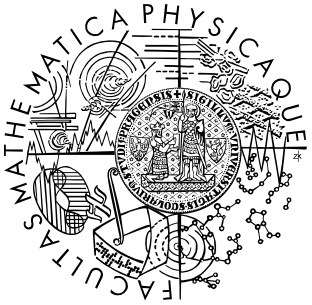
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

DIPLOMOVÁ PRÁCE



Ivana Šupalová

Orientace v prostoru zkoumaná ve virtuální realitě

Katedra softwarového inženýrství

Vedoucí diplomové práce: Mgr. Cyril Brom, Ph.D
Studijní program: Informatika

Na tomto mieste by som sa rada poďakovala vedúcemu diplomovej práce Mgr. Cyrilovi Bromovi, Ph.D, Mgr. Jiřímu Lukavskému,Ph.D z psychologického ústavu AV ČR a Mgr. Kamilovi Vlčkovi, Ph.D z oddelenia neurofyziológie pamäti AV ČR za ich rady, pripomienky a konzultácie. Ďalej ďakujem rodičom za podporu po celú dobu štúdia a všetkým priateľom.

Prehlasujem, že som svoju diplomovú prácu napísala samostatne a výhradne s použitím citovaných prameňov. Súhlasím so zapožičiavaním práce.

V Prahe dňa 15. apríla 2009 Ivana Šupalová

Obsah

Název práce: Orientace v prostoru zkoumaná ve virtuální realitě
Autor: Ivana Šupalová
Katedra: Katedra softwarového inženýrství
Vedoucí diplomové práce: Mgr. Cyril Brom, Ph.D
e-mail vedoucího: brom@ksvi.mff.cuni.cz

Abstrakt:
Na oddelení neurofyziológie pamäti Akadémie Vied Českej Republiky prebiehajú výskumy orientácie ľudí v priestore v experimentálnom reálnom prostredí zvanom Blue Velvet Arena. Táto diplomová práca sa v úvode zaoberá významom uvedených testov pre medicínu a stručným popisom doterajšieho riešenia. Jej hlavým cieľom je previesť toto reálne prostredie do virtuálnej reality, umožniť parametrizáciu experimentov a zaznamenávať výstupné dáta namerané počas ich priebehu.
Výsledný systém využíva prepojenie nástroju Unreal engine, aplikácie v jazyku Java a script enginu. Súčasťou je zakomponovanie prístroju Eyelink II na sledovanie očných pohybov.

Klíčová slova: orientace v prostoru, virtuální realita, unreal engine, oční pohyby

Title: Adaptation of spatial navigation tests to virtual reality
Author: Ivana Šupalová
Department: Department of Software Engineering
Supervisor: Mgr. Cyril Brom, Ph.D
Supervisor's email address: brom@ksvi.mff.cuni.cz

Abstract:
At the Department of Neurophysiology of Memory in the Academy of Sciences in Czech Republic are recently performed tests of spatial navigation of people in experimental real enviroment called Blue Velvet Arena. In introdution of this thesis is described importancy of these tests for medical purposes and the recent solution. The main aim is to adapt this real enviroment to virtual reality, allow it's configuration and enable to collect data retieved during experiment's execution.
Resulting system is using cooperation of Unreal engine, application in Java language and script engine. The work is also enabling to use Eyelink II device for measuring eye movements.

Keywords: spatial navigation, virtual reality, unreal engine, eye tracking

1 Úvod

1.1 Motivácia

Orientácia človeka v priestore sa skúma na zvieracích modeloch (potkanoch, vtákoch a podobne) a získané poznatky sa porovnávajú s chovaním ľudí v experimentálnom prostredí. V Českej Republike sa v súčasnej dobe takéto experimenty (na zvieratách a ľuďoch) vykonávajú vo fyziologickom ústave AV ČR. Jedno konkrétne experimentálne reálne prostredie, nazvané Blue Velvet Arena (BVA), tvorí stan s rotujúcou kruhovou plošinou.
Cieľom tejto práce je previesť toto prostredie do virtuálnej reality. Experimenty vo virtuálnej realite budú slúžiť hlavne k testovaniu rozdielov medzi zdravým človekom a človekom s poškodenými mozgovými štruktúrami zodpovednými za orientáciu v priestore.

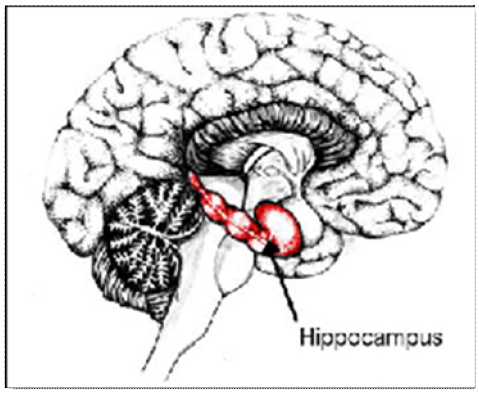
1.2 Štruktúra dokumentu

- **Kapitola 1** obsahuje úvod do problematiky, popis BVA a vymedzuje ciele tejto práce.
- **Kapitola 2** analyzuje návrh architektúry výsledného softwaru.
- **Kapitola 3** preberá možnosti výberu implementačných nástrojov.
- **Kapitola 4** obsahuje implementačné detaily a rieši konkrétne problémy.
- **Kapitola 5** sa venuje ukážkam prototypových experimentov.
- **Kapitola 6** je zhrnutím a zhodnotením výsledného riešenia.

1.3 Kognitívna mapa

Aby bolo možné lepšie pochopiť problematiku orientácie v priestore a jej význam pre medicínu je potrebné ozrejmiť pojem kognitívnej mapy. Kognitívna mapa je reprezentácia prostredia, ktorá vzniká jeho skúmaním. Táto mapa môže byť neskôr používaná v určitých krokoch plánovania cesty, vďaka čomu umožňuje nájsť subjektívne optimálnu trasu. Napomáha tak lepšej navigácii, napríklad pri plánovaní obchádzky alebo hľadaní skratiek.
Vlastníctvo kognitívnej mapy je prisudzované mnohým druhom živočíchov, hlavne cicavcom. Predpokladá sa, že je uložená v časti mozgu nazývanej hipokampus (obr. 1). Táto teória vznikla pri sledovaní mozgovej aktivity potkana v oblasti hipokampu počas špecializovaných

experimentov. Na základe týchto experimentov vznikol predpoklad, že miestom tvorby kognitívnej mapy je pravdepodobne aj u človeka hipokampus [1].



Obr. 1: Umiestnenie hipokampu v mozgu človeka (zdroj: Internet).

1.4 Výskum orientácie v priestore

Výskum orientácie v priestore môže pomôcť pri predikcii Alzheimerovej choroby (ďalej AD). Zlepšovanie skorej diagnózy AD je hlavným cieľom mnohých výskumov. Je to najčastejšia forma demencie a v poradí štvrtá alebo piata najčastejšia možná príčina smrti. Postihuje okolo 5-7% ľudí vo veku 65 rokov a so zvyšujúcim sa vekom jej pôsobenie rapídne rastie, až natoľko, že vo veku 85 rokov ňou trpí vyše 30% populácie [2].

AD je neurodegeneratívna porucha, ktorá zasahuje mozgové štruktúry, hlavne hipokampus a parahipokampálny gyrus [3]. Medzi symptómy jej počiatočného štádia patria poruchy epizodickej pamäti a orientácie v priestore. Pacienti majú problémy s učením sa nových informácií a s udržaním ich v pamäti viac ako pár minút. S postupom ochorenia sa schopnosť učenia stále viac zhoršuje a pacienti strácajú spomienky z minulosti. Poškodené sú tiež ďalšie kognitívne domény, ako úsudok a rozhodovanie.

Návrh experimentálneho prostredia BVA na výskum orientácie v priestore u ľudí vychádza zo zvieracích modelov. Popíšeme dva takéto modely:

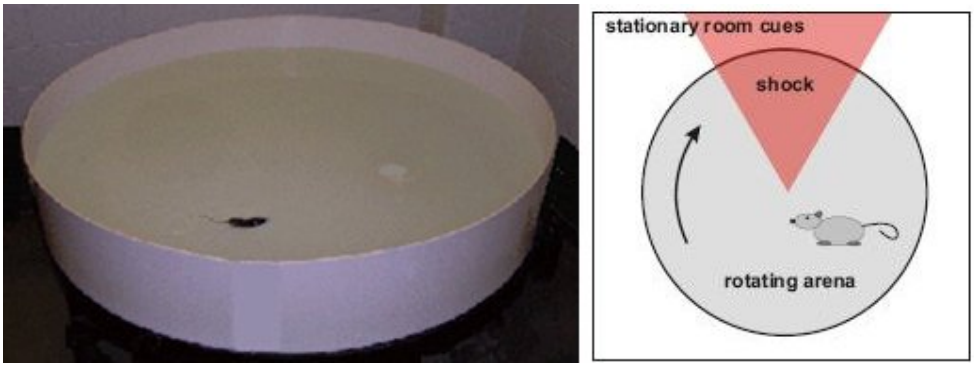
- Morrisove vodné bludisko (obr. 2). Je vytvorené pre potkanov, ktorých úlohou je hľadanie skrytej platformy. Potkan je umiestnený do bazéna s vodou, čo ho

inštinktívne núti nájsť vyvýšené miesto. Platforma je niekoľko centimetrov pod hladinou tak, aby nebola vidieť. Po niekoľkých pokusoch je potkan schopný nájsť plošinu z ktoréhokoľvek miesta v bazéne. Toto bludisko je tiež používané pri testovaní pracovnej pamäte a vytrvalosti.

- Úloha aktívneho vyhýbania sa miestu (obr. 2). Je určená na nájdenie rozdielov v úspešnosti spĺňania úlohy potkanmi pred a po čiastočnej deaktivácii hipokampu. Potkan je umiestnený na pomaly sa otáčajúcu plošinu. Jeho úlohou je vyhýbať sa miestu s miernym šokovým trestom, ktoré je stabilné v miestnosti. Aby sa mu úspešne vyhol, musí byť schopný rozlíšiť medzi statickým priestorom v miestnosti a vyhodnotiť jeho vzťah k otáčajúcej sa plošine. Potkan sa orientuje podľa značiek, ktoré môžu byť na stene (statické) alebo na plošine (pohybujúce sa).

  Tým, že sa plošina otáča, vytvára v miestnosti dva oddelené referenčné rámce: alocentrický a egocentrický. Alocentrický rámec tvorí zložku navigácie, ktorá potkanovi umožňuje zapamätať si miesto so šokom na základe pozície značky. Naopak, egocentrická navigácia značky nevyužíva. Potkan pri snahe vyhnúť sa miestu vníma len informácie, ktoré mu poskytuje vlastné telo.

Výsledky oboch experimentov ukázali, že potkany s jednostranne deaktivovaným hipokampom sa nedokážu naučiť, vyhodnotiť alebo získať z pamäti informácie dostatočné na to, aby boli schopné úlohy úspešne absolvovať[5].



Obr. 2: Morrisovo vodné bludisko a úloha aktívneho vyhýbania sa miestu. Na obrázku vľavo je Morrisovo vodné bludisko (prevzaté z [9]). Bludisko tvorí bazén s vodou a skrytou plošinou. Používa sa na testovanie priestorovej pamäte potkanov. Na obrázku vľavo je schéma úlohy aktívneho vyhýbania sa miestu pre potkany (prevzaté z [4]). Obsahuje otáčajúcu sa plošinu, na ktorej je miesto vysielajúce jemný šok. Potkan sa snaží miestu so šokom vyhnúť.

1.5 Blue Velvet Arena

BVA je kombinovaná verzia Morrisovho vodného bludiska a úlohy aktívneho vyhýbania sa miestu. Aplikuje sa na experimenty s ľuďmi. Vďaka tomu, že ľudia dokážu porozumieť zadaniu úlohy, je možné použiť ju zároveň na hľadanie skrytého cieľa aj na vyhýbanie sa zakázanému miestu. BVA sa používa na oddelení neurofyziológie pamäti na testovanie priestorovej navigácie pacientov v rôznych štádiách Alzheimerovej choroby. Výsledky týchto testov ukazujú, že schopnosť orientácie v priestore oddelenom dvomi referenčnými rámcami je u pacientov zhoršená už od skorého štádia AD. Z toho vyplýva, že podobné testy môžu byť cenné pri predikcii postupného vývoja AD.

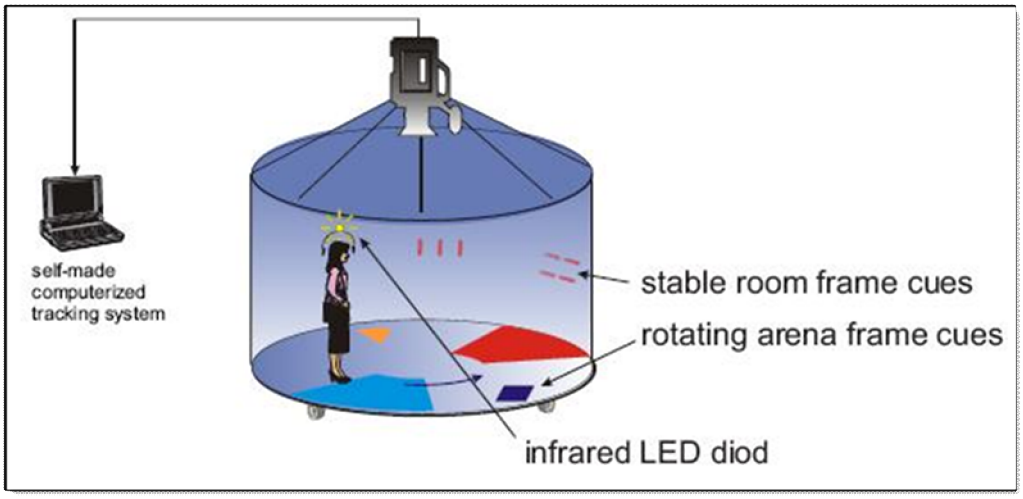
BVA sa využíva pre rôzne typy testov [3]:

- Test skrytého cieľa. Vyšetruje dva módy priestorovej navigácie. Alocentrický, ktorý používa vzdialené orientačné body a egocentrický, ktorý využíva súčasnú polohu subjektu.
- Test pamäti epizodického typu. Meria výkon v neverbálnej časti pamäti epizodického typu, ktorá sa skladá z informácie „kde“ a „kedy“ nejaká udalosť nastala a „čo“ sa stalo.
- Test oddelených referenčných rámcov. Hodnotí priestorovú navigáciu v nezávislých referenčných rámcoch (miestnosť a otáčajúca sa plošina).

1.5.1 Konštrukcia

Konštrukciu BVA (obr. 3) tvorí kruhová aréna s priemerom 2.9 metra, ktorá je obklopená závesom tmavomodrej farby (blue velvet) v tvare cylindra s výškou 2.8 metra. Nad stredom arény je upevnená kamera pripojená k počítačovému systému, ktorý nahráva pozíciu infračervenej LED diódy. Dióda sa nachádza v slúchadlách na hlave testovaného človeka. Okolo kamery je osem laserových ukazovateľov, ktoré umožňujú projekciu do ôsmich oblastí na podlahe arény v 45° intervaloch. Okrem toho aréna obsahuje osem orientačných značiek, ktoré sú rozmiestnené v 45° intervaloch, 1,5 metra nad podlahou arény. Viditeľnosť značiek a oblastí je možné vypnúť alebo zapnúť. Oblasti sú citlivé na vstup človeka.
Na podlahe arény je otáčajúca sa drevená plošina, podporovaná ôsmimi kolieskami, 18 cm nad zemou. Pomocou motora môže byť otáčaná v smere alebo proti smeru hodinových

ručičiek. Jej rýchlosť je plynule regulovateľná. Senzor na okraji arény zaznamenáva aktuálny uhol otočenia plošiny.



Obr. 3: Schéma Blue Velvet Arena (prevzatý z [6]). Obsahuje otočnú plošinu, ktorá je uzavretá v stane. Hore nad jej stredom je umiestnená kamera, ktorá sníma pohyb testovaného človeka. V stane je možné používať oblasti a značky, ktoré sú na stene alebo na podlahe.

1.5.2 Priebeh experimentu

Pri prevode akéhokoľvek reálneho prostredia do virtuálnej reality je dôležité dôkladne poznať spôsob, akým je toto prostredie používané. Preto v tejto časti popíšeme priebeh konkrétneho experimentu, ktorý je v BVA vykonávaný pri testovaní pacientov.

Ide o test na základe oddelených referenčných rámcov. Toto tzv. sedenie s pacientom má 5 častí, aby mal dostatočný priestor na pochopenie úlohy. Jednotlivé časti sú vykonávané v poradí, v akom sú popísané v nasledujúcej časti:

- Prvá úloha: Rozsvietime orientačné značky na stene stanu a vstúpime s testovaným človekom dovnútra. Rozmiestnime značky patriace na podlahu. Červený obdĺžnik na pozíciu 0° a biely trojuholník na pozíciu 135°.

  Pacienta poprosíme, aby sa počas experimentu prechádzal. Nesmie stáť na jednom mieste a nemá sa dlhšiu dobu zdržiavať v strede stanu. Vysvetlíme mu, že počas pokusu bude nosiť slúchadlá a pri vstupe do určitej oblasti mu v nich bude znieť nepríjemný výstražný tón. Úlohou pacienta je snažiť sa vyhnúť zneniu výstražného

tónu po dobu piatich minút. Čím menej krát zaznie výstražný tón, tým lepšie. Ak zaznie, tak je potrebné, aby to bolo na čo najkratšiu dobu. To, kde je zakázané miesto mu neprezradíme. Ukážeme mu, že sa môže orientovať v stane pomocou svetelných značiek na stene, ktoré sú stále na rovnakom mieste alebo značiek na podlahe, ktoré sa spolu s ňou otáčajú. Podlaha sa otáča pomalou rýchlosťou tak, aby sa cítil bezpečne.

- Druhá úloha: Je podobná prvej úlohe, ale pacient sa musí vyhýbať dvom zakázaným miestam súčasne. Jedno miesto zostáva rovnaké, ako predtým. K nemu pribudne ďalšie, pričom jedno stojí na mieste a druhé sa otáča spolu s plošinou.
- Tretia úloha: Opäť sú v stane dve miesta, v ktorých znie výstražný tón. Ich poloha je odlišná, ako v predchádzajúcej úlohe. Pacientovou úlohou je hľadať ich, vstupovať do nich a snažiť sa zistiť, ktoré miesto sa otáča spolu s podlahou, a ktoré nie. Na konci experimentu pacientovi vysvetlíme (pokiaľ na to neprišiel sám), ktoré značky sú k daným miestam najbližšie, aby sa podľa nich mohol orientovať.
- Štvrtá úloha: Pacient využije, čo sa naučil v tretej úlohe, ale miestam sa musí tentoraz vyhýbať.
- Piata úloha: Je rovnaká ako štvrtá, ale plošina sa neotáča.

Popísaný priebeh experimentu nám pomôže predstaviť si, akým spôsobom doktor s pacientom komunikuje, a akou formou sú testy vykonávané. Táto predstava je užitočná pri návrhu konfigurácie experimentov a rozhrania na ich ovládanie.

1.6 Eyelink

Súčasťou práce je umožniť sledovanie očných pohybov pacienta počas experimentu. Nazbierané údaje pomôžu pri spätnej rekonštrukcii experimentu a pri jeho vyhodnocovaní. Je z nich možné napríklad zistiť, kedy pacient využíval orientačné značky, na základe ktorých stimulov sa snažil orientovať, v akom poradí ich skúmal a koľko času im venoval.

Na meranie očných pohybov je použitý prístroj Eyelink II, vyrábaný spoločnosťou SR Research Ltd. [37]. Nasledujúca časť sa venuje stručnému popisu merania očných pohybov a prístroja Eyelink II.

Očné pohyby nie sú plynulé. Majú dve základné fázy. Prvá je fáza fixácie, kedy je oko relatívne v pokoji a získava informácie. Druhá je fáza sakád, čo sú veľmi rýchle priame

pohyby z jedného miesta fixácie na druhé. V priebehu sakád je oko „slepé“. K získavaniu informácií dochádza len v dobe fixácie. Jej doba závisí od náročnosti vnímaného materiálu a typu úlohy, ktorú skúmaná osoba plní. Predpokladá sa, že minimálna doba potrebná na získanie vizuálnych informácií je 50 ms, v priemere je to však až 250 – 300 ms [38].

Eyelink II pozostáva z troch miniatúrnych kamier umiestnených na čelenke (obr. 4) testovaného človeka. Dve kamery sledujú očí, tretia optická kamera, ktorá je v strede čelenky deteguje uhol natočenia oka vzhľadom na monitor. Použitie odrazu rohovky s kombináciou sledovania zrenice redukuje prípadné chyby, ktoré môžu vzniknúť posunutím čelenky, chvením svalov alebo vibráciami prostredia.

Prístroj je vhodný tiež pre osoby so zhoršeným zrakom. Je ho možné použiť s dioptrickými okuliarmi aj s kontaktnými šošovkami.

Prístup k súradniciam aktuálnej pozície oka je v *realtime* čase (len s 3 ms oneskorením). Toto bude využité na priame vykresľovanie pohľadu testovanej osoby vo virtuálnom prostredí.

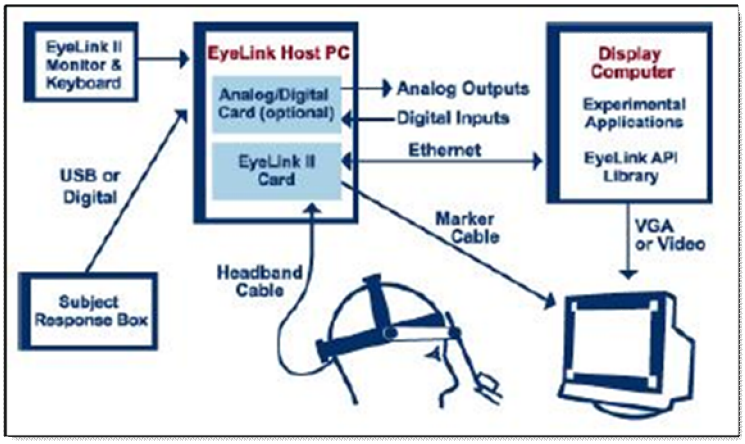
Eyelink II software obsahuje aplikáciu *Host PC Tracker* a rozhranie *C Developers Kit API*. Aplikácia *Host PC Tracker* riadi celý Eyelink II systém, vykonáva analýzu obrazu, spracováva udalosti a nahráva dáta. Umožňuje konfigurovať vlastnosti, nastavenia kamier, vykonávať kalibráciu a korekciu driftu. Kalibrácia slúži na vyhodnotenie presnosti očných pohybov. Počas nej sú testovanému človeku na obrazovke zobrazované body a jeho úlohou je sledovať ich. Korekcia driftu slúži na kontrolu posunu čelenky, na základe ktorej sa upravia súradnice. Kalibráciu je nutné vykonať vždy na začiatku práce s Eyelinkom. Potom stačí už len v určitých časových intervaloch urobiť korekciu driftu. Ak je posun čelenky príliš veľký, je potrebné opäť kalibrovať.

K spusteniu aplikácie *Host PC Tracker* je potrebný samostatný počítač so špeciálne upraveným operačným systémom, na ktorom beží len táto aplikácia.

