TECHNICKÁ UNIVERZITA V KOŠICIACH
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Hra pre experimentálne posúdenie kognitívnych funkcií vo virtuálnej realite: konfigurácia hry a zber údajov

Diplomová práca

2019 **Bc. Dominik Trojčák**

TECHNICKÁ UNIVERZITA V KOŠICIACH
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Hra pre experimentálne posúdenie kognitívnych funkcií vo virtuálnej realite: konfigurácia hry a zber údajov

Diplomová práca

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

2019 Košice **Bc. Dominik Trojčák**

Abstrakt v SJ

Cieľom práce je vytvoriť časť hry pomocou vizualizačného nástroja SuperEngine, ktorý je v čase písania práce jedinou plne funkčnou platformou pre virtuálnu jaskyňu laboratória LIRKIS. Táto časť hry pokrýva konfiguráciu a uchovávanie dát pre ďalšie účely spojené s vyhodnocovaním experimentu. Konfiguračná časť musí byť prispôsobená tak, aby poskytovala expertom z oblasti kognitívnej vedy plnú kontrolu nad prebiehajúcou časťou experimentu. Výstupom tejto časti hry sú dáta, s ktorými je možné ďalej pracovať. V práci je analyzovaná celková hra so zameraním na konfiguráciu a zber dát. Analýza sa zameriava predovšetkým na možnosti a spôsob nastavenia konfigurácie pomocou vstaveného grafického používateľského rozhrania a taktiež možnosti uchovávania dát čo najjednoduchším spôsobom. Ďalej sa práca zameriava na samotný návrh a implementáciu týchto častí. Súčasťou práce je aj webová aplikácia, pomocou ktorej je možné prezentovať dosiahnuté výsledky účastníkov experimentu. Táto webová aplikácia zároveň slúži aj ako spôsob motivácie účastníkov. Výsledkom tejto práce je funkčná časť hry, ktorá v spojení s prácou ďalšieho riešiteľa vytvára celkovú hru určenú na časť experimentu zameraného na posúdenie kognitívnych funkcií vo virtuálnej realite.

Kľúčové slova v SJ

Virtuálna realita, hra, SuperEngine, experiment, kognitívne funkcie

Abstrakt v AJ

The aim of the thesis is to create a part of the game using the visualization tool SuperEngine, which is the only fully functional platform for the virtual cave of the LIRKIS laboratory at the time of writing. This part of the game covers the configuration and storage of data for other purposes related to the evaluation of the experiment. The configuration part must be tailored to provide cognitive experts with full control over the ongoing part of the experiment. The output of this part of the game is data that can be further worked on. In the thesis is analyzed the overall game focused on configuration and data collection. The analysis focuses mainly on the possibilities and method of configuring the configuration using the embedded graphical user interface as well as the ability to keep data as simple as possible. Furthermore, the thesis focuses on the design and implementation of these parts. Part of the work is also a web application, through which it is possible to present the results of the experiment participants. This web application also serves as a way of motivating participants. The result of this work is a functional part of the game, which in

conjunction with the work of another solver creates an overall game designed for a part of an experiment aimed at assessing cognitive functions in virtual reality.

Kľúčové slova v AJ

Virtual realit, game, SuperEngine, experiment, cognitive functions

TECHNICKÁ UNIVERZITA V KOŠICIACH

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Katedra počítačov a informatiky

ZADANIE DIPLOMOVEJ PRÁCE

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

Názov práce:

Hra pre experimentálne posúdenie kognitívnych funkcií vo virtuálnej realite: konfigurácia hry a zber údajov

Game for Experimental Assessment of Cognitive Functions in Virtual Reality: Game Configuration and Data Collection

Študent: **Bc. Dominik Trojčák**
Školiteľ: **Ing. Štefan Korečko, PhD.**
Školiace pracovisko: **Katedra počítačov a informatiky**
Konzultant práce:
Pracovisko konzultanta:

Pokyny na vypracovanie diplomovej práce:

1. Oboznámiť sa so softvérovým a hardvérovým vybavením virtuálno-realitnej jaskyne LIRKIS CAVE.
2. Analyzovať požiadavky na hru sformulované expertmi z oblasti kognitívnej vedy.
3. Pri analýze sa zamerať na požiadavky na náročnosť hry, jej konfiguráciu a údaje, ktoré sa majú počas hrania zaznamenávať.
4. Na základe analýzy pre LIRKIS CAVE navrhnúť a implementovať komponenty hry, týkajúce sa konfigurácie a zohľadnenia náročnosti hry a zberu údajov.
5. Pri návrhu a implementácii spolupracovať s riešiteľom súvisiacej záverečnej práce.
6. Vypracovať dokumentáciu podľa pokynov vedúceho práce.

Jazyk, v ktorom sa práca vypracuje: slovenský
Termín pre odovzdanie práce: 26.04.2019
Dátum zadania diplomovej práce: 31.10.2018

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

TECHNICKÁ UNIVERZITA V KOŠICIACH FAKULTA ELEKTROTECHNIKY A INFORMATIKY -10-

prof. Ing. Liberios Vokorokos, PhD.
dekan fakulty

Čestné vyhlásenie

Vyhlasujem, že som celú diplomovú prácu vypracoval samostatne s použitím uvedenej odbornej literatúry.

Košice, 25. apríla 2019 .....

vlastnoručný podpis

Poďakovanie

Chcel by som sa poďakovať Ing. Štefanovi Korečkovi Phd., za jeho odborné vedenie, metodickú pomoc a cenné rady poskytnuté pri písaní diplomovej práce. Zároveň ďakujem svojej rodine a priateľke za pomoc a podporu počas celého štúdia.

Obsah

Zoznam obrázkov

Zoznam tabuliek

Zoznam symbolov a skratiek

API Aplication programming interface

CDT Change detection task

CSV Comma separated-values

EEG Elektroencefalografia/ Elektroencefalograf

GUI Graphic user interface

REST Reprezentational state transfer

SAV Slovenská Akadémia Vied

UI User interface

UK Univerzita Komenského

Úvod

Virtuálna realita sa vo svete technológií dostáva čoraz viac do popredia a to najmä vďaka mobilným zariadeniam, ktoré priblížili tento druh vnímania virtuálneho sveta. Virtuálna realita ako pojem v sebe zahŕňa vnímanie 3-rozmernného sveta, spojeného s interaktívnym ovládaním pomocou použitia špeciálnych periférií, ktoré zaisťujú obrazovú, hmatovú, zvukovú a polohovú interakciu. Virtuálna realita nám prináša celú škálu využití. Od zábavy cez realistickú prehliadku plánov budov až po edukačné účely v mnohých smeroch. Jedným zo spôsobov využitia virtuálnej reality je aj oblasť medicíny a psychológie.

Cieľom diplomovej práce (ďalej ako „práca“) je vytvoriť aplikáciu, ktorá bude spĺňať požiadavky experimentu. Experiment bude prebiehať v laboratóriu virtuálnej reality LIRKIS v spolupráci so Slovenskou Akadémiou Vied (ďalej ako ,,SAV“) a Univerzitou Komenského (ďalej ako ,,UK“). Cieľom tohto experimentu je zistiť vplyv virtuálnej reality na človeka. Súčasťou experimentu je hra, v ktorej musí hráč zničiť nepriateľské objekty. Hráč bude umiestnený do virtuálno-realitnej jaskyne (CAVE). CAVE v hre predstavuje obrannú vežu s kanónom, ktorým je možné zostreliť lietajúce objekty. Lietajúce objekty budú rozdelené do dvoch skupín a to na nepriateľov a objekty, ktoré hráč nesmie zostreliť. Tieto dve skupiny objektov bude hráč vedieť odlíšiť na základe indikátorov, ktoré sa mu zobrazia na začiatku každého herného kola. Hra bude pozostávať z úrovní a epizód. Experiment bude prebiehať na vzorke približne 15-20 ľudí v niekoľkých fázach. Každý účastník bude musieť absolvovať prípravu na experiment, kde vyplní vstupný formulár, zrealizuje behaviorálny test spolu s meraním mozgovej aktivity pomocou zariadenia EEG. Po príprave bude nasledovať niekoľko sedení, na ktorých bude účastník súčasťou virtuálnej reality ako hráč. Počas každého sedenia si hráč zahrá jednu úroveň, ktorá sa skladá z niekoľkých epizód. Ich počet, dĺžka a náročnosť bude závisieť od konfigurácie, ktorú bude možné meniť pre každého hráča individuálne. Z každej odohratej úrovne bude potrebné získavať štatistiky a celkový priebeh hry hráča. Sedenia účastníkov budú pravidelné. Približne v polovici experimentu sa znovu uskutoční meranie pomocou EEG. Na konci experimentu sa s účastníkom vyplní záverečný formulár a realizuje posledné meranie EEG prístrojom. V tejto fáze budú už k dispozícií všetky potrebné výstupy pre vyhodnotenie experimentu. O vyhodnotenie experimentu, teda spracovanie dát z meraní, formulárov a štatistík sa budú starať experti zo SAV a UK.

Úlohou tejto práce v experimente bude zabezpečiť zaznamenávanie údajov z hry a vytváranie štatistík každého hráča, teda načítanie daného hráča podľa predošlých sedení a následne jeho uloženie po skončení sedenia. Pri tejto časti bude potrebné brať do úvahy, ako sa budú štatistiky vytvárať, ako často a čo všetko budeme o hráčovi zaznamenávať a v akom formáte budú dáta

uložené. Ďalšou dôležitou časťou bude nastavenie a logika hry, kde bude potrebné pripraviť možnosti pre vedúcich experimentu tak, aby boli schopní meniť konfiguráciu pred každým jedným sedením. Nastavenie tejto konfigurácie bude prispôsobené podľa potrieb experimentu.

Na vývoji hry budeme pracovať spolu s kolegom Bc. Peterom Vasiľom, ktorého úlohou bude vytvorenie samotného jadra hry a taktiež s pracovníkmi SAV a UK, ktorí špecifikujú požiadavky na hru.

1. Formulácia úlohy a cieľ práce

Cieľom práce je navrhnutie a implementovanie súčasti hry, ktorá bude slúžiť ako prostriedok pre experimentálne posúdenie kognitívnych funkcií človeka v prostredí virtuálnej reality. Prvá časť hry sa zameriava na hernú logiku a dizajn. V rámci druhej časti hry sme sa zamerali na získanie potrebných štatistík z hry. V práci sa budeme venovať iba druhej časti, a to implementácii vhodnej konfigurácie hry a uchovávaniu štatistík z experimentu. Prvej časti hry, zameranie sa na hernú logiku a dizajn sa bude venovať kolega Bc. Peter Vasiľ.

Pre splnenie hlavného cieľa je potrebné splniť čiastkové ciele. Prvým čiastkovým cieľom je oboznámiť sa so softvérovým a hardvérovým vybavením virtuálno-realitnej jaskyne LIRKIS CAVE, v ktorej bude hra vytvorená, a kde sa bude celý experiment odohrávať.

Druhý čiastkový cieľ spočíva v analýze požiadavok na hru, ktorá je formulovaná expertmi z oblasti kognitívnej vedy. Tejto časti sa venuje predovšetkým kapitola 2. Celý experiment prebieha podľa inštrukcií expertov, na základe ktorých je potrebné zamerať sa výlučne na tieto požiadavky. Pri analýze je dôležité zamerať sa na časti, ktoré budú predmetom mojej časti riešenia a teda požiadavky na náročnosť hry, jej konfiguráciu a údaje, ktoré sa majú počas hrania zaznamenávať. Vyššie spomínaným častiam sme sa venovali v kapitole 3, kde sú zhodnotené možnosti vývoja a podpory, v kapitole 4, kde sú analyzované jednotlivé vstupné parametre na základe požiadavok a v kapitole 5, ktorá je zameraná na konkrétnejšie riešenie uchovávania dát

Na základe tejto analýzy môžeme následne prejsť k splneniu hlavného cieľa, teda vhodne navrhnúť a implementovať komponenty hry, týkajúce sa konfigurácie a zohľadnenia náročnosti hry a zberu dát, pričom tejto časti sa venuje návrh a implementácia v kapitolách 7 až 14. Pri návrhu a implementácií je potrebné spolupracovať s kolegom Bc.Petrom Vasiľom, ktorého úlohou je implementácia jadra hry. Na záver je potrebné vypracovať všetku dokumentáciu ku implementovanému systému.

V kapitole 7 sa už práca zaoberá konkrétnym návrhom riešenia používateľského rozhrania. V tejto časti sú navrhuté jednotlivé časti a procesy nastavenia konfigurácie úrovní experimentu spolu s celkovou podporou nastavenia. Výsledkom je koncepčný návrh používateľského rozhrania, ktoré je následne implementované v kapitole 12.

Na používateľské rozhranie priamo nadväzujú kapitoly 8, 9 a 10. V týchto kapitolách je postupne navrhnutá funkcionalita spojená so zberom a uchovávaním dát či už úrovní alebo samotný výstup účastníkov experimentu v podobe dát pripravených na analýzu. Tento návrh je následne implementovaný v kapitole 13.

Kapitola 11 sa venuje návrhu prostriedku pre motivovanie účastníkov experimentu formou prezentačnej webovej aplikácie, prostredníctvom ktorej sa zvyšuje motivácia účastníkov a tým zároveň môže dôjsť k zvýšeniu kvality výsledkov. Popis implementácie tejto aplikácie je následne uvedený v kapitole 14.

2. Analýza hry a špecifikácia experimentu

V tejto kapitole sa budeme venovať analýze hry, ktorá má slúžiť ako prostriedok pre experiment. Stručne opíšeme obsah danej hry, čo je jej cieľom a zmyslom. Súčasne v tejto kapitole stručne popíšeme čo je cieľom experimentu. Obsah tejto kapitoly vychádza z článku [1], kde je podrobne špecifikované ako bude daná hra vyzerať, jej ciele a celkový opis experimentu. Tieto údaje poskytli experti zo SAV a UK.

2.1. Popis hry

Hra, ktorá je predmetom experimentu vychádza z klasických hier, kde hráč musí brániť seba alebo inú entitu pred útokmi nepriateľov. Hra sa odohráva v priestore pripomínajúcom vesmír. Hráč je umiestnený do obrannej veže, kde má k dispozícií prostriedok (zbraň) na obranu proti náletom nepriateľov. Úlohou hráča je rozoznať nepriateľské objekty od priateľských a zneškodniť tieto objekty. Objekty sú reprezentované ako lietajúce telesá – drony prichádzajúce z priestoru pred hráčom. Objekty priateľských a nepriateľských dronov sú jasne odlíšené tvarom. Hráč používa na ovládanie veže vstupné zariadenie, ktoré mu prioritne umožňuje vežou otáčať, mieriť a strieľať. Hra na pozadí musí byť schopná zaznamenávať aktivitu hráča v jednotlivých fázach hry a poskytnúť vedúcim experimentu potrebné dáta pre vyhodnotenie hry/experimentu.

2.2. Rozdelenie cieľov hry

Je potrebné rozdeliť cieľe na dva pohľady a to ciele z pohľadu experimentu a z pohľadu hráča, ktoré sú podstatne odlišné.

2.2.1. Ciele hry z pohľadu experimentu

Cieľom experimentu je zistiť vplyv dlhodobejšieho hrania hry na kognitívne schopnosti človeka, ako napríklad , vzťahy medzi objektami v 3D priestore, priestorová pamäť, rozpoznávanie objektov v priestore, rozoznávanie tvarov a podobne.

V článku Assessment and training of visuospatial cognitive functions in virtual reality: proposal and perspective, sa spomína, že vizuálno-špecifické funkcie tvoria hlavnú úlohu v kognitívnom poznaní človeka. To v poslednej dobe vyvolalo množstvo výskumov a experimentov so zameraním na hodnotenie, posudzovanie a trénovanie práve kognitívnych funkcií človeka. Zaujímavosťou je, že aj keď nám naše vizuálno-špecifické schopnosti umožňujú pochopiť a odvodiť vzťahy 3D objektov v priestore, tieto 3D aspekty vizuálno-špecifické spracovania sú často v laboratórnych testoch zanedbávané a namiesto toho sa bežne používajú 2D návrhy. S cieľom zlepšiť a zvýšiť platnosť týchto testov je navrhnutý experiment na vyhodnotenie kapacity 3D priestoru so zameraním na stimuláciu kognitívnych funkcií človeka. Experiment zahŕňa viacero aspektov

kognitívnych funckií, EEG merania a úlohy stimulujúce kognitívne funkcie testovaného subjektu v 3D virtuálnom prostredí vytvoreného pomocou virtuálno realitnej jaskyne – Cave systému, s cieľom posúdiť možný účinok hry pred a po jej dlhodobejšom hraní.

2.2.2. Ciele hry z pohľadu hráča

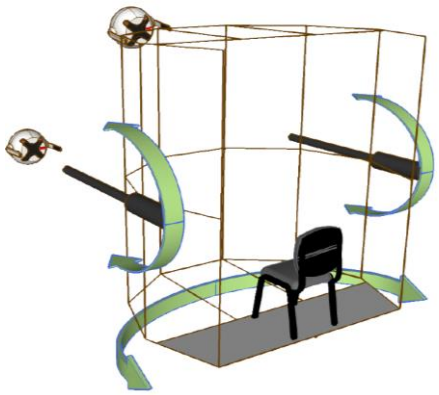
Cieľom hry je z pohľadu hráča zneškodniť všetky nepriateľské drony (tragety) za čas ich preletu z počiatočného bodu do bodu v ktorom sa nachádza hráč. Vedľajším cieľom je nezostreliť vlastné priateľské drony – distraktory. Hráč počas jedného sedenia(úrovne), ktoré trvá približne 30 minút, odohrá úroveň stanovenej obtiažnosti. Obtiažnosť je ovplyvnená vstupnými parametrami ako sú napr. počet targetov, počet distraktorov, ich rýchlosť, rozptyl a podobne. Jedna úroveň sa skladá z niekoľkých náletov(epizód). Tieto epizódy majú v rámci jednej úrovne konštantnú zložitosť. Medzi epizódami sa scéna hry mení následovne:

- mení sa počet targetov a distraktorov o variabilnú zložku;
- mení sa vzhľad targetov a distraktorov – hráčovi sa indikuje daná príslušnosť indikátorom;
- mení sa pozícia targetov a distraktorov – zmena trajektórie letu.

Hráčovi je počas hrania umožnené:

- streľba pomocou špeciálnych zbraní, ktoré ovláda ovládačom;
- pomocou ovládača natáčať smerom doprava a doľava, keďže sa vo virtuálnom svete nachádza na otočnej platforme. Uhol otočenia je limitovaný;
- pomocou ovládača nakláňať zbrane pripevnené na bokoch jeho kabíny smerom hore resp. dole, podobne s obmedzeným uhlom otočenia.

Na Obr. 1 je zobrazené ako približne bude scéna a priestor hry vyzerať v prostredí CAVE.



Obr. 1 Konceptuálny návrh hry Tower Defense

Zdroj: [1]

2.3. Účastníci (experimentálna skupina)

• Počet účastníkov – minimálny počet je 15, optimálny počet je 20 ľudí;

• Vek účastníkov – 21 až 24 rokov (pravdepodobne študenti 3. - 4. ročníka na KPI);

• Zastúpenie pohlaví – ideálne rovnaký počet mužov a žien.

2.4. Príprava na experiment

Pre každého účastníka bude potrebné zistiť vhodnú počiatočnú úroveň pre experiment (CDA/T – špeciálny test založený na pozorovaní a reagovaní na určité zmeny) a hru. To sa dosiahne vyplnením vstupného formuláru účastníkom a zopakovaním behaviorálneho testu CDT (Change Detection Task) - trikrát v priebehu 1 týždňa.

• Vstupný formulár- bude zisťovať základné demografické údaje, frekvenciu hrania hier, prípadne užívateľskú skúsenosť s technológiou virtuálnej, zmiešanej alebo rozšírenej reality + briefing. Odhad trvania je približne 10 - 15 minút;

• Behaviorálny test – tento test sa bude realizovať v koordinácii s odborníkmi zo SAV. Odhad trvania je približne 30 minút;

• Výstupný formulár – zisťujú sa zmeny vnímané používateľom a spätná väzba v podobe návrhov a možných zlepšení v hre a v priebehu celého experimentu. Odhad trvania je približne 10 - 15 minút.

2.5. Priebeh experimentu

Nižšie sú popísané kroky experimentu v presnom poradí, v akom musia byť realizované.

1. Vstupný formulár + EEG meranie (CDA/T)

2. 5 tréningových sedení

3. EEG meranie

4. 5 tréningových sedení

5. EEG meranie + výstupný formulár

Všetky tréningové sedenia sú rovnako dlhé. Každé pozostáva z troch hracích úsekov po 8 minút s krátkymi prestávkami (2 minúty). Spolu to potrvá 30 minút.

2.6. Špecifikácia požiadaviek pre hru

Z cieľov experimentu vyplynuli požiadavky. Ich presne popísaná špecifikácia od zadávateľa (experti zo SAV a UK, ktorí sú spolurealizátori experimentu) je popísaná nižšie v jednotlivých bodoch.

• Hra pozostáva z niekoľkých úrovní (levelov) s narastajúcou zložitosťou. Každá úroveň predstavuje vopred stanovený počet behov (trials) = "náletov". Každá úroveň je daná svojou zložitosťou zhluku – bod 3. (Požiadavka hry 1, ďalej len PH1)

• V rámci jedného behu sa blížia objekty v menšom počte, pričom vytvárajú priestorový zhluk, ktorý sa v čase nerozptýli (objekty majú podobný smer pohybu a rovnakú rýchlosť). Zhluk sa vynorí v ľubovoľnej (zadnej) časti priestoru a lineárne sa pohybuje smerom k rovine hráča, no nemusí smerovať priamo na hráča. Všetky objekty sa musia objaviť v rovnakom čase. (PH2)

• Veľkosť a zloženie zhluku je dané úrovňou hry, ale trochu sa mení aj v rámci úrovne. Označenie zhluku je #T/#D, kde #=počet, T=target, D=distraktor (napr. 3/2 znamená 3 cieľové objekty a 2 distraktory). Každá úroveň hry má stanovenú strednú hodnotu, pričom jednotlivé inštancie v rámci hry danej úrovne sa môžu líšiť o hodnotu +/-1 ks (čiže mierne sa mení počet T a/alebo D v daných epizódach). Vo vstupnej konfigurácii pre jednotlivé úrovne bude uvedená hodnota #T/#D, charakterizujúca danú úroveň, parametrom bude tiež variabilita okolo strednej hodnoty. (PH3)

• Objekty v rámci zhluku majú rovnakú farbu, líšia sa len tvarom (uvažujeme N rôznych 3D tvarov polygonálneho typu, nech N=5), pričom v rámci každého behu je jeden typ objektu cieľom (target), ostatných N-1 typov sú distraktory. Všetky typy objektov majú približne rovnakú tvarovú zložitosť a mali byť vzájomne zhruba rovnako nepodobné (odlíšiteľné). (PH4)

• Typ cieľového objektu sa mení v každom behu a indikuje sa na obrazovke (napr. nejakou značkou na cieľovom objekte) v čase, keď sa zhluk vynorí, aby si to hráč mohol všimnúť. (PH5)

• Počas behu raz (počas existencie zhluku) nastane situácia, že na obrazovke nastane "výpadok prúdu", keď časť obrazovky veľmi stmavne (okrem podlahy veže) na krátku dobu (600 až 900 ms). Objekty však pokračujú v pohybe a po výpadku hráč musí reagovať ako predtým (cieľové objekty budú o čosi bližšie). Výpadok prúdu má za cieľ zaťažiť vizuálnu pracovnú pamäť. (PH6)

• Po výpadku môžu nastať dva prípady: jeden z cieľových objektov sa trochu natočí alebo nenatočí. V oboch prípadoch by hráč mal cieľový objekt rozpoznať. Zmena by nemala byť výrazná (napr. natočenie v nejakej rovine o malý uhol). (PH7)

• Cieľom hráča je zostreľovať cieľové objekty a (podľa možnosti úplne) ignorovať distraktory (ktoré zbytočne zaťažujú vizuálnu pracovnú pamäť). (PH8)

• Kabína sa nemusí hýbať, postačí ak sa subjekt bude pohybovať očami, preto by malo stačiť renderovanie scény na displejoch vpredu. (PH9)

• Hra sa ovláda pomocou joysticka: pozícia cieľa (2D) a streľba na cieľ (gombíkom). (PH10)

• Hra by mala poskytovať report pre dané tréningové sedenie (počet behov = počet zhlukov), koľko mali T a koľko D, koľko a v akom čase po obnovení "výpadku prúdu" ich zostrelil a prípadne možnosť pridať ID subjektu a samozrejme čas sedenia. (PH11)

3. Systém virtuálnej reality

Pri vývoji hry je potrebné zvoliť správny systém pre tvorbu a aplikovanie experimentu. Experiment je zameraný na virtuálnu realitu, pričom je na výber viacero systémov a technológií. Jednou z možností je využitie laboratória LIRKIS a jeho virtuálno realitnej jaskyne (CAVE) [10]. Jedná sa o prenosné prostredie virtuálnej reality s 2.5x2.5x3 metrami zobrazovacej plochy. Vizuálny výstup je renderovaný na 20 LCD paneloch o veľkosti 55 palcov. 14 z nich je umiestnených vertikálne do 7-stranného desaťuholníka. Zvyšných 6 panelov je umiestnených horizontálne na vrchnej a spodnej strane. Spolu vytvárajú 250 stupňov panoramatického priestoru. CAVE podporuje viacero vstupno/výstupných zariadení, ako sú napríklad herné ovládače, klávesnica, myš ale aj komplikovanejšie zariadenia typu EEG, Myo.

