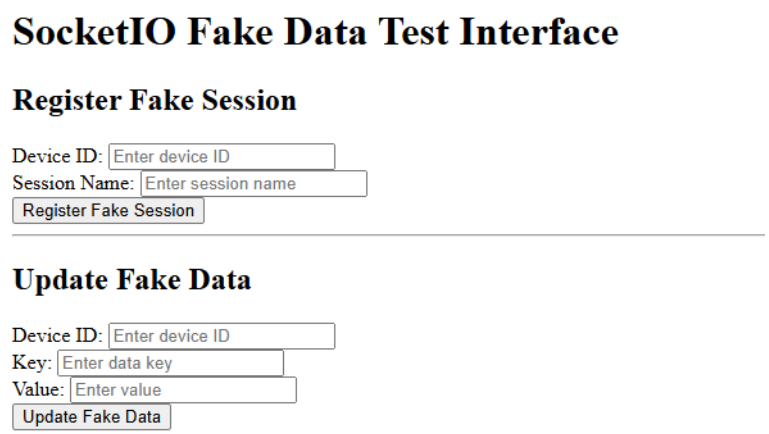
Mapa sa zobrazuje v ľavej dolnej časti na stránke detailu relácie. V reálnom čase zobrazuje pozície a rotácie všetkých sledovaných hráčov a objektov. Pri zmene úrovne sa obrázok mapy automaticky aktualizuje. Pre zobrazenie mapy je potrebné v konfiguračnom súbore definovať typ prijímaných dát „position“. Na mape sa zobrazujú posledné pozície a aj niekoľko predošlých pozícií, ktorých počet sa definuje v konfigurácii. Pozície sú spojené čiarou, ktorá zobrazuje cestu pohybu. Každá pozícia je zobrazená ako bodka so šípkou znázorňujúcou rotáciu. Staršie pozície sú vykresľované vždy menšími bodkami a šípkami. V balíčku Unity je možné definovať rôzne farby pre každý sledovaný objekt a taktiež rôzne farby pre najnovšiu pozíciu, staršie pozície a cestu pohybu. V balíčku Unity sa deifinuje aj interval odosielania pozícií.



Obrázok 7: Mapa aktuálnej úrovne zobrazuje pohyb hráča a polohy objektov v scéne

Testovanie s falošnými dátami

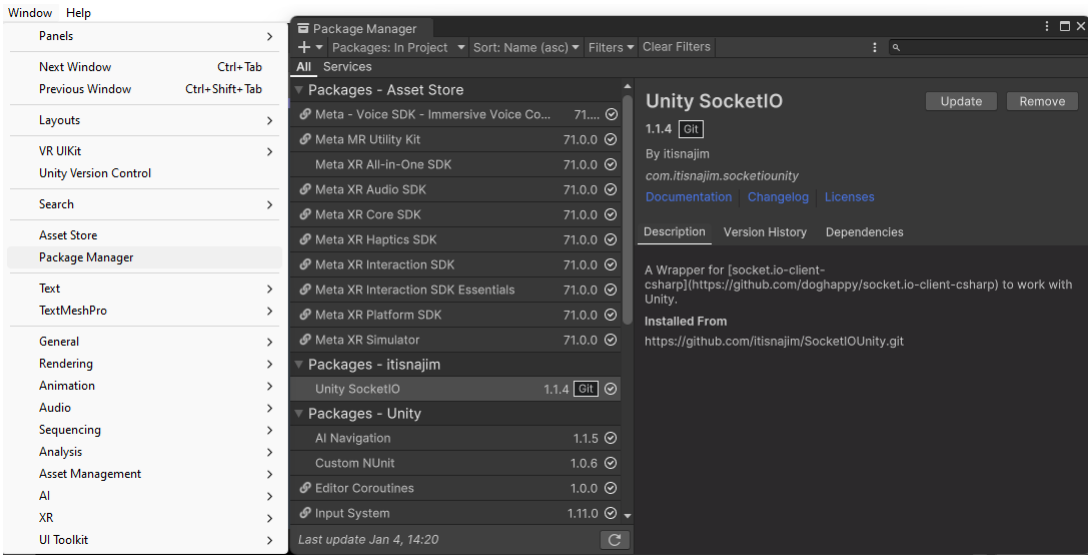
Komunikáciu frontendu s backendom a taktiež zobrazovanie dát vo frontende je možné zľahka otestovať na adrese *http://localhost:4000/test* pri lokálnom vývoji. Vyplnením falošného názvu experimentu a falošného identifikačného čísla zariadenia je možné na serveri založiť falošnú reláciu. Taktiež je možné na konkrétnu reláciu podľa identifikačného čísla zariadenia odoslať falošné dáta.