C Developers Kit API je rozhranie, ktoré dovoľuje ovládať *Host PC Tracker* z iného počítača a sprístupňuje aktuálne dáta o očných pohyboch. Rozhranie je napísané v jazyku C, ale je dostupná aj verzia v jazyku Python (Pylink).

K samotnému experimentu sú potrebné dva počítače. *Host PC* s upraveným operačným systémom a *Display PC*. Na *Display PC* sú testovanému človek prezentované vizuálne podnety. *Display PC* musí obsahovať rozhranie (*C Developers Kit API)* na komunikáciu s *Host PC*. Komunikácia jednotlivých komponentov systému Eyelink je znázornená na schéme na obrázku 5.

Obr. 4: Čelenka systému Eyelink II na sledovanie očných pohybov (prevzaté z [37]). Obsahuje tri kamery na sledovanie pohybu očí. Jej veľkosť je možné prispôsobovať.



Obr. 5: Schéma použitia systému Eyelink II (prevzaté z [38]). Monitor, na ktorom sú testovanému človeku premietané vizuálne podnety je pripojený k *Display Computer*. Ten obsahuje rozhranie na komunikáciu s *Eyelink Host PC*. K *Eyelink Host PC* je pripojená čelenka s tromi kamerami.

1.7 Virtuálna realita

Hlavným cieľom tejto práce je previesť prostredie BVA do virtuálnej reality (VR). Preto je potrebné definovať, čo pod pojmom virtuálna realita rozumieme.

Ľudia vnímajú realitu prostredníctvom piatich zmyslov - zraku, sluchu, hmatu, čuchu a chuti. Pokiaľ je niektorý z týchto zmyslov nahradený umelým zdrojom, vstupujeme viac alebo menej do umelej reality. Pokiaľ je väčšina vstupných (zmyslových) zdrojov umelá, hovoríme o virtuálnej realite. Jedným z charakteristických znakov virtuálnej reality je vytvorenie pocitu (ilúzie), že človek je súčasťou určitého prostredia, a nie len jeho pozorovateľom.

VR je úspešne integrovaná do viacerých aspektov medicíny a psychológie. Umelo vytvorené prostredie sa používajú pri liečbe fóbií (z výťahov, výšok, stiesnených priestorov, ľudí), porúch stravovania, post traumatického stresu a návykov. Tiež je využívaná ako pomôcka na rozptýlenie pacienta pri bolestivej liečbe alebo ako nástroj na trénovanie chirurgických výkonov [11].

Pre túto prácu je dôležité zohľadniť porovnanie navigačných úloh vo VR a v reálnom prostredí. Virtuálne prostredie sa totiž od reálneho líši užším zorným poľom a absenciou skutočného pohybu, ktorý pri navigácii v reálnom prostredí hrá významnú úlohu. VR je založená na báze vizuálnych znakov prostredia bez vestibulárnych informácií (vestibulárny aparát je orgán rovnováhy umiestnený v uchu). Napriek tomuto obmedzeniu bolo dokázané, že kognitívne mapy vytvorené vo VR sú porovnateľné s tými, ktoré vznikajú v reálnom prostredí. Mnohé štúdie navyše potvrdili, že reprezentácie rozsiahlych prostredí naučených vo VR sú v prípade potreby prenášané aj do reálneho prostredia.

Hlavnou príčinou, prečo sa VR stále častejšie používa ako nástroj pre výskum priestorovej navigácie je to, že umožňuje vytvoriť detailný dizajn prostredia a poskytuje podrobný záznam správania sa a pohybu testovaného človeka [7].

1.8 Cieľ práce

V časti 1.5 bolo popísané experimentálne zariadenie BVA. Jeho používanie v praxi je nákladné a skrýva mnoho obmedzení. Testovacie prostredie je príliš malé (2.9 metra), orientačných značiek je len 8, nie je možné sledovať natočenie tela, ani to kam sa testovaný človek pozerá a ktoré orientačné body využíva.

Cieľom tejto práce je previesť prostredie BVA do VR tak, aby zostala zachovaná možnosť pôvodného navrhovania experimentov, ale zároveň aby boli odstránené obmedzenia, ktoré plynú konštrukcie prostredia. Napriek tomu, že vychádzame z konkrétneho modelu BVA je z hľadiska budúceho využitia vhodnejšie vytvoriť prostredie na všeobecné testovanie orientácie v priestore.

Musíme teda vytvoriť virtuálne prostredie, ktoré užívateľ môže konfigurovať z vizuálneho hľadiska a tiež z hľadiska priebehu udalostí. Priebeh experimentu bude možné parametrizovať pomocou jednoduchého skriptovacieho jazyka (napríklad meniť viditeľnosť oblastí, ich aktivitu, viazanosť na plošinu a podobne).

V reálnom prostredí má doktor možnosť sledovať experiment pomocou kamery, ktorá je umiestnená nad stredom stanu. Náhľad na prostredie sa prenáša aj do 2D podoby na obrazovku monitora. Táto možnosť by mala ostať zachovaná aj vo virtuálnom prostredí.
Dôležitou súčasťou je aj zaznamenávanie dát z priebehu experimentu, vrátane očných pohybov a možnosť experiment spätne prehrať.
V pôvodnom riešení BVA sa experimenty spúšťajú z príkazového riadku. Táto práca by mala poskytnúť prívetivejšie užívateľské rozhranie.

2 Analýza návrhu architektúry

Z predchádzajúcej kapitoly vyplýva, že potrebujeme vytvoriť software, ktorý bude umožňovať nasledovné :

- vytváranie grafických prostredí,
- konfiguráciu interakcií užívateľa s objektmi v prostredí,
- zaznamenávanie dát, a
- spoluprácu s prístrojom Eyelink II.

V tejto kapitole preberieme, ako bude software v praxi využívaný, aké možnosti máme pri jeho zostavovaní a zvolíme výslednú architektúru. Pri návrhu budeme vychádzať z nasledujúcich poznatkov:

- Software budú používať minimálne dve osoby zároveň. Jednou osobou bude testovaný subjekt (pacient) a druhou bude človek dohliadajúci na experiment (doktor).
- Software by malo byť možné spustiť na bežnom počítači, a nemal by vyžadovať nadštandardný hardware.
- Do softwaru je potrebné zapojiť prístroj Eyelink II, ktorý funguje na samostatnom počítači so špeciálnym operačným systémom.

Prvým problémom sú teda dvaja užívatelia. Ak by sme navrhli software, ktorý by ako celok bežal na jednom počítači, plynuli by z toho nasledujúce nevýhody. Doktor by nemohol sledovať náhľad na priebeh experimentu a nemohol by doňho zasahovať, pretože počítač by bol v danej chvíli obsadený pacientom. Problém so sledovaním náhľadu by bolo možné odstrániť pripojením jedného monitoru navyše. Takéto riešenie však vyžaduje použitie viacerých grafických aplikácií (samotná aplikácia + náhľady) a komunikácie so systémom Eyelink II na jednom počítači. To nespĺňa požiadavku na bežné technické vybavenie, pretože už samotné grafické aplikácie sú náročné na výkon. Z uvedených dôvodov vyplýva rozhodnutie rozdeliť záťaž na dva alebo viac počítačov. Toto rozhodnutie so sebou prináša nutnosť vytvoriť minimálne dve aplikácie komunikujúce po sieti. Jednou bude grafická

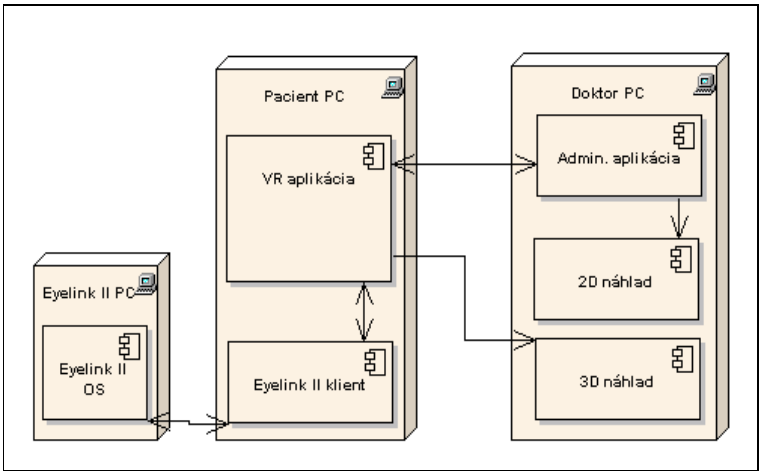
aplikácia a druhou aplikácia na administráciu a na zobrazovanie náhľadov na experiment. Výhodou takéhoto riešenia je, že možnosť použitia jedeného počítača zostáva stále zachovaná.

Ďalej je potrebné zvážiť, kde bude vytvorené užívateľské rozhranie (GUI) aplikácie na administráciu. Jedna možnosť je zaradiť ho do časti grafického nástroja pre 3D vizualizáciu experimentov. Problémom je, že tieto nástroje jeho jednoduchú a rýchlu tvorbu nepodporujú. Väčšinou umožňujú len základnú funkčnosť, a navyše bez prístupu k externým súborom, ktoré sú v prípade zaznamenávania výstupných dát potrebné. Vhodnejšie preto je vytvoriť samostatnú aplikáciu na administráciu experimentov v jazyku, ktorý má podporu vytvárania GUI a práce so súbormi. Tým, že v prechádzajúcom rozhodnutí bola zvolená komunikácia po sieti, je možné vybrať akýkoľvek jazyk, ktorý podporuje sieťovú komunikáciu.

Ďalšou úlohou, ktorá je spomenutá v úvode tejto kapitoly, je umožniť užívateľovi konfiguráciu experimentov vo virtuálnom prostredí. Pri návrhu architektúry je to v zásade otázka, kam zakomponovať spracovanie konfiguračného skriptu. Skriptovací jazyk môže byť použitý buď v grafickom nástroji alebo v aplikácii na administráciu. Z dôvodov, ktoré sú vysvetlené v časti 3.2.1, bola zvolená druhá možnosť.

Posledný problém pri návrhu architektúry je spolupráca s prístrojom Eyelink II. Ten používa špeciálne upravený operačný systém, ktorý musí byť spustený na samostatnom počítači. Opäť teda je využitá komunikácia po sieti. Komunikácia musí prebiehať medzi počítačom, na ktorom budú merané očné pohyby (*Display PC*) a Eyelink počítačom (*Host PC*). Sú dve možnosti. Buď bude grafický nástroj komunikovať s Eyelink počítačom priamo, alebo bude vytvorený klient, ktorý túto komunikáciu sprostredkuje. Vybraná bola druhá možnosť (bližšie v časti 3.3).

Na obrázku 6 na nasledujúcej strane je schéma zvolenej výslednej architektúry. Sú na nej tri počítače. Prvý počítač obsahuje operačný systém pre Eyelink II. Druhý počítač, ktorý obsahuje grafickú aplikáciu a klienta na komunikáciu so systémom Eyelink, bude používaný pacientom. Tretí počítač, používaný doktorom, obsahuje aplikáciu na administráciu a 2D a 3D náhľad na priebeh experimentu. Centrum sieťovej komunikácie je grafická aplikácia (na druhom počítači), ktorá komunikuje s aplikáciou na administráciu, 3D náhľadom a Eyelink klientom. Eyelink klient umožňuje spojenie s Eyelink počítačom.



Obr. 6: Schéma architektúry výsledného softwaru. Počítač vľavo obsahuje operačný systém pre Eyelink II. Počítač v strede, ktorý bude používaný pacientom, obsahuje grafickú aplikáciu a klienta na komunikáciu so systémom Eyelink. Počítač vpravo, ktorý bude používaný doktorom, obsahuje aplikáciu na administráciu, 2D a 3D náhľad na priebeh experimentu. Hlavným strediskom sieťovej komunikácie je grafická aplikácia, ktorá komunikuje s aplikáciou na administráciu, 3D náhľadom a Eyelink klientom. Eyelink klient sprostredkuje spojenie s Eyelink počítačom.

3 Výber jednotlivých nástrojov

Z predchádzajúcej kapitoly vyplýva, že potrebujeme:

- nástroj na zobrazovanie virtuálneho prostredia,
- nástroj na administráciu a konfiguráciu experimentov s 2D a 3D náhľadom na ich priebeh, a
- klienta na sprostredkovanie komunikácie medzi virtuálnym prostredím a prístrojom Eyelink II.

V tejto kapitole preberieme dostupné nástroje a zdôvodníme ich výber. Ako hlavná zložka výsledného systému bol zvolený robustný herný Unreal engine. Engine použijeme ako prostredie na vytvorenie virtuálneho sveta. S ním bude komunikovať aplikácia na administráciu experimentov v jazyku Java. Táto aplikácia bude obsahovať *script engine,* umožňujúci aktívnu komunikáciu s konfiguračným skriptom a Java 2D rozhranie na zobrazovanie 2D náhľadu na experiment. Na vytvorenie komunikačného mostu medzi virtuálnym prostredím a prístrojom Eyelink II použijeme jednoduchého klienta v jazyku C++. Nasledujúce podkapitoly sú venované vyššie uvedeným nástrojom.

3.1 Nástroj pre tvorbu virtuálnej aplikácie

Stojíme pred problémom, ako vybrať grafický nástroj na 3D vizualizáciu prostredia. Táto voľba je kľúčovým krokom pri tvorbe virtuálnej aplikácie, pretože od schopností grafického nástroja sa odvíja ďalší výber, či dokonca samotná potreba ostatných komponentov. Od grafického nástroja požadujeme:

- Možnosť 3D vizualizácie prostredia.
- Pohľad z pozície avatara. Avatar reprezentuje užívateľa vo virtuálnom svete. V istom zmysle slova sa užívateľ s avatarom stotožňuje a pomocou neho jedná. Pohľad z pozície avatara je potrebný, ak chceme dať užívateľovi pocit vlastnej prítomnosti vo virtuálnom prostredí.

- Rozumný pomer medzi grafickým zobrazením a rýchlosťou. Prezentovaná ilúzia virtuálneho sveta musí byť čo najvernejšia. Kritickým bodom je však rýchlosť vykresľovania, najmä ak sa avatar bude pohybovať na otáčajúcej sa plošine. Z toho plynie potreba zobrazovať inú časť scény v takmer každom snímku. Zaujíma nás teda snímková frekvencia, t.j. frekvencia, ktorou zobrazovacie zariadenie vykresľuje jednotlivé snímky. Obvykle sa udáva v jednotkách *fps* (z anglického *frames per second*, čo je počet snímok za sekundu). V počítačovej grafike sa ňou hodnotí napríklad obrazová odozva a plynulosť grafiky. Za dostatočnú hodnotu pre bezproblémový chod grafickej aplikácie sa považuje 60 *fps*.

Uvedené požiadavky na grafický nástroj pre tvorbu virtuálneho sveta možno považovať za minimálne. Bolo by však vhodné, aby mal navyše aj nasledujúce vlastnosti:

- Zabudovaný grafický editor na tvorbu prostredia experimentu. Táto vlastnosť by nebola potrebná v prípade, keby sme sa rozhodli vytvoriť určité, svojím spôsobom dané, prototypové prostredie a následne jeho vlastnosti (textúry, polohu objektov, ich tvar a rozmery) meniť z konfiguračného skriptu. Takéto riešenie však skrýva mnoho obmedzení, konfiguračný skript robí neprehľadným a je veľmi nepohodlné pre nastavovanie polohy objektov alebo celkového priameho náhľadu na vytváranú scénu. Pri grafickom editore je tiež dôležité mať možnosť vkladať komplexnejšie 3D modely, pretože samotné editory na vytváranie scény sa na túto oblasť nešpecializujú a poskytujú len základné geometrické tvary.
- Možnosť sieťovej komunikácie, pretože chceme, aby výsledný software mohol byť rozdelený na viac počítačov.
- Podporu tvorby interiérov. Predpokladané simulované prostredie má byť vo vnútri miestnosti. Tiež nás zaujíma napríklad možnosť realistického osvetlenia.
- Konfigurovatelnosť, v zmysle možnosti meniť dátami „z vonku“ chod grafickej aplikácie.
- Rozšíriteľnosť. V tomto prípade sa myslí možnosť vytvoriť požadované prostredia a konfigurovať ich.
- Podporu nahrávania dema. Je to vlastnosť vítaná, ale nie nevyhnutná.

- Podporu viacerých platforiem. Nakoľko samotný projekt bude vyvíjaný pre platformu Windows, ani táto vlastnosť nie je nevyhnutná, aj keď možnosť prejsť na iný operačný systém je vždy výhodou.
- Podporu skriptovacích jazykov a tvorby grafického užívateľského rozhrania. Aj keď sú to vítané vlastnosti, pretože chceme konfigurovať chod experimentov a mať nad nimi určitý administrátorský prehľad, pri konečnom riešení projektu neboli využité.
- Vysoký počet užívateľov. Predpokladá sa, že čím viac má nástroj užívateľov, tým je stabilnejší.
- Cenovú dostupnosť alebo licenciu.
- Dostatočnú dokumentáciu.

Aké grafické nástroje, priamo určené na vývoj virtuálnych prostredí vyhovujú uvedeným požiadavkám? Komerčné nástroje nevyhovujú, lebo sú príliš nákladné. Do úvahy preto prichádza voľne dostupný jazyk VRML (Virtual Reality Modeling Language), čo je štandardný formát, určený na popis trojrozmerných scén. Je navrhnutý hlavne pre svet Internetu. Užívateľovi dovoľuje komunikovať s objektmi prostredníctvom avatara a poskytuje mu pohľad z pozície avatara. Umožňuje pridanie programového kódu (v jazyku Java alebo JavaScript) do VRML súboru. Okrem toho som už s VRLM pracovala, takže by som ušetrila čas na jeho naštudovanie. Zdá sa, že máme všetko čo potrebujeme. Je ale naozaj VRML dostatočne robustný nástroj? V prvom rade si treba uvedomiť absenciu editoru na vytváranie scény. Od budúceho užívateľa, s veľkou pravdepodobnosťou doktora, nemožno očakávať, že si bude každé testovacie prostredie pracne vytvárať v textovom súbore. Tento nedostatok by bolo možné nahradiť nejakým externým editorom, napríklad *Flux Studio* od *Media Mechines*. Predstavme si však, že chceme vytvárať komplexné scény. Vo všeobecnom zmysle vytvárame software na skúmanie orientácie ľudí v priestore. Priestor môže byť čokoľvek. V prvom rade samozrejme stan s otočnou plošinou a niekoľkými oblasťami. Tento stan vznikol ako model prostredia pre experimenty v reálnom prostredí preto, lebo bolo potrebné vytvoriť uzavreté prostredie, v ktorom je možné merať prejdené vzdialenosti, zaznamenávať vstupy do oblastí a umožniť meniť ich pozíciu. Hlavným podnetom pre vznik myšlienky previesť reálne prostredie do virtuálnej reality bolo odstránenie faktorov, ktoré konfiguráciu prostredia obmedzovali. Predstavme si, že by sme chceli vytvoriť bazén ako interpretáciu Morrisovho vodného bludiska pre potkany prevedenú na ľudí. To by už v reálnom prostredí šlo veľmi

ťažko. Testovať ľudí v pokročilom veku s poruchami mozgových štruktúr v bazéne by bolo vo väčšine prípadov nemožné. Hľadáme preto nástroj, ktorý by nás pri konfigurácii priestoru neobmedzoval a umožnil vytvárať scény, predstavujúce akékoľvek prostredie a zároveň vyvolávajúce v užívateľovi pocit skutočnosti. To už vo VRML nie je jednoduché, pretože vznikol ako jazyk na popis 3D scény a nemá žiadnu vedľajšiu potrebu vtiahnuť človeka do deja. Je to viac-menej „školský" nástroj, na ktorom sa dá dobre demonštrovať vzťah objektov v scéne, ale jeho praktické využitie je v štádiu hľadania (ak si odmyslíme pekné interaktívne webové stránky). Hlavným problémom je však rýchlosť vykresľovania. V prípade trochu zložitejšej scény je VRML pomalé. Pokiaľ vezmeme do úvahy otáčajúcu sa plošinu, nutnosť výpočtov na pozadí kvôli meraniu dát, spoluprácu s Eyelink II a potrebu neobmedziť budúce experimenty limitovanou komplexnosťou scény je VRML pre tento účel nedostatočné. Všetky tieto dôvody boli podnetom k hľadaniu výkonnejšej a robustnejšej alternatívy.

Viacero článkov a textov hovorí v prospech herných enginov oproti virtuálnym nástrojom. Hoci spojenie počítačových hier a serióznej medicínskej aplikácie na prvý pohľad odrádza, postupom času mi táto myšlienka prišla zaujímavá.

3.1.1 Virtuálny nástroj vs. herný engine

Virtuálna realita bola dlhý čas štandardným pojmom pre počítačovú 3D reprezentáciu vo vede a architektúre. Spájala sa s vysokými nákladmi, vyšším stupňom informatického vzdelania a potrebou vlastniť špeciálne počítačové vybavenie. V dnešných časoch sú však herné enginy natoľko vyvinuté, že v mnohých prípadoch predbehnú VR enginy [16]. Hlavným dôvodom tohto obratu je, že obchod s počítačovými hrami sa v snahe udržať si čo najvyšší počet užívateľov, vždy snažil o čo najlepšie vizuálne efekty a grafické umenie. Výrobcovia počítačových hier preto vyvíjali enginy schopné zo štandardne dostupného hardwaru vyťažiť čo najviac.

Pokiaľ určitý projekt vyžaduje precíznu simuláciu, ako napríklad počítačová chirurgia, je špecializované riešenie nevyhnutné. Vo väčšine ostatných prípadov virtuálnych aplikácií však môže použitie herného enginu pridať na hodnote [17].

3.1.2 Výber enginu

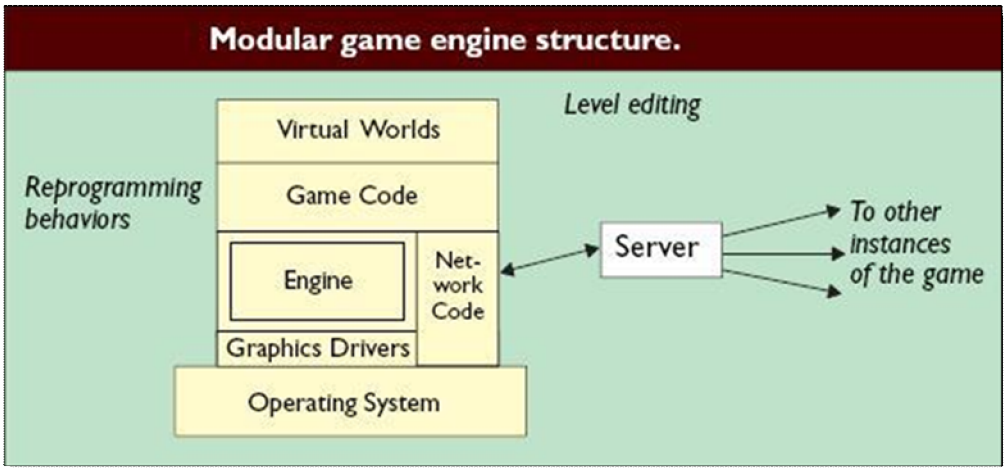
Na súčasnom trhu je značná ponuka rôznych herných enginov. Výber toho správneho je náročná úloha, najmä pre človeka, ktorý nie je v tomto obore odborníkom. V nasledujúcej časti preto najprv stručne zhrniem, čo je to herný engine a aké má základné vlastnosti.
Herný engine je súbor tried, ktoré poskytujú grafické API, sieťové pripojenie, vstup, výstup, zvuk a podobne [17]. Spúšťa hlavný cyklus hry, teda mechanizmus na vykresľovanie 3D rámcov, ktoré obnovuje v priemere 60 krát za sekundu.