Obr. 2 LIRKIS Cave

Virtuálna jaskyňa alebo CAVE (zobrazená na Obr. 2) obsahuje grafické jadro SuperEngine, ktoré umožňuje prispôsobenie scény. Okrem SuperEnginu je možné v CAVE využiť aj iné vizualizačné nástroje. Ich integrácia v čase experimentu nie je dokončená na takej úrovni, aby mohli byť plnohodnotne využité.

3.1. SuperEngine

SuperEngine je robustný zobrazovací systém schopný zobraziť veľké dátové sety v reálnom čase. Je vhodný pre revíziu vytváraných CAD projektov, prezentovanie vízie zákazníkovi, virtuálne manuály, prezentačné činnosti, ako aj prehrávanie multimediálnych dát. Umožňuje prezeranie v mono a rôznych 3D stereo režimoch. Pre zvýšenie výkonu je možné systém prevádzkovať v clustrovom režime. Súčasťou je skriptovací jazyk RUBY, ktorý umožňuje vytvárať zložité animácie a dotvárať funkčnosť celého systému bez nutnosti úpravy samotného softvéru. Ďalšou jeho zaujímavou vlastnosťou je možnosť synchronizovaného prezerania modelu vo vzdialených kanceláriách, prípadne využitia pre 3D konferencie. Tento systém je možné s výhodou použiť aj pre vytvorenie aplikácií virtuálnej reality. [12]

Jadro systému tvoria 3 programy:

- Remote Console - slúži na vzdialené ovládanie celého systému;
- Video Renderer - tvorí vizualizačnú časť. V systéme môže byť použitý viackrát na samotných počítačoch v závislosti od konkrétnej konfigurácie;
- Control Center - je hlavná časť systému, obsahuje kompletnú funkcionalitu. Ovláda sa pomocou konzoly a určuje činnosť a konfiguráciu renderov. [13]

Výhody SuperEnginu:

- Jednoduchá a rýchla tvorba 3D scén pomocou Ruby skriptovania;
- Podpora viacerých zobrazovacích jednotiek;
- Prispôsobený pre virtuálne zariadenia (v našom prípade CAVE systém).

Nevýhody SuperEnginu:

- Zastaralé jadro systému bez súčasnej podpory;
- Zobrazovací systém poskytuje iba úzky výber základných funkcií bez kvalitnejšej dokumentácie;
- Slabšia podpora pre prácu s 3D objektami v scénach;
- Výkonnostné limity;
- Obmedzená práca so súbormi.

Pre účely experimentu je SuperEngine vhodne využiteľný, najmä z hľadiska súčasnej podpory pre virtuálno realitnú jaskyňu laboratória LIRKIS, kde bude celý experiment uskutočnený. S využitím SuperEnginu bude možné pokryť jednotlivé požiadavky pre hru. Control Center spolu s Video Rendererom jadra SuperEnginu poskytnú vhodné virtuálne prostredie a Remote Console nám umožní vytvorenie jednoduchšieho používateľského rozhrania pre používateľov. Menším obmedzením v rámci implementácie budú nevýhody systému spomenuté vyššie.

3.2. Práca so súbormi v SuperEngine

V hre je potrebné pracovať s údajmi, ktoré nie sú priamou súčasťou zdrojových kódov. Tieto dáta budú ďalej využívané, vzhľadom na jednotlivé požiadavky. Je potrebné zabezpečiť aby hra bola schopná komunikovať, ukladať a čítať dáta z externých súborov automatizovane. V prípade SuperEnginu, ktorý využíva programovací jazyk Ruby s verziu 1.6 sú možnosti pre využitie rôznych typov súborov ako xml, yaml json a podobne. Avšak treba brať do úvahy aj limity a obmedzenia SuperEngine.

3.2.1. Typy súborov pre uchovávanie dát v SuperEngine

Ruby dokáže štandardne s využitím správnych knižníc pracovať so širokou škálou súborových typov od jednoduchých textových po komplikovanejšie. Pre našu hru a celkový experiment prichádzajú do úvahy typy ako xml, yaml, json, csv, xslt.

Po testovaní jednotlivých typov súborov sa ako jediný bezproblémovy ukázal typ CSV. Ostatné typy, xml, yaml a json, spôsobovali problémy pri opätovnom spustení scény. Tento problém sa prejavoval zaseknutím procesu renderera, čo malo za následok jeho ukončenie.

3.2.2. CSV v Ruby

Ruby využíva pre prácu s CSV súborom knižnicu `csv.rb`. Táto knižnica poskytuje celkové rozhranie pre CSV súbory a dáta. Ponúka nástroje, ktoré umožnujú čítať a zapisovať z a do reťazca alebo IO objektu.

Použitie:

- **Čítanie:**

```
CSV.foreach("cesta k súboru") do |row|
… operácia na úrovní riadku - row
end
csv = CSV.read("cesta k súboru")
```
– načíta kompletne celý súbor

- ***Zápis:***

```
CSV.open("cesta k súboru") do |csv|
csv << ["row","of","csv","data"]
csv << ["another","row"]
end
```
– vygeneruje riadky priamo do csv súboru *csv_string*

```
CSV.generate do |csv|
csv << ["row","of","csv","data"]
csv << ["another","row"]
end
```
– vygeneruje csv reťazec

- **Konvertovanie dát do csv formátu:**

```
csv_string = ["csv", "data"].to_csv
csv_array = "csv.string".parse_csv
```

4. Analýza vstupných parametrov

Z požiadaviek hry a experimentu vyplýva potreba konfigurácie hry používateľom, ktorý bude dohliadať na priebeh experimentu. Vstupné parametre sa menia na základe vlastností hráča. Berie sa do úvahy jeho aktuálna úroveň, nastavenie experimentu, prípadne iné sledované faktory. Pre jednoduché nastavovanie a automatizáciu experimentu je potrebné tieto vstupy zanalyzovať a preniesť do konfiguračnej podoby, ktorá bude jasne a zreteľné popisovať nastavovanú vlastnosť. Vstupné parametre vyplývajúce z požiadaviek:

- Počet úrovní(Levelov) – možnosť vedúceho experimentu nastaviť počet úrovní(levelov), ktoré sa odohrajú. (PH1)
- Počet behov(epizód) – možnosť vedúceho experimentu nastaviť počet epizód v danej úrovni. Každá úroveň môže obsahovať 1 a viac epizód. (PH1)
- Dĺžka trvania úrovne a epizódy – možnosť nastaviť trvanie každej epizódy prípadne úrovne, pričom dĺžka sedenia musí byť 30 minút.
- Počet objektov – možnosť nastaviť počet nalietavajúcich objektov v ľubovoľnom pomere T/D. (PH3)
    - Priateľské objekty D (distraktor).
    - Nepriateľské objekty T (target).
- Variabilná časť počtu objektov – možnosť nastavenia variabilnej zložky celkového počtu objektov. (PH3)
- Počet rôznych 3D tvarov polygonálneho typu pre targety a distraktory – hodnota udávajúca počet obsadených modelov targetmi alebo distraktormi, pričom jeden model môže byť pridelený iba jednej skupine. (PH4)
- Dĺžka trvania zobrazenia indikátora – možnosť nastavenia ako dlho bude na začiatku každého behu(epizódy) zobrazený indikátor udávajúci typ objektu (target, distraktor). (PH5)
- Trvanie „výpadku prúdu“ – možnosť nastaviť dĺžku trvania výpadku prúdu. (PH6)
- Čas nastatia udalosti „výpadok prúdu“ – možnosť nastaviť čas kedy dôjde k výpadku prúdu počas behu(epizódy). (PH6)
- Generovanie reportu – možnosť nastaviť zaznamenávanie sedenia. (PH11)

Všetky tieto základne parametre bude nutné nastavovať špeciálne pre každé sedenie účastníka experimentu. Preto je rozumné umožniť vedúcemu experimentu tieto dáta nastavovať bez zásahov do zdrojového kódu. Je viacero možností ako uchovávať a nastavovať tieto hodnoty.

4.1. Možnosti nastavenia a uchovávania konfigurácie

Konfiguráciu vstupných parametrov je možne nastavovať niekoľkými spôsobmi. Z pohľadu používateľa je ideálne, ak môže konfiguráciu nastaviť jednoducho a rýchlo. Na druhej strane by nastavenie konfigurácie nemalo povoľovať nastavenie nekorektných hodnôt. Napríklad, ak je potrebné zadať číslo z rozpätia 1 až 5, je vhodné ošetriť daný vstupný parameter konfigurácie práve na tieto krajné hodnoty.

4.1.1. Statické nastavenie vstupných parametrov konfigurácie

Statické nastavenie parametrov predstavuje druh konfigurácie, ktorá je pevne daná v kóde programu prípadne je oddelená v samostatnom súbore, ktorý je možné meniť avšak nato aby sa zmeny prejavili je potrebné program spustiť nanovo. V tomto prípade je možné konfiguračné premenné uložiť ako:

- Hodnoty priamo v kóde - výhodou je, že môžu byť zmenené v prípade, že iba daný používateľ má prístup k zdrojovým kódom. Táto výhoda môže byť aj nevýhodou v prípade, že potrebujeme konfiguráciu často meniť. Zmena sa prejaví až po uložení a opätovnom spustení;
- Hodnoty uložené ako externé dáta – využitie externého súboru, v ktorom sú zapísané jednotlivé konfiguračné hodnoty. Hlavnou výhodou je, že môžeme mať viacero druhov konfigurácie.

4.1.2. Dynamické nastavenie vstupných parametrov využitím Remote Console

Dynamické nastavovanie vstupných parametrov umožňuje vytvárať používateľovi jednoduchšie a okamžite zmeny v programe. Pod dynamickou zmenou rozumieme zmenu takú, ktorá nevyžaduje opätovné spustenie systému.

SuperEngine poskytuje podporu pre vytváranie základných prvkov používateľského rozhrania priamo v konzole, ktorá slúži na riadenie, spúšťanie a nastavovanie scén zobrazovaných pomocou renderera. Každá scéna môže byť riadená štyrmi tlačidlami (Stop, Start, Load, Pause), ktoré sú uložené v hlavnom paneli (Main Panel) a niekoľkými konfiguračnými časťami, ktoré nie sú primárne podstatné pre tvorbu hry. Okrem toho sa v konzole nachádza časť, ktorá poskytuje priestor a možnosť pre dotvorenie vlastného používateľského rozhrania, ktorým je možné ovplyvňovať samotnú scénu. Do tejto časti sa jednotlivé elementy vkladajú pomocou Ruby skriptu, kde sa definujú jednotlivé atribúty ako sú napríklad: typ, pozícia, hodnota(obsah), funkcia, ktorá sa má vykonať v zadanom prípade, prípad použitia. Elementy sa v ruby skripte vytvárajú využitím GUI manažéra, ktorého inštancia sa vytvorí pomocou volania ENGINE::GetGUIManager().

Po získaní tejto referencie je následne možné s touto premennou pracovať. Súčasná dokumentácia SuperEnginu neobsahuje časť venovanú používateľskému rozhraniu, preto je potrebné túto časť podrobne zanalyzovať a zmapovať jednotlivé funcionality komponentov a elementov.

Primárne funkcie GUI manažéra:

- Update() – funkcia, ktorá vyvolá prerenderovanie používateľského rozhrania, ak je potrebné aktualizovať dáta napríklad pri zmene údajov. Táto funkcia sa môže volať v každej hernej slučke, ale aj individuálne podľa potreby.
- GetMainPanel() – funkcia, ktorej návratová hodnota je hlavný panel(Main panel). Po získaní tejto referencie je možné pristupovať k základným tlačidlám.
- CreateObject() – pomocou volania tejto funkcie nad objektom gui manažéra je možné do panelu konzoly pridávať vlastné elementy. Návratovou hodnotou je konkrétny element. Parametre funkcie sú:
    - elementType – typ elementu, ktorý sa má vytvoriť ;
    - panel – referencia na panel, ktorý bude ďalej slúžiť ako jej rodičovský element.

4.1.2.1. Typy elementov

Konzola poskytuje štandradne niekoľko rôznych elementov, ktoré sa dajú využiť práve na účely spojené s experimentom. Typy elementov sú: panel, štítok (label), tlačidlo (button), textové pole (textField), zaškrtávacie pole (checkbox) a číselník (comboBox). Pre prácu s týmito typmi je potrebné ich dôkladne zanalyzovať pre chýbajúcu dokumentáciu.

Panel

Panel je základný prvok pri tvorbe grafického rozhrania v konzole. Panel slúži na logické a grafické rozdelenie celkovej štruktúry. Vytvára akýsi rodičovský komponent pre ďalšie elementy, ktoré sú do neho vkladané. Vloženie elementu do panelu sa vykoná priamo pri vytváraní elementu pomocou spomínanej funkcie gui manažéra CreateObject(), kde sa ako druhý parameter zadá referencia na príslušný panel, do ktorého bude tento element spadať. Panel môže obsahovať aj iné, ďalšie panely. V prípade, že panel nemá rodičovský panel je jeho rodičom hlavný panel.

Vytvorenie panelu: panelInstace = gm.CreateObject('panel') - kde 'panelInstace' predstavuje premennú do ktorej sa uloží referencia na vytvorený panel a 'gm' je gui manažér. Následne je možné vytvorenému panelu priradiť nadpis, nastaviť mu veľkosť a pozíciu:

- panelInstace.SetTitle(‘Názov’) – nastaví názov panelu na ‘Názov’.
- panelInstace.SetLayout(‘ABSOLUTE’) – nastaví pozíciu panelu ako absolútnu, ďalšou možnosťou je pozícia ‘RELATIVE‘ – relatívna.
- panelInstance.SetPosition(10,20) – nastaví pozíciu panela vzhľadom na rodičovský element na x=10 a y=20, pričom x udáva pozíciu v smere vertikálnom a y v smere horizontálnom.
- panelInstace.SetDimension(30,40) – nastaví veľkosť panelu na rozmery 30x40.

Štítok (Label)

Štítok slúži v tomto grafickom rozhraní ako element uchovávajúci textovú informáciu. Táto textová informácia sa môže meniť iba vplyvom zmeny hodnoty v programe a nie je ju možné meniť používateľom. Štítok môže zobrazovať rôzne informácie o chode hry, experimentu alebo pomôcť používateľovi zorientovať sa v prostredí konzoly.

Vytvorenie štítku: labelInstace = gm.CreateObject(‘label’) - kde ‘labelInstance’ predstavuje premennú, do ktorej sa uloží referencia na vytvorený štítok a ’gm’ je gui manažér. Následne je možné vytvorenému panelu priradiť hodnotu(text), nastaviť mu veľkosť a pozíciu:

- labelInstace.SetText(‘text’) – nastaví zobrazenú textovú hodnotu ’text’.
- labelInstace.SetPosition(10,20) – nastaví pozíciu štítku vzhľadom na rodičovský element na x=10 a y=20, pričom x udáva pozíciu v smere vertikálnom a y v smere horizontálnom.
- labelInstace.SetDimension(30,40) – nastaví veľkosť štítku na rozmery 30x40.

Funkciu labelInstance.SetText(), je možné použiť v každej hernej slučke čo pri využití Update funkcie gui manažéra zabezpečí aktualizovanie textu v reálnom čase. Takýmto spôsobom je možné zobrazovať napríklad aktuálny čas úrovne/epizódy.

Tlačidlo (Button)

Pomocou tlačidla je možné zabezpečiť používateľovi možnosť zasahovať do chodu programu tak, že tlačidlu sa priradí určitá akcia, ktorá sa vyvolá zmenou jeho stavu – stlačené/nestlačené. Nato aby tlačidlo v konzole správne fungovalo musí mať priradenú akciu (event).

Vytvorenie tlačidla: buttonInstace = gm.CreateObject(‘button‘) - kde ‘buttonInstace‘ predstavuje premennú do ktorej sa uloží referencia na vytvorené tlačidlo a ’gm’ je gui manažér. Následne je možné vytvorenému tlačidlu priradiť hodnotu(text), nastaviť mu veľkosť a pozíciu:

- buttonInstace.SetText(‘text’) – nastaví zobrazenú textovú hodnotu ’text’.
- buttonInstace.SetPosition(10,20) – nastaví pozíciu tlačidlu vzhľadom na rodičovský element na x=10 a y=20, pričom x udáva pozíciu v smere vertikálnom a y v smere horizontálnom.
- buttonInstace.SetDimension(30,40) – nastaví veľkosť štítku na rozmery 30x40.

Okrem týchto možnosti tlačidlo disponuje aj funkciami, ktoré ovplyvňujú jeho funkčnosť a vlastnosti:

- buttonInstace.SetEnable(false) – spôsobí, že tlačidlo nebude aktívne, hodnota ,,true“ spôsobí, že tlačidlo sa opäť aktivuje. Štandartne je táto hodnota pri vytvorení tlačidla nastavená na ,,true“.
- buttonInstace.AddEventListener(‘clicked‘, akcia()) – priradí tlačidlu akciu, ktorá sa aktivuje pri nastatí určitej udalosti, v tomto prípade ‘clicked‘ – pri kliknutí na tlačídlo zavolaj funkciu akcia().

Hodnotu(nadpis) tlačidla je možné meniť počas behu programu rovnako ako pri štítku.

Textové pole (textField)

Textové pole umožňuje používateľovi zadávať vstupné textové reťazce – čísla, znaky. Pomocou týchto textový polí dokáže SuperEngine spracovať informáciu počas behu programu, s ktorou je možné ďalej pracovať.

Vytvorenie textového poľa: textFieldInstace = gm.CreateObject(‘textField‘) - kde ‘textFieldInstace‘ predstavuje premennú, do ktorej sa uloží referencia na vytvorené textové pole a ’gm’ je gui manažér. Následne je možné vytvorenému textovému poľu priradiť premennú, nastaviť mu veľkosť a pozíciu:

- textFieldInstace.SetValue(textFieldValue) – nastaví textovému poľu referenciu na hodnotu, ktorú primárne zobrazuje, upravuje.
- textFieldInstace.SetPosition(10,20) – nastaví pozíciu textovému poľu vzhľadom na rodičovský element na x=10 a y=20, pričom x udáva pozíciu v smere vertikálnom a y v smere horizontálnom.
- textFieldInstace.SetDimension(30,40) – nastaví veľkosť štítku na rozmery 30x40.

Okrem týchto možnosti textové pole disponuje aj funkciami, ktoré ovplyvňujú jeho funkčnosť a vlastnosti:

- textFieldInstace.SetEnable(false) – spôsobí, že textové pole nebude aktívne a teda nebude do neho možné zapisovať text, hodnota ,,true“ naopak pole opäť aktivuje. Štandardne je táto hodnota pri vytvorení textového poľa nastavená na ,,true“.
- textFieldInstace.AddEventListener(‘valueChanged‘, akcia()) – priradí textovému poľu akciu, ktorá sa aktivuje pri zmene hodnoty.

Zaškrtávacie pole (checkbox)

Zaškrtávacím poľom sa rozumie element, ktorý umožnuje používateľovi zadávať vstup pre konkrétnu vlastnosť v rozmedzí pravda-nepravda (false-true).

Vytvorenie zaškrtávacieho poľa: checkboxInstance = gm.CreateObject(‘checkBox‘) - kde ‘checkboxInstance ‘ predstavuje premennú, do ktorej sa uloží referencia na vytvorené zaškrtávacie pole a ’gm’ je gui manažér. Následne je možné vytvorenému poľu priradiť hodnotu(text), nastaviť mu veľkosť a pozíciu:

- checkboxInstance.SetText(‘text’) – nastaví zobrazenú textovú hodnotu ’text’ – funguje ako štítok naviazaný na tento element.
- checkboxInstance.SetPosition(10,20) – nastaví pozíciu poľa vzhľadom na rodičovský element na x=10 a y=20, pričom x udáva pozíciu v smere vertikálnom a y v smere horizontálnom.
- checkboxInstance.SetDimension(30,40) – nastaví veľkosť poľa na rozmery 30x40.

Okrem týchto možnosti zaškrtávacie pole disponuje aj funkciami, ktoré ovplyvňujú jeho funkčnosť a vlastnosti:

- checkboxInstance.SetEnable(false) – spôsobí, že pole nebude aktívne, hodnota ,,true“ naopak tlačidlo opäť aktivuje. Štandardne je táto hodnota pri vytvorení tlačidla nastavená na ,,true“.
- checkboxInstance.setChecked(isChecked) – nastaví zaškrtávaciemu poľu referenciu na hodnotu pravda-nepravda(true-false).
- checkboxInstance.AddEventListener(‘stateChanged‘, akcia()) – priradí poľu akciu, ktorá sa aktivuje pri zmene stavu.

Číselník (combo box)

Číselník je typ elementu, ktorý poskytuje používateľovi možnosť výberu z presnej a konečnej množiny možností.

Vytvorenie zaškrtávacieho poľa: comboBoxInstance = gm.CreateObject(‘comboBox‘) - kde ‘comboBoxInstance ‘ predstavuje premennú, do ktorej sa uloží referencia na vytvorený číselník a ’gm’ je gui manažér. Následne je možné vytvorenému číselníku priradiť možnosti výberu, nastaviť mu veľkosť a pozíciu:

- comboBoxInstance.SetOptions([‘možnosť1’,’možnosť2’,’možnosť3’]) – nastaví alebo priradí číselníku množinu možností, ktoré má zobraziť;
- comboBoxInstance.SetPosition(10,20) – nastaví pozíciu číselníka vzhľadom na rodičovský element na x=10 a y=20, pričom x udáva pozíciu v smere vertikálnom a y v smere horizontálnom;
- comboBoxInstance.SetDimension(30,40) – nastaví veľkosť číselníka na rozmery 30x40.

Okrem týchto možnosti číselník disponuje aj funkciami, ktoré ovplyvňujú jeho funkčnosť a vlastnosti:

- comboBoxInstance.SetEnable(false) – spôsobí, že čísleník nebude aktívny, hodnota ,,true“ naopak čísleník opäť aktivuje. Štandardne je táto hodnota pri vytvorení tlačidla nastavená na ,,true“.
- comboBoxInstance.AddEventListener(‘indexChanged‘, akcia()) – priradí poľu akciu, ktorá sa aktivuje pri zmene zvolenej hodnoty z číselníka.

4.1.2.2. Spracovávanie dát z elementov

Pri použití grafického rozhrania a jednotlivých elementov je možné spracovávať hodnoty pomocou funkcie, ktorá je priradená k elementu využitím funkcie AddEventListener(viď kap. 4.1.2.1). Do tejto funkcie vstupuje event ako jediný vstupný parameter. Event obsahuje v sebe informácie o elemente, z ktorého pochádza. Pomocou toho je možné zistiť o akú zmenu a akého elementu sa jedná. Event ponúka metódy:

- GetSource – návratová hodnota je inštancia elementu;
- GetType – návratová hodnota je informácia o type akcie, ktorou element vstupuje do tejto funkcie. Napr. ‘valueChanged’ pre textové pole;
- GetParams – návratovou hodnotou sú aktuálne dáta prichádzajúce z elementu. Napríklad zmena hodnoty textovehé poľa z “abc” na “abcd”. Využíva sa pri ukladaní novej hodnoty do premennej, ktorá je naviazaná na element.

Využitie tohto používateľského rozhrania pri tvorbe hry je vhodné najmä z pohľadu, že koncový používateľ bude mať celú konfiguračnú časť priamo v Remote Console SuperEnginu a tak bude

schopný riadiť celý proces priamo z jedného používateľského okna. Podpora pre tvorbu používateľského rozhrania je vhodná a pokrýva požiadavky na hru a experiment. Z pohľadu efektívnejšej prípravy dát a experimentu sa javí ako najvhodnejšie využitie spojenia statickej a dynamickej konfigurácie, kde bude možné zvoliť predpripravenú konfiguráciu v spojení s používateľským prostredím, na okamžité zmeny počas vykonávania sedenia a celkového procesu experimentu.

5. Analýza zaznamenávania a uchovávania dát

V rámci tejto kapitoly sa budeme venovať oboznámeniu sa s hlavnými sa entitami v hre, ktorými sú hráči a zaznamenávaniu ich štatistík pomocou CSV formátu súboru.