Obrázok 8: Testovacie rozhranie

Integrácia Unity balíčka

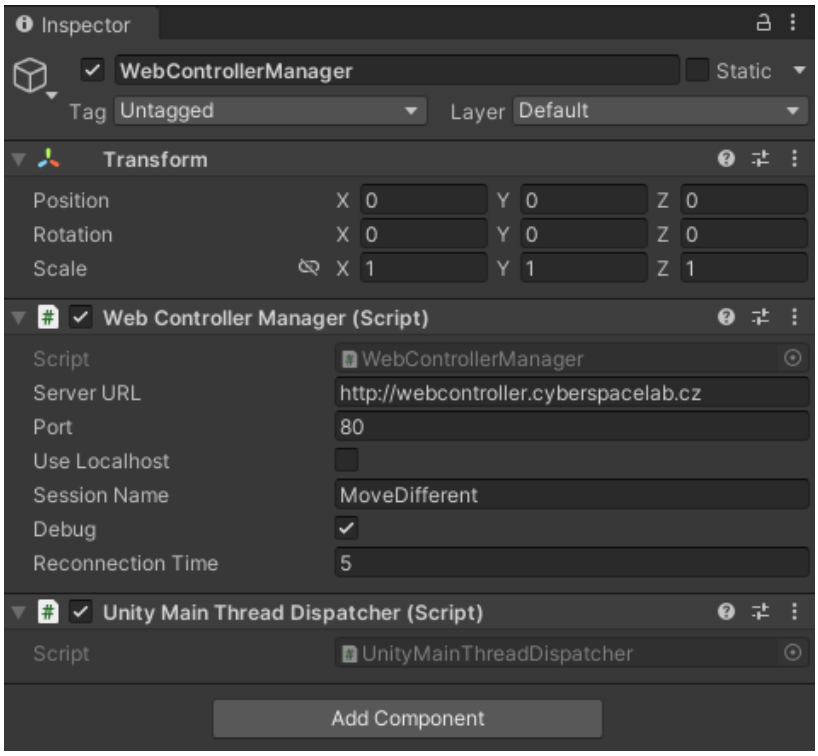
Pre komunikáciu hernej relácie so serverom slúži predpripravený Unity balíček. Balíček obsahuje päť skriptov. Tieto skripty je potrebné vložiť do daného projektu. Balíček je závislý na knižnici Socket.IO, ktorú je potrebné do Unity stiahnuť pomocou okna Package Manager. V ňom sa knižnica stiahne odkazom „https://github.com/itisnajim/SocketIOUnity.git“.



Obrázok 9: Unity Package Manager

Hlavným skriptom balíčka je **WebControllerManager**, ktorý je potrebné pripojiť k objektu v hlavnej začiatočnej scéne projektu. Do všetkých ostatných scén by sa malo prechádzať z tejto scény. Objekt, na ktorom je **WebControllerManager**