Typický engine obsahuje nasledujúce podsystémy:

- grafický modul (2D alebo 3D) na vykresľovanie scény a grafických efektov
- fyzický modul na simulovanie gravitácie a ďalších vlastností sveta
- detekciu kolízií
- vstupno-výstupný modul na prácu so súbormi, spoluprácu s klávesnicou, myšou ...
- zvuk
- umelú inteligenciu, ktorá je najčastejšie využívaná na protihráčov ovládaných systémom
- sieťový modul nevyhnutný pre hry po sieti
- databázový modul
- GUI systém pre vytváranie a prácu s menu a s oknami

Vzťah medzi hrou a *enginom* možno prirovnať ku vzťahu auta a motora. Rovnako, ako je možné zobrať motor a použiť ho v úplne odlišne navrhnutom aute, je možné použiť ten istý herný engine pri vytváraní militaristickej simulácie alebo hry pre deti založenej na rozprávke. Na obrázku 7 je znázornená schéma popisujúca začlenenie enginu do hry alebo grafickej aplikácie.
Spočiatku to môže vyzerať, že plávame v mori možností, ale nie je tomu tak. Mnoho hier nemožno editovať vôbec alebo len veľmi ťažko. Niektoré nemajú základ potrebný pre tvorbu reálne pôsobiacich experimentálnych prostredí a ďalšie zase nemajú dostatočnú dokumentáciu na to, aby sa v nich začínajúci programátor grafických hier zorientoval.
Enginy, ktoré spĺňajú väčšinu z požadovaných vlastností a ich použitie je zdarma, sú napríklad: Crystal space [21], Delta 3D [22], Irrlicht [23], Nebula [27], Unreal 2 [28]

a Quake 3 [29]. V tejto práci uvediem len stručnú tabuľku (tab. 1) s prehľadom ich základných výhod a nevýhod. Bližšie informácie je možné nájsť na stránkach výrobcov.
Okrem herného enginu existuje aj možnosť zvoliť si grafický engine, obsahujúci len modul na vykresľovanie scény a jej správu (vkladanie modelov, textúr, osvetlenie...). K takýmto grafickým enginom patria napríklad OGRE [30], OpenSceneGraph [31] a Blender [32]. Tieto nástroje však nedisponujú 3D editorom, nepodporujú všetky potrebné vlastnosti a väčšinou sú pomalé a preto sa pre náš projekt nehodia.



Obr. 7: Príklad štruktúry začlenenia herného enginu do grafickej aplikácie (zdroj [17]). Na spodnej vrstve je operačný systém spolu s grafickými ovládačmi. Nasleduje samotný engine a kód pre sieťovú komunikáciu. Nad nimi je kód konkrétnej hry s určitou témou, ktorý definuje jej možnosti. Celkom na vrchu tejto pyramídy stojí virtuálny svet, pripravený získať si čo najvyšší počet prívržencov.

Z množstva všeobecne používaných herných enginov najviac vyhovujú naším požiadavkám Unreal a Quake, najmä vďaka svojej profesionalite, obľúbenosti, stabilite a veľkému počtu užívateľov. Vybrať si jeden z nich na základe nejakej kľúčovej vlastnosti v podstate nie je možné. To čo dokáže jeden, dokáže aj druhý. Existujú však aj odlišnosti, ktorým sa bude venovať nasledujúca časť.

	Výhody	Nevýhody
Crystal space	Vyspelý engine s množstvom vyvinutých vlastností	Neobsahuje 3D editor Vyžaduje programovanie na nízkej úrovni enginu
Delta 3D	Kombinuje viacero open-source modulov do game enginu Obsahuje 3D editor	Vyžaduje programovanie na nízkej úrovni enginu Nedostatočná dokumentácia a počet príkladov
Irrlicht	Dobre dokumentované programátorské rozhranie	Neobsahuje 3D editor Nemá podporu zvuku
Panda3D	Jeho design podporuje rýchle učenie a vývoj	Neobsahuje 3D editor
Nebula	Používa sa v komerčných hrách	Neobsahuje 3D editor
Unreal 2	Robustný engine so všetkým potrebným Objektovo orientovaný skriptovací jazyk	Jeho licencia je zdarma len pre nekomerčné a vzdelávacie účely
Quake 3	Robustný engine so všetkým potrebným	Vyžaduje programovanie na nízkej úrovni enginu

Tab. 1: Výhody a nevýhody jednotlivých herných enginov, ktoré sú dostupné zdarma a spĺňajú podmienky kladené na vyspelosť nástroja.

3.1.3 Porovnanie enginov Unreal a Quake

Unreal engine je vyvinutý firmou Epic Games. Quake je od spoločnosti ID Software. Rozdiel medzi nimi je približne na úrovni *„čo komu viac vyhovuje“*. Každý z nich má mnoho nadšených užívateľov, ale aj odporcov. Pre aplikáciu na vedecké účely je síce vhodný jeden aj druhý, existujú však rozdiely, ktoré je možné pri výbere vziať v úvahu.

Hlavný rozdiel z hľadiska užívateľa - programátora, je v spôsobe modifikácie alebo vytvárania novej hry. Quake engine je napísaný v jazyku C a optimalizovaný na rýchlosť. Jeho zdrojové kódy sú plne dostupné a zmeny sú uskutočňované editáciou už vytvoreného kódu.

Unreal na rozdiel od Quake ťaží z výhod novších trendov a technológií. Jeho jadro, ktoré sa stará o kritické operácie náročné na výkon (napríklad vykresľovanie scény, sieťovú komunikáciu, zvuk a ďalšiu funkčnosť v nižšej vrstve) je napísané v jazyku C++. Táto časť kódu sa nazýva natívna a pre programátora nie je viditeľná. Jadro Unreal enginu vytvára virtuálny stroj, podobný *Java virtual machine*, ktorý tvorí programátorské rozhranie

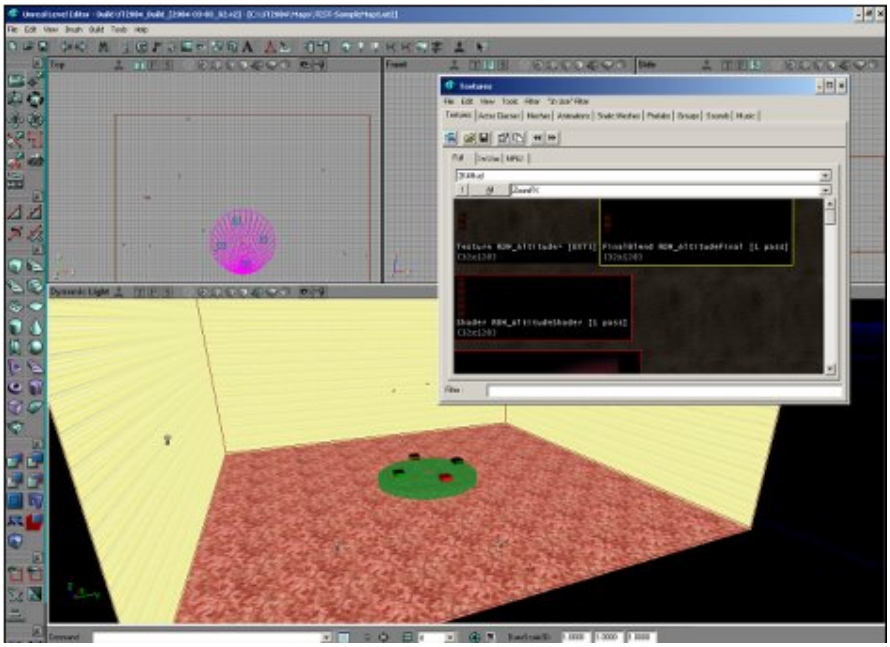
nad natívnym softwarom. V tomto prostredí sa používa objektovo-orientovaný jazyk UnrealScript, podobný Jave. Jeho zdrojový kód je dostupný a umožňuje robiť rozsiahle zmeny v hre, pričom jeho použitie je jednoduchšie. Pristupuje k programovaniu na úrovni objektov a ich interakcií a nie na úrovni bitov a pixlov tak, ako je tomu u Quake enginu. Z toho dôvodu bol pre mňa Unreal engine vhodnejší. Tiež zavážil fakt, že fakulta má zakúpenú licenciu na hru Unreal Tournament 2004 (UT2004), ktorej kód poskytuje mnoho užitočných príkladov.

3.1.4 Základné informácie o Unreal engine

Unreale engine je robustný nástroj, ktorý zahrňuje mnoho vlastností vyvinutých na vysokej úrovni a dotiahnutých až do konca. Zachováva pritom čo najvyššiu možnú rýchlosť vďaka prepracovaným optimalizáciám. Reprezentuje svet pomocou BSP (Binary Space Partitioning) geometrie. Využíva statické mriežky, animované skelety, terény, oblohu, pozadia a podporuje rovnako interiéry aj exteriéry. Umožňuje použiť viac druhov osvetlenia (dynamické, priame, bodové, blikanie, tiene, projektované textúry), ktoré pri rozumnom začlenení do scény výrazne zvyšujú jej realistický zjav. Podporuje simuláciu vody a hmly, ktoré práve pri navigačných úlohách môžu mať zaujímavé využitie.

Jazyk UnrealScript je objektovo-orientovaný a využíva sa na implementáciu funkčnosti jednotlivých objektov, prípadne na rozšírenie alebo zmenu už existujúcich častí hry. Tiež podporuje programovanie sieťového rozhrania.

Hra UT2004 zahrňuje editor máp UnrealEd 3 (obr. 8), ktorý je plne integrovaný do herného enginu a slúži na vytváranie alebo modifikovanie úrovní. Používa sa na umiestnenie objektov, textúr, geometrických tvarov, zvukov a osvetlenia. Môže byť použitý na operačných systémoch Windows, Linux a Mac OS X. V tabuľke 2 sú uvedené základné hardwarové požiadavky potrebné pre použitie Unreal enginu.



Obr. 8: UnrealEd editor. Používa sa na vytváranie máp, umiestňovanie a nastavovanie vlastností objektov. V náhľade je vidieť 3D pohľad na scénu a tri 2D náhľady zhora, zboku a spredu. Ďalej disponuje prehliadačom, v ktorom je možné vybrať textúry, statické mriežky, zvuky a priradiť ich objektom. V editore je takisto možné modelovať základné tvary objektov, prípadne importovať zložitejšie tvary z iného grafického nástroja (napr. Maya alebo 3D Max Studio).

Základné hardwarové požiadavky pre Unreal engine	
CPU	Pentium® III 1.0GHz alebo AMD Athlon 1.0GH (a vyššie)
RAM	256 MB (a viac)
Hard disk	min. 5.5 GB voľného miesta
Grafická karta	32 MB Windows® 98/Me/2000/XP alebo iná kompatibilná video karta (64 MB NVIDIA® GeForce 2, ATI Radeon Hardware T&L alebo DirectX® od verzie 9.0b)

Tab. 2: Základné HW požiadavky pre použitie Unreal enginu.

3.2 Administrácia experimentov a zobrazovanie 2D náhľadu

Ďalšou časťou návrhu architektúry popísaného v kapitole 2 je aplikácia na administráciu experimentov. Táto časť priblíži, prečo bol na jej tvorbu vybraný jazyk Java. Okrem toho sa venuje popisu zobrazovania 2D náhľadu a konfiguračnému skriptu, pretože sú vo výslednom riešení súčasťou aplikácie na administráciu.

V kapitole 2 sme dospeli k rozhodnutiu, že užívateľské rozhranie na administráciu experimentov môže byť naprogramované v akomkoľvek jazyku, ktorý dovoľuje komunikáciu po sieti. Okrem toho by mal umožňovať jednoduchú tvorbu grafického užívateľského rozhrania a prácu so súbormi. To všetko spĺňa jazyk Java. Nie je samozrejme jediný, ale jeho výhodou je, že podporuje grafické 2D rozhranie a skriptovacie jazyky, čo bude využité v tomto projekte.

Možnosti na zobrazovanie 2D náhľadu sú dve. Buď bude spojený s 3D náhľadom, alebo bude súčasťou aplikácie na administráciu. Najprv treba uviesť, že 3D náhľad nemôže byť riešený inak, ako pomocou Unreal enginu vybraného v predchádzajúcej časti. Netreba zdôvodňovať, že iné riešenie 3D náhľadu by zahrňovalo pripojenie ďalšej 3D aplikácie a znamenalo by nevyužitie možností, ktoré Unreal engine poskytuje.

Spomínaná možnosť, spojiť 2D a 3D náhľad, má výhodu komplexného náhľadu na experiment v jednom okne. Nevýhodou tejto možnosti je, že Unreal engine dovoľuje spustiť len jedno grafické okno na jednom počítači. Možnosť 2D náhľadu by teda pri použití jedného počítača bola stratená. Druhá nevýhoda je, že Unreal engine nepodporuje prácu so súbormi. To vedie k problému, že výslednú 2D mapu nie je možné uložiť. A 2D mapa je pritom požadovaným výstupom.

Druhá možnosť je spojiť 2D náhľad s aplikáciou na administráciu. Takéto riešenie odstraňuje obe predchádzajúce nevýhody. Pripomeňme si, že 2D náhľad na experiment má zobrazovať pohyb avatara, tvary oblastí a plošiny, a textové údaje o priebehu. Okrem tvarov oblastí sú to všetko dáta, ktoré sú zaznamenávané do výstupných súborov. Pred uložením je možné ich využiť a vykresliť pomocou nich aktuálny 2D náhľad. Pre vyššie uvedené výhody bola zvolená druhá možnosť. Problém, ktorý z nej plynie je získavanie tvaru oblastí. Ten bude riešený v časti 4.5.1.

3.2.1 Skriptovací jazyk

Súčasťou aplikácie na administráciu je práca s konfiguračným skriptom. Skript má umožňovať definovanie priebehu experimentov vo virtuálnom prostredí.

Pri rozhodovaní, do ktorej časti softwaru bude zaradená konfigurácia experimentov boli dve možnosti. Prvou možnosťou bolo začleniť ho do grafickej aplikácie, druhou do aplikácie na administráciu.

Výhodou prvej možnosti je, že udalosti aj reakcie na ne sú spracovávané priamo v prostredí, v ktorom vznikajú. Konfiguráciu experimentov v grafickej aplikácii je možné riešiť nasledujúcimi spôsobmi:

- Využiť priamo jazyk, v ktorom je aplikácia naprogramovaná – UnrealScript. Výhodou je, že by sa s objektmi v prostredí pracovalo priamo. Nevýhodou je, že každá zmena v kóde aplikácie vyžaduje rekompiláciu. Okrem toho by neexistovalo rozhranie na výber rôznych konfiguračných skriptov.
- Použiť externý skriptovací jazyk. Túto možnosť Unreal engine bohužiaľ nemá.
- Využiť tzv. „triggery“ [33]. Trigger je objekt Unreal enginu, ktorého úlohou je reagovať na udalosti. Konkrétne udalosti, aj reakcie sa nastavujú v UnrealEd editore. Výhodou oproti predchádzajúcej možnosti je, že triggery sú definované pre každú mapu (grafické prostredie) zvlášť. To znamená, že s výberom mapy je zároveň vybraná aj konfigurácia prostredia. Tým by sa vytvorilo jednoduché rozhranie pre výber rôznych konfigurácií. Nevýhodou je, že nie je možné použitie štruktúry „for“ a „while“ cyklu alebo definovanie premenných. Tiež by nebolo možné nastaviť globálne parametre prostredia alebo avatara.

Ani jedno z uvedených riešení nie je pre tento projekt dostatočné.

Druhou možnosťou ako začleniť prácu s konfiguračným skriptom do projektu je využiť aplikáciu na administráciu. Existuje nasledujúci spôsob, ako to docieliť. Jazyk Java, od verzie 6, podporuje použitie skriptovacích jazykov. Pomocou skriptov môže užívateľ rozšíriť alebo parametrizovať funkčnosť aplikácie. Hlavným prvkom je *script engine*, ktorý tvorí štandardné Java rozhranie pre interpretáciu a vyhodnocovanie skriptov. Umožňuje integrovať skriptové súbory ako časť Java programov a naopak, dovoľuje programom pristupovať k premenným v skriptoch.

Java obsahuje zabudovaný *Mozilla Rhino engine* [34], ktorý je implementáciou JavaScriptu. Môžu byť použité aj iné skriptovacie jazyky. Stačí ak spĺňajú požiadavky špecifikácie [35].

Na základe porovnania výhod a nevýhod vyššie popísaných riešení bolo vybrané použitie script enginu v rámci aplikácie na administráciu. Konkrétny skriptovací jazyk bude JavaScript, pretože po prvé je už v Jave zabudovaný, a po druhé je dostačujúci pre tvorbu potrebných konfigurácií.

Na záver treba spomenúť, že je tiež možnosť vytvoriť vlastný skriptovací jazyk. To by však pri existencii script enginu bolo kontraproduktívne.

3.3 Sledovanie očných pohybov

Na sledovanie očných pohybov je použitý prístroj Eyelink II (1.6). Voľba iného prístroja nebola zvažovaná, pretože vo všeobecnosti sú príliš nákladné a Eyelink je k dispozícii (vďaka spolupráci s Psychologickým ústavom AVČR).

Eyelink rozhranie je tvorené knižnicami, z ktorých najdôležitejšie sú *eyelink_core.dll* a *eyelink_core_graphics.dll*. Tieto knižnice implementujú ovládanie prístroja Eyelink a umožňujú prístup k funkciám na počítači, na ktorom je špeciálne upravený operačný systém a Eyelink software.

Rozhranie Eyelink musí byť umiestnené na počítači, na ktorom sú merané očné pohyby. Najlepšie priamo v grafickej aplikácii. Problém je, že Unreal engine neumožňuje prístup ku knižniciam vo formáte DLL. Preto je potrebné vytvoriť komunikačný most, ktorý by tento prístup sprostredkoval. Unreal engine komunikuje s okolím pomocou sieťového rozhrania. To je možné využiť a vytvoriť klienta, ktorý by pracoval s Eyelink knižnicami a Unreal enginom zároveň. Klient môže byť napísaný v akomkoľvek jazyku, ktorý podporuje tvorbu užívateľského grafického rozhrania, sieťovú komunikáciu a prístup k DLL knižniciam. Okrem iných, tomuto popisu vyhovuje jazyk C/C++, ktorý bol zvolený a použitý na tvorbu klienta sprostredkujúceho komunikáciu medzi Unreal enginom a softwarom Eyelink.

4 Implementácia

Táto kapitola sa zaoberá popisom implementácie riešenia vzhľadom na zvolené vývojové prostredia a nástroje.
Hlavné problémy riešené pri implementácii sú nasledujúce:

- vytvorenie virtuálneho prostredia pomocou Unreal enginu,
- komunikácia Unreal enginu s okolím,
- interakcia avatara s objektmi v prostredí,
- konfigurácia experimentov,
- návrh rozhrania aplikácie na administráciu,
- 2D a 3D náhľad na priebeh experimentu,
- meranie očných pohybov,
- meranie výstupných dát, a
- spätná rekonštrukcia experimentu.

4.1 Vytvorenie virtuálneho prostredia

V časti 3.1 bol ako nástroj pre vytvorenie grafickej aplikácie zvolený Unreal engine. Obsahuje UnrealEd editor, ktorý umožňuje navrhovať mapy, rozmiestňovať v nich objekty a priradiť im rôzne vlastnosti. To ale nestačí. Je potrebné vytvoriť samostatný programový celok, na základe ktorého bude možné zasahovať do kódu prostredia tak, ako to bude potrebné. Preto riešime problém, ako do Unreal enginu zakomponovať oddelenú aplikáciu, v ktorej bude možné spustiť a ovládať špecifické virtuálne prostredie. Existujú nasledujúce spôsoby, ako to docieliť:

- Vytvoriť úplnú konverziu hry. Treba pripomenúť, že Unreal engine nevyužívame ako samostatný nástroj, ale je k dispozícii ako súčasť hry UT2004. Úplná konverzia hry znamená ponechať len samotný engine a zmeniť celú hru na inú. Výhodou takéhoto riešenia je, že grafické prostredie experimentov nie je rušené prvkami herného rozhrania. Pri štúdiu možnosti úplnej konverzie som sa dočítala, že vyžaduje nie jedného, ale celý tím programátorov a veľmi veľa času. Preto nie je dobré sa do nej

púšťať, ak to nie je nevyhnutné. Je to nevýhoda, ktorá značne prevyšuje výhody plynúce z tohto riešenia.

- Použiť mutátor. Mutátor je trieda alebo objekt, pomocou ktorého je možné zmeniť určité vlastnosti hry. Predstavme si napríklad, že by sme chceli, aby náš avatar bol nesmrteľný alebo mal nejakú výnimočnú vlastnosť. Naprogramujeme teda mutátor, ktorý môžeme následne zvoliť v menu ako doplnok hry. Táto možnosť je zo všetkých najjednoduchšia a vyžaduje najmenej programovania. Nevýhodou je, že mutátor tvorí len akýsi prídavok k hre. Hrou je tu myslený konkrétny typ. V UT2004 napríklad *Death match* alebo *Capture the flag*. Znamenalo by to teda, že by sme sa doplnkom hry snažili celú hru zmeniť.
- Vytvoriť nový typ hry. Táto možnosť predstavuje zlatú strednú cestu medzi dvoma prechádzajúcimi. Pre vytvorenie nového typu hry nie je potrebný tím programátorov a pritom umožňuje získať virtuálne prostredie oddelené od celej hry. Jedinou nevýhodou je, že k tomu, aby sme sa do nášho prostredia dostali, je potrebné preklikať sa cez menu hry UT2004. Táto nevýhoda je estetickej povahy a iný vplyv na prostredie nemá.

Po zvážení výhod a nevýhod uvedených troch možností sa ukázal ako najvhodnejší tretí spôsob. Grafické prostredie je vytvorené ako nový typ hry.
Aby sa z bojovej hry stal mierumilovný simulátor vedeckých experimentov, bolo potrebné urobiť niekoľko úprav. Medzi hlavné patrilo odobranie zbrane, odstránenie útočníkov, vypnutie akčných zvukov a časového limitu. Okrem toho bola doplnená možnosť z konfiguračného skriptu nastaviť rýchlosť pohybu avatara. Priložené CD obsahuje voľne dostupný skin (výzor) avatara v ľudskej podobe, ktorý je možné avatarovi priradiť v menu hry.