5.1. Hráči

Hlavnou entitou v hre a v experimente budú subjekty, na ktorých bude experiment vykonávaný. Týmto subjektom budú hráči, pri ktorých bude potrebné uchovávať rôzne dáta. Ako vyplýva z kap. 4.1 najvhodnejšou alternatívou v systéme je využitie CSV formátu súboru, ktorý navyše poskytuje prijateľnú formu dát pre používateľa z hľadiska čitateľnosti a použitia externých tabuľkových programov. Úlohou systému je umožniť vedúcim experimentu jednoduchú správu hráčov. To znamená, že používateľ by mal byť schopný hráča v systéme vytvoriť, odstrániť, editovať a ďalej pracovať s týmito dátami. Na tieto účely je vhodné využiť niektorý z tabuľkových editorov, ktoré používateľovi poskytujú jednoduchú a rýchlu manipuláciu s dátami. Uchovávanie dát je pre experiment dôležité najmä k spätnému vyhodnocovaniu, pričom musí byť jasné, aké dáta patria konkrétnemu hráčovi.

5.2. Štatistiky

Podobne ako uchovávanie hráčov je potrebné zabezpečiť, aby sa zaznamenávali štatistiky hráčov. Táto požiadavka vychádza z požiadavky PH11. Hra musí byť schopná uchovávať reporty z jednotlivých sedení a to tak, že zaznamená každú špecifickú udalosť, akou môže byť napríklad: výstrel, zostrelenie target-a, zostrelenie disktraktor-a, výpadok prúdu a podobne. Report by mal taktiež obsahovať sumárne štatistiky z každej úrovne. Tieto štatistiky by mali obsahovať dáta ako počet zostrelených targetov a distraktorov z celkového počtu.

5.3. Využitie CSV

Z hľadiska podpory SuperEngine je najvhodnejšou formou uchovávanie dát v systéme využite CSV formátu. CSV je jednoduchý súborový formát, určený na ukladanie tabuľkových dát. Súbor vo formáte CSV pozostáva z ľubovoľného počtu záznamov oddelených znakom nového riadku. Tento formát je široko podporovaný rôznymi aplikáciami. CSV formát obsahuje čistý text. V hre bude potrebné o hráčovi zoznamovať dáta ako: ID hráča, meno, priezvisko prípadne doplňujúce informácie. CSV formát by teda v prípade takýchto dát vyzeral ako 1;Meno;Priezvisko;..;..;. Každý jeden riadok by predstavoval jedného hráča. Pre štatistiky by to následne mohlo vyzerať ako Úroveň;Počet zostrelených targetov;Celkový počet targetov;Počet zostrelených distraktorov;Celkový počet distraktorov;Časová známka.

5.4. Zabezpečenie integrity a dostupnosti dát

Pri využití súboru ako zdroja a cieľa dát je dôležité zabezpečiť integritu a dostupnosť dát. SuperEngine je schopný pracovať s dátami, ktoré sú súčasťou balíčka spustenej scény. Na základe toho je potrebné zabezpečiť, aby dáta boli súčasťou scény, v našom prípade hry tak, aby boli vždy keď si ich program vyžiadá dostupné, prípadne, aby ich systém bol schopný automaticky vytvoriť.

6. Analýza spôsobu motivácie účastníkov experimentu

Zo štúdie [11] vyplývajú výsledky, ktoré potvrdzujú, že výsledky experimentu sú závislé práve od motivácie jeho účastníkov. Štúdia taktiež preukázala rozdiely medzi dobrovoľnými účastníkmi a účastníkmi, ktorí sú povinní vykonať daný experiment. Najlepšie a najpresnejšie výsledky boli dosiahnuté práve pri účastníkoch s povinnou účasťou, ktorí boli správne motivovaní vo vykonávaní experimentu. V prípade tohto experimentu je teda potrebné správne motivovať jeho účastníkov za účelom dosiahnutia čo najlepších výsledkov. Kedže v čase vývoja aplikácie ešte nie je známa presná skupina účastníkov, nie je možné z našej strany určiť či sa jedná o dobrovoľníkov alebo nie. Preto je potrebné, zamerať sa práve na vytvorenie motivačného nástroja. Jednou z možností je dosiahnutie súťaže medzi účastníkmi. Táto súťaž sa dá docieliť porovnaním všetkých účastníkov(hráčov), napríklad pomocou vytvorenia skóre, kde motívaciou bude dosiahnúť čo najvyššie hodnotienie a tým sa dostať na jej vrchol. Takto sa zvýši sústredenosť hráča na dosiahnutie lepšieho výsledku, čo môže priaznivo vplývať na posúdenie kognitívnych funkcií.

Z kapitoly 4 a 5 taktiež vyplýva potreba spôsobu, akým budú výstupy z hry zobrazované. Z používateľského pohľadu sa ako vhodná alternatíva javí webová aplikácia, ktorá bude schopná vziať dáta z hry a tie následne pretransformovať do podoby grafických tabuliek a grafov. Vhodným nástrojom by teda mohla byť webová aplikácia, ktorá bude zobrazovať výsledky účastníkov a zaraďovať ich od najlepších po najhoršie a taktiež zobrazovať aj detailné štatistiky každého hráča. Aplikácia by bola následne publikovaná v priestoroch OpenLab-u Technickej Univerzity v Košiciach (https://kpi.fei.tuke.sk/sk/openlab), prípadne na inom verejne prístupnom mieste. Aplikácia by zároveň mohla slúžiť aj ako propagačným nástrojom celého experimentu.

6.1. Využitie frameworku pri tvorbe webovej aplikácie

Framework je softvérová štruktúra, ktorá slúži ako podpora pri programovaní a vývoji softvérových projektov. Môže obsahovať rôzne podporné programy, knižnice, návrhové vzory, alebo postupy pri vývoji. Cieľom frameworku je prevzatie typického problému danej oblasti, čím sa uľahčí vývoj tak, aby sa dizajnéri a vývojari mohli sústrediť výhradne na svoje zadanie. V oblasti webových technológií existuje mnoho framworkov určených na vývoj aplikácie. Počas tohto experimentu sú najvhodnejšími React, Vue a Angular. Tieto frameworky sú postavené na programovacom jazyku javascript s využitím ďalších technológií ako sú HTML, CSS a podobne.

7. Návrh používateľského rozhrania

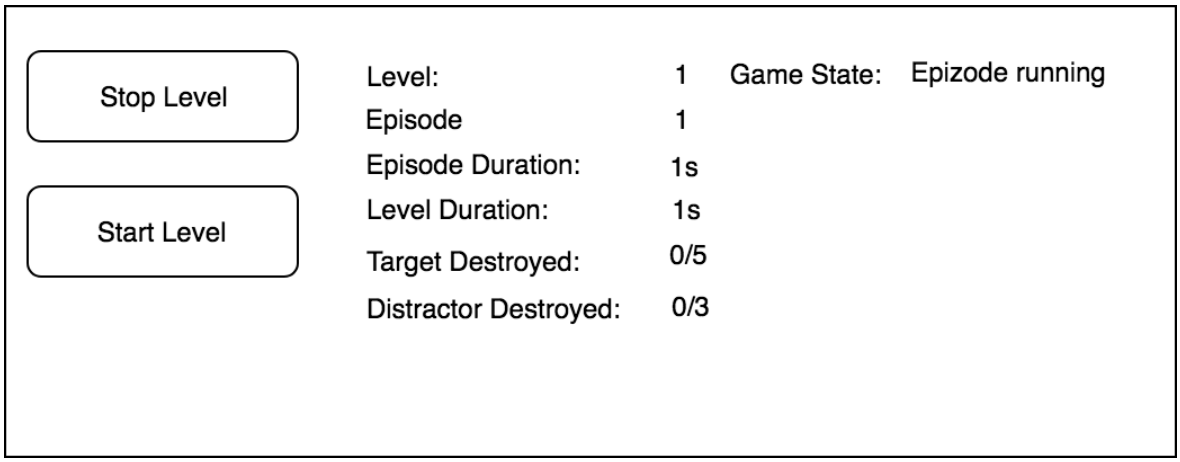
Pri návrhu používateľského rozhrania vychádzame z analýzy vstupných parametrov. Požívateľské rozhranie bude vytvorené v remote console SuperEnginu. Toto používateľské rozhranie bude používateľovi poskytovať potrebnú kontrolu nad priebehom jednotlivých častí experimentu. Používateľské rozhranie sa bude skladať zo 4 samostatných častí. Každá z týchto častí bude zastrešovať špecifickú oblasť čo v konečnom dôsledku sprehľadní a zjednoduší proces výberu hráča, nastavenia úrovne a sledovania experimentu.

Časti používateľského rozhrania:

- Informácie o aktuálnej úrovni
- Nastavenie úrovni
- Informácie o hráčovi
- Nápoveda

7.1. Informácie o aktuálnej úrovni

Táto časť má za úlohu zobrazenie informácií o aktuálne prebiehajúcej úrovni. Zobrazuje informácie ako poradové číslo úrovne, poradové číslo epizódy, trvanie epizódy (aktuálny čas/celkový čas), trvanie úrovne (aktuálny čas/celkový čas), počet zostrelených distraktorov (počet zostrelených/celkový počet), počet zostrelených targetov (počet zostrelených/celkový počet), stav hry, v akom sa práve nachádza. V tejto časti používateľského rozhrania budú taktiež umiestené tlačidlá pre spustenie a ukončenie úrovne. Rozmiestenie elementov je zobrazené na Obr. 3.



Obr. 3 Návrh časti: informácie o prebiehajúcej úrovni

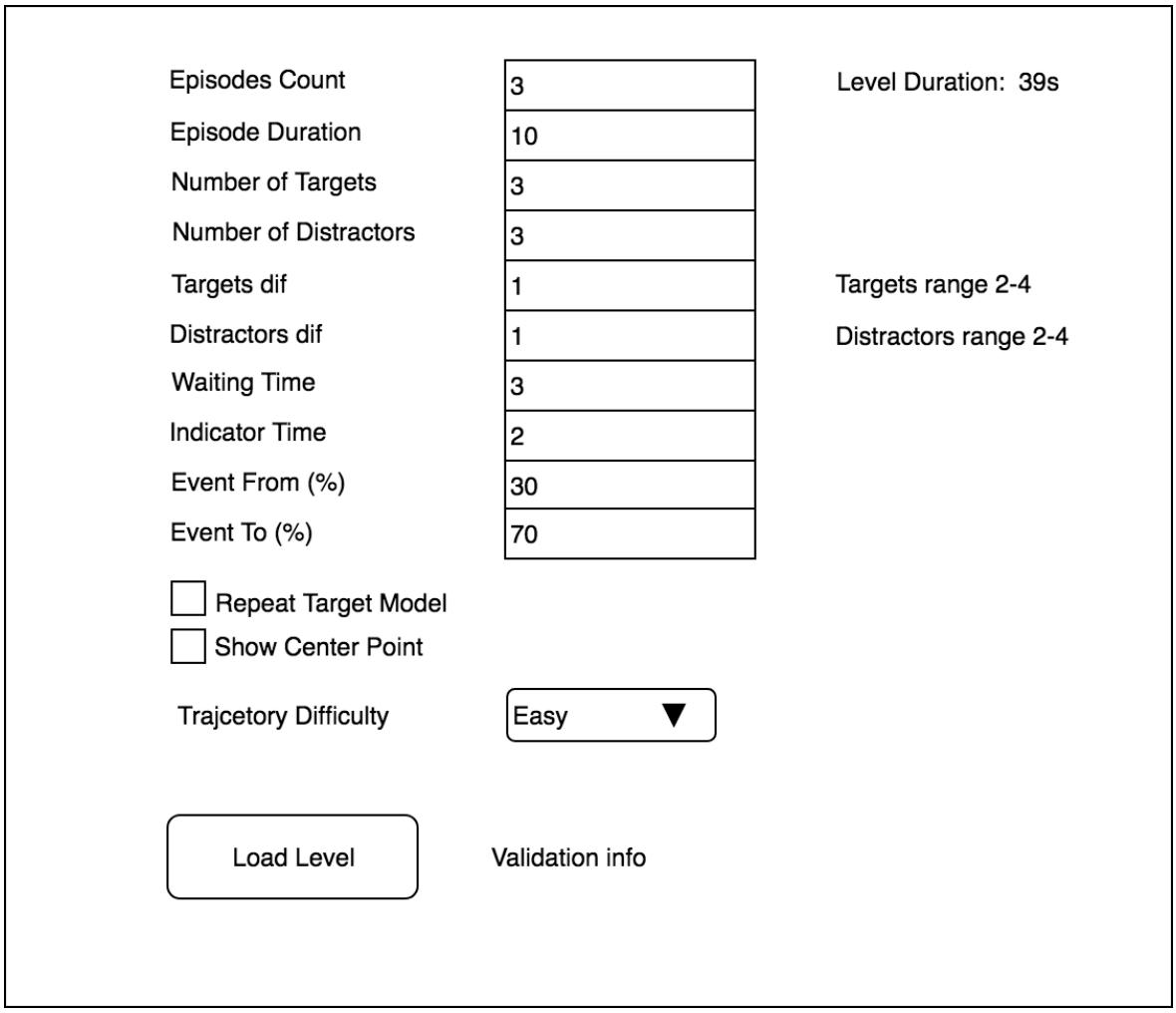
7.2. Nastavenie úrovne

V tejto časti sa nachádzajú všetky potrebné prvky pre nastavenie súčasnej úrovne. Je hlavnou časťou používateľského rozhrania, ktorá umožní používateľovi, v tomto prípade vedúcemu experimentu správne nakonfigurovať jednotlivé parametre potrebné pre experiment. Používateľ bude môcť nakonfigurovať:

- Počet epizód (Episodes count) – celkový počet epizód, textové pole, ktoré bude validované ako číslo;
- Trvanie epizódy (Episode duration) – trvanie jednej epizódy, textové pole, ktoré bude validované ako číslo;
- Počet tragetov (Number of targets) – počet targetov v jednej epizóde, textové pole, ktoré bude validované ako číslo;
- Počet distraktorov (Number of distractors) – počet distraktorov v jednej epizóde, textové pole, ktoré bude validované ako číslo;
- Target dif (Targets dif) - variabilná zložka počtu targetov, textové pole, ktoré bude validované ako číslo, pričom veľkosť tohto čísla musí byť väčšia ako počet targetov. Ak je počet targetov 5 a variabilná zložka target dif je 2, v epizóde sa môže objaviť 5 ± 2, číže 3 až 7;
- Disktraktor dif (Disktractors dif) - podobne ako target dif to bude varibilná zložka počtu distraktorov ,pričom veľkosť tohto čísla musí byť väčšia ako počet distaktorov.
- Čas medzi epizódami (Waiting time) – čas od skončenie jednej epizódy a začiatku ďalšej, textové pole, ktoré bude validované ako číslo;
- Čas trvania indikátora (Indicator time) – doba pokiaľ bude zobrazený indikátor, ktorý zobrazuje pri modeli či sa jedná o disktraktor alebo target, textové pole, ktoré bude validované ako číslo;
- Nastatie udalosti od (Event from) – percento udávajúce, od ktorej časti epizódy môže dôjsť k udalosti (zatmeniu). Napríklad, ak epizóda má dobu trvania 10 sekúnd a nastatie udalosti je 30%, táto udalosť môže nastať od tretej sekundy, textové pole, ktoré bude validované ako číslo;
- Nastatie udalosti do (Event to) – podobne ako nastatie udalosti od, ale v tomto prípade sa jedná o časť epizódy, do ktorej môže nastať udalosť, textové pole, ktoré bude validované ako číslo;
- Pravdepodobnosť nastatia udalosti – (Event Probability) – pravdepodobnosť nastatia udalosti v percentách, textové pole, ktoré bude validované ako číslo;

- Zobrazenie stredu (Show center point) - nastavuje zobrazenie/nezobrazenie stredového bodu, zaškrtávacie pole;
- Opakovanie sa modelov targetov – povoľuje použitie rovnakého modelu targetu v dvoch po sebe nasledujúcich epizódach;
- Zložitosť trajektórií – veľkosť rozptylu trajektórií, číselník s hodnotami ľahké(easy),stredné(medium),tažké(hard).

Súčasťou tejto časti je aj tlačidlo, pomocou ktorého sa konfigurovaná úroveň načíta. Táto časť bude taktiež zobrazovať pomocné informácie ako sú celková dĺžka úrovne(počet epizód*(dĺžka epizódy + čas medzi epizódami)), rozsah tartgetov a distraktorov. Rozmiestenie elementov je zobrazené na Obr. 4 .



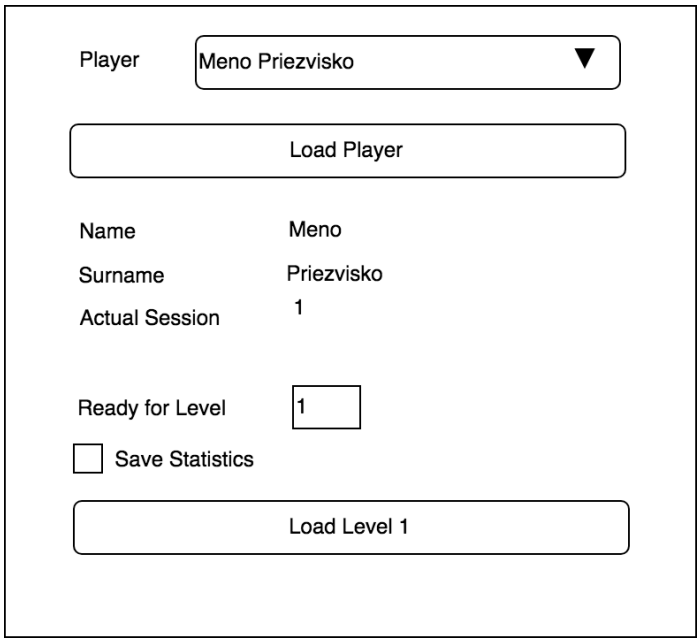
Obr. 4 Návrh časti: konfigurácia úrovne

7.3. Informácie o hráčovi

Tretia časť používateľského rozhrania je zameraná na hráča. Táto časť obsahuje všetko potrebné k tomu, aby bolo možné hráča načítať a nastaviť mu požadovanú úroveň. Taktiež by mala poskytnúť informácie vedúcim experimentu o tom, akú úroveň hral hráč naposledy a ako sa mu v ňom darilo. Vedúci experimentu musí následne vedieť vybrať ďalšiu úroveň, ktorú hráčovi nastaví. Rovnako môže zobrazovať informácie o hráčovi ako meno, priezvisko, aktuálne sedenie. Táto časť sa bude skladať z:

- Hráč (Player) – číselník hráčov;
- Načítanie hráča (Load Player) – tlačidlo, ktoré načíta hráča a informácie o ňom;
- Meno (Name) – informácia o krstnom mene hráča;
- Priezvisko (Surname) - informácia o priezvisku hráča;
- Aktuálne sedenie (Actual session) – informácia o tom, ktoré sedenie aktuálne absolvuje;
- Úspešnosť predchádzajúcej úrovne (Last game stats) – percentuálne úspešnosť zostrelení v poslednej hranej úrovni pre tagety a distraktory;
- Úroveň (Level) – textové pole pre načítanie úrovne, 1-n, kde n je posledná úroveň;
- Ukladanie štatistík (Save statistics) – zaškrtávacie pole, pomocou ktorého je možné nastaviť ukladanie/neukladanie štatistík;
- Načítaj úroveň (Load Level) – načíta úroveň zadanú v textovom poli úroveň.

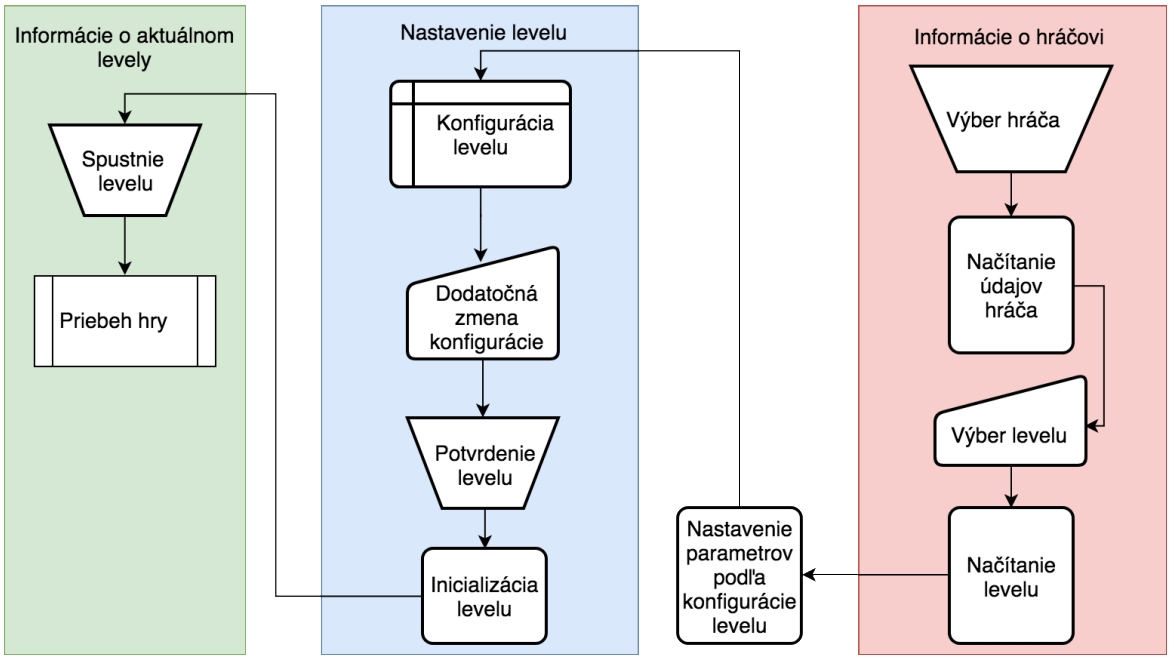
Rozloženie elementov sekcii informácie o hráčovi je zobrazené na Obr. 5.



Obr. 5 Návrh časti: výber hráča

7.4. Návrh postupnosti operácií v rámci používateľského rozhrania

Okrem návrhu rozloženia elementov na obrazovke je potrebné navrhnúť aj proces a postupnosť jednotlivých krokov v rámci výberu a nastavenia úrovne. Používateľské rozhranie musí byť navrhnuté tak, aby spĺňalo požiadavky na experiment, a zároveň musí poskytnúť jasnú postupnosť krokov bez toho, aby došlo k nesprávnej konfigurácií, čo by mohlo mať za následok nesprávne fungovanie programu, prípadne jeho zlyhanie. V tomto smere bude používateľské rozhranie poskytovať presne definovaný postup. Tento postup je znázornený na diagrame, ktorý môžeme vidieť na Obr. 6.



Obr. 6 Postupnosť operácií v rámci používateľského rozhrania

Ako vyplýva z Obr. 6, prvým krokom je výber hráča vedúcim experimentu. Po zvolení hráča sa automaticky načítajú jednotlivé informácie. Následne je potrebné zvoliť úroveň, ktorú bude hrač hrať. Zvolením úrovne sa prednastaví konfigurácia úrovne tak, že príslušné polia sa predvyplnia a znemožní sa ďalšie nastavovanie. Používateľ môže v tomto kroku zmeniť iba parametre, ktoré sú globálne nastavené pre všetky úrovne(primárne meniť nemusí nič). Úroveň sa definitívne inicializuje stlačením tlačidla “Load Level”, pričom sa sprístupní tlačidlo “Start Level”. Po kliknutí na tlačidlo “Start Level” sa hra spustí.

7.5. Návrh používateľského rozhrania pre ukážku experimentu v desktopovej verzii

Pre prezentačné účely je potrebné navrhnúť používateľské rozhranie, pomocou ktorého bude možné spustiť ľubovoľnú uroveň. Táto zmena sa týka najmä remote console SuperEnginu. Okrem tejto zmeny je potrebné taktiež navrhnúť zobrazenie informácie pre hráča po ukončení úrovne, kde sa mu zobrazia informácie o odohratej úrovni. Prezentačná verzia sa bude využívať pre prezentovanie projektu mimo laboratória LIRKIS. Hráči si tak budú môcť vyskúšať hru na akomkoľvek mieste.

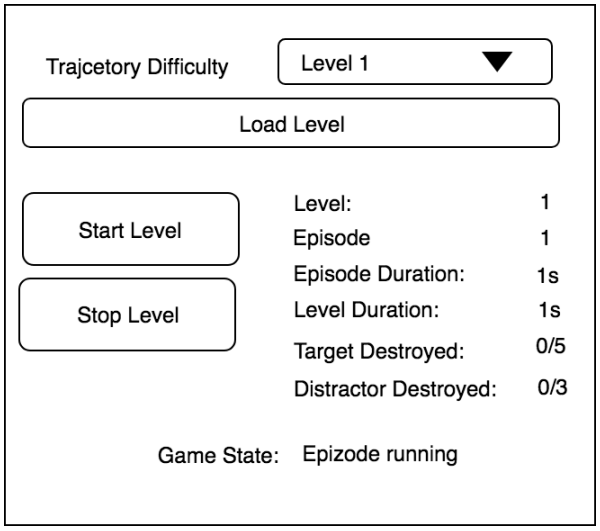
7.5.1. Spúšťanie úrovne

Úroveň pre prezentačnú verziu bude možné spustiť pomocou výberu požadovanej úrovne z číselníka, kde predvolená hodnota bude 1. Po výbere úrovne sa hra spustí podobne ako pri základnej verzii stlačením tlačidla „Start level“.