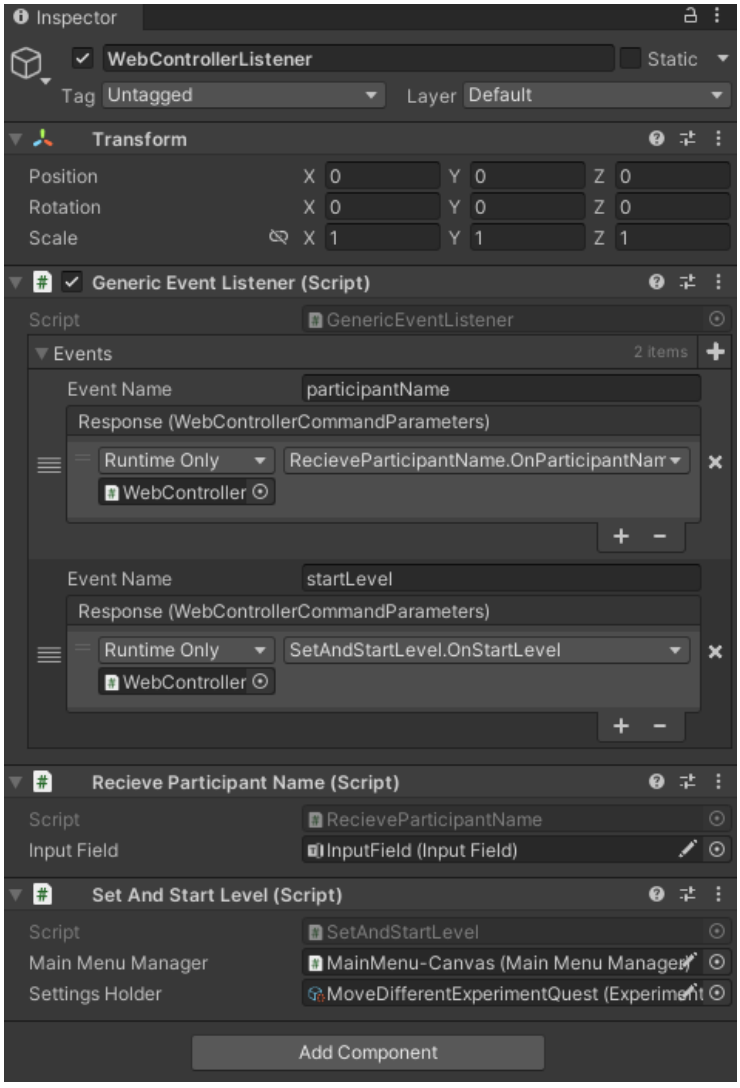
sa pri prechode na inú scénu nezničí a vždy existuje maximálne jedna inštancia. Tým pádom stačí založiť len jeden takýto objekt. Cez inšpektor sa vo **WebControllerManager** nastavujú všetky hlavné parametre pripojenia. Je potrebné nastaviť URL adresu serveru (**serverURL**), **port**, meno experimentu (**sessionName**) a interval v sekundách, po ktorom sa v prípade straty pripojenia relácia bude pokúšať k serveru znova pripojiť (**reconnectionTime**). Príklad pre adresu je „http://localhost“. K dipozícii sú aj dve políčka k zaškrtnutiu, **debug** a **useLocalhost**. Ak bude zaškrtnutý **debug**, Unity bude v konzole vypisovať hlášky informujúce o pripojení. Zaškrtnutím políčka **useLocalhost** za namiesto adresy definovanej v poli **serverURL** bude vždy používať „http://localhost“. Parameter **sessionName** nemusí byť vyplnený, a ak nie je, ako meno experimentu sa použije meno aplikácie definované v nastaveniach Unity. Toto meno, či už ako parameter alebo nastavenie projektu Unity, musí byť totožné s menom nastaveným v konfigurácií webovej aplikácie. Po tomto nastavení sa bude herná relácia automaticky pripájať k serveru.



Obrázok 10: Príklad použitia a nastavenia skriptu WebControllerManager (skript UnityMainThreadDispatcher sa k nemu automaticky pripojí)

Pre odosielanie dát na server stačí v kóde zavolať statickú metódu **WebControllerManager.SendData(string key, string value)**. Parameter **key** je meno typu odosielaných dát, ktoré musí byť totožné s menom definovaným

v konfigurácii webovej aplikácie v sekcii „receivers“. Aplikácia má rezervované dva mená typov dát, ktoré nemožno používať a to „position“ a „context“. Parameter **value** je hodnota dát, podporovaný je iba typ reťazca (string).