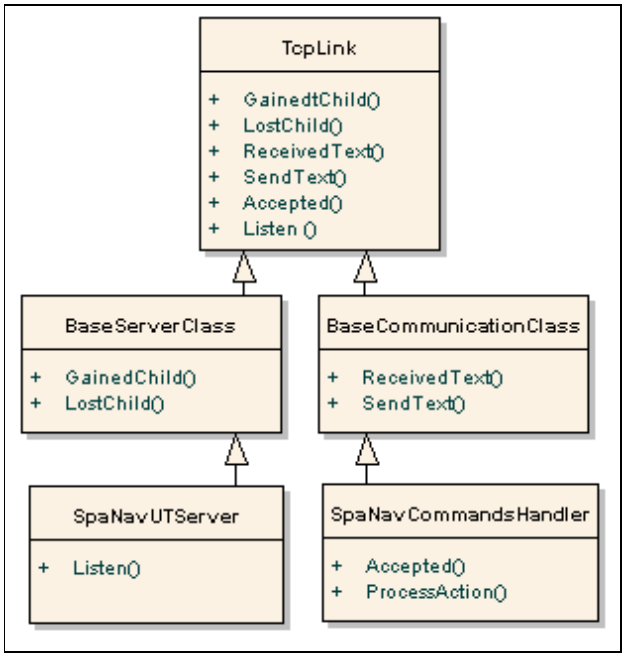
4.2 Komunikácia Unreal enginu s okolím

Z popisu architektúry tohto projektu v kapitole 2 plynie, že grafické prostredie musí komunikovať s ďalšími dvomi aplikáciami. Rozhranie pre sieťovú komunikáciu v Unreal engine tvorí trieda *TcpLink.* Poskytuje rozhranie na nadviazanie spojenia, prijímanie a posielanie správ. Pomocou nej bol vytvorený server starajúci sa o spojenie a trieda

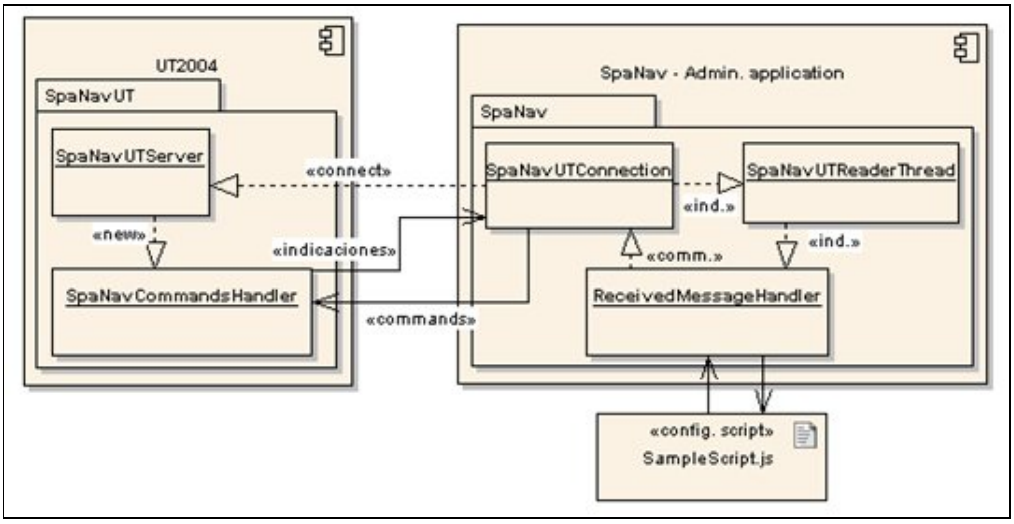
obsluhujúca komunikáciu. Hierarchia a funkcie tried sú zobrazené na schéme na obrázku 9.
Existuje aj jej verzia *UdpLink*, ale pretože správy z Unreal enginu budú informovať o udalostiach, na ktoré je potrebné hneď reagovať, je vhodnejšia jej tcp verzia.
Aby server fungoval správne je potrebné hru spúšťať v menu pomocou voľby *Host game*. Táto voľba však okrem správnej funkcie serveru umožňuje ostatným hráčom pripojiť sa k hre on-line. Aby sa tomu zabránilo bol maximálny počet hráčov nastavený na 1.
Na obrázku 10 je schéma toku komunikácie medzi grafickým prostredím a aplikáciou na administráciu experimentov. Na výmenu informácii medzi nimi slúži jednoduchý textový protokol, ktorý má nasledujúci formát.

```
TYP_PRÍKAZU {názov_atribútu1 hodnota_atribútu1} {názov_atribútu2 hodnota_atribútu2} …
```

Všetky medzery v predpísanom formáte sú povinné, pričom TYP_PRÍKAZU a názov_atribútu nesmú obsahovať medzery.
Napríklad na nastavenie viditeľnosti oblasti s menom „oblasť“ sa pošle do grafickej aplikácie príkaz v tvare VISIBLE {oblast true}.



Obr. 9: Hierarchia tried použitých na sieťovú komunikáciu. V unreal engine tvorí rozhranie sieťovej komunikácie trieda *TcpLink*. Obsahuje udalosti *GainedChild()* a *LostChild()*, ktoré vznikajú pri pripojení alebo odpojení klientov. Tieto funkcie dedí trieda *BaseServerClass*, ktorá ich rozširuje. Trieda *SpaNavUTServer* pri svojom vlastnom vzniku začne počúvať na porte 3000. Rozhranie *TcpLink* ďalej obsahuje udalosť *ReceivedText()*, ktorá vznikne pri príchode textu a funkciu *SendText()*, ktorá umožňuje odosielať informácie na druhú stranu nadviazaného spojenia. Trieda *BaseCommunicationClass*, rozdeľuje prichádzajúce dáta tak, aby k jednotlivým hodnotám bolo možné pristupovať pomocou funkcie *GetArgVal(String názov_atribútu)*, ktorá vracia hodnotu atribútu zadaného v parametri (viz. textový protokol popísabý vyššie). Keď vznikne spojenie, *SpaNavUTServer* vytvorí obslužnú triedu *SpaNavCommandsHandler*, ktorá slúži na spracovanie požiadaviek klienta. Pri zrušení spojenia trieda *SpaNavCommandsHandler* zanikne. Návrh oboch tried, *BaseServerClass* a *BaseCommunicationClass* je prebratý z projektu Pogamut [40].



Obr. 10: Komunikácia medzi grafickým prostredím a aplikáciou na administráciu experimentov. Grafické prostredie je začlenené do hry UT2004 a pomenované *SpaNavUT*. Komunikuje pomocou tried *SpaNavUTServer* a *SpaNavCommandsHandler*, popísaných na obrázku 14. Aplikácia na administráciu je pomenovaná *SpaNav*. Obsahuje triedy *SpaNavUTConnection* a *SpaNavUTReaderThread*, ktoré obsluhujú spojenie a príjem správ. Správy spracováva trieda *ReceivedMessageHandler*. Program aplikácie na administráciu vyhodnocuje konfiguračný skript pri príchode každej relevantnej správy.

4.3 Objekty vo virtuálnom prostredí

Pri skúmaní orientácie v priestore sú okrem samotného priestoru potrebné aj objekty, ktoré v určitom zmysle definujú úlohu. Táto časť sa bude venovať popisu, akým spôsobom je možné definovať a konfigurovať priebeh experimentov.

Úlohou môže byť napríklad prejsť od jedného objektu k druhému a cestou sa vyhnúť tretiemu. Aby experimentátor mohol úlohu hodnotiť, potrebuje nazbierať dáta z jej priebehu. Zbieranie dát je popísané v časti 4.8.

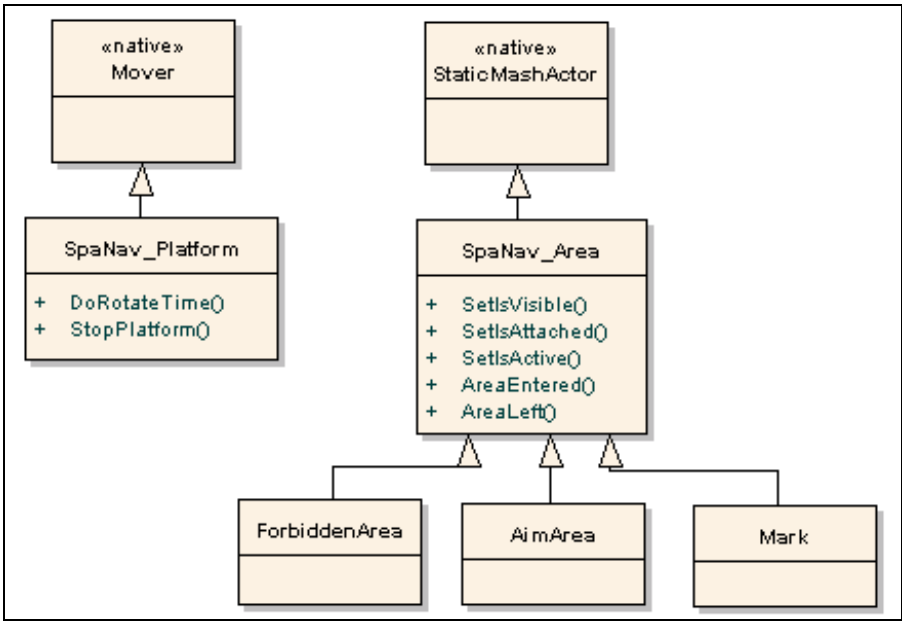
Z popisu úlohy v BVA aréne vieme, že obsahuje plošinu, ktorá sa otáča a oblasti. Oblasti sú podľa významu v úlohe rozdelené na cieľové (tie, ktoré hľadáme), zakázané (tie, ktorým sa snažíme vyhnúť) a značky (orientačné body).

Pri prevode BVA arény do virtuálnej reality nás preto zaujíma hlavne plošina, oblasti a značky. Stan nie je podstatný, pretože slúži len na vymedzenie a ohraničenie testovacieho prostredia. Vo virtuálnej realite môže byť testovacím prostredím celý priestor. Riešime teda, akými vlastnosťami by mali objekty disponovať. V prvom rade, by mali mať určitý tvar a farbu. V reálnom prostredí je plošina kruh, oblasti kruhové výseky a značky jednoduché tvary. Vo virtuálnom prostredí si môžeme dovoliť tieto vlastnosti rozšíriť. Najlepšie bude, ak

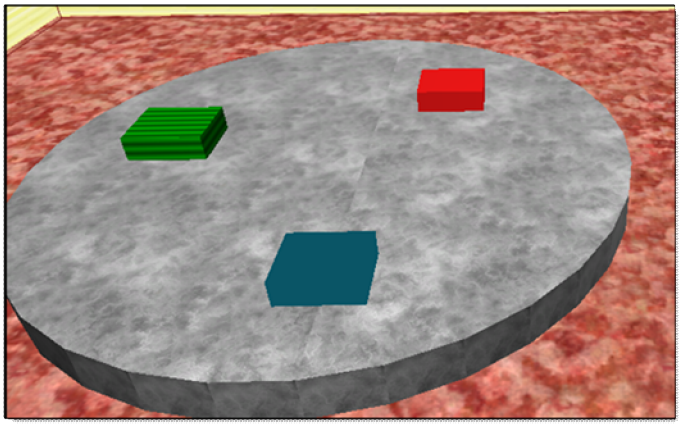
umožníme objektom priradiť akýkoľvek tvar a farbu. V druhom rade je potrebné, aby bolo možné objektom meniť viditeľnosť, viazanosť na plošinu (otáčanie alebo neotáčanie spolu s ňou), aktivitu a umožniť spúšťanie výstražného signálu. Oblasť je aktívna, ak je schopná detegovať, či do nej avatar vstúpil. Tieto vlastnosti chceme meniť počas experimentu pomocou konfiguračného skriptu. K tomu nestačí nadefinovať prostredie v UnrealEd editore.

Za týmto účelom boli vytvorené nové objekty, ktorým bola explicitne vytvorená vyššie uvedená funkčnosť. Tieto objekty a ich hierarchia sú znázornené na obrázku 11. Sú vytvorené tak, aby ich bolo možné v UnrealEd editore vložiť do prostredia. Výhodou takéhoto riešenia je, že umožňuje definovať ich tvar a polohu („viditeľné“ vlastnosti) v editore a ich ostatné vlastnosti ovládať programovo. Objekty pri použití v editore vyzerajú tak, ako je ukázané na obrázku 12, ale nie je problém im tvar aj textúru zmeniť. Nový tvar je možné vytvoriť v samotnom editore, ktorý poskytuje základné tvary. Pretože v pôvodnom riešení BVA sú oblasti definované ako kruhové výseky a UnrealEd editor ich nemá, bol tento nedostatok doplnený o voľne dostupný *„Tarquin brush builder pack“* [41], ktorý rozširuje množinu základných tvarov v editore. Pokiaľ by ani to nestačilo, je možné importovať tvary z aplikácií určených na ich vytváranie (3D Max Studio a Maya).

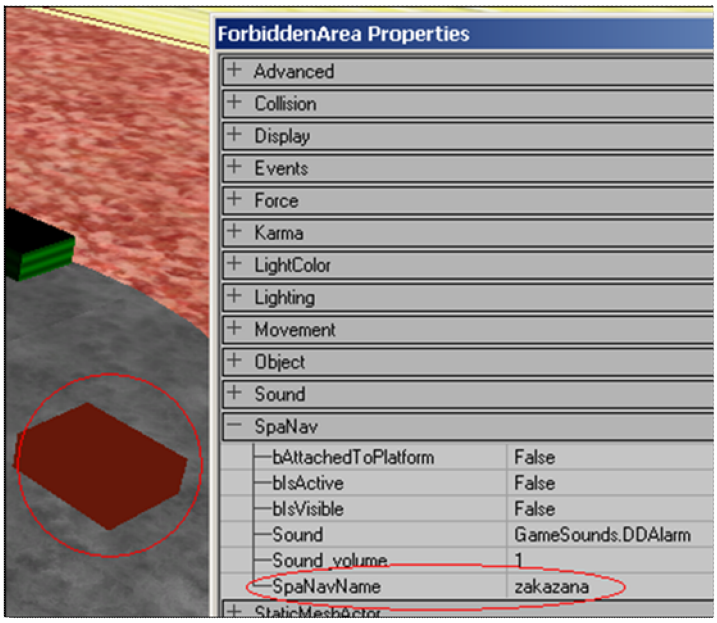
V UnrealEd editore je možné objektom nastaviť (obr. 13) aj ostatné vlastnosti a priradiť zvuk, ktorý bude spúšťaný ako výstražný signál. Zvuky sú súčasťou editoru alebo ich je možné tiež importovať. Okrem toho je potrebné každému objektu nastaviť meno, ktoré slúži ako jednoznačný identifikátor. Pomocou neho sa na objekt v konfiguračnom skripte odkazuje.



Obr. 11: Schéma hierarchie tried *SpaNav_Platform* a *SpaNav_Area*. *SpaNav_Platform* reprezentuje otočnú plošinu. Je zdedená od natívnej triedy *Mover*, ktorá pre Unreal engine reprezentuje pohybujúce sa objekty a deteguje, kedy sa na ne hráč postaví, čo spôsobí, že sa pohybuje spolu s nimi. Obsahuje okrem iného funkciu na definovanie času, rýchlosti a smeru jej otáčania. *SpaNav_Area* reprezentuje akúkoľvek oblasť alebo značku. Je zdedená od natívnej triedy *StaticMashActor*, ktorá enginu hovorí, že počas priebehu programu nemení svoj tvar, čím zefektívňuje grafické výpočty. Obsahuje hlavne funkčnosť na definovanie vlastností (viditeľnosť, citlivosť na vstup, viazanosť na plošinu) a detekciu kolízií.



Obr. 12: Plošina, zakázaná oblasť (červená), cieľová oblasť (zelená) a značka (modrá) s ich základnými tvarmi a textúrami v UnrealEd editore.



Obr. 13: Priradenie mena k objektu v UnrealEd editore. Okrem toho je možné objektu inicializovať viazanosť na plošinu, aktívnosť, viditeľnosť, priradiť zvuk a nastaviť hlasitosť výstražného signálu.

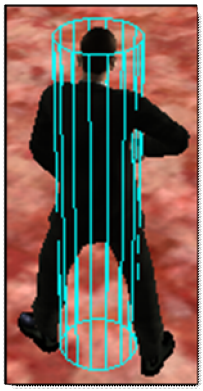
4.3.1 Interakcia avatara s objektmi

Jediný spôsob, ktorým avatar komunikuje s objektmi v prostredí je dotyk. Návrh priebehu experimentu preto v zásade pozostáva z definícií typu „ak avatar vstúpil do oblasti X, tak zmeň vlastnosti oblastí Y a Z“.

Unreal engine obsahuje rozhranie na detekciu dotyku dvoch objektov: aby sa dva objekty dotkli sa im musia dotknúť ich tzv. „kolízne obaly“. V počítačovej grafike je detekcia kolízií náročná na výpočet. Objekty môžu mať rozlične zložité tvary. Preto sú im priradené kolízne obaly, ktoré celý objekt uzavrú do jednoduchého geometrického útvaru nenáročného na výpočet. Presne takýmto spôsobom deteguje kolízie aj Unreal engine. V editore umožňuje objektom priradiť kolízne teleso, neskôr použité pri výpočtoch. Pretože sa môže stať, že objekt má tak neforemný tvar, že ho nie je možné uzavrieť do jednoduchého telesa, poskytuje Unreal engine aj tzv. detekciu kolízií vzhľadom na materiál. Materiál je objektu priradená textúra alebo farba. Kolízny obal objektu vtedy tvoria všetky jeho plochy, pokryté daným materiálom.

V tomto projekte sú použité oba druhy detekcie kolízií. Avatar má priradený kolízny obal v tvare cylindru (obr. 14), pretože výzor avatara najčastejšie reprezentuje model človeka, ktorý má veľmi komplexný tvar. Toto riešenie si môžeme dovoliť, lebo v reálnom prostredí je

pohyb človeka reprezentovaný jedným bodom (senzor je umiestnený v helme, ktorú má testovaný človek na hlave). Okrem toho užívateľ môže nastaviť veľkosť kolízneho cylindra avatara, čo rozširuje pôvodné riešenie v BVA aréne.
Pre objekty reprezentujúce oblasti je dôležitý ich tvar, preto majú nastavenú detekciu kolízií vzhľadom na materiál.



Obr. 14: Kolízny cylinder avatara. Tvar cylindra rozhoduje, či avatar vstúpil do nejakej oblasti. Jeho výšku a priemer a je možné nastaviť pomocou konfiguračného skriptu.

Rozhranie Unreal enginu postačuje na to, aby sme vedeli zareagovať, keď sa avatar dotkne oblasti. Užívateľ však potrebuje merať do výstupných súborov dáta o dráhe alebo čase strávenom v oblasti. Tým vzniká problém, pretože Unreal engine nevie určiť, či sme vo vnútri oblasti alebo či sme ju opustili. Z moji pozorovaní vyplýva, že Unreal engine rieši detekciu kolízií nasledovne: rozpoznáva vstup do alebo výstup z oblasti pomocou udalostí *Touch()* a *Untouch()*. To ešte vyhovuje. Udalosti *Touch() a Untouch()* sú obe však volané pri každom preťatí kolízneho cylindru s materiálom. To spôsobuje problém, lebo napríklad už pri vstupe avatara do oblasti vznikne niekoľko desiatok takýchto udalostí. Kým je avatar vo vnútri oblasti a nedotýka sa jej okrajov, nevzniká žiadna udalosť.
Riešenie Unreal enginu nemožno použiť v tomto projekte, preto bolo upravené nasledujúcim spôsobom:

- Ak vzniknú udalosti *Touch()* a *Untouch()* naraz znamená to, že avatar vstúpil do oblasti.
- Ak po prvom vstupe už udalosti *Touch()* a *Untouch()* nevzniknú, treba vykonať test pomocou vyslania lúču (raytracingu) smerom od avatara do stredu oblasti. Ak lúč

pretne hranu danej oblasti znamená to, že avatar ju opustil, v opačnom prípade je vo vnútri. Ak je vo vnútri opäť testujeme buď vznik udalostí *Touch()* a *Untouch()* alebo raytracingom.

- Ak vznikne samostatný *Untouch()*, tak avatar oblasť opustil (nie je potrebné použiť raytracing).

Popísaným algoritmom vieme presne určiť, kedy avatar do oblasti vstúpil, kedy je vo vnútri a kedy oblasti opustil.

4.3.2 Konfigurácia experimentov

V kapitole 3.2.1 sme si zvolili, že experimenty bude možné parametrizovať pomocou konfiguračného skriptu v jazyku JavaScript. V tejto časti bolo treba navrhnúť, akou formou bude skript komunikovať s objektmi a s prostredím vo virtuálnom svete a premyslieť, čo všetko bude umožňovať.
Skript by mal užívateľovi dovoľovať:

- definovať konkrétne prostredie (mapu), ktorá sa má spustiť,
- inicializovať vlastnosti objektov na začiatku experimentu,
- zastaviť alebo pozastaviť experiment,
- konfigurovať vlastnosti avatara, napríklad veľkosť kolízneho cylindru, rýchlosť pohybu, či sa má pohybovať behom alebo krokom,
- nastaviť ako často chceme zaznamenávať dáta do výstupných súborov,
- nastaviť časový interval, po ktorého vypršaní sa vykoná nejaká akcia,
- nastaviť kláves, ktorého stlačenie vykoná nejakú akciu,
- reagovať na vstup avatara do oblasti,
- konfigurovať čas, rýchlosť a smer otáčania plošiny,
- meniť vlastnosti oblastí alebo reagovať na ich zmenu (vlastnosťami sú viditeľnosť, viazanosť na plošinu a aktívnosť),
- spúšťať zvukový signál pri vstupe do oblasti alebo samostatne na určitý časový interval.

V prvom rade je potrebné vyriešiť problém, ako sa informácia zo skriptu v textovom súbore dostane do grafickej aplikácie a späť. Script engine (3.2.1) dovoľuje definovať objekty v programe aplikácie tak, že je k nim možné pristupovať z konfiguračného skriptu. Týmto spôsobom boli vytvorené „kľúčové slová“, ktoré užívateľ používa v skripte a definuje pomocou nich priebeh experimentov. Sú to napríklad slová *experiment* (vlastnosti avatara a experimentu), *timer* (časovač), *key* (definovanie funkčného klávesu), *preference* (cieľová oblasť), *avoidance* (zakázaná oblasť) a *mark* (značka). Každé slovo je vlastne objekt, ktorý má určité funkcie. Napríklad:

```
preference.get("ciel").setActive(true);
```

V príklade je pomocou funkcie *get()* určená cieľová oblasť s menom „ciel“ a nastavená ako aktívna pomocou funkcie *setActive()*.

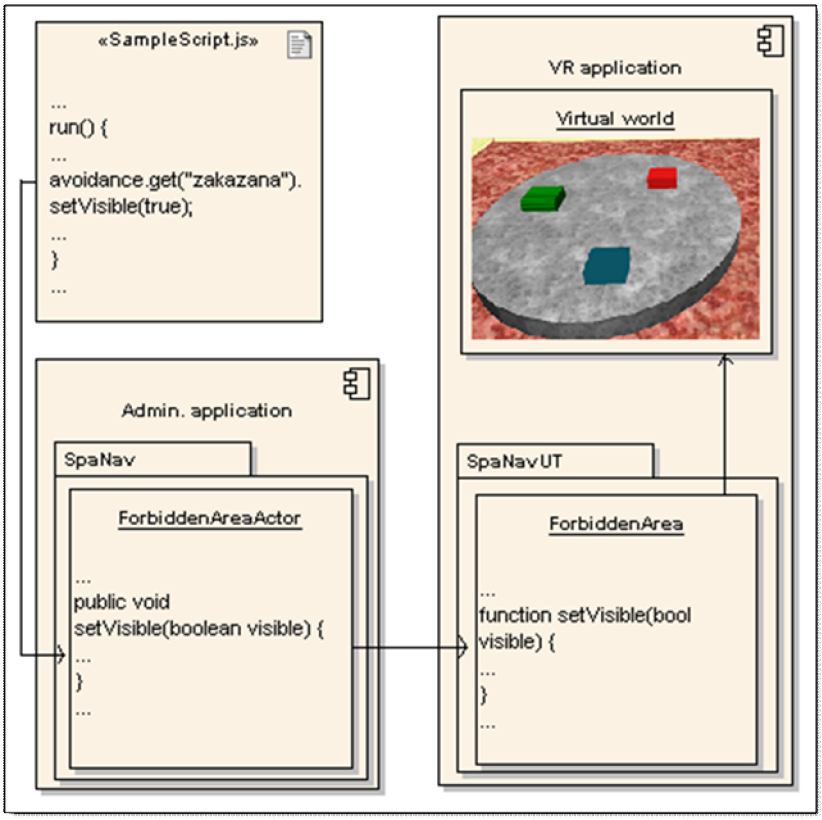
Pri definovaní funkčného klávesu, bolo potrebné vziať do úvahy situáciu, kedy je na experiment použitý len jeden počítač. V takom prípade sa predpokladá, že pacient vykonáva experiment a na obrazovke je aktívne len okno grafickej aplikácie. Preto treba zaznamenávať stlačenie funkčného klávesu nie len z aplikácie na administráciu, ale aj z grafickej aplikácie a informovať konfiguračný skript.