7.5.2. Ukončenie úrovne

Po ukončení úrovne sa používateľovi zobrazia priamo na obrazovke jeho štatistiky o tom ako sa mu darilo v danej úrovni. Tieto štatistiky budú obsahovať informáciu o počte targetov a distraktorov spolu s počtom zostrelení. Úspešnosť úrovne bude reprezentovaná percentuálnou hodnotou.

Rozloženie rozhrania pre prezentačnú verziu je zobrazené na Obr. 7.



Obr. 7 Návrh rozhrania pre prezentačné účely

Po zvolení úrovne z číselníka sa daná úroveň načíta stlačením tlačidla “Load Level” a následne sa spustí tlačidlom “Start Level”. Po ukončení úrovne sa používateľovi zobrazia výsledky, ktoré v danej úrovni dosiahol.

8. Návrh ukladania dát a používateľských štatistík

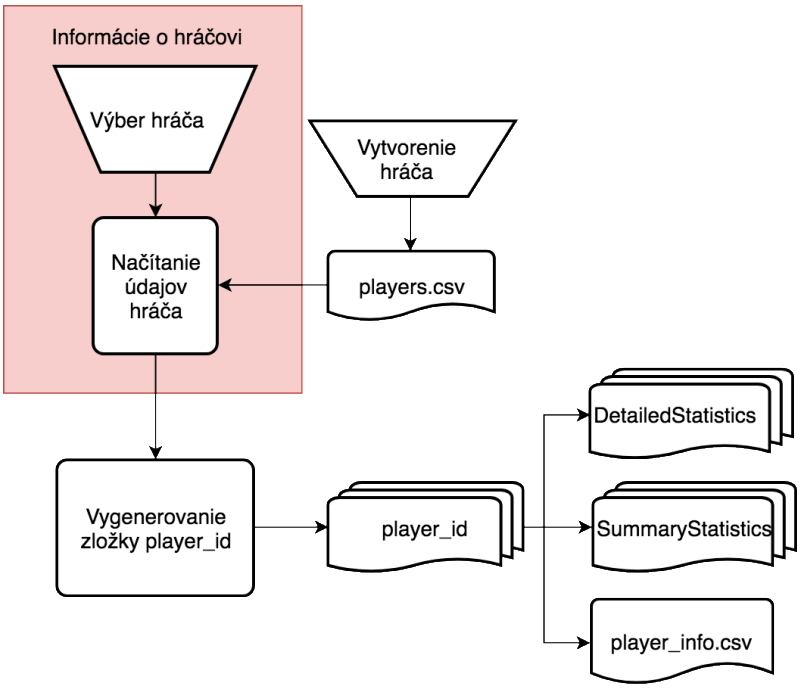
Prácu s dátami bude zabezpečovať súborová štruktúra, ktorá bude súčasťou hry. Hra bude obsahovať priečinok „Players“, kde budú uchované všetky potrebné dáta o jednotlivých hráčoch, podľa potrieb experimentu. Ako vyplýva z analýzy, koncové dáta budú ukladané ako tabuľky vo formáte csv.

8.1. Návrh súborovej štruktúry

Pre uchovávanie dát bude slúžiť súborová štruktúra, ktorá bude logicky rozčlenená na jednotlivé časti. Primárne sa bude skladať zo súborov “`players`”, kde budú uchované dáta o hráčoch a “levels”, kde budú uložené jednotlivé úrovne. Zložka “players” bude v sebe uchovávať jednu hlavnú tabuľku so všetkými hráčmi. Do tejto tabuľky bude možné pridávať nových hráčov, ktorí sa následne zobrazia v používateľskom rozhraní, používateľ už ďalej do tejto štruktúry zasahovať nemôže. Zložka “levels” bude mať v sebe csv súbor s názvom levels.csv. Tento súbor bude uchovávať dáta o každej úrovni.

8.1.1. Automatické generovanie súborov

O ďalšie dotvorenie dát sa bude starať služba, ktorej úlohou bude po zvolení hráča dogenerovať zvyšné súbory, pokiaľ ešte nie sú vytvorené. To znamená, ak používateľ vytvorí v players/player.csv nového hráča a následne ho v používateľskom rozhraní načíta, systém automaticky vytvorí v zložke players podzložku s názvom player_id, kde id je jedinečné číslo hráča. V novovytvorenej zložke následne pripraví zložky detailedStatistics a summaryStatistics. Pre uľahčenie vyhľadávania informácií o hráčovi sa v zložke players/player_id vytvorí csv súbor s názvom player_info.csv, kde sa uložia všetky informácie o hráčovi. Do zložky detailedStatistics sa budú ukladať záznamy pre každú úroveň osobitne, pričom po skončení každej úrovne sa vygeneruje nový csv súbor s detailnými štatistikami. Názov tohto csv súboru sa vyskladá ako leve_n_timestamp, kde n je číslo úrovne a timestamp aktuálna jedinečná časová známka. V zložke summaryStatistics sa bude nachádzať jeden csv súbor s názvom summaryStats.csv, kde každý riadok predstavuje jednu osobitnú úroveň. Súborovej štruktúre je takto zabezpečená dostupnosť. V prípade, že niektorá časť tejto štruktúry chýba, alebo nie je možné k nej pristupovať, systém ju automaticky doplní. Proces vygenerovania súborov je znázornený v diagrame na Obr. 8.



Obr. 8 Proces generovania súborov

8.1.2. Štruktúra CSV súborov

Ako bolo spomenuté v predošlej časti, bude potrebné pripraviť csv súbory: players.csv, summaryStatistics.csv, level_id_timestamp.csv, player_info.csv, levels.csv. Súbory players.csv a levels.csv budú vytvárané používateľom, ostatné sa budú generovať automaticky. Zloženie jednotlivých súborov bude nasledovné:

- players.csv – meno;priezvisko;id – (name;surname;id);
- levels.csv – Úroveň;trvanie_epizódy;obtiažnosť_trojektórií;targety;distraktory – (Level;Episode time;Swarm angle;Targets;Distractors);
- level_id_timestamp – Epizóda;Čas v episode (s);Udalosť;Zničené;Časová známka – (Episode;Time In Episode (s);Event;Destroyed;Timestamp);
- summaryStats.csv – Úroveň; Počet výstrelov;Počet targetov;Počet zničných targetov;Počet Disktraktorov;Počet zničených distraktorov;Časová známka – (Level; Number of Shoots;Number of Targets;Number of Targets Destroyed;Number of Distractors;Number of Distractors Destroyed;Timestamp);
- player_info.csv - meno;priezvisko;id – (name;surname;id);

8.2. Návrh procesu načítania hráčov a detail hráča

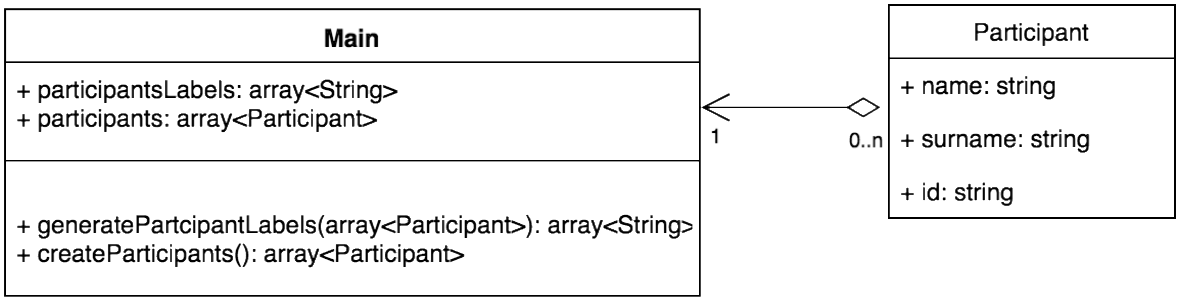
Pre experiment je potrebné zabezpečiť správne načítanie hráčov. Prvou úlohou je vytvorenie číselníka, na základe ktorého bude používateľ vedieť zvoliť hráča. Ako už bolo spomenuté, tento

číselník bude v používateľskom rozhraní vytvorený ako combo box element remote console, ktorý je potrebné naplniť správnou štruktúrou dát. Systém musí pri spustení nájsť súbor players.csv, z ktorého využitím knižnice csv.rb otvorí a prečíta dáta uložené v tomto súbore. Spôsob, akým sa načítajú dáta z player.csv:

- `CSV.open("players/players.csv",'r',' ;') do |row|`
  … operácia nad riadkom
  `end`

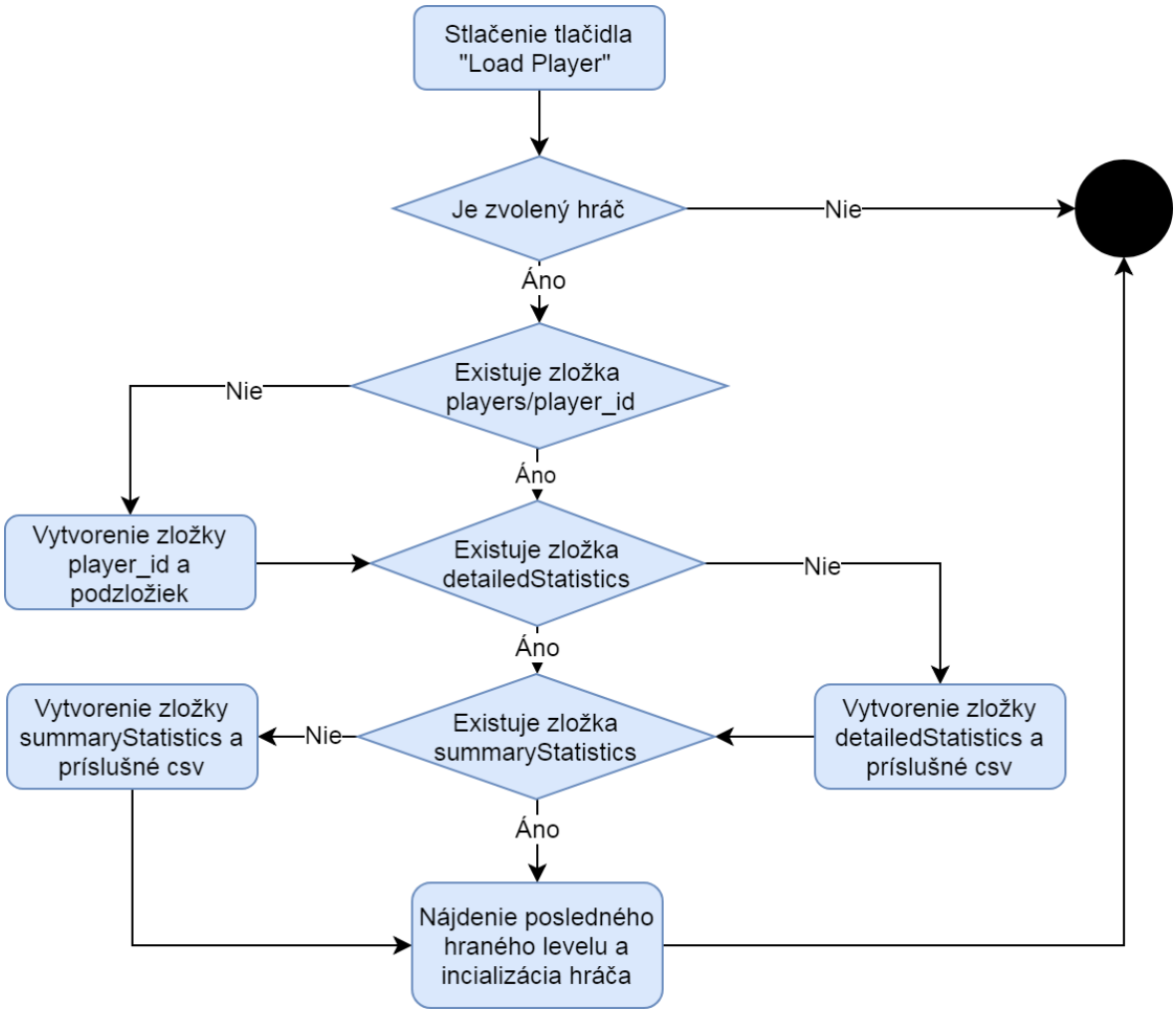
- r = prepínač, ktorý zabezpečí, že je možné zo súboru iba čítať
- ; = znak oddeľujúci stĺpce v csv súbore

Funkcionalita bude oddelená do samostatnej funkcie, ktorá pri volaní vráti aktuálny zoznam hráčov. Každý hráč bude v pamäti následne uložený ako samostatná trieda. Následne je potrebné vytvoriť štruktúru, s ktorou dokáže pracovať číselník. Číselník pracuje s poľom reťazcom. Tento reťazec bude tvorený z mena a priezviska hráča. Generovanie reťazca pre číselník bude rovnako oddelené v samostatnej funkcii, ktorú bude možné použiť pre aktualizáciu hráčov. Vzťah medzi hlavnou triedou Main.rb a Participant.rb je popísaný na diagrame na Obr. 9.



Obr. 9 Diagram vzťahu medzi hlavnou scénou a navrhovanou triedou Participant

Načítanie detailu bude možné vykonať po zvolení konkrétneho hráča z číselníka stlačením tlačidla "Load Player", ktoré je umiestnené pod číselníkom (viď kap. 7.3). Toto tlačidlo bude možné stlačiť až po zvolení hráča. Dovtedy bude v stave "disabled". Po stlačení tlačidla sa spustí proces, ktorého úlohou bude vygenerovať súborovú štruktúru a uložiť si informácie o aktuálnom hráčovi do pamäte, pričom aktuálny hráč bude inštanciou triedy Participant. Proces je popísaný na diagrame na Obr. 10



Obr. 10 Proces načítania hráča a príslušnej úrovne

Po načítaní hráča sa následne sprístupní časť pre načítanie úrovne (nastaví sa zo stavu blokovaný(disabled) na neblokovaný(enable)).

9. Návrh procesu načítania úrovne

Podobne ako načítanie hráča je potrebné zabezpečiť, aby sa hráčovi nastavila správna úroveň. Ako už bolo spomenuté, úrovne sú preddefinované v súbore level.csv. Zo súboru sa budú následne načítavať podľa zadaného čísla úrovne v používateľskom rozhraní. Stlačením tlačidla "Load Level" sa spustí proces hľadania a načítania úrovne. Pri načítaní úrovni je potrebné zohľadniť, že niektoré hodnoty v nastavení úrovne sú nečíselné a je potrebné im priradiť správnu hodnotu. Zo špecifikácií experimentu vyplýva, že čas epizódy je zadaný ako enumeračný typ obťažnosti s hodnotami: Short, Middle, Long. V programe sa musia tieto hodnoty pretransformovať na číslo udávajúce dĺžku epizódy v sekundách. Hodnoty budú pridelené podľa vzorca:

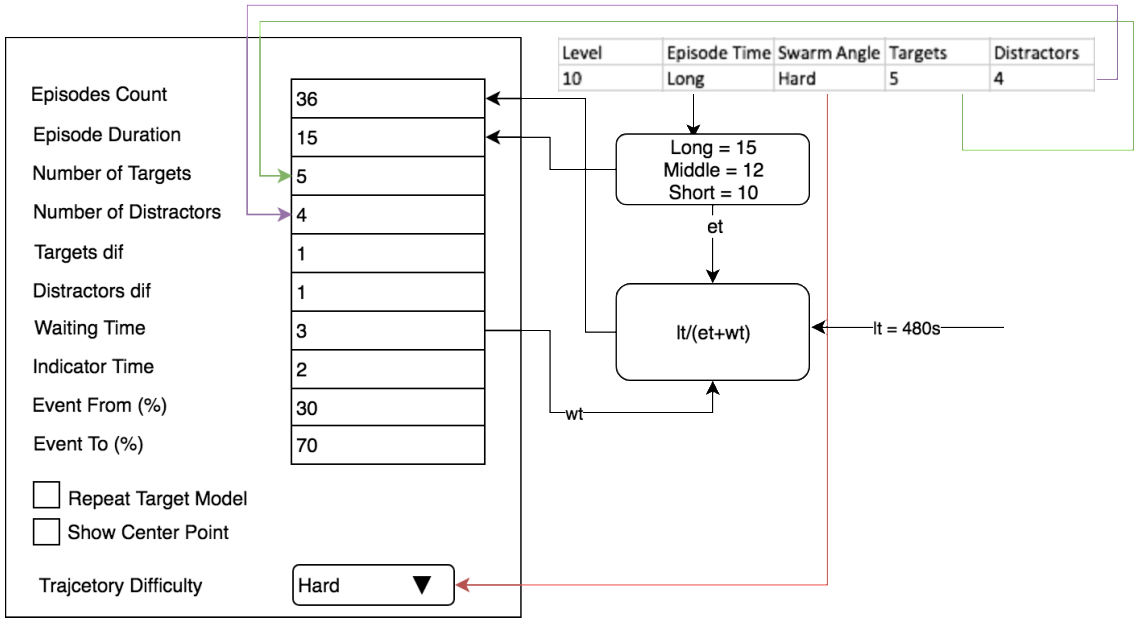
- Short = 10 sekúnd
- Middle = 12 sekúnd
- Long = 15 sekúnd

Ďalšou nečíselnou hodnotu je obťažnosť trajektórii, ktorá je definovaná hodnotami Easy, Medium, Hard. Obťažnosť trajektórií v hre predstavuje veľkosť rozptylu jednotlivých objektov. Jedná sa o vertikálny, ale aj horizontálny rozptyl. Zo špecifikácií experimentu taktiež vyplýva, že dĺžka jedného sedenia je 30 minút, počas ktorých subjekt odohrá 3 úrovne. Odhadovaný čas na jednu úroveň je 8 minút a 2 minúty prestávka pred ďalšou úrovňou. Počet epizód sa teda vyráta ako lt/(et+wt), kde lt je konštantná dĺžka úrovne 8 minút(480s), et je dĺžka jednej epizódy získaná zo súboru a wt je čas medzi epizódami. Spôsob akým sa načítajú dáta z levels.csv:

- ```
  CSV.open("levels/levels.csv",'r',' ;') do |row|
  ```
  … operácia nad riadkom
  ```
  end
  ```

- r = prepínač, ktorý zabezpečí, že je možné zo súboru iba čítať
- ; = znak oddelujúci stĺpce v csv súbore
  ```

Na diagrame na Obr. 11 je zobrazený celý proces, ktorým sa načíta a nastaví úroveň.



Obr. 11 Proces nastavenia úrovne

Po načítaní úrovne a preklopení hodnôt z csv do používateľského rozhrania sa príslušné nastavené hodnoty stávajú ďalej needitovateľné. Používateľ môže editovať iba globálne premenné pre celý experiment v prípade potreby.

9.1. Náročnosť úrovne

Náročnosť úrovne bude priamo vychádzať z konfigurácie danej úrovne. Preddefinované úrovne budú mať teda fixnú náročnosť. Náročnosť bude závislá od nastavenia dĺžka epizódy, veľkosti uhla rozptylu trajektórií, počtu targetov a distraktorov. Návrh úrovní 1-30 je plne v kompetencii expertov z oblasti kognitívnej vedy, ktorí túto náročnosť definujú. Pri spomínanom spôsobe načítavania úrovne je možné hocikedy v prípade požiadavky zmeniť csv súbor obsahujúci dané úrovne.

10. Návrh zaznamenávania dát – zapisovania aktivity

Zaznamenávanie štatistických dát je najdôležitejšou časťou experimentu, pretože na základe týchto dát bude možné spätne vyhodnotiť ako sa účastníkovi experimentu darilo v jednotlivých úrovniach. Podobne pri návrhu zaznamenávania aktivity vychádzame z požiadaviek experimentu. Je potrebné navrhnúť spôsob, akým bude možné jednoducho zaznamenať požadovanú aktivitu tak, aby bolo možné ľahko reagovať na zmeny aj v priebehu samotného experimentu. Ako bolo spomenuté (kap. 5), štatistiky sú rozdelené na detailné a sumárne.

10.1. Proces spracovania detailných štatistík

Na zaznamenanie detailných štatistík bude vytvorená premenná, do ktorej sa budú vkladať jednotlivé záznamy v podobe reťazcov. Primárne sa budú zaznamenávať aktivity:

- Začiatok epizódy (episodeStart) = informácia o tom že začala epizóda;
- Zatmenie (eclipse) = informácia o nastatí udalosti zatmenia v príslušnom čase epizódy;
- Výstrel (shoot) = informácia o tom, že hráč vystrelil, pri výstrele môžu nastať 3 situácie:
    - Netrafil nič (‘ ’)
    - Trafil target (target)
    - Trafil distraktor (distraktor)

Každá z týchto udalostí sa zaznamená tak, že sa pridá reťazec v tvare:

“episode_x_ns_event_destroyed_timestamp”

- x = poradové číslo epizódy (0+);
- n = sekunda epizódy, v ktorej nastala udalosť (0-dĺžka epizódy);
- event = type udalosti, ktorý nastal: episodeStart,eclipse,shoot;
- destroyed = cieľ, ktorý bol zničený, zapĺňa sa iba pri výstrele (shoot): target, distractor;
- timestamp = časová známka nastatia udalosti vo formáte hhmmss_dd-mm-yyyy .

Po skončení každej úrovne sa tieto dáta preložia do štatistík umiestených v zložke hráča tak, že sa vytvorí nový csv súbor s názvom level_x_timestamp. X je číslo úrovne a timestamp časová známka. Časová známka zabezpečí, že jeden hráč môže mať v štatistikách viackrát rovnakú úroveň v prípade, že nesplní podmienky pre postup do ďalšej. Na konci každej úrovne bude potrebné pretransformovať reťazec spomínaný vyššie na csv dáta, ktoré bude možné uložiť. O to sa bude starať funkcia, ktorá na vstupe dostane pole štatistických záznamov, meno súboru, do ktorého tieto dáta uloží, identifikačné číslo hráča, ktorý aktuálne hrá a separátor, ktorý bude primárne nastavený na znak ‘;’. Funkcia bude vďaka týmto údajom schopná vytvoriť nový súbor v ceste:

„players/player_id/detailedStatistics/názov_súboru.csv“

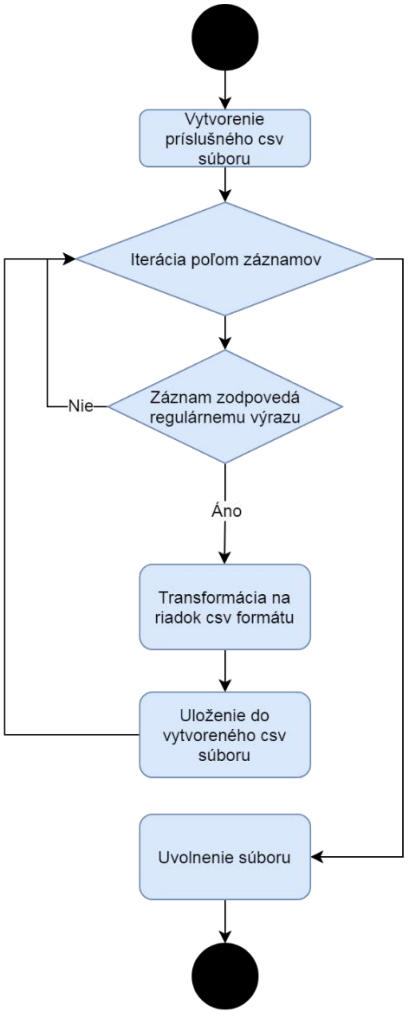
Nato sa využije knižnica csv.rb a súbor sa vytvorí príkazom:

- CSV.Open(cestaSúboru,’wb’), kde prepínač wb umožní zapisovanie do tohto súboru

Do tohto novovytvoreného súboru sa následne vloží hlavička pre detailné štatistiky. Následne sa iteračným cyklom bude prechádzať každý jeden záznam z poľa detailných štatistík a pomocou regulárneho výrazu sa overí správnosť týchto dát. Tento regulárny výraz bude v tvare:

/^(episode[1-9][0-9]*)_(\d+s)_(shoot|eclipse|episodeStart)_(target|distractor|\s*)_(.*)$/).

V prípade, že záznam zodpovedá regulárnemu výrazu, vloží sa celý záznam do vytvoreného csv súboru pričom sa znak “_” nahradí separátorom. Celý tento proces je znázornený na diagrame na Obr. 12.



Obr. 12 Proces uloženia detailných štatistík

10.2. Proces spracovania sumárnych štatistík

Podobne ako pri detailných štatistikách bude vytvorená metóda, ktorá zabezpečí po ukončení úrovne spracovanie a uloženie štatistik. V prípade sumárnych štatistík sa ale bude jednať iba o jeden záznam do súboru summaryStatistics.csv, kde sa budú zhromažďovať všetky sumárne štatistiky pre jedného hráča. Do sumárnych štatistík sa budú zapisovať dáta:

- Poradové číslo úrovne (level);
- Celkový počet výstrelov (Number of Shoots);
- Celkový počet targetov (Number of Targets);
- Celkový počet zničených targetov (Number of Targets Destroyed);
- Celkový počet distraktorov (Number of Distractors);
- Celkový počet zničených distraktorov (Number of Distractors Destroyed)
- Časová známka (Timestamp);

Na spracovanie dát bude slúžiť samostatná metóda, ktorá na vstupe dostane vyskladaný reťazec v tvare:

“level_x_s_t_td_d_dd_timestamp”

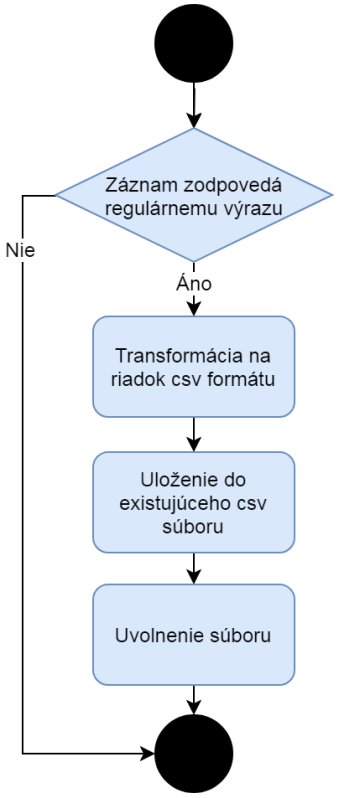
- x = odohratý level;
- s = počet výstrelov, ktorý sa bude zaznamenávať;
- t = počet targetov za celú úroveň;
- td = počet zničených targetov za celú úroveň;
- d = počet distraktorov za celú úroveň;
- dd = počet zníčených distraktorov za celú úroveň;
- timestamp = časová známka.

Ďalšími parametrami bude identifikačné číslo hráča a separátor, ktorý je štandardne nastavený na znak ‘;’. Keďže pri spustení programu sa zabezpečila dostupnosť súboru summaryStastics.rb a nepredpokladá sa manuálny zásah používateľa do tohto súboru, nemusíme už v tejto fáze overovať jeho dostupnosť. Overíme iba, či je vstupný reťazec validný. To dosiahneme tak, že ho overíme regulárnym výrazom:

(/^(level[1-9][0-9]*)_([0-9][0-9]*)_([0-9][0-9]*)_([0-9][0-9]*)_([0-9][0-9]*)_([0-9][0-9]*)$/)

K súboru potom pristúpime podobne ako pri detailných štatistikách:

CSV.open(cestaSuboru,'a'), kde prepínač umožní zapisovanie do už existujúceho súboru. Znak "_" sa nahradí separátorom. Celý tento proces je znázornený na diagrame na Obr. 13.



Obr. 13 Proces ukladania sumárnych štatistík

11. Návrh prostriedku pre motivovanie účastníkov experimentu

Ako vyplýva z analýzy, motiváciu účastníkov dosiahneme prostriedkom, pomocou ktorého bude možné porovnať výsledky hráčov. Výsledky hráčov uložené na disku v laboratóriu LIRKIS je možné publikovať pomocou webovej aplikácie, kde sa budú zobrazovať jednotliví účastníci experimentu. Každý účastník bude vo výsledkoch vystupovať pod svojim jedinečným používateľským menom. Toto používateľské meno si zvolí sám pred začiatkom experimentu. Vzhľadom nato že prístup do siete v rámci laboratória LIRKIS je obmedzený a dáta sú dostupné iba v prípade, že je cave zapnutý, bude aktualizácia a synchronizácia dát riešená pomocou ručného vkladania dát do pripraveného repozitára servera, odkiaľ si bude aplikácia vedieť tieto dáta vziať. To znamená, že obsluha bude musieť po každej iterácií sedení ručne aktualizovať dáta v repozitári. Takéto riešenie je v poriadku z hľadiska, že nie je potrebné vytvárať ďalšiu službu, ktorá by sa starala o automatické aktualizovanie a obsluha má túto akciu pod plnou kontrolou. Celkovo bude teda aplikácia pre zobrazovanie výsledkov obsahovať serverovú časť, ktorá bude načítavať dáta zo spomínaných csv súborov a preposielať ich ďalej do front-endovej časti, ktorá bude tieto dáta zobrazovať. Aplikácia sa bude teda skladať zo serverovej časti a klientskej aplikácie.

11.1. Serverová aplikácia

Úlohou serverovej aplikácie bude prečítať csv súbory jednotlivých hráčov a pretransformovať jednotlivé dáta do formátu json. Tieto dáta sa následne budú pomocou REST API preposielať do klientskej časti. Serverová aplikácia bude vytvorená pomocou Node.js a OpenAPI. API bude disponovať niekoľkými requestami. Návrh requestov:

- `getScoreTable - /scoretable`
  - formát dát:
```
{
'players' : [
{
    '_id': 'string',
    'name': 'string',
    'score': 'number',
    'level_played': 'number',
}
]
}
```
  - Objekt v ktorom sa nachádza pole hráčov, kde každý hráč má svoje jedinečné id, meno, príslušné skóre a počet úrovní, ktoré už odohral

- getPlayerDetail: /player/{player_id}
  - formát dát: {