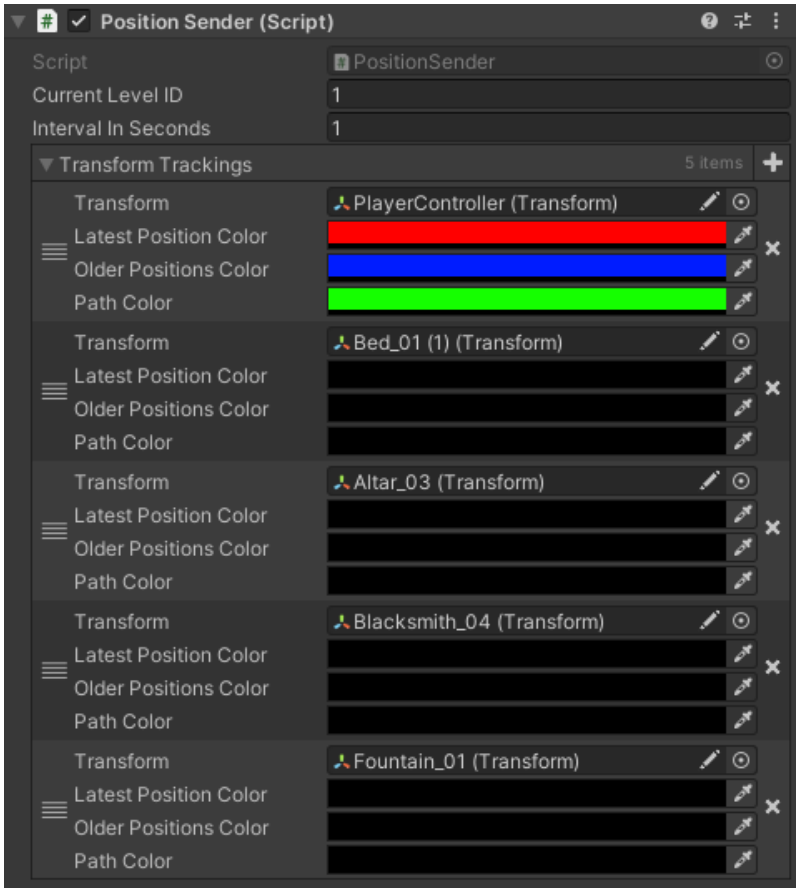
Obrázok 11: Príklad použitia skriptu GenericEventListener

Pre prijímanie príkazov zo serveru je potrebné na objekt v scéne pripojiť skript **GenericEventListener**, ktorý bude aktívne naslúchať serveru a v prípade, že zaznamená nejaký príkaz, tak bude adekvátne reagovať. Týchto skriptov môže byť v scéne ľubovoľný počet. V inšpektore nájdete položku **events**, do ktorej možno pridávať udalosti. Jedna takáto udalosť pozostáva z jej mena a z následnej akcie. Meno musí byť rovnaké s tým, ktoré je definované v konfigurácii tlačidla webovej aplikácie. Akcia je potom metóda, ktorá sa zavolá v prípade, že herná relácia daný príkaz prijme. Táto metóda je buď bez parametru alebo s jediným parametrom typu

WebControllerCommand. Tento typ je obalením obyčajného slovníka. Ak je príkaz očakávaný aj s parametrami, z tohoto slovníka sa podľa názvu parametru, čo bude kľúč, vytiahne jeho hodnota. Hodnota je typu **object**.

Zmena kontextu pre tlačidlá vo webovej aplikácii sa prevedie zavolaním statickej metódy **WebControllerManager.SetContext(string[] values)**. Jej parametrom je pole názvov aktuálnych kontextov. Tlačidlá, ktoré majú v konfigurácii webovej aplikácie definovaný aspoň jeden z týchto kontextov sa budú zobrazovať (a taktiež tlačidlá bez definovaného kontextu).

Pre odosielanie pozícií hráča a objektov je potrebné do danej scény na prázdny objekt pripojiť skript **PositionSender**. V inšpektore sa v ňom definuje identifikačné číslo aktuálnej úrovne, pre ktorú sa budú pozície posielať (parameter **currentLevelID**). Parametrom **intervalInSeconds** sa nastaví interval v sekundách, v ktorom sa pozície budú posielať. Do poľa **transformTrackings** sa pridajú referencie na všetky objekty, ktorých pozície sa majú odosielať a definujú sa tu aj farby poslednej pozície, predošlých pozícií a cesty pohybu pre daný objekt.



Obrázok 12: Príklad nastavenia skriptu PositionSender