Zoznam všetkých objektov, ich funkcie a popis je možné nájsť v užívateľskej dokumentácii na priloženom CD. Na obrázku 15 je ukážka príkladu konfiguračného skriptu.

Vieme teda, že zo skriptu je možné pristupovať k objektom v aplikácii na administráciu. Z nej sú odoslané cez sieťové spojenie do grafickej aplikácie. Tento proces je znázornený na obrázku 16. Skript je vyhodnocovaný pri každom príchode relevantnej udalosti.

```
function init() {
        experiment.setMap("TEST-SampleMap");
}
function run() {
        if (experiment.isStarted()){
                experiment.setTrackRate(0.3);
                timer.set("cas",20);
                preference.get("cielova").setActive(true);
                preference.get("cielova").setVisible(true);
                avoidance.get("zakazana").setActive(true);
                avoidance.get("zakazana").setVisible(true);
                platform.get("plosina").doRotateTime(10000,5,-1);
        }
        if (preference.get("cielova").entered()){
                preference.get("cielova").setActive(false);
                preference.get("cielova").setVisible(false);
                preference.get("cielova2").setActive(true);
                preference.get("cielova2").setVisible(true);
        }
        if (preference.get("cielova2").entered()){
                preference.get("cielova2").setActive(false);
                preference.get("cielova2").setVisible(false);
                preference.get("cielova").setActive(true);
                preference.get("cielova").setVisible(true);
        }
}
function timerTask(name) {
        if (name == "cas"){
                platform.get("plosina").doRotateTime(10000,5,1);
        }
}
```

Obr. 15: Príklad konfiguračného skriptu. Vo funkcii *init()* je definované meno mapy, ktorá sa má spustiť. Vo funkcii *run()* je popis priebehu experimentu. Na začiatku experimentu sa nastaví požadovaná frekvencia získavania výstupných dát (každých 0.3 sekúnd) a spustí sa odpočítavanie *timeru* (20 sekúnd). Inicializujú sa počiatočné hodnoty oblastí s menami „cielova“ a „zakazana“ a spustí sa otáčanie plošiny (po dobu 10000 sekúnd, rýchlosťou 5 stupňov za sekundu, proti smeru hodinových ručičiek). Po vypršaní časového limitu *timeru* sa spustí funkcia *timerTask*, ktorá zmení smer otáčania plošiny. Ďalej skript definuje, že vždy keď avatar vstúpi do oblasti s menom „cielova“ (pokiaľ je aktívna), tak sa deaktivuje a nastaví ako neviditeľná, pričom druhá oblasť „cielova2“ sa aktivuje a zviditeľní. Pri vstupe do oblasti „cielova2“ je reakcia opačná.

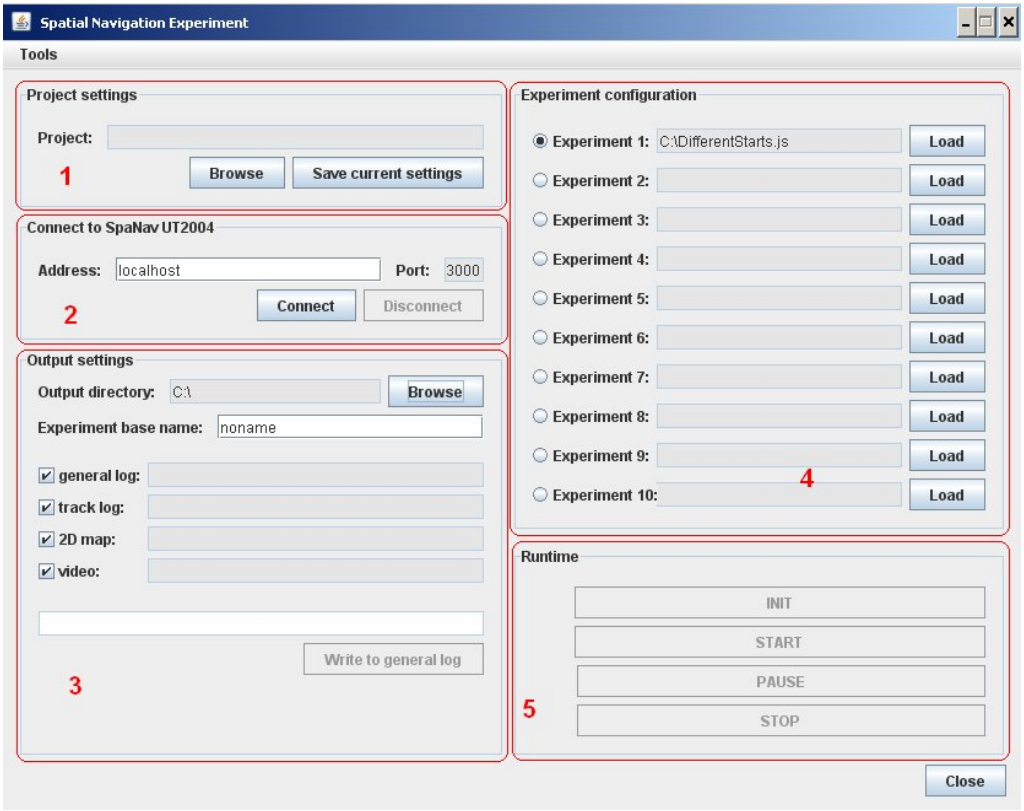


Obr. 16: Schéma toku informácií z konfiguračného skriptu do grafickej aplikácie pri nastavení vlastnosti objektu v skripte. Objekt *avoidance* obsahuje pole všetkých zakázaných oblastí v mape, ktoré sa naplní pri inicializácii experimentu. Pomocou funkcie *get(<meno_oblasti>)* získame konkrétny objekt s týmto menom. Funkcia volaná na získanom je vlastne funkcia volaná na objekte *ForbiddenAreaActor* v aplikácii na administráciu. Tá odošle príkaz do grafického prostredia, že chceme zmeniť viditeľnosť oblasti „zakázaná“, a to túto správu odovzdá cieľovému objektu. Pri vzniku udalosti vo virtuálnom svete, napríklad pri vstupe do oblasti, tečú informácie opačným smerom. Zároveň každá relevantná udalosť vyvolá spustenie skriptu, aby na ňu bolo možné reagovať.

4.4 Aplikácia na administráciu

Pri návrhu grafického rozhrania aplikácie na administráciu experimentov bolo potrebné zistiť, ako prebiehajú reálne experimenty v BVA aréne z pohľadu doktora a čo by bolo možné vylepšiť. V pôvodnej aplikácii grafické rozhranie neexistuje a experimenty sa spúšťajú z príkazového riadku. Takéto riešenie nie je veľmi pohodlné. Doktor počas jedného sedenia s pacientom väčšinou vykoná viac experimentov, pričom niektoré podľa potreby zopakuje. Z toho vyplýva, že nová aplikácia by mala mať priestor na načítanie rôznych konfiguračných súborov, z ktorých každý definuje jeden experiment. Konfiguračný skript určuje, ktorá mapa sa má v grafickej aplikácii spustiť a ako má experiment prebiehať. Ďalej by mala byť možnosť určiť adresár, do ktorého sa uložia namerané dáta. Výstupné súbory by mali byť

pomenované jednoznačne, a malo by byť zabezpečené, aby namerané dáta neboli prepísané pri novom experimente. Vhodná je aj možnosť všetky nastavenia uložiť. Obrázok 17 popisuje jednoduché grafické rozhranie, ktoré bolo vytvorené na základe týchto poznatkov.



Obr. 17: Aplikácia na administráciu experimentov. Jej grafické rozhranie tvoria nasledujúce časti, ktoré umožňujú: 1. Uloženie aktuálnych nastavení do súboru a ich opätovné načítanie. 2. Pripojenie ku grafickej aplikácii s virtuálnym prostredím. 3. Zvolenie adresára a základného mena pre výstupné súbory, určenie požadovaných výstupov, priamy zápis poznámok do logu počas priebehu experimentu. 4. Načítanie konfiguračných skriptov, ktoré chce doktor použiť počas sedenia s pacientom. 5. Ovládanie experimentu – inicializácia (načítanie mapy), spustenie, pozastavenie, ukončenie. Menu Tools obsahuje položky Track log reconstruction a Show debug lines.
Mená výstupných súborov sú tvorená automaticky zo základného mena (*Experiment base name*) nasledovným spôsobom: General log: <základné_meno>.log; Track log: <základné_meno>_<číslo_experimentu>_<číslo_opakovania_experimentu>.tr; 2D mapa: <základné_meno>_<číslo_experimentu>_<číslo_opakovania_experimentu>.png; Video: <základné_meno>_<číslo_experimentu>_<timestamp>. demo. Pridaním čísla opakovania sa predíde premazaniu súborov s dátami.

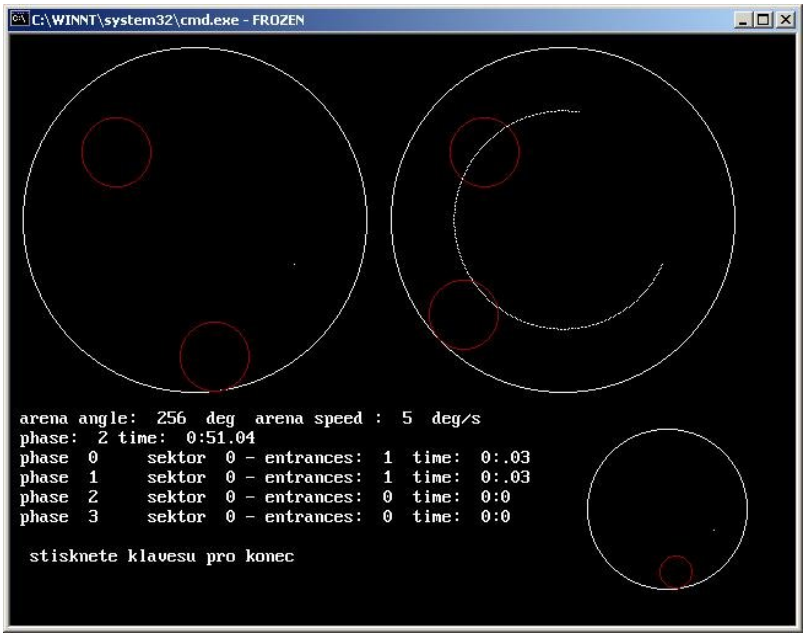
4.5 2D náhľad na experiment

Pri riešení 2D náhľadu na experiment mi bol poskytnutý obrázok (obr. 18) doterajšieho riešenia. Obsahuje tri pohľady zhora na plošinu. Na jednom je staticky vykreslená plošina a oblasti, ktoré k nej nie sú viazané, teda sú statické v miestnosti. Vykresľuje tiež dráhu

pohybu testovaného subjektu vzhľadom na miestnosť. To znamená, že ak subjekt stojí nehybne na plošine a tá sa s ním otáča, tak sa vykreslí kruh. Na druhom obrázku je staticky vykreslená plošinu a oblasti, ktoré k nej sú viazané a zobrazuje dráhu pohybu subjektu vzhľadom na plošinu To znamená, že ak subjekt stojí nehybne na plošine a tá sa s ním otáča, tak sa vykreslí len jeden bod s polohou subjektu. Tretí obrázok je aktívny a ukazuje, ako sa plošina otáča spolu s aktuálnou polohou subjektu. Tiež zobrazuje práve aktívne oblasti. Grafické rozhranie poskytuje aj textové informácie o priebehu experimentu v podobe údajov, koľko krát bolo do jednotlivých oblastí vstúpené a ako dlho trvalo nájsť oblasť. Toto riešenie je v podstate vyhovujúce a neurológovia, s ktorými som prácu konzultovala sú naň zvyknutí. Nie je preto potrebné veľa vecí meniť.

Vhodné je farebne odlíšiť typy oblastí (cieľová, zakázaná a značka) a dráhu testovaného subjektu. Tiež by bolo dobré na treťom aktívnom obrázku zobrazovať viditeľné oblasti s výplňou a neviditeľné bez výplne, aby ich mohol doktor rozlíšiť. Okrem toho rozšírime textové informácie o viac dostupných údajov.

Pri vytváraní 2D náhľadu bolo potrebné riešiť získavanie tvaru oblastí.



Obr. 18: Ukážka pôvodného riešenia 2D náhľadu v BVA aréne. Obsahuje tri obrázky plošiny. Dva horné sú statické. Obrázok vľavo hore zobrazuje oblasti neviazané k plošine a dráhu človeka vzhľadom na miestnosť. Obrázok vpravo hore zobrazuje oblasti viazané k plošine a dráhu človeka vzhľadom na plošinu. Na treťom obrázku dole je plošina vykresľovaná aktívne, podľa toho, ako sa otáča, spolu s práve aktívnymi oblasťami a aktuálnou polohou človeka. Textové pole obsahuje údaje o priebehu experimentu.

4.5.1 Získanie tvaru oblasti

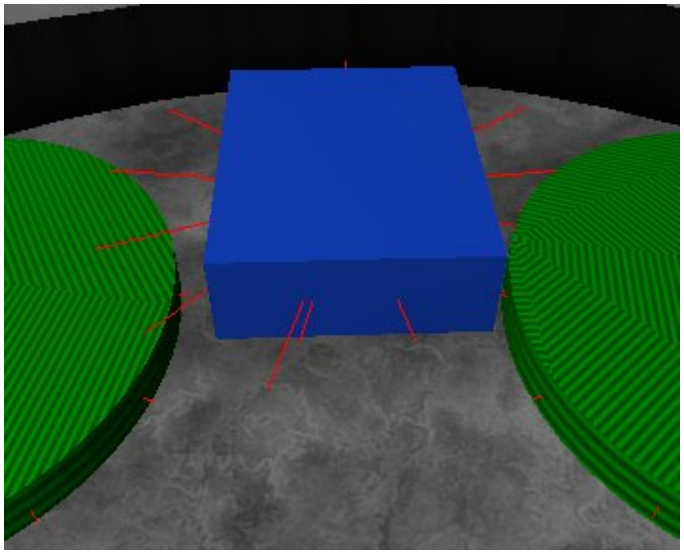
Jediný údaj, ktorý Unreal engine o objekte vo virtuálnom svete môže pre získanie tvaru poskytnúť, je jeho relatívny stred. Tento stred sa nazýva pivot a je to vlastne os, okolo ktorej sa objekt otáča (v prípade, že sa otáča). Je teda treba vyriešiť, ako je možné zo znalosti polohy pivota získať tvar oblasti.

V kapitole 4.3.1 bola pri detekcii kolízií využitá technika raytracingu. Podobný postup je použitý aj v tomto prípade. Lúče, pomocou ktorých zisťujeme tvar oblasti, sú mierené z určitej vzdialenosti do stredu oblasti. Prvá vzdialenosť je vždy konštantná hodnota, ktorá sa zväčšuje alebo zmenšuje na základe výsledku predchádzajúceho raytracingu. Napríklad, pokiaľ vyslaný lúč nepretne hranu oblasti vieme, že je vzdialenosť miesta vyslania od stredu oblasti príliš krátka. Zväčšíme ju teda a skúsime šťastie znovu. Pokiaľ pretneme hranu oblasti a získame súradnice, posunieme štartovacie miesto lúču o určitý uhol. Potom opakujeme raytracing so vzdialenosťou od stredu oblasti, ktorá bola úspešná pri predchádzajúcom získaní súradníc, pretože predpokladáme, že väčšina oblastí mení svoj tvar plynule. Takto postupujeme po celej pomyslenej kružnici okolo oblasti.

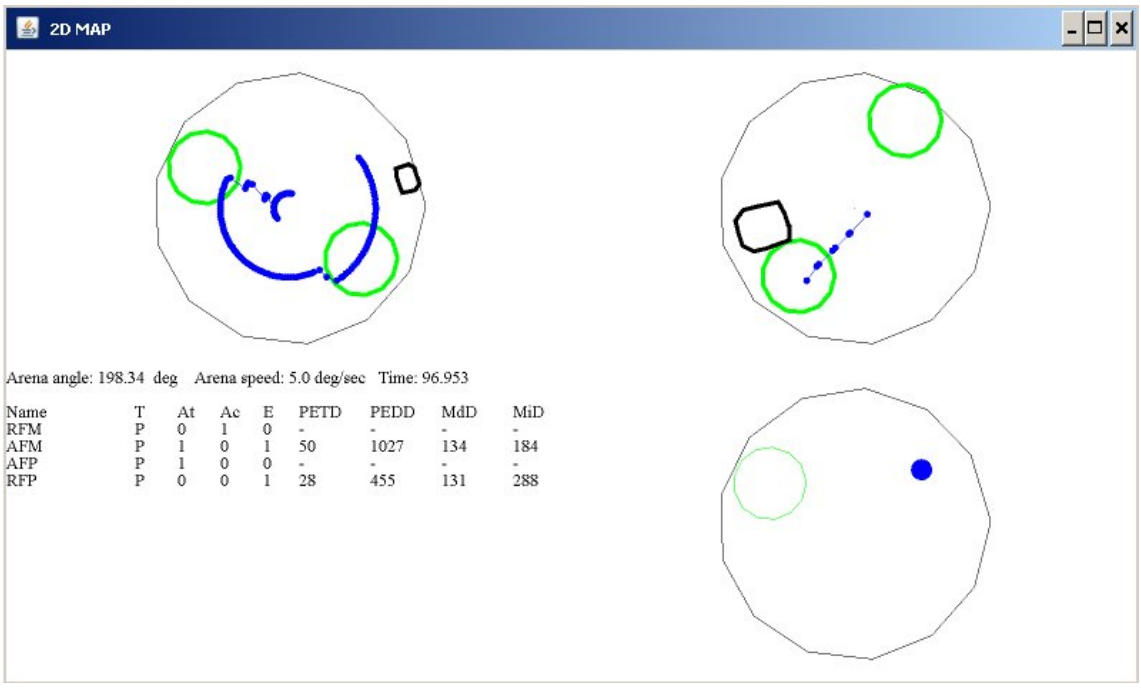
Treba si uvedomiť, že popísaný algoritmus je zjednodušený. V skutočnosti sú riešené aj prípady, kedy do oblasti zasahujú iné objekty. Preto musíme sledovať všetky objekty, ktoré lúč na svojej trase pretne, a upravovať pozíciu štartovacieho miesta lúču na základe viacerých informácií. Pokiaľ je oblasť mimo miestnosť, alebo pivot nie je v jej vnútri, tvar nie je možné zistiť. Ak je objektom plošina, tak raytracingu vypočítame jej polomer. Ten je následne použitý na škálovanie veľkosti obrázku v 2D mape.

Ak pri navrhovaní virtuálneho prostredia nedáme pozor na polohu pivota, môžeme neskôr naraziť na problémy pri 2D náhľade, alebo pri zisťovaní, či avatar vstúpil do oblasti. Pri používaní tohto softwaru neurológmi som sa stretla s pripomienkami, že majú občas problém zistiť, kde sa pivot v skutočnosti nachádza. Preto bola aplikácie na administráciu doplnená o voľbu viditeľnosti lúčov použitých pri získavaní tvaru. Takýmto spôsobom môže užívateľ pozorovať, kde sa lúče zhlukujú a odhaliť nesprávnu polohu pivota. Obrázok 19 ukazuje, príklad ako takéto čiary môžu vyzerať.

Na obrázku 20 je vidieť výsledné riešenie 2D náhľadu na experiment.



Obr. 19: Získavanie tvaru oblasti. Červené čiary mieria z určitej vzdialenosti do miesta, kde sa nachádza pivot. Je vidieť, že čiary sú kratšie (zelená oblasť) alebo dlhšie (modrá oblasť) v závislosti od tvaru oblasti a od toho, či čiara náhodou nepreťala iný objekt. Vtedy je potrebné jej vzdialenosť od pivota upraviť.



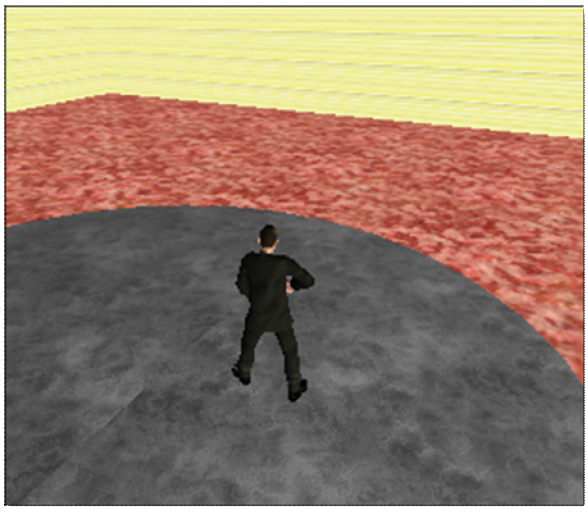
Obr. 20: 2D náhľad na priebeh experimentu. Obsahuje tri obrázky plošiny. Dva horné sú statické. Obrázok vľavo hore zobrazuje dve cieľové oblasti (zelené) a značku (čierna) neviazané k plošine. Zakázaná oblasť by bola červenou farbou. Ďalej obrázok znázorňuje dráhu avatara vzhľadom na miestnosť. Obrázok vpravo hore zobrazuje oblasti a značku viazané k plošine a dráhu avatara vzhľadom na plošinu. Je vidieť, že na obrázku vpravo sa avatar pohyboval len jedným smerom vpred. Na obrázku vľavo však pozorujeme, že miestami zastal a plošina sa s ním zatiaľ otočila. Na treťom obrázku dole je plošina vykresľovaná aktívne, podľa toho, ako sa otáča, spolu s práve aktívnou jednou oblasťou a aktuálnou polohou avatara. Textové pole obsahuje údaje o priebehu experimentu.

4.6 3D náhľad na experiment

Pre tvorbu 3D náhľadu na experiment je použitý Unreal engine. Iné riešenie by znamenalo tvorbu novej 3D aplikácie a nevyužitie možností, ktoré Unreal engine ponúka.

Unreal engine podporuje hru po sieti, a teda umožňuje pripojiť sa k vzdialenému počítaču ako nový hráč alebo ako pozorovateľ. Pre 3D náhľad je použitá druhá možnosť. Pre takéto riešenie bolo potrebné naštudovať hru po sieti, pretože je optimalizovaná na minimalizovanie prenosu informácií. To práve tomuto projektu nevyhovuje. Preto bolo potrebné rozšíriť kód o blok nazývaný „*Replication*“ a špeciálne upraviť funkcie tak, aby boli získané všetky relevantné informácie pre správne zobrazenie experimentu pozorovateľovi. Na obrázku 21 je ukážka 3D náhľadu na experiment.