```
'_id': 'string',
'name':'string',
'score':'number',
'levels_played': [
  {
    'level':'number',
    'shoots': 'number,
    'distractors':'number,
    'distractors_destroyed': 'number',
    'targets': 'number',
    'targets_destroyed':'number',
  }
]
}
```

Serverová aplikácia si bude dáta načítavať zo zložky players, umiestnenej na disku servera. Pre aktualizáciu týchto dát bude nutné po každom dni, v ktorom prebehnú sedenia aktualizovať zložku tak, že sa táto zložka "players" nahradí novou.

11.1.1. Výpočet skóre hráčov

Keďže špecifikácie experimentu neobsahujú informácie o skóre a o tom, ako by mohlo byť vypočítavané, je potrebné navrhnúť ako bude skóre vznikať. Skóre bude vychádzať zo sumárnych štatistík, kde vezmeme jednotlivé údaje z úrovne a pomocou vhodného vzorca vypočítame dosiahnuté skóre. Maximálne skóre bude možné dosiahnuť 1000 bodov + 100 bodov za presnosť streľby za jednu úroveň. Skóre 1100 bodov dosiahne hráč, ktorý zostrelí všetky targety, žiaden distraktor a počet výstrelov sa bude rovnať počtu zostrelených targetov.

- Ts = max * (Td/T) – skóre za zostrelené targety
- Ds = Ts * (Dd/D) – skóre za zostrelené distraktory
- Ss = shoots_max * (Td/ts) – skóre za presnosť výstrelov
- S = Ts – Ds + Ss – celkové skóre

Max= maximálny počet bodov, T = celkový počet targetov, Td = celkový počet zostrelených targetov, D= celkový počet distraktorov, Dd = celkový počet zostrelených distraktorov, ts= počet výstrelov.

V dosiahnutom skóre bude potrebné zohľadniť aj úroveň, ktorú hráč odohral. To zohľadníme tak, že k celkovému skóre prirátame body: 10 * najvyššia odohratá úroveň. Celkové skóre hráča bude teda vyzerať nasledovne:

$$\sum_{1}^{maxLevel} S + maxLevel * 10$$

Rovnica 1 - Výpočet skóre

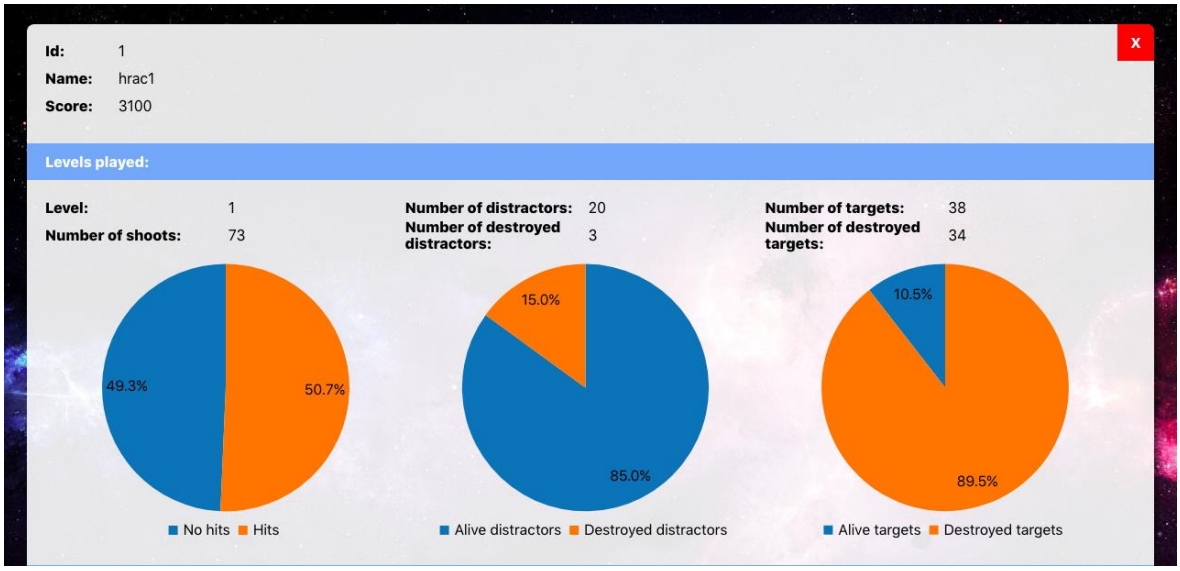
11.2. Klientská aplikácia

Klientská aplikácia bude zobrazovať dáta, ktoré prostredníctvom GET requestov dostane zo servera. Aplikácia bude vytvorená prostredníctvom technológie React.js. Aplikácia sa bude skladať z dvoch podstránok(obrazoviek). Na jednej obrazovke bude zobrazená tabuľka hráčov s ich celkovým skóre. Hráči budú zoradení podľa skóre zostupne. Po kliknutí na príslušný riadok sa zobrazí detail daného hráča, kde sa zobrazí každá odohratá úroveň s informáciami. Pre každý riadok budú zobrazené tri grafy:

- Graf úspešností streľby – pomer výstrelov a zostrelených targetov + zostrelených distraktorov;
- Graf zostrelených distraktorov – pomer distraktorov a zostrelených distraktorov;
- Graf zostrelených targetov – pomer targetov a zostrelených targetov.

Grafický návrh klientskej webovej aplikácie je zobrazený na Obr. 14 a Obr. 15.

Obr. 14 Grafický návrh obrazovky s celkovým skóre



Obr. 15 Grafický návrh obrazovky detail hráča

12. Implementácia používateľského rozhrania a konfigurácie

Používateľské rozhranie vychádza predovšetkým z návrhu (kap. 7) a analýzy jednotlivých elementov grafického rozhrania, kde je presne zdokumentované ich používanie. Konfigurácia je implementovaná statickým a dynamickým prístupom. Do statickej konfigurácie sú zaradené predovšetkým premenné potrebné pri vývoji, ktoré nie je nutné nastavovať dynamicky priamo z používateľského rozhrania. Prevažná časť konfigurácie úrovne sa nachádza v hlavnom skripte „main.rb“. Tento skript zabezpečuje hlavnú funkcionalitu SuperEnginu. Obsahuje taktiež hernú slučku.

12.1. Staticky nastavované premenné konfigurácie

Medzi staticky nastavované premenné patria všetky premenné, ktoré si nevyžadujú častú zmenu. Častou zmenou v našom prípade rozumieme každé sedenie účastníka experimentu, prípadne každý deň sedenia. Väčšina týchto premenných sa počas jednej iterácie experimentu vôbec nezmení. Prvou konfiguračnou premennou je premenná „CAVE“. Táto premenná môže nadobúdať hodnoty 0 a 1. V prípade, že je nastavená na hodnotu 0 znamená to, že kód chceme spustiť mimo virtuálnej jaskyne a tým sa neinicializujú špecifické operácie, ktoré sa vykonávajú v jaskyni ako sú napríklad OptiTrack, joystick a podobne. Naopak hodnota premennej nastavená na 1 tieto funkcie povolí. Ďalšou premennou je premenná s názvom „LEVEL_DURATION“. Táto premenná v sebe uchováva dĺžku jednej úrovne v minútach. Primárne vychádzajúc zo špecifikácie experimentu je nastavená na hodnotu 30 minút.

- $CAVE = 1
- $LEVEL_DURATION = 30

Ďalšími premennými sú:

- **$maxDistanceOfDroneForFocusing**: najväčšia vzdialenosť, na ktorú je možné zamerať dron, prednastavená hodnota je 1200
- **$towerLaserDurationTime**: dĺžka zobrazenia laséra, prednastavená hodnota je 0.1
- **$requiredTowerShootPower**: sila, hodnota potrebná aby bolo možné vystreliť, prednastavená hodnota je 100
- **$towerLaserRecoilSpeed**: rýchlosť akou sa predošlá premenná inkrementuje v prípade výstrelu, prednastavená hodnota je 150
- **$towerRotationSpeedVertical**: rýchlosť vertikálneho otáčania sa veže, prednastavená hodnota je 100

- **$towerRotationSpeedHorizontal**: rýchlosť horizontálneho otáčania sa veže, prednastavená hodnota je 55
- **$percentToStartTrajectory**: percentuálna hodnota údavajúca, v ktorej časti trajektórie sa zobrazí dron, prednastavená hodnota je: 50
- **$rangeOfRotation**: uhol o aký je možné vežou otáčať, prednastavená hodnota je: 50

Väčšina z týchto premenných a ich hodnoty vznikli počas vývoja samotnej hry. Ich správnemu nastaveniu predchádzalo viacero fáz testovania počas implementácie a po nej.

12.2. Dynamicky nastavované premenné konfigurácie

V návrhu používateľského rozhrania konfigurácie levelu sú presne definované jednotlivé nastavované premenné a ich rozmiestenie v rámci grafického rozhrania. Pri implementácií bol tento návrh spolu s vypracovanou dokumentáciu základnou časťou pre tvorbu funkčného rozhrania. Nato aby toto používateľské rozhranie fungovalo je potrebné mať vytvorenú premennú pre každý jeden vstupný parameter. Každá z týchto premenných v sebe uchováva aktuálnu hodnotu zadanú v parametri. Pred spustením scény majú prednastavené základne hodnoty.

12.2.1. Implementácia elementov UI

Ako vyplýva z návrhu, grafické rozhranie obsahuje niekoľko elementov, ktorými sa scéna nastavuje Každý jeden element musí byť pridaný do hlavného skriptu osobitne. Spôsob akým sú všetky elementy vytvorené je popísaný na príklade nižšie, kde implementujeme textové pole pre počet epizód.

```
$inputEpidesCount = $gm.createObject(“textField”, $panel2)
$inputEpidesCount.SetPosition(180,15)
$inputEpidesCount.SetValue($numberOfEpisodesTemp)
$inputEpidesCount.SetDimension(100,20)
$inputEpidesCount.AddEventListener(“valueChanged”,
method(:handleInputChange))
```

Takto sú vytvorené všetky elementy. Pozícia vychádza z návrhu, vstupným elementom sa priradia premenné a metóda reagujúca na príslušnú zmenu elementu. Priradenie elementov k premenným a ich prednastavenou hodnotou sú zobrazené v

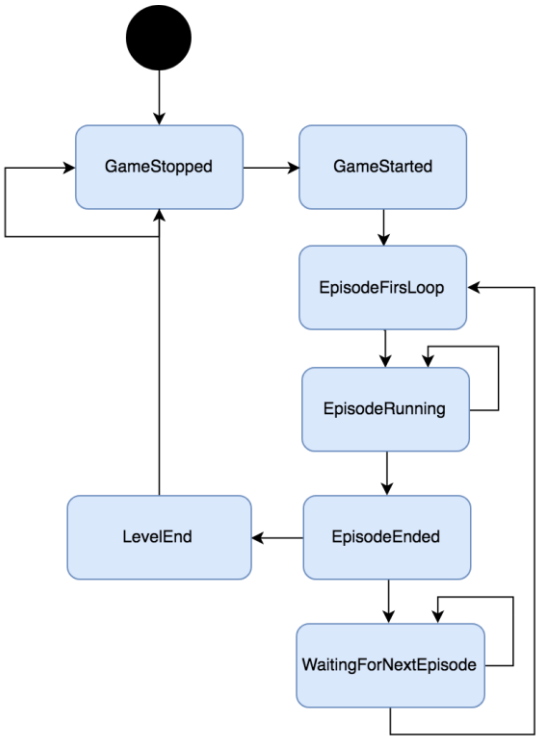
Tab. 1.

Štítok (label)	Názov premennej elementu	Názov premennej dát	Prednastavená hodnota
Episodes count	inputEpidesCount	numberOfEpisodesTemp	3
Episodes duration	inputEpideDuration	episodeDurationTemp	10
Number of targets	inputNumberOfTargets	numberOfTargetsTemp	3
Number of distractors	inputNumberOfDistractors	numberOfDistractorsTemp	3
Targets dif	inputTargetsDif	numberOfTargetsDiffTemp	1
Distractors dif	inputDistractorsDif	numberOfDistractorsDiffTemp	1
Waiting time	inputEpisodeWaitingTime	waitingTimeTemp	3
Indicator time	inputDroneIndicatorDuration	droneIndicatorTimeDurationTemp	1
Event from(%)	inputEventTimeFrom	evenTimeFromTemp	30
Event to(%)	inputEventTimeTo	eventTimeToTemp	70
Show center point	showCenterPoint	showCenterPointValue	true
Repeat target model	sameTargetModel	sameTargetModelIsPossible	true
Trajectory difficulty	trajectoryDifficultyInput	actualTrajectoryDifficulty	0
Even probability	inputEventProbablity	eventProbabalityTemp	30

Tab. 1 Premenné používateľskeho rozhrania

12.2.2. Stav hry v závislosti od konfigurácie

Spolu s riešiteľom druhej časti tejto hry bolo potrebné implementovať základne stavy hry. Tieto stavy ďalej uľahčujú prácu a vykonávanie programu v jednotlivých fázach. Okrem toho výrazne pozitívne vplývajú na prehľadnosť riešenia. Stavy hry sú implementované ako enumeračný typ triedy „GameState“ modulu „gameStates.rb“. Prechody a závislostí týchto stavov sú zobrazené na diagrame na Obr. 16.



Obr. 16 Diagram prechodov stavov hry

Každý zo stavov hry ma pridelenú príslušnú akciu, ktorá sa v danom stave vykonáva. Táto funkcionalita je bližšie popísaná v časti práce Bc.Petra Vasiľa.

12.2.3. Spracovanie dát elementov

Nato aby elementy správne fungovali je potrebné dáta spracovávať a validovať. Na zber dát z elementov slúži vytvorená metóda handleInputChange(event), ktorá je použitá v predošlej kapitole. Táto metóda použitá pre všetky elementy, ktoré vykonávajú nejakú zmenu. Hlavnou kostrou tejto metódy je sekvencia podmienok, ktoré identifikujú o aký vstupujúci element sa jedná.

Metóda vyzerá nasledovne:

```
def handleInputChange(event)
  src = event.GetSource – vráti inštanciu objektu, z ktorého event pochádza
  case(src)
     when $numberOfEpisodesTemp
      … operácia s príslušnou inštaciou elementu
  end
end
```

V prípade, že sa jedná o tlačidlá, metóda volá príslušnú priradenú metódu obsahujúcu funkcionalitu pre dané tlačidlo. Každé z tlačidiel používateľského rozhrania má priradenú funkcionalitu. V

Tab. 2 sú zobrazené jednotlivé prepojenia tlačidiel a funkcií s ich popisom.

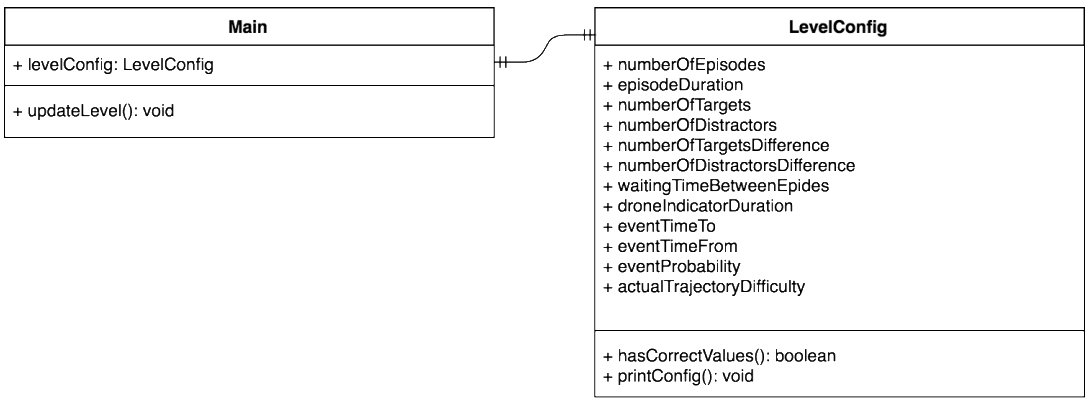
Tlačidlo	Funkcia	Popis
Stop Level	cleanGameField	Vyčistí hracie pole, re-inicializuje konfiguráciu zmena stavu hry na „GameStopped“
Start Level	initLevel	Inicializuje level a zmení stav hry na „GameStarted“
Load Level	loadLevel	Načíta úroveň z konfigurácie a pripraví hu
Load Player	loadPlayer	Načíta hráča z csv a nastaví ho ako aktuálneho hráča
Load Level from CSV	updateLevel	Načita úroveň z csv a predvyplní konfiguráciu

Tab. 2 Prepojenie tlačidiel a funkcií

12.2.4. Trieda „LevelConfig“

Trieda „LevelConfig“ nachádzajúca sa v module „levelConfig.rb“ má za úlohu držať v sebe dáta o aktuálne načítanej úrovni. Tejto metóde sa po správnom vyplnení konfigurácie posunú dáta, ktoré ďalej uchováva a poskytuje hernej slučke v jednotlivých stavoch hry. Okrem toho, táto trieda poskytuje funkciu, ktorá validuje vypĺňanú konfiguráciu tak, že kontroluje každú zadanú hodnotu.

Vzťah medzi hlavným skriptom „Main“ a triedou „LevelConfig“ je zobrazený na diagrame na Obr. 17. Využitím metódy hasCorrectValue vieme dovoliť respektíve zakázať spustiť level používateľovi tak, že príslušné tlačidlo na načítanie levelu sprístupníme až v prípade validného nastavenia dát.



Obr. 17 Vzťah medzi triedou LevelConfig a hlavným skriptom

12.2.5. Obnova používateľského rozhrania v hernej slučke

Obnova alebo aktualizovanie používateľského rozhrania prebieha v každej iterácií hernej slučky. Pri obnove rozhrania sa využívajú stavy hry, podľa ktorých sa príslušne vykreslí grafické rozhranie. Rozlišujú sa dva stavy hry a to „GameStopped“ a „GameStarted“.

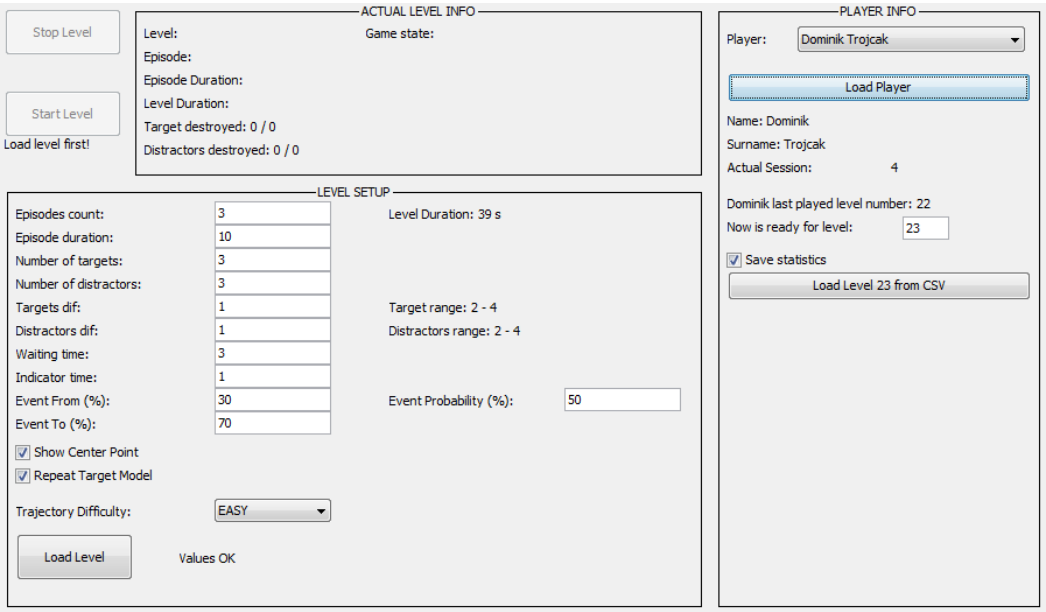
GameStopped

V prípade, že je hra v tomto stave, je možné editovať konfiguráciu úrovne. To znamená, že všetky príslušné elementy sú v stave „enabled" a to volaním funkcie SetEnabled(true) nad každým textovým elementom. Primárne je v tom stave prístupné iba tlačidlo „Load Player“ – podmienka, že musí byť zvolený a načítaný hráč. Ak je hráč načítaný sprístupní sa tlačidla „Load Level from CSV“ – medzikrok v prípade, že chceme danú úroveň načítavať z csv súboru a „Load Level“ – preklopenie dát využitím triedy „LevelConfig".

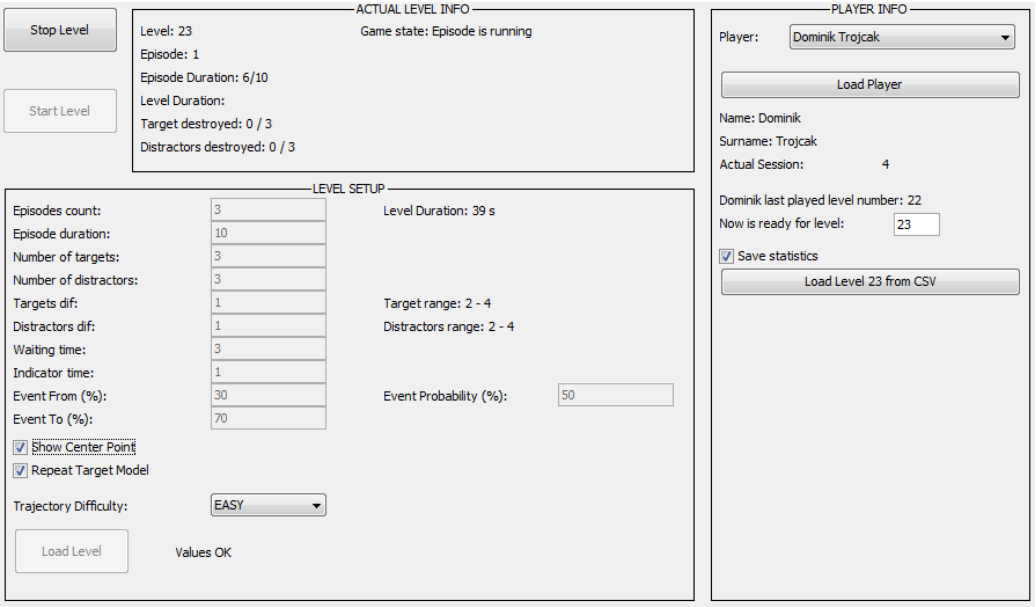
GameStarted

V tomto stave hry, už nie je možné žiadnym spôsobom zasahovať do konfigurácie a preto sa príslušné textové elementy nastavia na „disabled“ a to volaním funkcie SetEnabled(false). V porovnaní s predchádzajúcim stavom je teraz k dispozícii jediné tlačidlo „Stop Level“, ktoré vráti hru do stavu „GameStopped“.

Používateľské rozhranie je zobrazené na Obr. 18 (stav hry „GameStopped“) a Obr. 19 (stav hry „GameStarted“).



Obr. 18 Používateľské rozhranie – nespustená hra



Obr. 19 Používateľské rozhranie – spustená hra

13. Implementácia úrovní a hráčov

Implementácia tejto časti zahŕňa funkcionalitu, ktorá pokrýva zber dát a uchovávanie dát experimentu. Ako vyplýva z návrhu, dáta sú uchované v súborovej štruktúre. Inicializovanie tejto súborovej štruktúry je nasledovné:

```
$playersDirectory = './players'
$levelsDirectory = './levels'
```

Tieto dve zložky sú potrebné pre uchovávanie dát a musia byť v danej scéne dostupné. Nato aby mohla byť scéna spustená je taktiež nutné mať vytvoreného najmenej jedného hráča v súbore /players/players.csv (štruktúra vychádza z návrhu, kap. 8.1.2).

13.1. Hráči

Na základe kap. 8 o návrhu spôsobu načítania a prípravy jednotlivých procesov je implementovaná ich funkcionalita.

13.1.1. Načítanie a zobrazenie

Prvým krokom pri spúšťaní úrovne je výber hráča. Ako už bolo vyššie spomenuté bez tohto kroku nie je možné v rámci postupu pokračovať. O načítanie hráčov sa stará funkcia

- getParticipants() : array<Participant> - vráti aktuálny zoznam hráčov.

Funkcia si načíta csv súbor a prechádza jednotlivé záznamy pričom si hráčov(účastníkov) po jednom ukladá do poľa. Výsledkom je zoznam hráčov, ktorý sa uloží do premennej $participants. Následne sa tieto dáta zobrazia v číselníku „Player“ časti „Výber hráča“ používateľského rozhrania(viď obr. 7, 18). Na napĺnenie číselníka sa dáta normalizujú tak, aby ich číselník vedel zobraziť. Nato slúži funkcia:

- getParticipantLabels(participants: array<Participant>): array<String>

Funkcia teda vráti pole reťazcov, ktoré vyhovuje elementu číselník a zobrazí ich v našom rozhraní.

13.1.2. Zvolenie hráča a príprava dát

Po vylistovaní číselníka a zvolení konkrétneho hráča (kliknutí na hráča) sa zavolá funkcia „handleInputChange“ (bližsie špecifikovaná v 12.2.3) a zvolený hráč sa uloží do premennej $actualParticipant: Participant. To znamená, že v tomto momente sú k dispozícii všetky potrebné informácie o hráčovi a je možné hráča inicializovať v súborovej štruktúre využitím funkcie:

- `loadPlayer(): void`

Funkcia má prístup ku globálnej premennej „actualParticipant“. Vďaka tomu vie presne určiť, ktoré súbory má hľadať. Pri tomto procese sa kontroluje či požadovaná cesta k súborom a súbory vôbec existujú. V prípade, že niektoré časti chýbajú, systém ich dotvorí podľa návrhu z kap. 8.1.1. Pri generovaní súborov sa využíva knižnica Dir.

Proces kontroly a dotvárania štruktúry

1. Neexistuje zložka pre príslušného hráča

```
Dir.mkdir($playersDirectory + "player_" + $actualParticipant.id)

    Dir.mkdir($playersDirectory + "player_" + $actualParticipant.id +
"/detailedStatistics")

    Dir.mkdir($playersDirectory + "player_" + $actualParticipant.id + "/summaryStatistics")

    CSV.open($playersDirectory + "player_" + $actualParticipant.id + "/player_info.csv",
"wb", ";") do |csv|

      csv << ["NAME", "SURNAME", "ID"]

      csv << [$actualParticipant.name, $actualParticipant.surname, $actualParticipant.id]

    end
```

2. Neexistuje podzložka hráča /detailedStatistics

```
Dir.mkdir($playersDirectory + "player_" + $actualParticipant.id +
"/detailedStatistics
```

3. Neexistuje zložka hráča /summaryStatistics

```
Dir.mkdir($playersDirectory + "player_" + $actualParticipant.id +
"/sumaryStatistics
```

4. Neexistuje niektorý z csv súborov

```
CSV.open($playersDirectory + "player_" + $actualParticipant.id +
"/player_info.csv", "wb", ";") do |csv|

    csv << ["NAME", "SURNAME", "ID"]

    csv << [$actualParticipant.name, $actualParticipant.surname,
    $actualParticipant.id]
```

Táto funkcionalita zabezpečí, že vždy budú dostupné potrebné súbory, pokiaľ obsluha fyzicky dané súbory nezničí počas behu hry. Po pregenerovaní súborov sa získajú ďalšie informácie ako maximálne dosiahnutá úroveň, počet sedení.

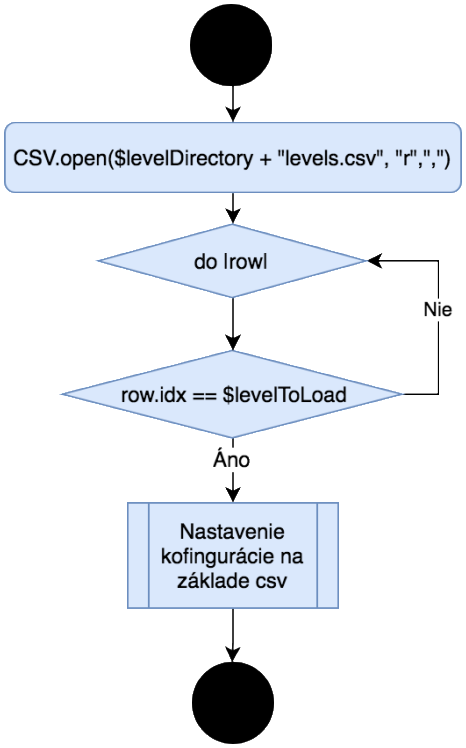
- `$actualLevel = levelsDone.max – levelsDone: array<Number>` - zoznam odohratých úrovní
- `$actualSitting = ((levelsDone.length/3) + 1)` – číslo sedenia

13.2. Úroveň

Na načítanie úrovne slúži metóda:

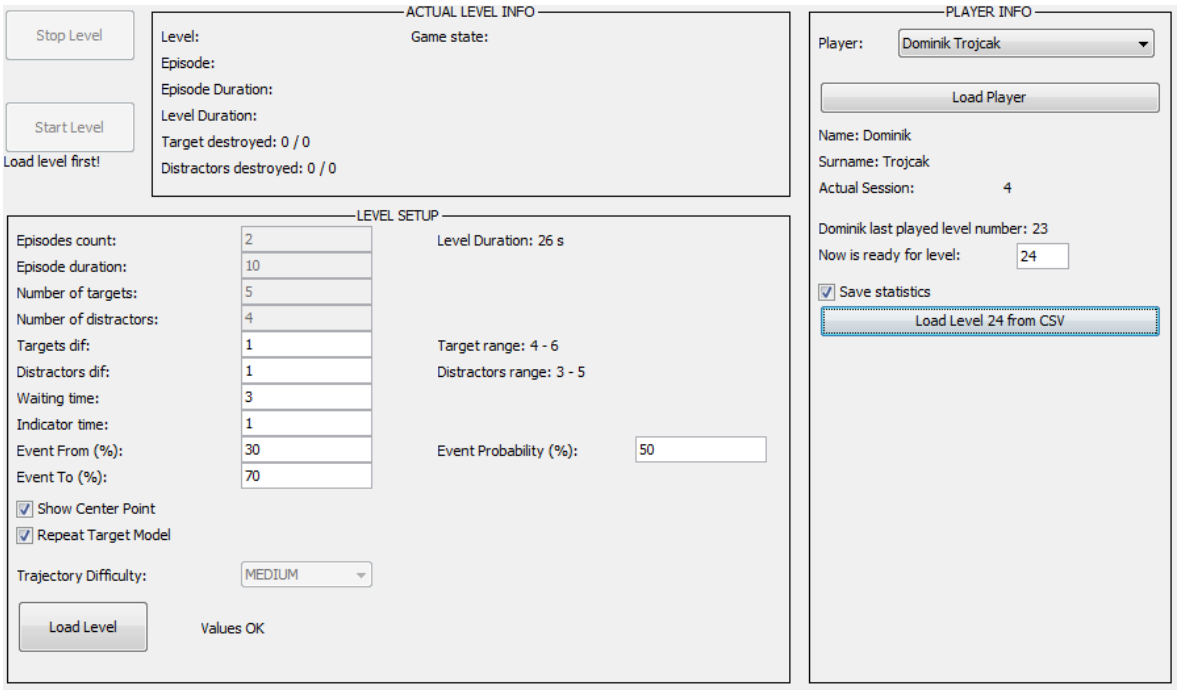
- `loadLevel(): void`

Táto metóda zo zložky /levels vyberá súbor level.csv. Na základe zadaného čísla používateľom následne nájde a vyberie príslušnú úroveň. Proces nastavenia a výberu levelu je detailnejšie popísaný v návrhu procesu načítania úrovne (kap. 9). Algoritmus načítania úrovne je popísaný na Obr. 20. Z grafického rozhrania sa načítavaný level vyberá pomocou textového poľa. Táto hodnota sa následne uloží do premennej $levelToLoad, pričom táto hodnota musí byť 1-maximálna úroveň.



Obr. 20 Algoritmus nastavenia úrovne z csv

Výsledkom tohto procesu je prednastavená konfigurácia získaná z csv súboru danej úrovne. Následne je možné túto konfiguráciu mierne upraviť. Časti konfigurácie, ktoré používateľ nesmie upravovať sú neaktívne. Vzhľad používateľského rozhrania po zvolení úrovne je zobrazený na Obr. 21.



Obr. 21 Vzhľad používateľského rozhrania po načítaní úrovnelevel z csv

13.3. Štatistické dáta

Na ukladanie štatistických dát slúži modul „statisticsParserUtils.rb“. Tento modul poskytuje kompletnú podporu pre parsovanie a ukladanie štatistických dát do formátu csv.

13.3.1. Detailné štatistiky

Na zozbieranie detailných štatistík sa využíva pole „$detailedStatisticsArray“. Do tohto poľa sa ukladá každý potrebný záznam, bližšie popísaný v kap. 10.1. V stave hry „levelEnded“ sa následne pole pretransformuje na csv súbor, pričom platí, že veľkosť poľa „$detailedStatisticsArray“ + 1 (hlavička csv) sa rovná počtu vygenerovaných riadkov csv súboru.

- `parseDetailedStatistics(statisticsArray:array<String>, nameOfFile: String, playerId: String, ):void`
  - statisticsArray – pole záznamov
  - namOfFile – názov súboru, ktorý sa vygeneruje
  - playerId – id aktuálneho hráča, pre ktorého sa štatistika generuje

Funkcia na základe mena vytvorí nový csv súbor a následne prechádza pole záznamov. Každý záznam prejde regulárnym výrazom, kde sa overí jeho správnosť a v prípade kladného výsledku sa záznam vloží do súboru. Do csv súboru sa dáta vkladajú následovne:

```
csv << [episode.gsub!("episode", ""), time.gsub!("s", ""), event, destroyed, timestamp]
```

13.3.2. Sumárne štatistiky

Na zozbieranie sumárnych dát sa využíva viacero premenných, ktoré zaznamenávajú aktivitu hráča.

- $shootCount – premenná uchováva počet výstrelov hráča, inkrementuje sa každý výstrelom
- $numberOfTargetsInLevel – premenná uchováva informáciu o celkovom počte targetov v úrovni, kedže toto číslo nie je dopredu známe a je čiastočne náhodné, inkrementuje sa pri každej vygenerovanej episode o príslušný počet targetov
- $numberOfDistractorsInLevel – podobne ako premenná „$numberOfTargetsInLevel" s tým rozdielom, že počíta distraktory
- $numberOfDestroyedTargets – uchováva informáciu o počte zničených targetov, inkrementuje sa každý zostrelom
- $numberOfDestroyedDistractors – uchováva informáciu o počte zničených distraktorov, inkrementuje sa každý zostrelom

V stave hry „levelEnded" sa tieto dáta vložia do csv súboru. Narozdiel od detailných štatistík sa sumárne ukladajú do jedného spoločného súboru. Bližšia špecifikácia je popísaná v kap. 10.2.

- parseSummeryStatistics(summaryStatistics:String, playerId: String):void
    - summaryStatistics – reťazec vytvorený z premenných zaznamenavajúcich aktivitu hráča
    - playerId - id aktuálneho hráča, pre ktorého sa štatistika generuje

Funkcia na základe id hráča nájde správny súbor, do ktorého zapíše dáta v prípade, že zodpovedajú príslušnému regulárnemu výrazu. Do csv súboru sa dáta vkladajú následovne:

```
csv << [level.gsub!("level", ""), shoots, numberOfTargets,
targetsDestroyed, numberOfDistractors, distractorsDestroyed,
GameUtils.timeStamp]
```

14. Implementácia webovej aplikácie

Architektúra webovej aplikácie vychádza z návrhu (kap. 11), podľa ktorého je rozdelená na serverovú a klientskú.

14.1. Serverová aplikácia

Serverová aplikácia je vytvorená pomocou technológie node.js s využitím swagger dokumentácie. Aplikácie poskytuje prostredníctvom volaní dáta. Aplikácia bude dostupná na URL, ktorá v čase písania tejto práce ešte nie je známa. Zoznam volaní:

- <URL_API>/v1/scoreboard – vráti pole hráčov s príslušným skóre
- <URL_API>/v1/player/{playerId} – vráti detail hráča spolu s detailom každej úrovne

Kompletná dokumentácia k aplikácií bude dostupná po publikovaní na adrese <URL_API>/v1/docs.

Na to aby aplikácia preposielala požadované dáta je nutné nahrať aktuálny priečinok z balíku hry ´./players´ v neporušenej forme. Aplikácia následne z týchto súborov vyberá a transformuje dáta, ktoré ďalej poskytuje. Predokladá sa, že každý hráč bude mať rovnakom počte sedení, rovnaký počet odohratých úrovní(levelov).

14.2. Klientská aplikácia

Klientská aplikácia je vytvorená pomocou technológie react. Skladá sa z dvoch obrazoviek, ktoré fungujú pod samostatnými URL adresami.

14.2.1. Tabuľka hráčov - /scoreBoard

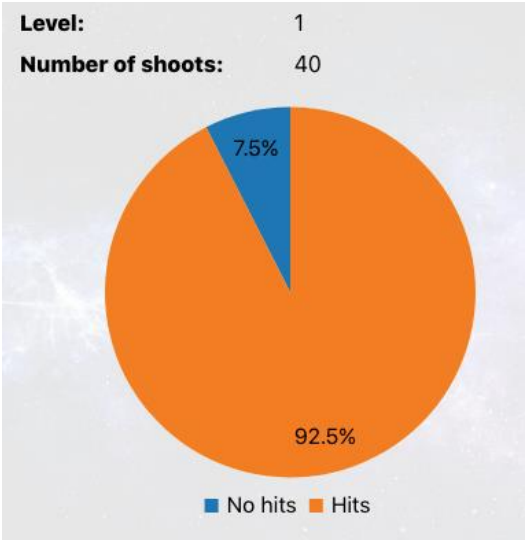
Na tejto ceste sa používateľovi zobrazí tabuľka hráčov. Dáta sa načítajú zo serverovej aplikácie a pretranaformujú do podoby zoznamu, ktorý je zoradený od najvyššieho dosiahnutého skóre po najnižšie. Úspešnosť je taktiež zobrazená šírkou elementu hráča, pričom 100% šírku má hráč so 100% úspešnosťou hrania. Každý hráč má náhodne vygenerovaný obrázok profilu. Vzhľad tabuľky je zobrazený na Obr. 22.

Obr. 22 Tabuľka zobrazujúca skóre hráčov

14.2.2. Detail hráča - /scoreDetail/:id

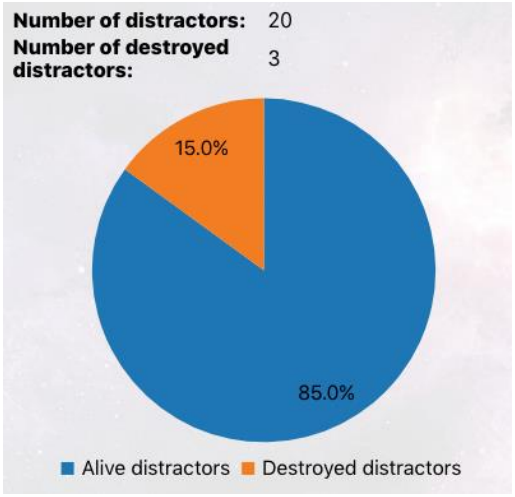
Na tejto adrese sa používateľovi zobrazí detail daného hráča: meno,id, skóre a zoznam jednotlivých odohratých úrovni. Každá odohratá úroveň predstavuje 3 grafy.

- Graf zásahov – pomer všetkých výstrelov a všetkých zostrelov (Obr. 23).



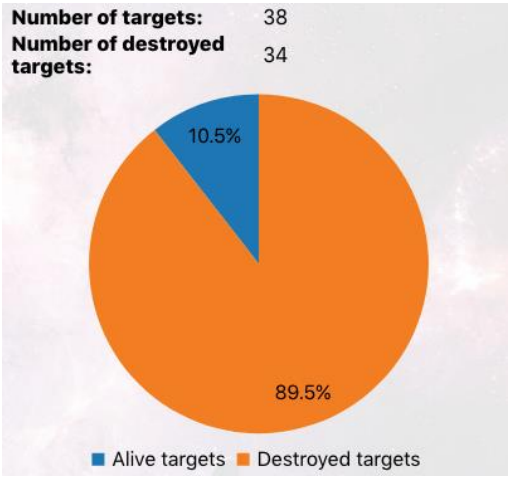
Obr. 23 Graf zásahov

- Graf zosrelených distraktorov – pomer zostrelených a nezosrelených distraktorov(Obr. 24).



Obr. 24 Graf zosrelených distraktorov

- Graf zostrelených targetov – pomer zostrelených a nezostrelených targetov (Obr. 25).



Obr. 25 Graf zosrelených targetov

15. Testovacia fáza experimentu

Výsledná hra je v čase písania práce použitá v rámci testovacieho experimentu, kde sa preukázala funkčnosť celkového riešenia. Spolu s druhou časťou hry vytvorenou Bc. Petrom Vasiľom tak vznikla plne funkčná hra pre experimentálne posúdenie kognitívnych funkcií vo virtuálnej realite. Výsledky tohto experimentu budú známe až po skončení oficiálnej fázy experimentu, avšak tieto výsledky nie sú predmetom tejto práce.

Záver

V rámci prvej časti práce sme sa venovali oboznámeniu sa s danou problematikou, pochopeniu požiadavok experimentu a prípravy krokov k dosiahnutiu hlavného a čiastkových cieľov práce. V analytickej časti práce sú postupne analyzované jednotlivé časti, na ktoré neskôr nadväzuje návrh a implementácia. Práca obsahuje analýzu hry a špecifikáciu experimentu, kde je jasne popísané, čo má daná hra priniesť a aký má mať cieľ spolu s podrobnou špecifikáciou zadanou expertami z oblasti kognitívnej vedy zo SAV a UK. Z tejto analýzy vychádza analýza zastrešujúca konfiguráciu a zber dát so zameraním sa na virtuálnu jaskyňu v laborátoriu LIRKIS. Analýza sa ďalej predovšetkým venuje možnostiam využitia používateľského rozhrania pre účely experimentu vo vizualizačnom nástroji SuperEngine. Analýza poskytuje prijateľné základy pre návrh jednotlivých komponentov, ktoré súvisia s cieľami tejto práce.

V rámci motivácie účastníkov experimentu bola ako súčasť práce vytvorená webová aplikácia pre získanie presnejších výsledkov, ktorá slúži ako nástroj pre motivovanie účastníkov formou súťaže v dosiahnutí čo najvyššieho skóre.

Hlavný prínos vidíme v podpore experimentu, ktorý má za úlohu posúdiť kognitívne funkcie človeka vo virtuálnej realite. Vďaka tejto práci bude možné uskutočniť daný experiment, ktorý prinesie expertom výsledky. Vedľajším prínosom práce je zmapovanie a vytvorenie chýbajúcej dokumentácie k časti vizualizačného nástroja SuperEngine – remoteConsol, kde je konkrétne popísané fungovanie jednotlivých elementov.

Hlavným cieľom práce bolo vytvoriť hru určenú pre experimentálne posúdenie kognitívnych funkcií vo virtuálnej realite so zameraním sa na konfiguráciu a zber údajov. Môžeme povedať, že hlavný cieľ a čiastkové ciele sme splnili.

Výsledkom tejto práce je funkčná časť hry, ktorá v spojení s prácou Bc. Petra Vasiľa, riešiteľa druhej časti, vytvára kompletnú hru určenú pre experimentálne účely. Prostredníctvom tejto hry je teda možné vykonať experiment tak, ako je popísaný v špecifikácii. Ako vyplýva z kapitoly 15, experiment prebieha v testovacej fáze, kde sa momentálne upravujú viaceré detaily časti spojenej s meraním EEG účastníkov zo strany expertov zo SAV a UK.

Zoznam použitej literatúry

[1]. KOREČKO, Štefan; HUDÁK, Marián; SOBOTA, Branislav; MARKO, Martin; CIMROVÁ, Barbora; FARKAŠ, Igor; ROSIPAL, Roman. *Assessment and training of visuospatial cognitive functions in virtual reality: proposal and perspective.* 9th IEEE International Conference on Cognitive Infocommunications CogInfoCom 2018: PROCEEDINGS: 22. – 24.8.2018: Budapest P. 39 – 43 Danvers: IEEE, 2018.

[2]. BURDEA, Grigore C.; COIFFET, Philippe. *Virtual reality technology*. John Wiley & Sons, 2003.

[3]. SCHULTHEIS, Maria T.; RIZZO, Albert A. The application of virtual reality technology in rehabilitation. *Rehabilitation psychology*, 2001, 46.3: 296.

[4]. YOUNGBLUT, Christine. *Educational Uses of Virtual Reality Technology*. INSTITUTE FOR DEFENSE ANALYSES ALEXANDRIA VA, 1998.

[5]. GAO, Zaifeng, et al. Contralateral delay activity tracks object identity information in visual short term memory. *Brain research*, 2011, 1406: 30-42.

[6]. BOWMAN, Doug A.; MCMAHAN, Ryan P. Virtual reality: how much immersion is enough?. *Computer*, 2007, 40.7.

[7]. SOBOTA, Branislav; HROZEK František. *Systémy virtuálnej reality.* 2015.

[8]. CRUZ-NEIRA, Carolina, et al. Scientists in wonderland: A report on visualization applications in the CAVE virtual reality environment. In: *Virtual Reality, 1993. Proceedings., IEEE 1993 Symposium on Research Frontiers in*. IEEE, 1993. p. 59-66.

[9]. BROOKS, Frederick P. What's real about virtual reality?. *IEEE Computer graphics and applications*, 1999, 19.6: 16-27.

[10].M. Hudák, Š. Korečko and B. Sobota, "On architecture and performance of LIRKIS CAVE system," *2017 8th IEEE International Conference on Cognitive Infocommunications (CogInfoCom)*,Debrecen,2017,pp.000295-000300.doi:10.1109/CogInfoCom.2017.8268260

[11].Kelly Jo Hubers, Elizabeth R. Graf, and Sherri B. Lantinga, "The Influence of Meta-Experimental Factors on Compliance and Attitudes: Participant Motivation and Experimenter Demeanor“, AMERICAN JOURNAL OF UNDERGRADUATE RESEARCH ,2003,

[12]. Dominik Trojčák, Hra pre SSC SuperEngine s využitím EEG snímača, Bakalárska Práca, 2017, 10.4: 22-23

[13]. SLOVAKIA SUPER COMPUTERS, http://www.supercomputers.sk/sk/superengine, 2013

[14]. D. Kravitz, K. Saleem, C. Baker, and M. Mishkin, “A new neural framework for visuospatial processing,” Nature Reviews Neuroscience, vol. 12, no. 4, pp. 217–230, 2011.

[15]. N. de Bruin, D. Bryant, J. MacLean, and C. Gonzalez, “Assessing visuospatial abilities in healthy aging: A novel visuomotor task,” Frontiers in Aging Neuroscience, vol. 8, pp. 1–9, 2016.

[16]. Baddeley, “Working memory: Theories, models, and controversies,” Annual Review of Psychology, vol. 63, no. 1, pp. 1–29, 2012.

[17]. R. Shepard and J. Metzler, “Mental rotation of three-dimensional objects,” Science, vol. 171, no. 3972, pp. 701–703, 1971.

[18]. R. Polana, M. Nitsche, C. Korman, G. Batsikadze, and W. Paulus, “The importance of timing in segregated theta phase-coupling for cognitiveperformance,” Current Biology, vol. 22, no. 14, pp. 1314–1318, 2012.

[19]. P. Toril, J. Reales, J. Mayas, and S. Ballesteros, “Video game training enhances visuospatial working memory and episodic memory in older adults,” Frontiers in Human Neuroscience, vol. 10, pp. 1–14, 2016.

[20]. Barman, A. Chatterjee, and R. Bhide, “Cognitive impairment and rehabilitation strategies after traumatic brain injury,” Indian Journal of Psychological Medicine, vol. 38, no. 3, pp. 172–181, 2016.

[21]. N. Dijkstra, P. Zeidman, S. Ondobaka, M. V. Gerven, and K. Friston, “Distinct top-down and bottom-up brain connectivity during visual perception and imagery,” Scientific Reports, vol. 7, no. 1, pp. 1–9, 2017.

[22]. Z. Shipstead, T. Harrison, and R. Engle, “Working memory capacity and visual attention: Top-down and bottom-up guidance,” Quarterly Journal of Experimental Psychology, vol. 65, no. 3, pp. 401–407, 2012.

[23]. W. Paulus, “Transcranial electrical stimulation (tes - tdcs; trns, tacs) methods,” Neuropsychological Rehabilitation, vol. 21, no. 5, pp. 602–617, 2011.

[24]. M. I. Posner, M. K. Rothbart, and Y. Y. Tang, “Enhancing attention through training,” Current Opinion in Behavioral Sciences, vol. 4, pp.1–5, 2015.

[25]. C. Meneghetti, E. Borella, and F. Pazzaglia, “Mental rotation training: transfer and maintenance effects on spatial abilities,” Psychological Research, vol. 80, no. 1, pp. 113–127, 2016.

[26]. C. H. Li, X. He, Y. J. Wang, Z. Hu, and C. Y. Guo, “Visual working memory capacity can be increased by training on distractor filtering efficiency,” Frontiers in Psychology, vol. 8, no. FEB, pp. 1–12, 2017.

[27]. Neubauer, S. Bergner, and M. Schatz, “Two- vs. three-dimensional presentation of mental rotation tasks: Sex differences and effects of training on performance and brain activation,” Intelligence, vol. 38, no. 5, pp. 529–539, 2010.

[28]. M. T. Schultheis, J. Himelstein, and A. A. Rizzo, “Virtual reality and neuropsychology: upgrading the current tools,” The Journal of head trauma rehabilitation, vol. 17, no. 5, pp. 378–394, 2002.

[29]. R. J. Matheis, M. T. Schultheis, L. A. Tiersky, J. DeLuca, S. R. Millis, and A. Rizzo, “Is learning and memory different in a virtual environment?” The Clinical Neuropsychologist, vol. 21, no. 1, pp. 146–161, 2007.

[30]. T. D. Parsons, A. R. Carlew, J. Magtoto, and K. Stonecipher, “The potential of function-led virtual environments for ecologically valid measures of executive function in experimental and clinical neuropsychology,” Neuropsychological rehabilitation, vol. 27, no. 5, pp. 777–807, 2017.

[31]. M. Hudak, v. Kore ´ cko, and S. Branislav, “On architecture and perfor- ˇ mance of lirkis cave system,” in 8th IEEE International Conference on Cognitive Infocommunications (CogInfoCom), 2017, pp. 295–300. ThalmicLabs. (2018) Myo armband homepage, https://www.myo.com. [Online]. Available: https://www.myo.com NaturalPoint. (2018) Optitrack homepage, https://optitrack.com. [Online]. Available: https://optitrack.com

[32]. M. Hudak, Š. Korečko, and B. Sobota, “Special input devices integration to lirkis cave,” Open Computer Science, vol. 8, no. 1, pp. 1–9, 2018.

[33]. E. Vogel and M. Machizawa, “Neural activity predicts individual differences in visual working memory capacity,” Nature, vol. 428, no. 6984, pp. 748–751, 2004.

[34]. S. Luck, An introduction to the Event-related Potential Technique. London: The MIT Press, 2014.

[35]. R. Luria, H. Balaban, E. Awh, and E. Vogel, “The contralateral delay activity as a neural measure of visual working memory,” Neuroscience and Biobehavioral Reviews, vol. 62, pp. 100–108, 2016.

[36]. P. Baranyi and A. Csap ´ o, “Definition and synergies of cognitive ´ infocommunications,” Acta Polytechnica Hungarica, vol. 9, no. 1, pp. 67–83, 2012.

[37]. Horvath and A. Sudar, “Factors contributing to the enhanced performance of the maxwhere 3d vr platform in the distribution of digital information,” Acta Polytechnica Hungarica, vol. 15, no. 3, pp. 149–173, 2018.

[38]. V. Kovecses-Gosi, “Cooperative learning in vr environment,” ¨ Acta Polytechnica Hungarica, vol. 15, no. 3, pp. 205–224, 2018.

Prílohy

Príloha A CD médium –záverečná práca v elektronickej podobe, prílohy v elektronickej podobe, vizualizačný nástroj spolu s spustiteľným programom a zdrojové kódy aplikácie.

Príloha B Používateľská príručka

Príloha C Systémová príručka

TECHNICKÁ UNIVERZITA V KOŠICIACH
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Používateľská príručka
Príloha B k diplomovej práci

2019

Bc. Dominik Trojčák

Obsah

Zoznam obrázkov

Zoznam tabuliek

1. Program a jeho funkcia

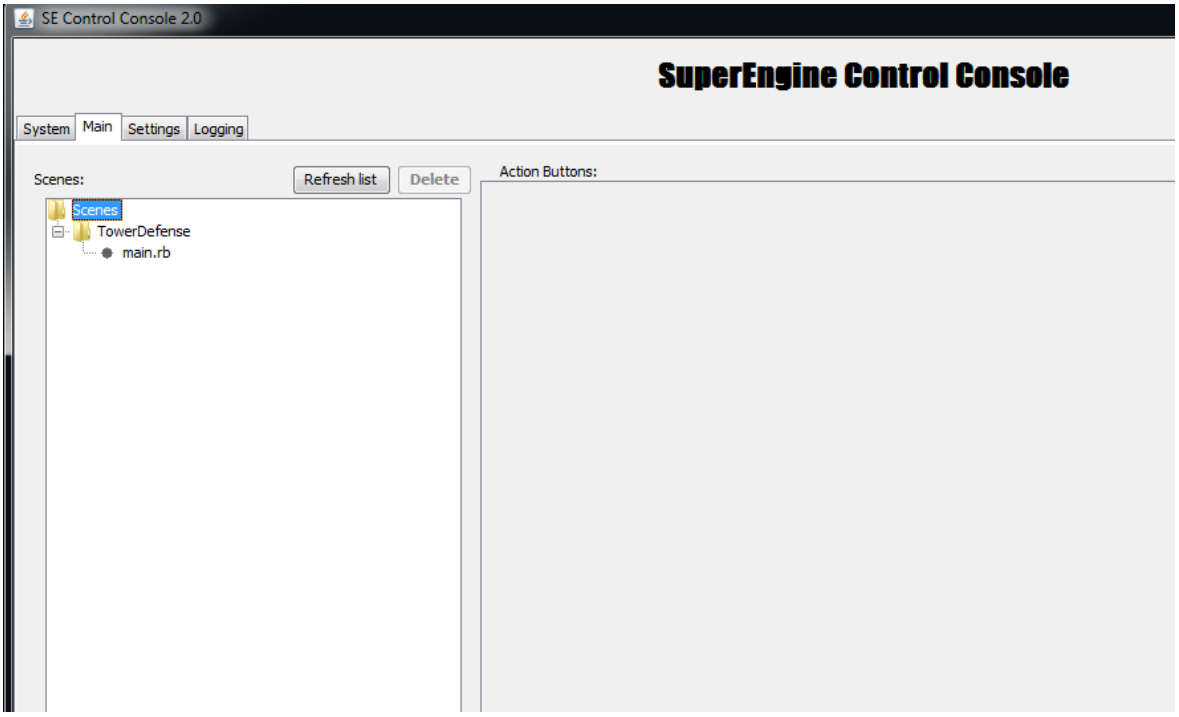
Hlavnou úlohou systému je simulácia hry. Táto hra sa odohráva vo virtuálnom priestore reprezentovanom vesmírom, kde hráč sa nachádza na otočnej plošine (veži) a pomocou zbraní sa snaží zničiť nepriateľské objekty (ciele). Používateľ je v tomto prípade človek, ktorý sa nachádza v prostredí virtuálno-realitnej jaskyne a zabezpečuje chod hry a jej nastavenie. V tejto príručke je teda definovaný postup a spôsob, akým používateľ zabezpečí, aby hra bola spustená korektne a ako výstup produkovala požadované dáta. Taktiež je v tejto príručke popísaný návod k samotnej hre.

2. Inštalácia

Hra je spustiteľná pomocou SuperEnginu, ktorý musí byť nainštalovaný podľa príručky k samotnému nástroju. Na spustenie celého systému je potrebné mať na lokálnom disku umiestnenú celú zložku Slovakia Supercomputers a nainštalované potrebné softvérové vybavenie (Java Runtime Enviroment 8, 3DxSoftware_v3-8-1, ImageMagick-6.3.7-8-Q8-windows-dll, LAVFilters-0.55.3, vcredist_x86).

2.1. Inštalácia a spustenie hry

Inštalácia samotnej hry pozostáva zo skopírovania adresára „TowerDefense“ do adresára SuperEnginu („Slovakia Supercomputers/SuperEngine/Packages/“). Po prekopírovaní a spustení systému sa adresár zobrazí priamo v konzole, ako je možné vidieť na obrázku 1 nižšie. Hra je dostupná v dvoch verziách, ktoré sú rozdelené samostatne. Jedna verzia je určená pre CAVE a druhá pre prezentačné účely. Verzia 2 je prispôsobená pre bežný počítač. V oboch prípadoch adresár obsahuje spustiteľnú verziu v podobe skriptu „main.rb“, ako je znázornené na obrázku č.1.



Obr. 1 Zobrazenie zložky hry v konzole

3. Konfigurácia levelov

Vytvorenie a konfigurácia levelov je nevyhnutná pre obe verzie. Štandardne má hra preddefinovaných 30 levelov, pričom je možné ich pridávať, odoberať a modifikovať v stanovenom formáte. Levely je možné konfigurovať v zložke hry („/levels/levels.csv“). Po zmene levelov a opätovnom načítaní scény sa zmena v používateľskom prostredí prejaví tak, že používateľ bude môcť vyberať z aktuálne nakonfigurovaných levelov.

Základná konfigurácia levelov:

Level	Episode time	Swarm angle	# Targets	# Distractors
1	Long	Easy	2	2
2	Long	Easy	2	3
3	Long	Easy	3	3
4	Long	Medium	3	3
5	Long	Medium	3	4
6	Long	Medium	4	3
7	Long	Medium	4	4
8	Long	Hard	4	4
9	Long	Hard	4	5
10	Long	Hard	5	4
11	Middle	Easy	3	3
12	Middle	Easy	3	4
13	Middle	Easy	4	4
14	Middle	Medium	4	5
15	Middle	Medium	5	4
16	Middle	Medium	5	5
17	Middle	Hard	4	5
18	Middle	Hard	5	5
19	Middle	Hard	5	6
20	Middle	Hard	5	7
21	Short	Easy	3	4
22	Short	Easy	4	4
23	Short	Medium	4	5
24	Short	Medium	5	4
25	Short	Medium	5	5
26	Short	Hard	4	5
27	Short	Hard	5	5
28	Short	Hard	5	6
29	Short	Hard	5	7
30	Short	Hard	6	7

Tab. 1 Základná konfigurácia levelov

4. Konfigurácia hráčov

Pri plnej verzii hry, ktorá sa spúšta v laboratóriu LIRKIS, je potrebné nadefinovať hráčov, ktorí sa zúčastňujú experimentu. Podobne ako pri leveloch je potrebné týchto hráčov vytvoriť v csv súbore, ktorý sa nachádza v adresári hry („/players/players.csv“). Používateľ môže hráčov pridávať, odoberať a editovať. Neodporúča sa zmena ID hráčov počas experimentu, pretože na základe ID sa následne každému hráčovi prideľuje skóre. Po zmene hráčov sa táto zmena v používateľskom prostredí prejaví tak, že bude možné daných hráčov vybrať zo zoznamu, ktorý sa zobrazuje pomocou číselníka.

Príklad vytvorených hráčov v csv súbore players.csv:

Meno	**Priezvisko**	**Id**
Dominik	Trojcak	1
Peter	Vasil	2

Tab. 2 Základná konfigurácia hráčov

5. Spustenie hry

5.1. Postup spustenia hry

5.1.1. Výber hráča

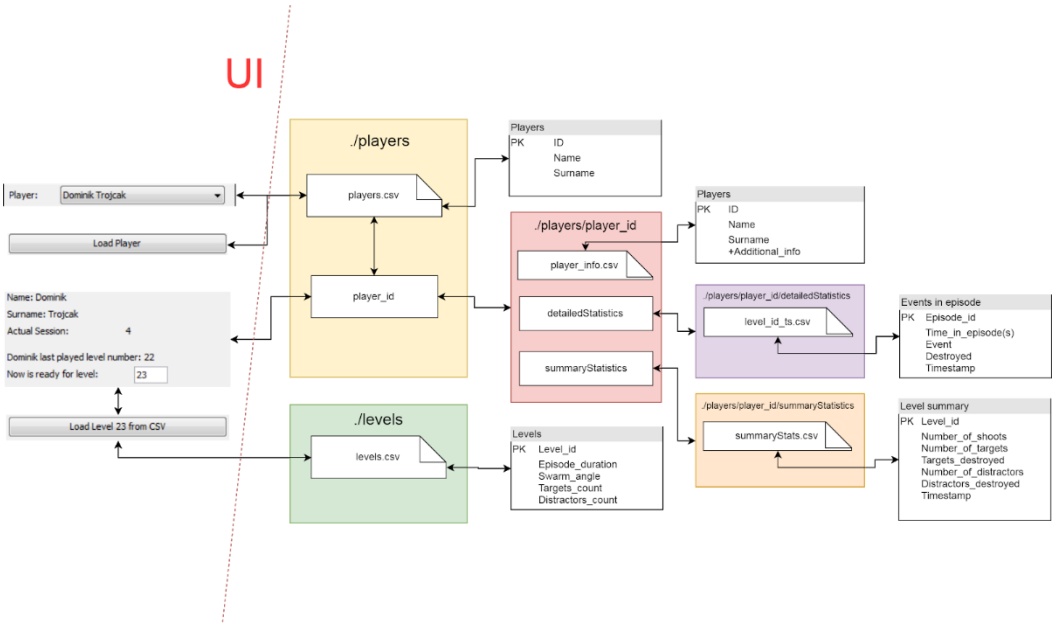
- Výber hráča pomocou číselníka v sekcii “Player info”.
- Po zvolení hráča je potrebné stlačiť tlačidlo “Load Player”, čím sa inicializuje hráčov profil. Zobrazia sa informácie o aktuálnom sedení a o poslednej absolvovanej úrovni. V prípade, že daný hráč hrá hru prvýkrát, na pozadí systém dotvorí všetky potrebné súbory.

5.1.2. Načítanie levelu

- Používateľské rozhranie umožnuje spustiť ľubovoľný level v rozsahu 1 až *n*, kde *n* je celkový počet levelov definovaný v csv súbore levels.csv.
- Po zvolení čísla levelu sa level inicializuje stlačním tlačidla “Load level x from CSV”
- Systém načíta dáta z csv a preklopí ich do časti “Level setup”, pričom dáta z csv nie je možné ďalej upravovať. Čas levelu je stanovený na 8 minút. Podľa tohto času systém vyráta počet epizód na základe dĺžky epizódy a dĺžky prestávky medzi epizódami.

5.1.3. Spustenie levelu

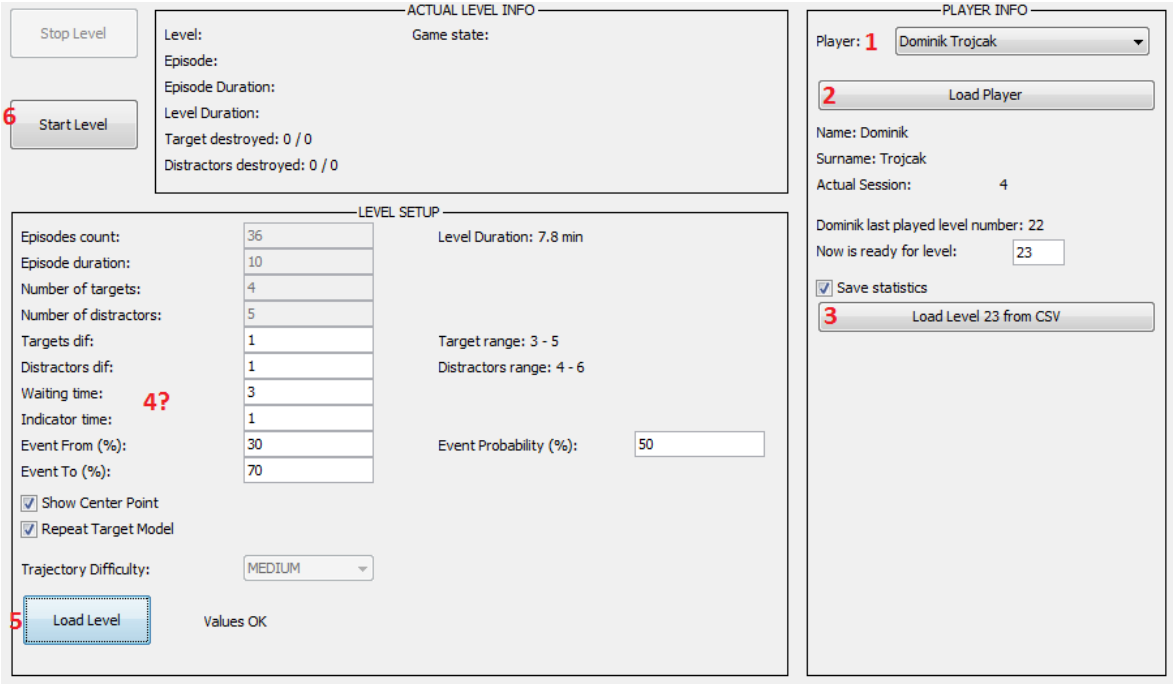
- Pomocou tlačidla “Load level” v časti “Level setup” sa nastaví celková konfigurácia v hre.
- Následne sa sprístupní tlačidlo “Start level”, ktorým sa spustí daný level.



Obr. 2 Schéma výberu hráča a levelu

5.1.4. Postup v bodoch (obr. č. 3) - CAVE

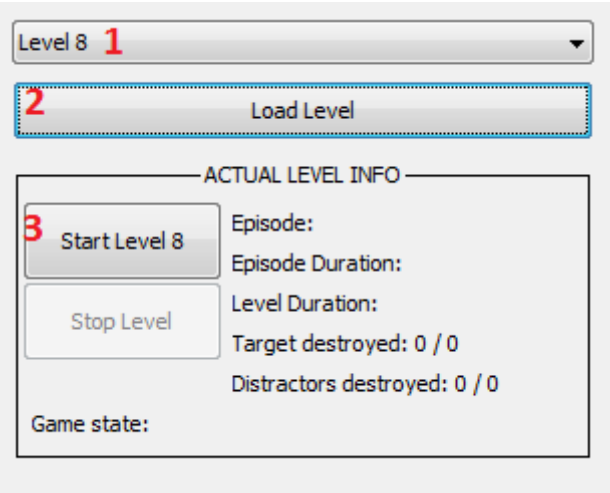
1. Výber hráča z číselníka
2. Načítanie hráča – “Load Player”
3. Načítanie levelu z CSV – “Load Level *x* from CSV”
4. Zmena konfigurácie – nepovinná
5. Načítanie levelu z konfigurácie do hry – “Load Level”
6. Spustenie levelu – “Start Level”



Obr. 3 Postup spustenia levelu - CAVE

5.1.5. Popstup v bodoch (obr. č. 4) - Desktop

1. Výber levelu z číselníka
2. Načítanie zvoleného levelu – “Load Level”
3. Spustenie levelu – “Start Level”

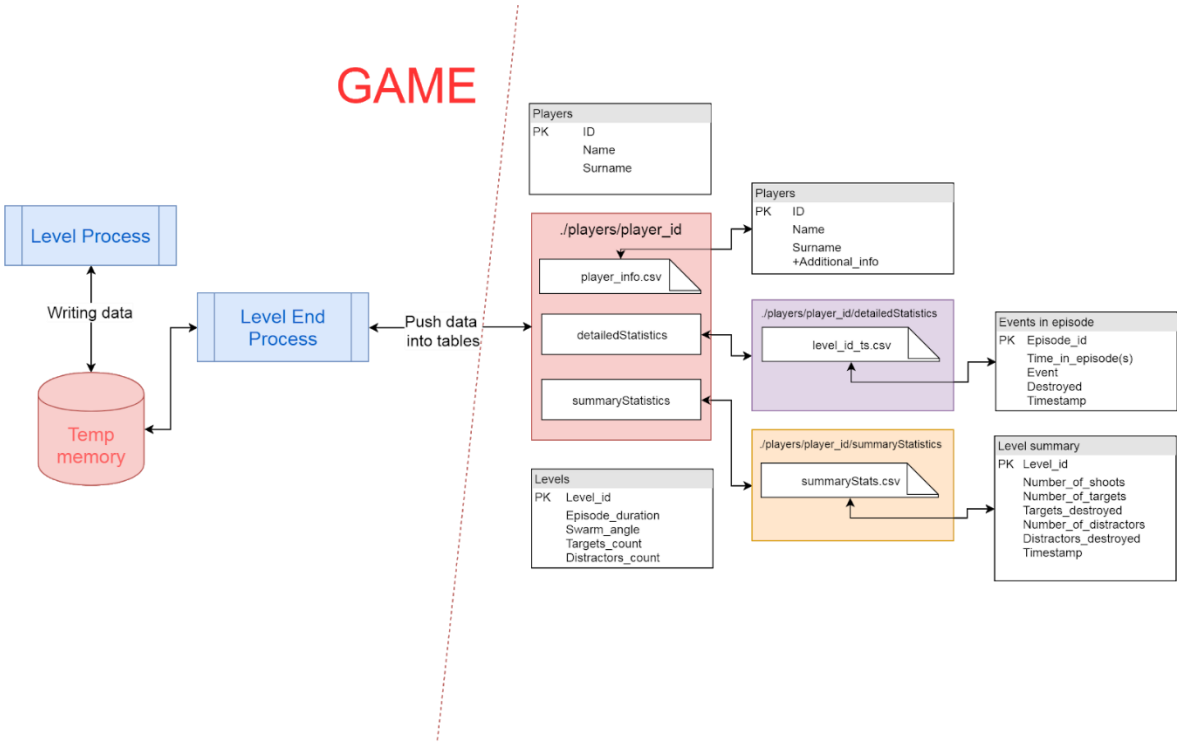


Obr. 4 Postup spustenia levelu - Desktop

6. Zaznamenávanie a ukladanie štatistík

Každému hráčovi sa po každom leveli vygenerujú príslušné štatistiky, pričom platia nasledujúce pravidlá:

- Zaznamenávanie hráčových štatistík prebieha automaticky na pozadí hry.
- Hráčovi sa štatistiky zapíšu do jeho zložky po korektnom ukončení levelu.
- Štatistiky sú rozdelené na sumárne a detailné**.**