V projekte je ošetrené, aby sa k nemu nemohli pripojiť noví hráči, ale len pozorovatelia.



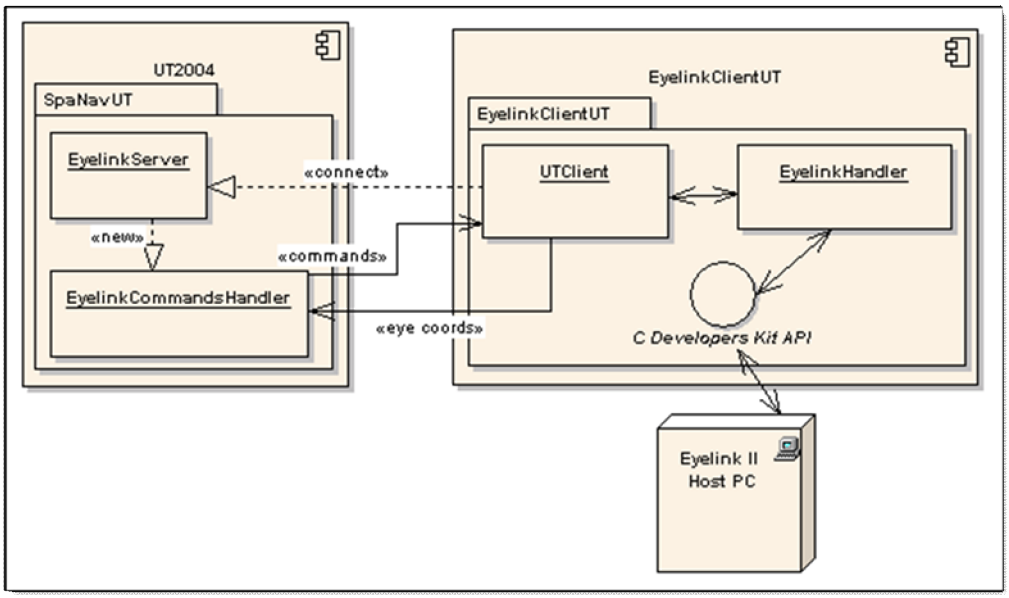
Obr. 21: 3D náhľad na experiment. Umožňuje voľne meniť uhol výhľadu.

4.7 Meranie očných pohybov

Očné pohyby sú merané pomocou prístroja Eyelink II. V kapitole 3.3 sme zvolili, že na komunikáciu medzi Unreal enginom a Eyelinkom použijeme klienta v jazyku C++, ktorý sprostredkuje výmenu informácií.

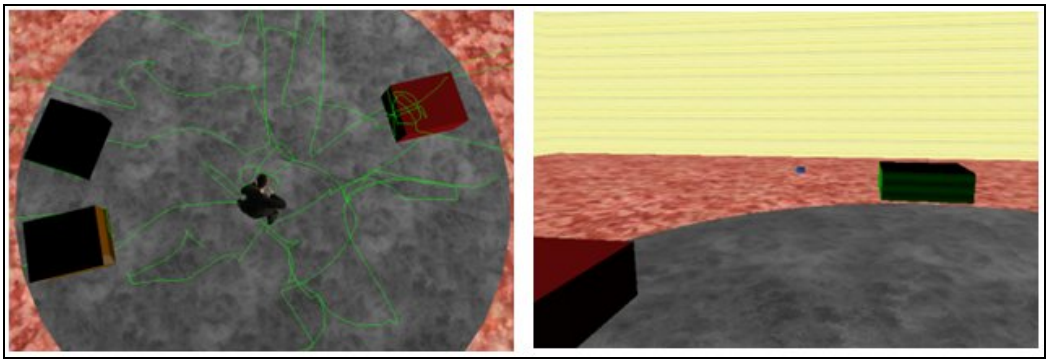
Komunikácia Unreal enginu s okolím bola popísaná v časti 4.2. Rovnaký spôsob je použitý aj pri komunikácii s klientom. V podstate mohol byť použitý rovnaký server, ktorý obsluhuje aplikáciu na administráciu. To by však dovoľovalo merať očné pohyby len v našej grafickej

aplikácii. Aby sa zvýšila modularita tohto softwaru, bol vytvorený samostatný server na obsluhu Eyelink klienta. Ten je do Unreal enginu zakomponovaný ako mutátor (4.1). Výhodou mutátoru je, že môže byť použitý ako nezávislý prídavný modul v akomkoľvek inom type Unreal hry, v ktorej je potrebné sledovať očné pohyby. V prípade, že je mutátor pre Eyelink zapnutý sú 2D aj 3D súradnice zaznamenávané do výstupného súboru. Ak k nemu nie je pripojený Eyelink klient sú 2D súradnice simulované ako stred obrazovky. Komunikácia grafickej aplikácie, klienta a prístroju Eyelink II je schematicky zobrazená na obrázku 22.



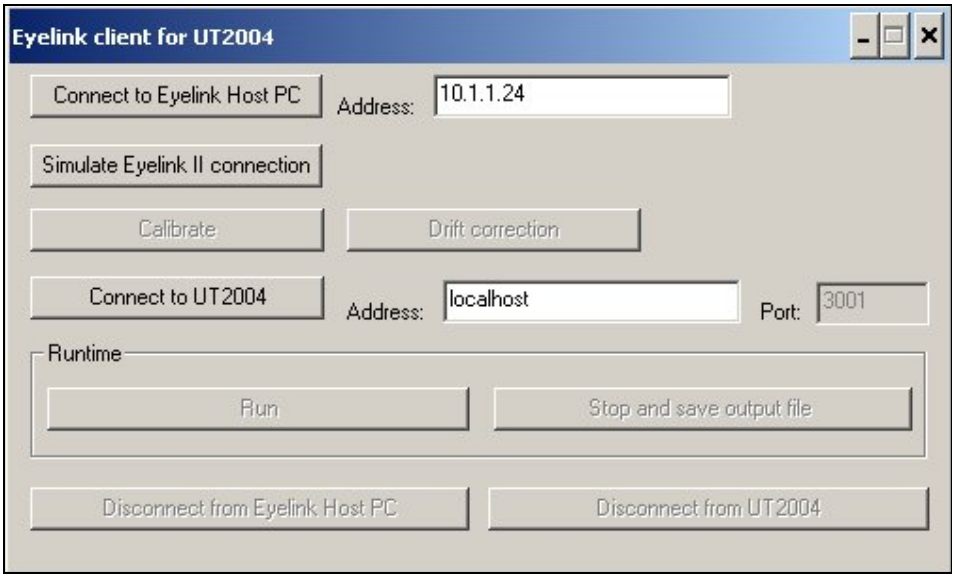
Obr. 22: Schéma komunikácie medzi grafickou aplikáciou, Eyelink klientom a prístrojom Eyelink II. Mutátor vytvára server (*EyelinkServer)*, ktorý beží na porte 3001. Po pripojení klienta k serveru vznikne trieda *EyelinkCommandsHandler*, ktorá prijíma 2D súradnice aktuálneho pohľadu testovaného subjektu od Eyelink klienta. So znalosťou 2D súradníc, aktuálnej pozície a rotácie avatara získame 3D súradnice bodu, kam sa človek pozerá vo virtuálnom prostredí. Hlavné triedy v Eyelink klientovi sú *UTClient*, ktorá komunikuje s Unreal enginom a *EyelinkHandler*, ktorá prostredníctvom Eyelink *C Developers Kit API* komunikuje s programom na ovládanie prístroju Eyelink II na Host počítači.

Súradnice pohľadu je možné zobrazovať počas experimentu, buď pomocou kreslenej čiary, bodu, alebo kombináciou oboch (obr. 23). Rozdiel je v tom, že kreslenú čiaru Unreal engine nevie uložiť na video záznam, ktorý je jedným z výstupných súborov. Aj z tohto dôvodu bol projekt rozšírený o spätnú rekonštrukciu dát nameraných počas experimentu (4.9).



Obr. 23: Zobrazenie očných pohybov vo virtuálnom prostredí. Na obrázku vľavo sú kreslené zelenou čiarou. Na obrázku vpravo je v strede bod, znázorňujúci očné pohyby.

Na manipuláciu s Eyelink klientom je vytvorené jednoduché rozhranie (obr. 24), ktoré umožňuje pripojenie k Eyelink prístroju a k Unreal enginu. Okrem toho obsahuje tlačítka na zahájenie zberu a posielania 2D súradníc a jeho ukončenie. Pri ukončení je možné uložiť výstupný súbor, ktorý vytvára Eyelink prístroj vo formáte EDF. Ďalej rozhranie dovoľuje prístroj kalibrovať alebo vykonať korekciu driftu.



Obr. 24: Eyelink klient. Umožňuje pripojenie k prístroju Eyelink (*Connect to Eyelink Host PC*) a simuláciu pripojenia pre účely testovania (*Simulate Eyelink II connection*). Ďalej sprístupňuje rozhranie pre kalibráciu (*Calibrate*) a pre korekciu driftu (*Drift correction*). Následne je možné sa pripojiť k Unreal enginu (*Connect to UT2004*), spustiť nahrávanie dát (*Run*) a ukončiť ho(*Stop and save output file*).

Jedným z problémov, ktorý vyplýva z použitia prístroja Eyelink, je že musí byť spustený Eyelink klient aj Unreal engine na jednom počítači. Obe tieto aplikácie spotrebúvajú veľké množstvo operačnej pamäti. Výrobcovia Eyelinku dokonca odporúčajú nastaviť jeho priority do režimu „*Realtime*“ a vypnúť všetky ostatné aplikácie, ktoré by mohli počítač zaťažovať.
Pri testovaní sa ukázalo, že aj priorita v režime „*Normal*“ je priveľa a v grafickej aplikácii sa takmer nedá pohybovať. Preto je v Eyelink klientovi automaticky nastavovaná priorita na „*Below normal*“. Hre Unreal musí byť nastavená ručne. Pri výkonnejšom počítači (v porovnaní s počítačom použitom pri experimentoch v kapitole 5) by pravdepodobne neboli potrebné takéto opatrenia.

4.8 Meranie výstupných dát

Výstupnými dátami z experimentu sú namerané hodnoty z virtuálneho prostredia, výsledný obrázok dráhy avatara z 2D mapy a video záznam z Unreal enginu.
Namerané dáta z virtuálneho prostredia môžu byť uložené v rôznych formátoch alebo dokonca v databáze. V pôvodnom prevedení BVA arény boli dáta zaznamenávané do textového súboru. Pri konzultácii možností s neurológmi sa ukázalo, že im pôvodné riešenie vyhovuje, pretože sú naň zvyknutí a dáta si podľa potreby transformujú do tabuliek v Exceli. Preto som sa rozhodla pôvodné riešenie zachovať. Následne sme sa dohodli na tom, ktoré dáta sa budú zaznamenávať.
Dáta sú ukladané do dvoch textových súborov. Prvý, s názvom *General log*, obsahuje informácie týkajúce sa vstupov avatara do jednotlivých oblastí. Pri každom vstupe do oblasti sa do logu zaznamená čas vstupu, meno oblasti, viazanosť na plošinu, typ (zakázaná/cieľová), počet vstupov, čas od predchádzajúceho vstupu, dráha prejdená od predchádzajúceho vstupu. Predchádzajúcim vstupom sa pri cieľovej oblasti myslí vstup do akejkoľvek cieľovej oblasti, ale pre zakázanú oblasť je to vstup do jednej konkrétnej oblasti. Takéto rozlíšenie bolo navrhnuté preto, lebo cieľové oblasti sa väčšinou hľadajú v určitom poradí a zaujíma nás ako človek postupoval. Naopak zakázaná oblasť býva konkrétna, preto nás zaujímajú dáta len vzhľadom na ňu. Ďalej sa podľa typu oblasti zaznamenáva: pre cieľovú oblasť vzdialenosť od stredu oblasti pri vstupe a minimálna vzdialenosť k dosiahnutiu stredu oblasti (od pozície predchádzajúceho vstupu), a pre zakázanú oblasť čas strávený v oblasti a vzdialenosť prejdená v oblasti. Meranie

prejdenej vzdialenosti pre oba typy oblastí sa rozlišuje nasledovne. Ak je oblasť viazaná na plošinu, tak sa merajú prejdené kroky avatara, teda ako keby sa plošina nepohybovala. Ak je oblasť statická v miestnosti, meria sa celková vzdialenosť (tzn. kroky + pohyb plošiny). *General log* je vytváraný len jeden , pre všetky experimenty s pacientom.

Druhý výstupný súbor, s názvom *Track log*, obsahuje dáta z celého priebehu experimentu. Po konzultácii a neurológmi som rozhodla, že bude obsahovať nasledujúce výstupy: čas, pozícia avatara vzhľadom na miestnosť, pozícia avatara vzhľadom na plošinu, uhol otočenia plošiny, uhol výhľadu avatara, stlačený funkčný kláves, zakázaná oblasť (ak sa v nej avatar nachádza), práve hľadaná cieľová oblasť, viazanosť na plošinu práve hľadanej cieľovej oblasti, zoznam všetkých cieľových oblastí s počtom vstupov do nich, 2D a 3D súradnice pohľadu. Tieto údaje sú zaznamenávané s určitou časovou frekvenciou, ktorú je možné nastaviť v konfiguračnom skripte pre každý experiment zvlášť.

Oba výstupné súbory majú údaje rozdelené do stĺpcov oddelených tabulátorom. V textovej podobe takéto riešenie pôsobí neprehľadne, ale umožňuje jednoduchú konverziu do tabuliek v Exceli. Ukážky general a track logu sú priložené na CD.

Textové súbory a 2D mapa sa ukladajú do výstupného adresára zvoleného v rozhraní aplikácie na administráciu. Video záznam je možné nájsť v adresári UT2004\ Demo, kam je ukladaný Unreal enginom. Z dôvodu jednoznačného rozlíšenia je jeho názov rozšírený o časovú známku. Je ho možné prehrať pomocou zabudovaného prehrávaču, ktorý má voľbu voľného pohybu kamery po scéne počas prehrávania.

4.9 Prehrávanie experimentu

Súčasťou zadania je možnosť zaznamenávať priebeh experimentu vo virtuálnej realite tak, aby sa dal spätne prehrať. Existuje niekoľko možností, ako tento problém riešiť.

Prvou je využiť nahrávanie videa, ktoré bolo spomenuté v predchádzajúcej časti. Nevýhodou tohto riešenia je, že s každou novou skompilovanou verziou grafickej aplikácie už nie je možné staré videá prehrať. Pri bližšom testovaní výsledkov nahrávania bolo tiež zistené, že niektoré objekty sú občas odlišne škálované v porovnaní s ich rozmermi v pôvodnej mape. Obe tieto nevýhody nie sú závažné a okrem toho sú napevno zabudované v Unreal engine, takže ich nie je možné opraviť. Okrem toho sa vyskytol ďalší problém: očné pohyby vykreslené pomocou čiar nie sú na video zaznamenávané. Aby boli čiary alebo bod zaznamenávané, musia byť počas experimentu viditeľné. Na testovaného človeka tak môžu

pôsobiť ako rozptyľujúci a rušivý element. Je preto potrebné nájsť iný spôsob, ako očné pohyby zaznamenať.

Jednoduchým riešením by bolo nastaviť bod viditeľný len v okne 3D náhľadu na experiment a nahrávať video tam. Stále však chýba zaznamenávanie kreslených čiar, pomocou ktorých je lepšie porovnateľné, ktoré miesta testovaný človek najčastejšie využíval ako orientačné body.

Druhou možnosťou ako zaznamenávať očné pohyby je „vložiť“ ich do videa. To je možné pomocou využitia HUDu (Head-Up Display). HUD je v podstate priesvitná obrazovka, ktorú má užívateľ stále pred sebou a ktorá sa pohybuje spolu s ním. V Unreal engine sa používa na zobrazovanie informácií, napríklad skóre hráča, jeho zdravie alebo počet životov. Očné pohyby môžu byť na HUD vykresľované pomocou čiar tak, že by sa s každým otočením hlavy premazali a vykreslili vždy nanovo podľa aktuálnych údajov. Nevýhodou je, že k zmazaným čiaram by sa nedalo vracať, keďže Unreal nemá možnosť prehrávať video dozadu.

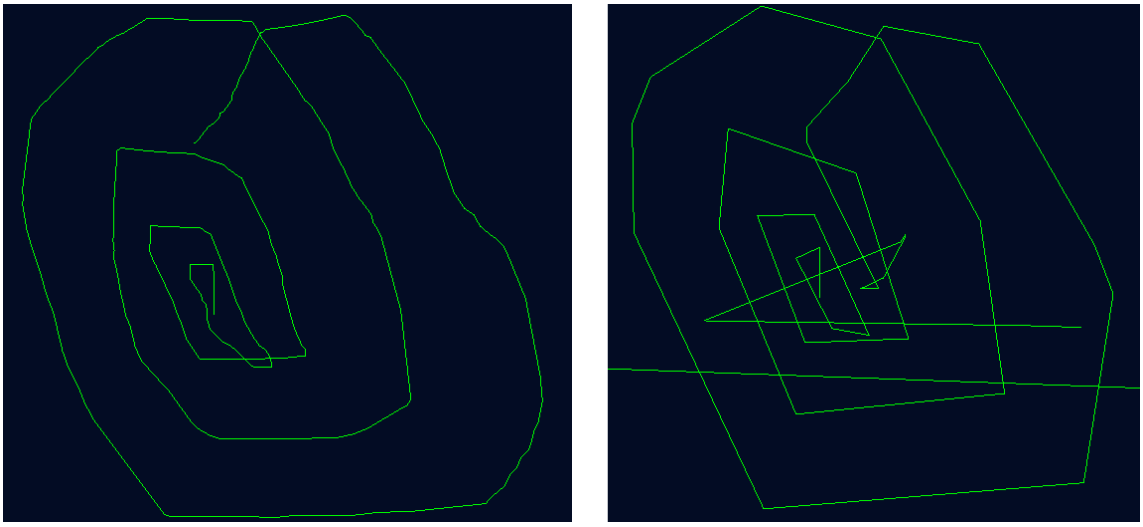
Treťou možnosťou je prehrať dáta bez použitia videozáznamu. Súradnice očných pohybov sú uložené v súbore *track log,* ktorý je vytváraný pre každý experiment. Obsahuje čas, pozíciu avatara, jeho uhol otočenia, otočenie plošiny aj očné pohyby. Všetky tieto dáta môžu byť využité na spätnú rekonštrukciu priebehu experimentu. Užívateľ sa musí len pripojiť ku grafickej aplikácii a vybrať *track log* súbor, ktorý chce rekonštruovať.

Pri rekonštrukcii sa čiary kreslia podľa toho, ako sa avatar pohybuje (červenou farbou) a podľa očných pohybov (zelenou farbou). Simuluje sa aj otáčanie tela, uhol výhľadu a otáčanie plošiny.

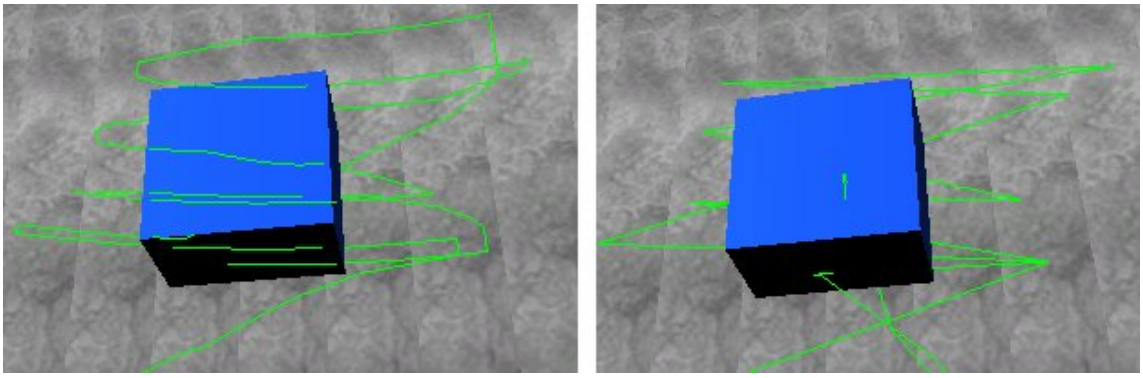
Rekonštrukciu možno kedykoľvek pozastaviť a užívateľ sa môže poprechádzať po scéne a prezrieť si postupný vývoj kreslenia čiar v jednotlivých miestach. Všetky použité oblasti v mape sú počas rekonštrukcie nastavené ako viditeľné.

Rekonštrukcia až na pár detailov (aktivovanie a deaktivovanie oblastí, zaznamenávanie vstupov a podobne) simuluje priebeh experimentu. To navodzuje otázku, prečo by sme nemohli zároveň aktívne vykresľovať aj 2D obrázok, ako tomu je počas normálneho priebehu experimentu. Samozrejme jeden obrázok už je uložený ako výstupný, ale pre užívateľa môže byť užitočná možnosť opätovne si prehrať ako obrázok vznikol. Počas rekonštrukcie teda nasimulujeme tok informácii do 2D mapy tak, aby „mala pocit“, že prebieha klasický experiment a vykresľovala ho, až na to, že na konci obrázok neuloží.

Na rozdiel od nahraného videozáznamu sme teda získali aj čiary, možnosť voľného pohybu a opätovné vykresľovanie 2D mapy. Na druhej strane toto riešenie má aj chybu. Je ňou sekanie obrazu pri vykresľovaní, v závislosti od frekvencie, s ktorou časovač obnovuje dáta prostredia (to znamená s akou frekvenciou sme ich zaznamenávali do *track logu*). S frekvenciou vykresľovania súvisí aj rozdiel medzi kreslením čiar očných pohybov počas experimentu a počas rekonštrukcie. Počas experimentu sa jednotlivé súradnice menia a spájajú do čiar s každým prekreslením scény. Pri rekonštrukcii máme zaznamenané hodnoty len s určitou, vo väčšine prípadov nižšou frekvenciou. Preto napríklad čiara počas experimentu vykreslená na povrchu nejakého objektu, môže v rekonštrukcii obsahovať len dva body, ktorých spojenie vedie poza objekt alebo cez objekt. Tento rozdiel je názorne ukázaný na obrázkoch 25 a 26.



Obr. 25: Porovnanie vykreslenia čiar, ktoré znázorňujú očné pohyby. Obrázok vľavo je zachytený počas priebehu experimentu, kedy sa dáta vykresľovali s najvyššou možnou frekvenciou, v tomto konkrétnom prípade každých 0.016 sekundy. Obrázok vpravo je zachytený počas rekonštrukcie *track logu*, do ktorého boli dáta zaznamenávané s frekvenciou 0.5 sekundy. Na obrázku vpravo tiež vidíme čiary, nenachádzajúce sa na obrázku vľavo. Je to spôsobené rýchlymi očnými pohybmi a spojením dvoch bodov čiarou, ktorá pôvodne viedla cez viac bodov inou cestou.



Obr. 26: Porovnanie vykreslenia čiar znázorňujúcich očné pohyby na povrchu oblasti. Obrázok vľavo je zachytený počas priebehu experimentu (frekvencia vykresľovania každých 0.016 sekundy). Obrázok vpravo je zachytený počas rekonštrukcie *track logu* (frekvencia raz za 0.5 sekundy). Na obrázku je vidno, že čiary pri rekonštrukcii prechádzajú cez objekt a na povrchu na neobjavia.