- Sumárne štatistiky uchovávajú informácie pre celkové skóre levelu. Jeden level = jeden riadok v tabuľke. Záznam obsahuje informácie, koľkokrát hráč vystrelil, koľko bolo v leveli cieľov, koľko bolo v leveli distraktorov, koľko cieľov zničil, koľko distraktorov zničil a zároveň časovú známku pre každý z týchto údajov.
- Detailné štatistiky predstavujú konkrétne udalosti v danom leveli. Jeden level = jeden csv súbor.
- Csv súbor má v názve číslo levelu a časovú známku.
- Záznam obsahuje číslo epizódy, čas od začiatku epizódy v sekundách, udalosť ktorá nastala, info o zostrelení/nezostrelení objektov a časovú známku.
- Udalosti sú následovné: výstrel, zatmenie, začiatok epizódy.



Obr. 5 Schéma zaznamenávania štatistík

7. Hra

7.1. Ovládanie

Hru je možné ovládať pomocou klávesnice alebo herným ovládačom v závislosti od toho, či je hra spustená v desktopovej verzií alebo vo verzií pre virtuálnu jaskyňu.

7.1.1. Ovládanie – desktopová verzia

Ovládanie dekstopovej verzie je možné pomocou klávesnice:

- Otočenie vpravo – šípka vpravo
- Otočenie vľavo – šípka vľavo
- Otočenie hore – šípka hore
- Otočenie dole – šípka dole
- Streľba – medzerník

7.1.2. Ovládanie – verzia pre virtuálnu realitu

Vo virtuálnej jaskyni je ovládanie možné využitím herného ovládača. Po zapojení ovládača do niektorých z USB portov riadiaceho počítača je potrebné spustiť príslušný program slúžiaci na prepojenie ovládača a SuperEnginu.

Ovládanie (obrázok č. 6):

- Otočenie vpravo – 2
- Otočenie vľavo – 1
- Otočenie hore – 4
- Otočenie dole – 3
- Streľba – 5 – tlačidlo umiestnené na opačnej strane ovládača

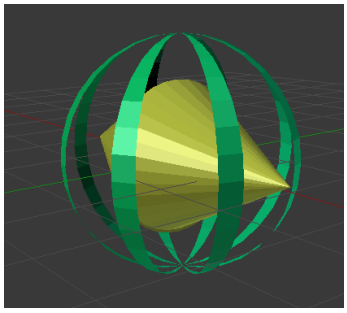
Obr. 6 Ovládanie herným ovládačom (joystick)

7.2. Postup hry

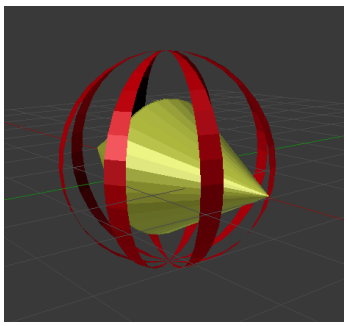
Postup hry je popísaný v nasledujúcich bodoch:

1. **Začiatok náletu**

   V tejto fáze sú hráčovi zobrazené identifikátory, ktoré označujú, či sa jedná o cieľ (červený – obrázok č. 8) alebo distraktor (zelený – obrázok č. 7).



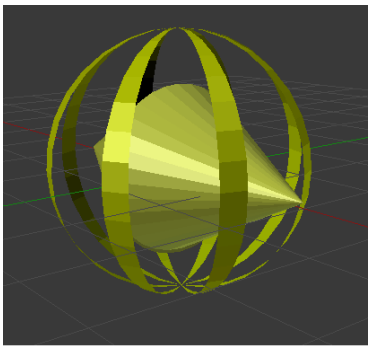
Obr. 7 Identifikátor distraktora



Obr. 8 Identifikátor targetu

2. **Priebeh epizódy**

   Počas epizódy sa hráč snaží zostreliť nepriateľské objekty (ciele) a nezosreliť priateľské (distraktory). Pri zameraní jednej alebo druhej skupiny sa v okolí objektu zobrazí žltý identifikátor (obrázok č. 9). Zameraný objekt je možné zostreliť stlačením príslušného tlačidla na príslušnom ovládači.



Obr. 8 Identifikátor zamerania objektu

3. **Záver epizódy**

   Epizóda končí v okamihu, kedy uplynie jej čas. Čas epizódy je pevný a od neho sa odvíja dĺžka Fjednotlivých rýchlosť dronov.

4. **Čakanie na ďalšiu epizódu**

   Po skončení epizódy hráč čaká na ďalšiu, až kým neskončí daná úroveň (level). V prípade, že nasleduje ďalšia úroveň pokračuje sa bodom č.1.

5. **Ukončenie levelu**

   Po uplynutí poslednej epizódy úroveň (level) končí. V prípade desktopovej verzie sa hráčovi zobrazí skóre a sumárna štatistika priamo na obrazovke. V prípade verzie pre virtuálnu jaskyňu sa skóre zapíše do príslušných csv súborov so štatistikami.

8. Webová aplikácia

Webová aplikácia slúži pre publikáciu výsledkov hráčov. Z používateľského hľadiska je potrebné pre zobrazenie aktuálnych výsedkov nahrať zložku „/players“, ktorá je súčasťou balíka hry, do pripraveného repozitára servera. Odtiaľ si už aplikácia automaticky berie dáta a nie je potrebný žiaden ďalší manuálny zásah.

8.1. Inštalácia

Pre spustenie webovej aplikácie je potrebné mať k dispozícií server, na ktorom bude možné spustiť react.js aplikáciu a node.js server.

Požiadavky na server:

- nainštalovaný webServer – Apache, Nginx - https://www.nginx.com/
- nainštalovaný node - https://nodejs.org/en/
- nainštalované npm - https://www.npmjs.com/

8.2. Spustenie serverovej aplikácie – back end

1. Otvoriť terminál v zložke projektu - Zdrojové kódy\Webová aplikácia\Server
2. Inštalácia balíčkov – npm install
3. Build projektu – npm build
4. Presun node_modules do buildu - mv node_modules ./dist/
5. Synchronizácia adresára rsync -avzc --no-times --no-perms --no-owner --no-group --delete ./dist/ <server>:<doména> /api/api/
6. Reštart servera - sudo systemctl restart – priamo na serveri
7. Test - = <doména API>/v1/docs – zobrazí dokumentáciu

8.3. Spustenie klientskej aplikácie – front end

1. Otvoriť termínal v zložke projektu - Zdrojové kódy\Webová aplikácia\Klient
2. Nastavenie .env premenných
   - v projekte je potrebné nastaviť globálne premenné na správne hodnoty nasledovne:
     - REACT_APP_DOMAIN= <doména API>/v1, lokálne = localhost:9999/v1
3. Inštalácia balíčkov – npm install
4. Build projektu (POZOR – API musí bežať) – npm build
5. Synchronizácia adresára rsync -avzc --no-times --no-perms --no-owner --no-group --delete ./build/ <server>:<doména> /toverDeffense /public/

TECHNICKÁ UNIVERZITA V KOŠICIACH
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Systémová príručka
Príloha C k diplomovej práci

2019 **Bc. Dominik Trojčák**

Obsah

1. Program a jeho funkcia

Hlavnou úlohou systému je simulácia hry. Táto hra sa odohráva vo virtuálnom priestore reprezentovanom vesmírom, kde hráč sa nachádza na otočnej plošine (veži) a pomocou zbraní sa snaži zničiť nepriateľské objekty (ciele). Používateľ je v tomto prípade človek, ktorý sa nachádza v prostredí virtuálno-realitnej jaskyne a zabezpečuje chod hry a jej nastavenie. V tejto príručke sú opisané jednotlivé moduly, ich funkcie a premenné potrebné pre požadovaný chod systému.

2. Program a jeho riešenie

2.1. Zoznam modulov

- **Hlavný modul**
  - main.rb
- **Controllers - ovládače**
  - gamepadController.rb
  - joystickController.rb
  - keyboardController.rb
- **Drones - drony**
  - drone.rb
  - droneDesign.rb
  - droneSpecification.rb
  - trajectories.rb
  - droneModelPool.rb
- **Utils – utility**
  - gameUtils.rb
  - intersectUtils.rb
  - soundUtils.rb
  - statisticsParserUtils.rb
- **Player – hráč**
  - player.rb
- **Ostatné**
  - gameStates.rb
  - levelConfig.rb
  - paricipant.rb

2.2. Popis metód v hlavnom module main.rb

Hlavný skript programu, v ktorom prebieha celý životný cyklus programu.

- **guiInit()**

  - inicializovanie prvkov grafického rozhrania

- **refreshGUI(gameState)**

  - zmena používateľského rozhrania na základe stavu hry „gameState“

- **hasTempCorrectValue**

  - kontrola vstupných hodnôt. Návratové hodnoty false/true.

- **updateLevel()**

  - aktualizuje konfiguráciu úrovne vytvorením novej inštancie objektu LevelConfig

- **loadLevel()**

  - načíta úroveň z csv súboru využitím globálnej premennej $levelToLoad

- **loadPlayer()**

  - načíta hráča z csv súboru využitím globálnej premennej $actualParticipant

- **getParticipants()**

  - načíta zoznam hráčov z players.csv. Návratová hodnota je zoznam objektov „Participants“

- **getParticipantLabels(participants)**

  - vytvorí list hráčov (z participants), ktorý je použiteľný pre číselník. Návratová hodnota je zoznam reťazcov

- **handleInputChange(event)**

  - odchytáva jednotlivé udalosti elementov používateľského rozhrania

- **initLevel()**

  - inicializácia úrovne. Zavolá sa pri stlačení tlačidla „Start Level“

- **updateGameState()**

  - aktualizuje stav hry na základe stanovených podmienok

- **logStateChange(actualState, previousState)**

  - zaznamenáva zmenu stavu previousState na actualState a v prípade zmeny zapíše do konzoly SE

- **ENGINE::OnSceneStart()**

  - herná slučka

- **ENGINE::OnSceneStart()**

  - začiatok scény

- **ENGINE::OnExit()**

  - ukončenie scény

- **episodeRunningAction()**

  - akcia vykonávaná v prípade stavu hry „EpisodeRunning“

- **episodeEndAction()**

  - akcia vykonávaná v prípade stavu hry „EpisodeEnded“

- **episodeWaitingAction()**

  - akcia vykonávaná v prípade stavu hry „WaitingForNextEpisode“

- **episodeFirsLoopAction()**

  - akcia vykonávaná v prípade stavu hry „EpisodeFirstLoop“

- **levelEndAction()**

  - akcia vykonávaná v prípade stavu hry „EpisodeFirstLoop“

- **generateEpisodeData()**

  - vygeneruje dáta pre aktuálnu epizódu

- **cleanGameField()**

  - na konci epizódy vyčistí hracie pole a uvoľní premenné

2.2.1. Opis premenných v hlavnom module main.rb

Opis premenných, ktoré nie sú štandardnou súčasťou skriptu SuperEnginu. Nie sú zahrnuté pomocné premenné. Konštanty majú pridelenú aj hodnotu.

- **$gm** – GUI manažér
- **$osd** – debug foreground
- **$camera** – kamera
- **$cameraStartX** = 1000 – počiatočná poloha kamery X
- **$cameraStartY** = 100 – počiatočná poloha kamery Y
- **$cameraStartZ** = 10 – počiatočná poloha kamery Z
- **$LEVEL_DURATION** = 480 – dĺžka jednej úrovne (levelu)
- **$numberOfDestroyedTargets** – počítadlo zostrelených cieľov
- **$numberOfDestroyedDistractors** – počítadlo zostrelených distraktorov
- **$detailedStatisticsArray** – pole na uloženie štatistík
- **$numberOfEpisodesTemp** – počet epizód
- **$episodeDurationTemp** – trvanie epizódy
- **$numberOfTargetsTemp** – počet cieľov
- **$numberOfTargetsDifferenceTemp** – variabilná zložka počtu cieľov
- **$numberOfDistractorsTemp** – počet distraktorov
- **$numberOfDistractorsDifferenceTemp** – variabilná zložka počtu distraktorov
- **$waitingTimeBetweenEpisodesTemp** – čas medzi epizódami
- **$droneIndicatorDurationTemp** – čas zobrazenia indikátora

- **$eventTimeFromTemp** – začiatok možnosti nastatia udalosti v %
- **$eventTimeToTemp** – koniec nastatia udalosti v %
- **$eventProbabilityTemp** – pravdepodobnosť nastatia udalosti
- **$actualLevel** – aktuálna úroveň
- **$levelToLoad** – úroveň, ktorá sa má načítať
- **$showCenterPointValue** – zobrazený stredový bod = false/true
- **$sameTargetModelIsPossible** – opakovanie sa rovnakého modelu cieľu = false/true
- **$lastTargetModel** – posledný model cieľového objektu
- **$saveStatistics** – indikátor uloženia štatistík = false/true
- **$numberOfTargetsInLevel** – počet cieľov
- **$numberOfDistractorsInLevel** – počet distraktorov
- **$shootCount** – počet výstrelov
- **$actualTrajectoryDifficulty** – obtiažnosť rozptylu trajektórií
- **$playersDirectory** = './players/' – adresár hráčov
- **$levelsDirectory** = './levels/' – adresár úrovní
- **$actualParticipant** – aktuálny hráč
- **$isLevelLoadedFromCSV** – indikátor, či bol hráč načítaný z csv súboru = false/true
- **$participants** – zoznam hráčov

Hlavný skript main.rb obsahuje aj ďalšie premenné týkajúce sa používateľského rozhrania.

2.3. Ovládače

Tento balíček obsahuje moduly, ktoré umožnujú ovládanie hry pomocou rôznych zariadení.

2.3.1. Modul gamepadController.rb

2.3.1.1. Popis metód

- **dataAxisX()** – načíta hodnotu X gamepadu
- **dataAxisY()** – načíta hodnotu Y gamepadu
- **dataAxisZ()** – načíta hodnotu Z gamepadu
- **dataAxisZrot()** – načíta hodnotu rotácie gamepadu
- **button1-8()** – načita stlačenie tlačidla 1-8 (každé samostatne)
- **handleControl**() – vykonáva požadovanú akciu pri zmene stavu tlačidiel

2.3.1.2. Popis premenných

- **$dataAxisX** – uchováva hodnotu X
- **$dataAxisY** – uchováva hodnotu Y

- **$dataAxisZ** – uchováva hodnotu Z
- **$dataAxisZrot** – uchováva hodnotu otočenia
- **$button1-8** – uchováva hodnotu tlačidiel 1-8 (každú samostatne)

2.3.2. Modul joystickController.rb

Identické s gamepadController.rb, s rozdielom, že sa jedná o iný typ zariadenia, pri ktorom sa zohľadnuje aj sila otáčania pomocou premenných **powerX** a **powerY**.

2.3.3. Modul keyboardController.rb

2.3.3.1. Popis metód

- **handleControl()** - vykonáva požadovanú akciu pri zmene stavu tlačidiel

2.3.3.2. Popis premenných

- **$dev_keyboard** – načítaná klávesnica

2.4. Drony

Balíček poskytuje moduly potrebné pre prácu s cieľmi a distraktormi.

2.4.1. Modul drone.rb

Predstavuje objekt drona.

2.4.1.1. Popis metód

- **initialize()** – konštruktor, pomocou ktorého sa inicializuje dron
- **reinitialize()** – reštartovanie vlastností drona
- **update()** – aktualizácia drona
- **initInterpolator(trajectory)** – inicilizácia interpolátora pohybu po stanovenej trajektórii (trajectory)
- **move()** – pohyb, zmena polohy drona na trajektórii
- **destroy()** – zničenie drona
- **revive()** – oživenie drona
- **setActive(value)** – aktivácia/deaktivácia drona podľa hodnoty (value = false/true)
- **droneShouldOccur()** – pomocná funkcia, či by sa mal dron zobraziť
- **selectTrajectory()** – náhodne vyberie dostupnú trajektóriu pre drona
- **reselectTrajectory()** – nanovo vyberie trajektóriu
- **droneIsBehindBorder()** – indikuje, či sa dron nachádza za hranicou kamery
- **showIndicator(show)** – aktivovanie/deaktivovanie indikátora (show = false/true)

2.4.1.2. Popis premenných

- **@type** – typ drona
- **@baseDroneModel** – model drona
- **@droneMarker** – ukazovateľ zameraného drona
- **@targetIndicator** – cieľový indikátor
- **@distractorIndicator** – distraktor indikátor
- **@actualSpeed** – @droneSpecification.speedInit
- **@trajectories** – trajektórie
- **@trajectory** – zvolená trajektória drona
- **@minInterpolatorKey** – minimálny interpolačný klúč
- **@maxInterpolatorKey** – maximálny interpolačný kľuč
- **@actualInterpolatorKey** – aktuálny interpolačný kľuč
- **@active** – aktívny/neaktívny
- **@lastRespawnedTime** – čas posledného respawnu
- **@lastShootOnTowerTime** – čas posledného výstrelu

2.4.2. Modul droneDesign.rb

Modul v sebe nesie modely potrebné pre drony.

2.4.2.1. Popis metód

- **initialize()** – konštruktor, pomocou ktorého sa inicializujú modely

2.4.2.2. Popis premenných

- **@targetIndicator** – model indikátora cieľa
- **@distractorIndicator** – model indikátora distraktora
- **@baseDroneModel** – model drona
- **@droneMarker** – model ukazovateľa zameraného drona

2.4.3. Modul droneModelPool.rb

Modul slúžiaci na uchovanie zoznamu dostupných dronov.

2.4.3.1. Popis metód

- **initialize()** – konštruktor, pomocou ktorého sa inicializuje zoznam
- **getFreeModel()** – vráti dostupný model
- **resetModels()** – reštartuje modely

2.4.3.2. Popis premenných

- **@modelType** – typ modelu
- **@models** – zoznam modelov

2.4.4. Modul trajectories.rb

2.4.4.1. Popis metód

- **initialize()** – konštruktor, pomocou ktorého sa incializujú trajektórie
- **getFreeTrajectory(difficulty)** – vráti pole trajektórií pre danú obtiažnosť (difficulty)
- **resetTrajectories()** – resetuje trajektórie

2.4.4.2. Popis premenných

- **@trajectories** – zoznam vytvorených trajektórií
- **@freeTrajectoriesIndexes** – pole dostupných trajektórií

2.5. Utility

Balíček utils obsahuje sadu pomocných funkcií

2.5.1. Modul gameUtils.rb

2.5.1.1. Popis metód

- **checkEndOfGame()** – kontroluje skončenie hry = false/true
- **elapsedTime()** – vráti čas od začiatku hry
- **elapsedTimeMili()** – vráti čas od začiatku hry v milisekundách
- **timeStamp()** – vráti aktuálnu časovú známku
- **timeStampMili()** – vráti aktuálnu časovú známku v milisekundách
- **selectRandomFromArray(array)** – vyberie náhodný prvok z poľa

2.5.2. Modul intersectUtils.rb

2.5.2.1. Popis metód

- **intersect(lineStartPoint,lineEndPoint, focusedObject)** – vráti informáciu, či je daný objekt (focusedObject) v prieniku s priamkou (lineStartPoint - lineEndPoint) = false/true
- **intersect2D(lx,ly,px,py,omega,factor,error)** – kontrola prieniku v 2D

2.5.3. Modul soundUtils.rb

2.5.3.1. Popis metód

- **initialize()** – incializácia zvukov v hre

2.5.3.2. Popis premenných

- **$bkg_sound_volume** – intenzita zvuku v pozadí
- **$fireSound** – zvuk výstrelu
- **$exploxionSound** – zvuk výbuchu
- **$droneFlySound** – zvuk drona

2.5.4. Modul statisticsParserUtils.rb

2.5.4.1. Popis metód

- **parseDetailedStatistics(statisticsArray, nameOfile, playerId, separator)** – zabezpečí uloženie detailných štatistík hráča (statisticsArray)
- **parseSummaryStatistics(summaryStatistics,playerId,sparator)** – zabezpečí uloženie sumárnych štatistík hráča (summaryStatistics)

2.6. Hráč

Modul predstavujúci triedu hráča. Oproti modulu „participant.rb“, táto trieda obsahuje celkovú funkcionalitu potrebnú pre hranie hry.

2.6.1. Modul player.rb

2.6.1.1. Popis metód

- **initialize()** – koštruktor, pomocou ktorého sa incializuje hráč
- **rotateHorizontal(right,power)** – horizontálna rotácia
- **rotateRight(power)** – rotácia vpravo
- **rotateLeft(power)** – rotácia vľavo
- **rotateUp (power)** – rotácia hore
- **rotateDown(power)** – rotácia dole
- **setLaserVerticalRotation()** – rotácia lasera
- **shoot()** – výstrel hráča
- **update()** – aktulizácia stavu hráča
- **focusingOnDrones()** – informácia o tom, či bol zameraný nejaký dron, false/true
- **updateLaserVisibility()** – zmena viditeľnosti lasera

2.6.1.2. Popis premenných

- **@marker** – model ukazovateľa mierenia
- **@bullet** – model náboja
- **@laser** – model lasera
- **@tower** – model veže

- **@actualTowerShootPower** – aktuálna sila lasera
- **@actualTowerLaserDurationTime** – aktuálny čas zobrazenia lasera
- **@focusedDroneIdx** – index zameraného drona
- **@actualHorizontalAngle** – aktuálny uhol otočenia

2.7. Ostatné

2.7.1. Modul gameStates.rb – enum

2.7.1.1. Hodnoty

- GameStarted = 'GameStarted'
- EpisodeFirstLoop = 'EpisodeEnded '
- EpisodeEnded = 'EpisodeEnded'
- WaitingForNextEpisode = 'WaitingForEpisode'
- EpisodeRunning = 'EpisodeRunning'
- LevelEnded = 'LevelEnded'
- GameEnded = 'GameEnded'
- GameStopped = 'GameStopped'

2.7.2. Modul levelConfig.rb

Modul uchováva informácie o konfigurácií aktuálnej úrovne(levelu).

2.7.2.1. Popis metód

- Initialize() – konštruktor, pomocou ktorého sa inicializuje konfigurácia úrovne
- hasCorrectValues() – kontrola správnosti hodnôt
- printConfig() – vypíše akutálnu konfiguráciu do kozoly SE

2.7.2.2. Popis premenných

- **@numberOfEpisodes** – počet epizód
- **@episodeDuration** – trvanie epizódy
- **@numberOfTargets** – počet cieľov
- **@numberOfDistractors** – počet distraktorov
- **@numberOfTargetsDifference** – variabilná zložka počtu cieľov
- **@numberOfDistractorsDifference** – variabilná zložka počtu distraktorov
- **@waitingTimeBetweenEpisodes** – čas medzi epizódami
- **@droneIndicatorDuration** – dĺžka trvania zobrazenia indikátoru

- **@eventTimeTo** - čas nastatia eventu do, v %
- **@eventTimeFrom** - čas nastatia eventu od, v %
- **@eventProbability** - pravdepodobnosť nastatie eventu, v 5
- **@actualTrajectoryDifficutly** – obtiažnosť trajektórií

2.7.3. Modul participant.rb

Modul reprezentujúci triedu hráča.

2.7.3.1. Popis metód

- **initialize()** – konštruktor, pomocou ktorého sa inicializuje hráč

2.7.3.2. Popis premenných

- **@name** – meno hráča
- **@surname** – priezvisko hráča
- **@id** – id hráča