Z uvedeného vyplýva: ak chceme získať kvalitnú rekonštrukciu experimentu je potrebné zaznamenávať dáta do track logu s čo najvyššou možnou frekvenciou.

5 Testovanie aplikácie

V takmer závere tejto práce stojíme pred úlohou celý software otestovať. To zahŕňa viacero činností. V prvom rade je to základné funkčné testovanie, počas ktorého je vyskúšaná každá funkcia samostatne. Takéto testovanie bolo vykonané už počas samotného vývoja aplikácie, hoci vstupmi boli zväčša len očakávané hodnoty. Druhým stupňom je doménové testovanie, pri ktorom množinu vstupov tvoria vzorky všetkých hodnôt, ktoré užívateľ môže zadať. Výstupom z tejto fázy sú tabuľky v prílohe programátorskej dokumentácie. Ďalším typom testovania je kontrola splnenia požiadaviek uvedených v špecifikácii. V našom prípade sa hlavne potrebujeme uistiť, či dáta zaznamenané v logoch alebo v 2D mape skutočne zodpovedajú požadovaným dátam. Pri práci s virtuálnou aplikáciou je to náročnejšia úloha a niektoré hodnoty je možné overiť len pozorovaním (napríklad dráha avatara v 2D mape alebo prejdená vzdialenosť v niektorej oblasti, kedy sa počas testovania v oblasti môžeme zastaviť, rýchlo ňou viac krát prebehnúť, alebo sa jej len zľahka dotknúť). Ďalším typom je rizikové, alebo záťažové testovanie, pri ktorom sa zamyslíme nad možnosťami, alebo situáciami, v ktorých by mohol program zlyhať a tie otestujeme. Prípadne sa snažíme určiť limit, ktorý už nie je únosný. Pri práci s prístrojom Eyelink II napríklad bolo zistené, že sa veľmi rýchlo (približne po 2 až 10-tich minútach) rozkalibruje, čo znamená, že prístroj sníma súradnice pohľadu s určitým posunom. Môže to byť kvôli posunu čelenky, ktorý vzniká tým, že pri prechode virtuálnym svetom testované osoby hýbu hlavou so snahou vidieť „za roh“. Tento problém je riešený pridaním možnosti rýchleho prepnutia do režimu korekcie driftu, pomocou ktorého zistíme, či je potrebné prístroj znovu kalibrovať. Možnou prevenciou voči posunu čelenky je použitie fixačného nástavca, ktorý zabráni posunu hlavy. Jeho nevýhoda je nepohodlnosť pre testovanú osobu.

Aby bolo testovanie kompletné, je vhodné vytvoriť scenáre jedného, alebo viacerých experimentov, ktoré bude skupina užívateľov prechádzať. Problémom v tejto časti je navrhnúť okruh experimentov tak, aby rozumne pokrývali potencionálnu množinu testov, pre ktoré bude aplikáciou v budúcnosti používaná.

Opýtala som sa teda pána neurológa, ktorý v dobe písania tejto kapitoly už s aplikáciou pracuje, aké typy experimentov plánuje robiť, či už v kratšom alebo dlhšom časovom horizonte. Zo všetkých testov je vhodné zvoliť niekoľkých špecifických zástupcov. Jeden by mal reprezentovať typický test, napríklad práve v tejto chvíli pripravovaný. Druhý, by mohol

byť menej typický, ale tiež je o ňom v blízkej budúcnosti uvažované. Tretí by sa mohol celkovo vymykať z pôvodného kontextu testovacej úlohy v prostredí BVA arény.
Konkrétny popis a prevedenie týchto experimentov je uvedené v nasledujúcich troch podkapitolách.

5.1 Experiment č.1: Štyri ciele na otáčajúcej sa plošine

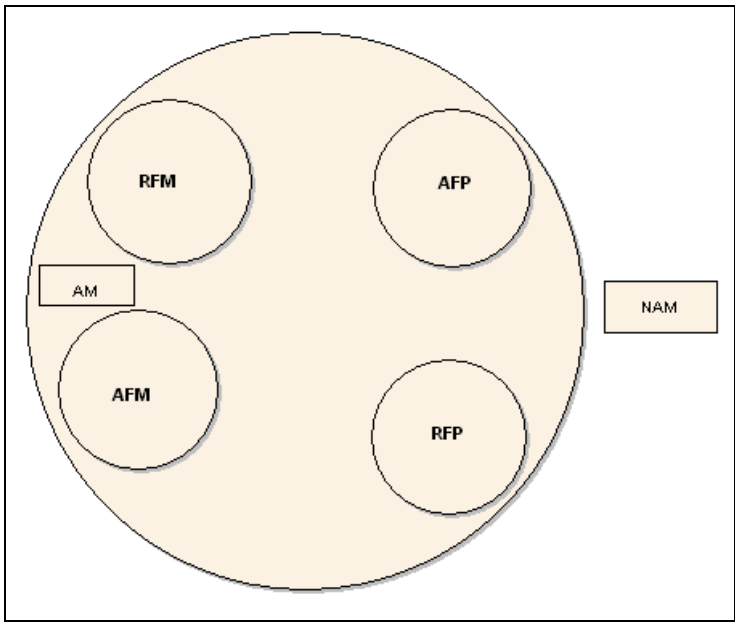
Popis experimentu:

Tento test je rekonštrukciou reálneho experimentu, ktorý prebiehal v BVA aréne. Umožňuje sledovať schopnosť ľudí oddeliť statickú časť miestnosti od točiacej sa plošiny.
Experiment prebieha na otáčajúcej sa plošine, na ktorej sú umiestnené štyri neviditeľné kruhové oblasti. Dve sa otáčajú spolu s plošinou (AF) a dve sa neotáčajú (RF). Okrem nich aréna obsahuje dve orientačné značky. Jedna značka je na podlahe a otáča sa spolu s ňou, druhá je statická na stene stanu. Dve kruhové oblasti sú situované v blízkosti značky umiestnenej na podlahe (AFM a RFM) a ďalšie dve sú niekde ďalej od nej (AFP a RFP). Úlohou testovaného človeka je naučiť sa chodiť medzi týmito oblasťami v danom poradí:
1. AFM, 2. RFP, 3. RFM, 4. AFP, ... atď. dookola.
To, že do niektorej oblasti vstúpil zistí pomocou dvojsekundového zvukového signálu, ktorý zaznie v prípade, že daná oblasť je aktívna, teda že je práve na rade, aby ju človek hľadal. Hodnotí sa čas chôdze a dráha prejdená medzi jednotlivými oblasťami (má sa chodiť čo najpriamejšou a najrýchlejšou cestou). Tieto hodnoty je možné získať priamo z *general logu*.
Čím kratší čas a prejdenú vzdialenosť osoba dosiahne, tým lepšie. Experiment končí, keď bola každá oblasť navštívená práve dva krát.
Statické aj viazané oblasti sú viac-menej diagonálne oproti sebe, takže stačí pamätať si polohu dvoch a ich vzťah k plošine. Táto pomôcka nebola testovaným ľudom prezradená.

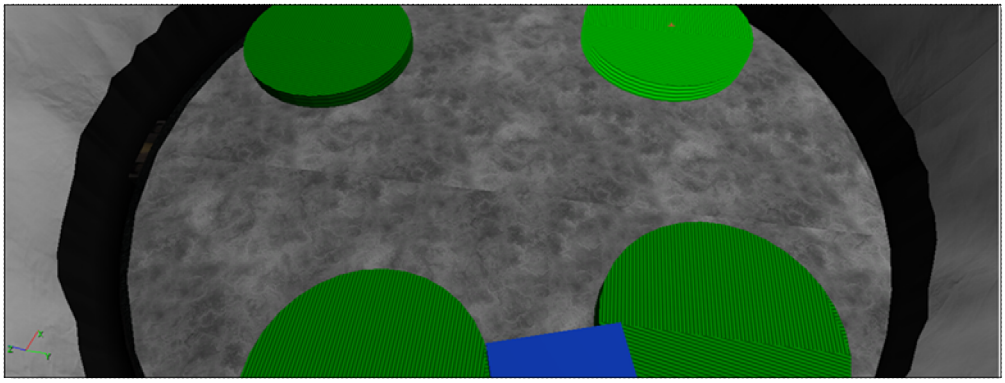
Návrh experimentu:

Na obrázku 27 na nasledujúcej strane je schéma ukazujúca rozmiestnenie vyššie popísaných objektov v aréne. Kruhové cieľové oblasti sú reprezentované objektom typu *„Preference“*, pretože chceme získať merania medzi jednotlivými oblasťami. To znamená chôdzu a čas od jedného miesta k druhému (od AFM, ku RFP, ...). Pri zvolení typu *„Avoidance“*, by sme merali hodnoty pre konkrétne miesto, napríklad čas od vstupu do AFM až po opätovný vstup

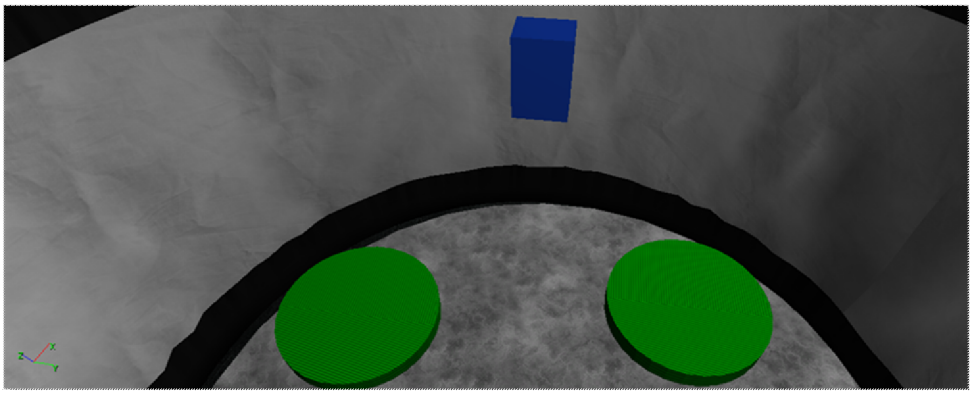
do AFM. Na obrázkoch 28 a 29 na nasledujúcej strane je prostredie experimentu vytvorené v UnrealEd editore (toto prostredie je pomenované TEST-Search4Areas a je súčasťou CD priloženého k tejto práci). V mape sú použité objekty *SpaNav_Platform*, *AimArea* a *Mark*. Všetkým objektom sú priradené mená tak, ako je to znázornené na obrázku 27 (mená sa priradzujú v editore spôsobom zobrazeným na obr. 13).



Obr. 27: Rozmiestnenie objektov v aréne. Schéma obsahuje kruhovú plošinu, na ktorej sa nachádzajú štyri kruhové cieľové oblasti. Tie sa buď otáčajú spolu s plošinou (AFM, AFP), alebo sú statické (RFM, RFP). Ďalej je na plošine umiestnená značka AM, ktorá sa s ňou otáča a mimo plošiny (na stene stanu) je statická značka NAM.



Obr. 28: Testovacie prostredie TEST-Search4Areas. Pohľad na cieľové oblasti RFM, AFM a značku AM v popredí a na oblasti RFP a AFP v pozadí.



Obr. 29: Testovacie prostredie TEST-Search4Areas. Pohľad na cieľové oblasti RFP, AFP a na statickú značku NAM na stene stanu.

Ovládanie experimentu v tomto prípade zahrňuje hlavne prepínanie poradia aktivity cieľových oblastí. K tomuto bol vytvorený konfiguračný skript Search4areas.js, ktorého zdrojový kód je v nasledujúcom odstavci.

```
function init() {
	//inicializácia mapy
	experiment.setMap("TEST-Search4Areas");
}

function run() {
	//tento blok sa vyhodnotí len prvý krát pri spustení experimentu,
	//používa sa na inicializáciu premenných
	if (experiment.isStarted()){
		experiment.setCollisionCylinder(20,88);
		experiment.setWalk(true);
		experiment.setTrackRate(0.3);
		experiment.setPlayerSpeed(280);

		//nasledujúce inicializačné nastavenia sú tu uvedené z dôvodu názornosti,
		//ale môžu byť zvolené priamo v UnrealEd editore
		preference.get("AFM").setActive(true);
		preference.get("AFM").setVisible(false);
		preference.get("AFM").setAttached(true);

		preference.get("RFP").setActive(false);
		preference.get("RFP").setVisible(false);
		preference.get("RFP").setAttached(false);

		preference.get("RFM").setActive(false);
		preference.get("RFM").setVisible(false);
		preference.get("RFM").setAttached(false);

		preference.get("AFP").setActive(false);
		preference.get("AFP").setVisible(false);
		preference.get("AFP").setAttached(true);

		mark.get("AM").setVisible(true);
		mark.get("AM").setAttached(true);
		mark.get("NAM").setVisible(true);

		//plošina rotuje dlhú dobu rýchlosťou 5 stupňov za sekundu
		//v protismere hodinových ručičiek
		platform.get("plosina").doRotateTime(10000,5,-1);
	}
	//pri vstupe do každej oblasti ju deaktivuj spusti zvukový signál a aktivuj
	//nasledujúcu oblasť v poradí
```

```
        if (preference.get("AFM").entered()){
                preference.get("AFM").setActive(false);
                preference.get("AFM").beep(2);
                preference.get("RFP").setActive(true);
        }
        if (preference.get("RFP").entered()){
                preference.get("RFP").setActive(false);
                preference.get("RFP").beep(2);
                preference.get("RFM").setActive(true);
        }
        if (preference.get("RFM").entered()){
                preference.get("RFM").setActive(false);
                preference.get("RFM").beep(2);
                preference.get("AFP").setActive(true);
        }
        if (preference.get("AFP").entered()){
                preference.get("AFP").setActive(false);
                preference.get("AFP").beep(2);
                preference.get("AFM").setActive(true);
        }
        //pri treťom vstupe do poslednej oblasti experiment skončí
        if (preference.get("AFP").enteredTimes(3)){
                experiment.setStop();
        }
}
```

Výsledné dáta:

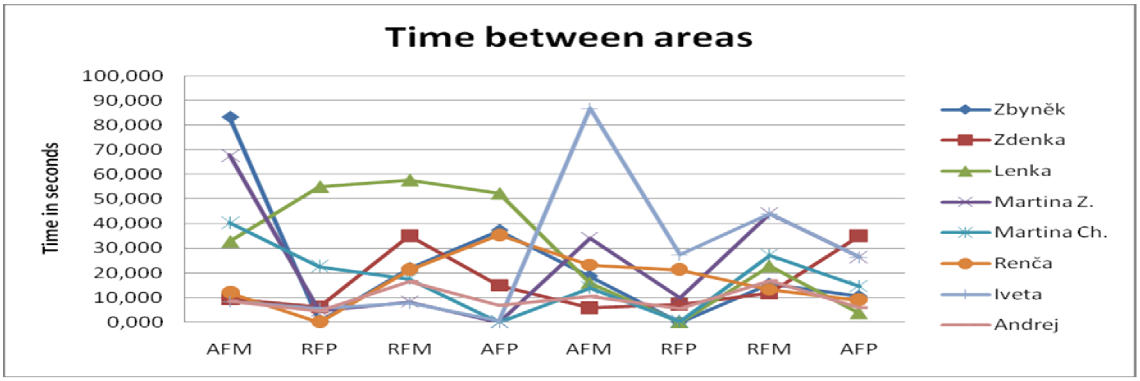
Test prebiehal tak, že testovaným osobám bol najprv ukázaný nákres rozmiestnenia oblastí v aréne a vysvetlený ich vzťah k plošine (oblasť je viazaná alebo statická). Všetkým bol poskytnutý určitý čas na uvedomenie si relácií medzi objektmi. V nasledujúcich výsledných tabuľkách je v tab. 3 zachytený čas chôdze testovaných ľudí medzi jednotlivými oblasťami (v sekundách) a v tab. 4 je uvedená dráha prejdená medzi jednotlivými oblasťami (v cm). Pri meraní prejdenej dráhy sa rozlišuje nasledovné: ak je oblasť viazaná na plošinu, meria sa vzdialenosť prejdená vzhľadom k plošine (teda ako keby sa plošina neotáčala), ak je oblasť statická, meria sa celková prejdená vzdialenosť (pohyb človeka + pohyb plošiny). Hodnoty z tabuliek sú graficky znázornené na obrázkoch 30 a 31 na nasledujúcej strane.

A/N	Zbyněk	Zdenka	Lenka	Martina Z.	Martina Ch.	Renča	Iveta	Andrej
AFM	83.256	9.406	32.703	67.875	40.546	12.087	8.312	8.593
RFP	0.437	6.375	55.000	4.546	22.750	0.000	5.453	4.546
RFM	21.828	34.984	57.718	8.187	17.703	21.375	7.718	16.343
AFP	37.265	15.000	52.281	0.000	0.000	35.468	0.453	6.828
AFM	18.640	5.921	15.906	34.093	14.109	23.171	86.812	10.437
RFP	0.000	7.265	0.000	9.984	0.453	21.359	27.281	5.453
RFM	15.453	11.812	22.734	44.109	27.281	13.171	44.078	16.828
AFP	10.453	35.000	3.640	26.359	14.531	9.109	26.375	5.906
Priemer	23.417	15.720	29.998	24.394	17.172	16.968	25.810	9.367

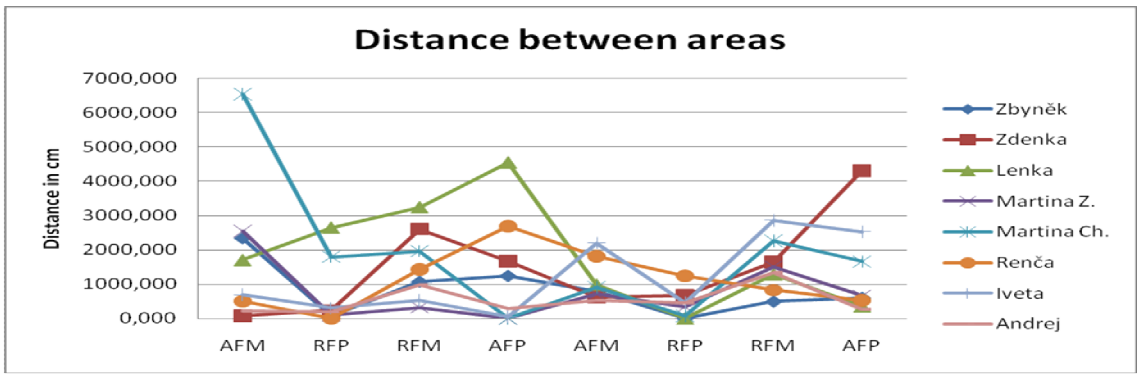
Tab. 3: Čas chôdze medzi jednotlivými oblasťami (v sekundách). Stĺpce tabuliek zodpovedajú údajom nameraným u jednotlivých testovaných osôb. Riadky zachytávajú nameraný čas chôdze od jednej oblasti k druhej v poradí tak, ako do nich bolo vstupované. V prípade prvej oblasti, je to čas od začiatku experimentu po vstup do oblasti. Posledný riadok obsahuje priemerný čas chôdze medzi oblasťami pre každého človeka.

A/N	Zbyněk	Zdenka	Lenka	Martina Z.	Martina Ch.	Renča	Iveta	Andrej
AFM	2336.922	74.126	1705.963	2561.358	6534.432	502.604	694.264	215.562
RFP	81.477	229.182	2654.511	84.905	1792.767	0.000	305.229	196.075
RFM	1070.198	2604.038	3251.459	322.132	1960.144	1429.720	523.855	973.758
AFP	1239.002	1674.671	4555.136	0.000	0.000	2702.441	53.309	291.817
AFM	781.720	619.911	987.509	711.380	910.110	1819.647	2195.619	538.709
RFP	0.000	669.530	0.000	343.022	76.894	1245.266	453.214	459.937
RFM	490.929	1649.697	1286.484	1518.391	2264.206	830.765	2854.920	1342.492
AFP	599.053	4307.415	352.027	666.133	1657.725	525.898	2522.823	269.414
Primer	824.931	1478.571	1849.136	775.915	1899.535	1132.043	1200.404	535.971

Tab. 4: Dráha prejdená medzi jednotlivými oblasťami (v cm). Stĺpce tabuliek zodpovedajú údajom nameraným u jednotlivých testovaných osôb. Riadky zachytávajú nameranú dráhu chôdze od jednej oblasti k druhej v poradí tak, ako do nich bolo vstupované. V prípade prvej oblasti, je to dráha chôdze od začiatku experimentu po vstup do oblasti. Posledný riadok obsahuje priemernú dráhu medzi oblasťami pre každého človeka.



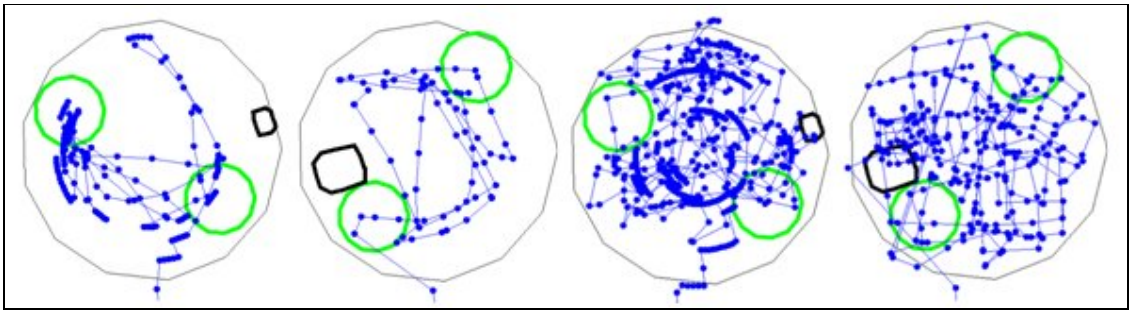
Obr. 30: Graf znázorňujúci čas chôdze medzi jednotlivými oblasťami (v sekundách) počas experimentu číslo 1. Na vodorovnej osi sú uvedené mená oblastí v poradí tak, ako do nich bolo vstupované. Na osi *Time in seconds* je uvedený čas chôdze medzi jednotlivými oblasťami. Každá čiara v grafe zobrazuje jednu testovanú osobu.



Obr. 31: Graf znázorňujúci dráhu prejdenú medzi jednotlivými oblasťami (v cm) počas experimentu číslo 1. Na vodorovnej osi sú uvedené mená oblastí v poradí tak, ako do nich bolo vstupované. Na osi *Distance in cm* je uvedená dráha prejdená medzi jednotlivými oblasťami. Každá čiara v grafe zobrazuje jednu testovanú osobu.

Na grafoch na obrázkoch 30 a 31 je možné pozorovať, že čas aj prejdená vzdialenosť majú u väčšiny testovaných osôb tendenciu klesať pri každej druhej navštívenej oblasti. Teda pri oblastiach RFP a AFP. Čo majú tieto dve oblasti spoločné? Obe boli na začiatku experimentu bližšie k značke umiestnenej na stene. Naproti tomu sa líšia tým, že RFP je statická a AFP sa otáča spolu s plošinou. Preto sa nedá z tohto pozorovania vyvodzovať, či sa osobám lepšie pamätala poloha statických alebo viazaných oblastí. Môžeme však usúdiť, že lepšie ako orientačný bod poslúžila statická značka na stene.

V priemere najnižší čas a aj prejdenú dráhu medzi jednotlivými oblasťami má Andrej. Najvyšší priemerný čas má Lenka, jej priemerná dráha je druhá najvyššia. Pozrime sa preto na obrázok 32, ktorý ukazuje, akú stratégiu pohybu v aréne si zvolili. Prvé dva obrázky znázorňujú pohyb Andreja. Jeden zachytáva celkový pohyb a statické oblasti. Druhý zachytáva pohyb len vzhľadom na plošinu a oblasti k nej viazané. Druhé dva obrázky patria Lenke. Je možné vidieť, že Andrej volil premyslené priame pohyby. Naopak Lenka sa len chaoticky premiestňovala v snahe nájsť nejakú oblasť. Toto pozorovanie podporuje domnienku, že rýchlosť a priemerná prejdená dráha pri presunoch môže u testovaného subjektu odrážať jeho znalosť o rozmiestnení objektov v aréne.



Obr. 32: Porovnanie stratégie pohybu najrýchlejšej a najpomalšej osoby počas experimentu č.1. Popis obrázkov v poradí zľava doprava: 1. Celkový pohyb Andreja a statické oblasti v miestnosti. 2. Pohyb Andreja vzhľadom na plošinu a oblasti k nej viazané. 3. Celkový pohyb Lenky a statické oblasti v miestnosti. 4. Pohyb Lenky vzhľadom na plošinu a oblasti k nej viazané.

Experiment bol testovanými osobami hodnotený ako náročný a ťažko naučiteľný. Kritizovaný bol malý priestor virtuálneho stanu, v ktorom bolo potrebné často sa otáčať, čím osoba strácala prehľad o aktuálnej polohe neviditeľných oblastí na plošine.

5.2 Experiment č.2: Hľadanie oblasti na presnosť

Popis experimentu:

Cieľom tohto experimentu je u testovanej osoby skúmať presnosť odhadu polohy vzdialenej oblasti, a to z rôznych východiskových bodov v miestnosti. Na rozdiel od predchádzajúceho experimentu sa plošina neotáča a je väčších rozmerov. Je na nej umiestnená jedna cieľová a jedna zakázaná oblasť, obe sú na začiatku viditeľné. Testovaný človek má určitý čas na to, aby si zapamätal ich polohu. Orientovať sa môže pomocou dvoch značiek, pričom jedna je na podlahe a druhá na stene. Ďalšími značkami v aréne sú štyri štartovacie pozície. Úlohou človeka je pokúsiť sa z každej štartovacej pozície prísť čo najbližšie k neviditeľnej cieľovej oblasti. Keď si bude myslieť, že už je v nej, tak stlačí kláves „a“. Počas hľadania cieľovej oblasti by sa mal snažiť vyhnúť zakázanej, ktorá pri vstupe vydá výstražný signál, čo mu umožňuje naučiť sa jej polohu aj v priebehu experimentu.

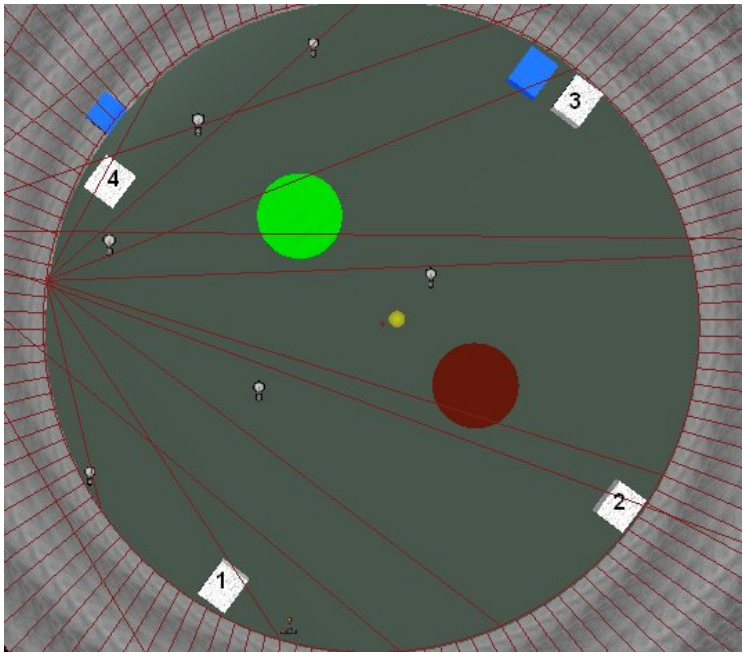
Experiment sa odkláňa od pôvodnej myšlienky BVA arény tým, že nevyužíva otáčanie sa plošiny a zameriavame sa na odhadovanie polohy objektu vo väčšom prostredí. V reálnom prostredí BVA by nebol možný, pretože je príliš malé. Tu má modulárne virtuálne prostredie výhodu. Okrem toho je netypický tým, že nás zaujíma presnosť priblíženia sa k oblasti, teda poloha subjektu v čase stlačenia klávesu „a“ vzhľadom na cieľovú oblasť. Tieto dáta nie sú počas experimentu nikam zaznamenávané, pretože náš projekt nepočítal s takýmto typom výstupu. Zo znalosti polohy oblasti a subjektu je však možné jednoducho vzdialenosť dopočítať.

Návrh experimentu:

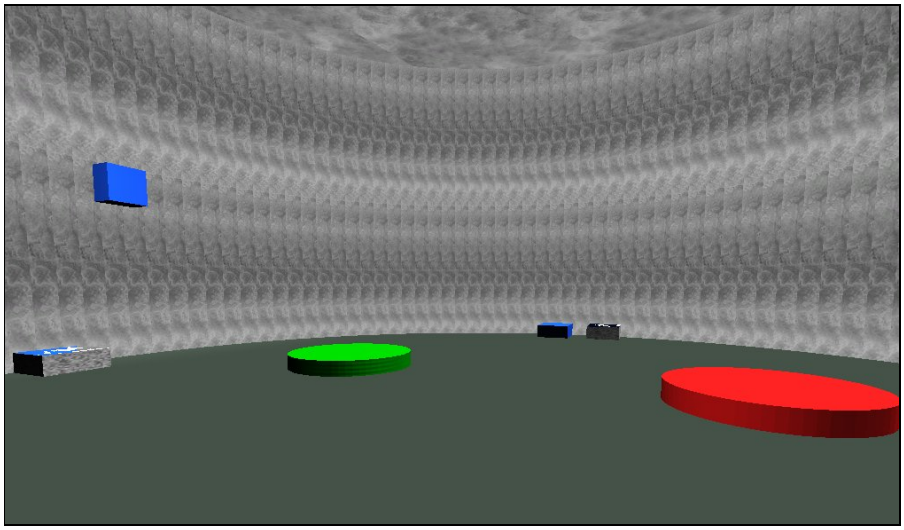
Na obrázkoch 33 a 34 na nasledujúcej strane je znázornené rozmiestnenie objektov v aréne. Zelenou farbou je označená cieľová oblasť typu *Preference,* červenou zakázaná oblasť typu *Avoidance.* Dve orientačné značky (typ *Mark*) majú modrú farbu a ďalšie štyri biele značky označujú štartovacie pozície. Toto prostredie je pomenované TEST-DifferentStarts a je súčasťou CD priloženého k tejto práci.

Konfiguračný skript je v tomto prípade veľmi jednoduchý. Stačí si rozmyslieť, ako chceme, aby experiment prebiehal. Nepotrebujeme riešiť žiadne reakcie na udalosti, len občas zasiahnuť jeho chodu podľa situácie. Na to sa nám hodí, ak si zadefinujeme pár klávesov na ovládanie. Konkrétne to bude kláves „h“ na zviditeľnenie alebo naopak skrytie cieľovej a zakázanej oblasti. Ďalej klávesy „1“, „2“, „3“ a „4“, pomocou ktorých zviditeľníme vždy

jednu určitú štartovaciu pozíciu a ostatné skryjeme. Do skriptu pripíšeme aj prázdnu podmienku s klávesom „a“, čím zabezpečíme, že jeho stlačenie bude zaznamenané do *track logu*.



Obr. 33: Pohľad zhora na testovacie prostredie TEST-DifferentStarts. Obsahuje plošinu, cieľovú (zelená) a zakázanú (červená) oblasť. Ďalej dve orientačné značky (modrá) a štyri štartovacie pozície (biela). Štartovacie pozície sú očíslované v poradí, v akom z nich bude testovaný subjekt štartovať. Na obrázku je tiež možné si všimnúť rozmiestnenie svetiel (počas priebehu experimentu nie sú viditeľné).



Obr. 34: Pohľad na prostredie mapy TEST-DifferentStarts z perspektívy testovaného človeka. Môžeme vidieť cieľovú a zakázanú oblasť. Dve orientačné značky a dve štartovacie pozície.

Všetky oblasti a značky sú v UnrealEd editore nastavené ako viditeľné a zakázaná a cieľová oblasť ako aktívne, takže to do konfiguračného skriptu už nie je potrebné písať.
V nasledujúcom odstavci sa nachádza kód skriptu. Je na priloženom CD je pomenovaný DifferentStarts.js

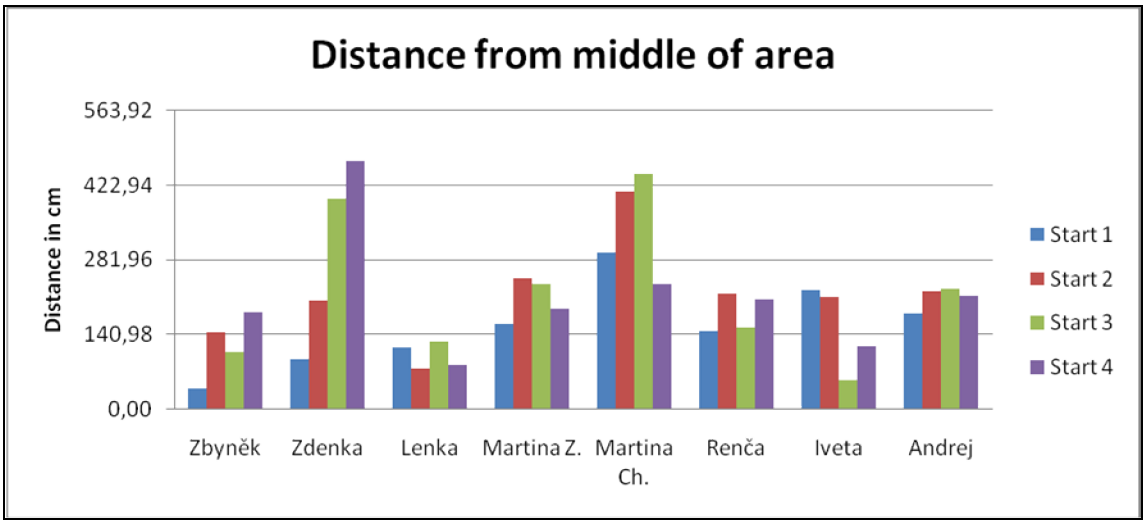
```
function init(){
	//incializácia príslušnej mapy
	experiment.setMap("TEST-DifferentStarts");
}

function run(){

	//prázdna podmienka pre kláves
	if(key.pressed("a")){
	}

	//pri stlačení „1", „2", „3" a „4" chceme zviditeľniť len jednu štartovaciu pozíciu
	if(key.pressed("1")){
		mark.get("start4").setVisible(false);
		mark.get("start3").setVisible(false);
		mark.get("start2").setVisible(false);
		mark.get("start1").setVisible(true);
	}

	if(key.pressed("2")){
		mark.get("start4").setVisible(false);
		mark.get("start3").setVisible(false);
		mark.get("start1").setVisible(false);
		mark.get("start2").setVisible(true);
	}

	if(key.pressed("3")){
		mark.get("start4").setVisible(false);
		mark.get("start1").setVisible(false);
		mark.get("start2").setVisible(false);
		mark.get("start3").setVisible(true);
	}

	if(key.pressed("4")){
		mark.get("start1").setVisible(false);
		mark.get("start3").setVisible(false);
		mark.get("start2").setVisible(false);
		mark.get("start4").setVisible(true);
	}

	//kláves „h" prepína viditeľnosť cieľovej a zakázanej oblasti
	if(key.pressed("h")){
		preference.get("ciel").setVisible(!preference.get("ciel").isVisible());
		avoidance.get("zakaz").setVisible(!avoidance.get("zakaz").isVisible());
	}

}
```

Výsledné dáta:

Experiment prebiehal tak, že testovaným osobám bola na začiatku, na dobu asi 5 sekúnd, ukázaná poloha oblastí. Prezrieť si ich mohli len zo štartovacej pozície číslo 1, ďalší pohyb nebol dovolený. Graf na obrázku 35 znázorňuje vzdialenosť medzi človekom a stredom cieľovej oblasti v čase stlačenia klávesu „a". Hodnota „140.98" na zvislej osi určuje deliaci bod. Ak je stĺpec určujúci veľkosť vzdialenosti pod týmto deliacim bodom, tak človek vstúpil

do oblasti, ak nad ním, tak nevstúpil. Deliaci bod je súčet polomeru oblasti (121.93 cm) a polomeru kolízneho cylindru avatara (19.05 cm). Keď sa okraj oblasti dotkne s kolíznym cylindrom, tak je daná udalosť chápaná ako vstup. Čím je hodnota vzdialenosti menšia, tým bol testovaný človek bližšie k stredu oblasti. Konkrétne hodnoty sú uvedené v tabuľke 5 na nasledujúcej strane. Najlepšie výsledky získala Lenka, ktorá bola paradoxne v predchádzajúcom experimente najhoršia. Môže to byť tým, že točiaca sa plošina ju zmiatla, ale v stabilnej miestnosti vedela polohu oblasti odhadnúť veľmi dobre. Za ňou nasledoval Zbyňek a Iveta. Andrej, ktorý bol v predchádzajúcom experimente najlepší skončil až na piatom mieste. Z grafu je vidieť, že pravdepodobne hneď na začiatku odhadol polohu oblasti zle, a tohto odhadu sa držal počas celého priebehu experimentu. Do zakázanej oblasti vstúpila len Martina Ch. a Zdenka, ktoré obsadili dve posledné miesta.

Pri sledovaní rekonštrukcie track logov bolo vidieť, že ku orientácii ľudia viac využívali značku na zemi. Mohlo to byť spôsobené tým, že výškou bola značka na úrovni cieľovej oblasti, čo umožňovalo lepšie odhadnúť vzdialenosť. Ako záchytné body boli používané aj viditeľné štartovacie pozície.



Obr. 35: Porovnanie vzdialeností testovaných osôb od stredu cieľovej oblasti v čase stlačenia klávesu „a“. Zvislá os udáva vzdialenosť v centimetroch. Hranica určujúca, či do oblasti bolo alebo nebolo vstúpené je hodnota 140.98. Vzdialenosti menšie ako táto hodnota sú chápané ako vstup do oblasti. Na vodorovnej osi sú testované osoby. Pre každú sú znázornené štyri pokusy o vstup z rôznych štartovacích pozícií.

	Zbyněk	Zdenka	Lenka	Martina Z.	Martina Ch.	Renča	Iveta	Andrej
Start 1								
Location.X	-309,93	-184,12	-371,24	-325,93	-454,18	-394,88	-435,63	-155,48
Location.Y	238,00	225,61	191,07	105,15	20,86	170,22	99,98	120,99
Distance	40,44	95,56	117,32	160,55	296,98	148,65	225,24	180,75
Radius dist.	-100,54	-45,42	-23,66	19,57	156,00	7,67	84,26	39,77
In area	**True**	**True**	**True**	**False**	**False**	**False**	**False**	**False**
Avoidance	No	No	No	No	Yes	No	No	No
Start 2								
Location.X	-356,51	-413,46	-261,35	-433,7	50,98	-119,16	-432,15	-111,01
Location.Y	376,61	106,56	334,13	66,25	4,21	412,15	113,80	105,72
Distance	144,98	205,05	78,04	248,71	412,08	219,30	213,23	222,73
Radius dist.	4,00	64,07	-62,94	107,73	271,10	78,32	72,25	81,75
In area	**False**	**False**	**True**	**False**	**False**	**False**	**False**	**False**
Avoidance	No	Yes	No	No	No	No	No	No
Start 3								
Location.X	-357,92	-48,37	-219,87	-478,3	-111,04	-193,47	-326,70	-87,87
Location.Y	326,40	584,79	374,07	375,06	-157,26	389,24	276,89	128,07
Distance	108,54	397,98	128,95	235,59	445,44	154,85	55,97	226,79
Radius dist.	-32,44	257,00	-12,03	94,61	304,46	13,87	-85,01	85,81
In area	**True**	**False**	**True**	**False**	**False**	**False**	**True**	**False**
Avoidance	No	No	No	No	No	No	No	No
Start 4								
Location.X	-295,91	-383,62	-191,23	-442,76	-38,06	-187,04	-384,89	-117,06
Location.Y	439,63	-198,51	269,89	166,61	279,96	445,87	212,87	110,49
Distance	183,73	468,61	84,06	191,24	237,36	207,90	119,12	215,05
Radius dist.	42,75	327,63	-56,92	50,26	96,38	66,92	-21,86	74,07
In area	**False**	**False**	**True**	**False**	**False**	**False**	**True**	**False**
Avoidance	No	No	No	No	No	No	No	No

Tab. 5: Tabuľka hodnôt nameraných počas experimentu číslo 2. Stĺpce zodpovedajú jednotlivým testovaným osobám. Riadky sú rozdelené do štyroch skupín podľa štartovacích pozícií. Každá skupina obsahuje riadky: Location.X a Location.Y – pozícia (v cm) testovaného človeka v miestnosti v čase stlačenia klávesu „a“, Distance – vzdialenosť človeka od stredu oblasti, Radius dist. – je počítaná ako „Distance - polomer oblasti (121.93) – polomer kolízneho cylindru avatara (19.05)“. Teda ak je menšia ako nula, tak kolízny cylinder prešiel okrajom oblasti a testovaný človek je vo vnútri. Čím je hodnota bližšie k – 140.98, tým je človek bližšie k stredu cieľovej oblasti. In area – udáva prehľad o tom, či bolo alebo nebolo do oblasti vstúpené, v čase stlačenia klávesu „a“, Avoidance – hovorí, či pri hľadaní cieľovej oblasti z danej štartovacej pozície bolo vstúpené do zakázanej oblasti.

5.3 Experiment č.3: Bludisko

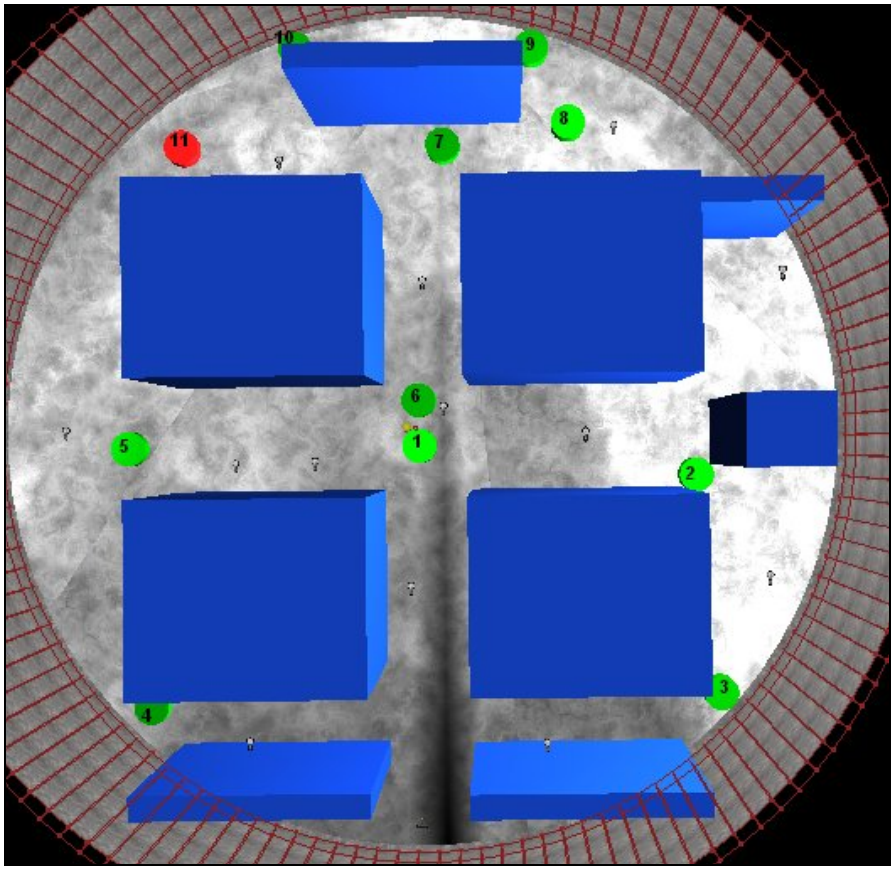
Popis experimentu:

Pre tretí experiment bol zvolený námet, ktorý v pôvodnom kontexte testov v prostredí BVA arény nebol zahrnutý. Jeho hlavným zámerom je poukázanie na skutočnosť, kam až môže navrhovanie testovacích prostredí zájsť v rámci tohto projektu. Námet experimentu tiež ukazuje, že požiadavka na robustnosť grafického nástroja bola opodstatnená. Treba si uvedomiť, že mnou navrhnuté bludisko je veľmi jednoduché, ale jeho možné interpretácie

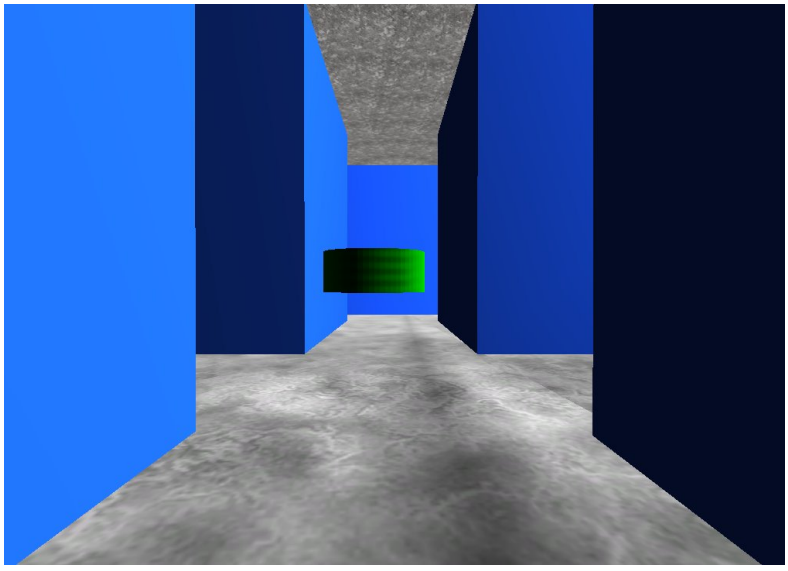
siahajú oveľa ďalej. Predstavme si napríklad krajinu tvorenú ako exteriér, mesto alebo budovu. V každom z nich je možné rozmiestniť oblasti, či značky a merať užitočné dáta.
Pre experiment bol zvolený nasledujúci námet. V bludisku budú umiestnené značky. Vždy bude viditeľná len jedna a vstupom do nej sa zapne nasledujúca (pričom pôvodná sa vypne). Úlohou človeka je postupovať po trase vyznačenej značkami podľa toho, v akom poradí sa budú zobrazovať a snažiť sa zapamätať si cestu. Po zobrazení poslednej značky sa bude musieť vrátiť naspäť rovnakou trasou akou prišiel, pričom značky už nebudú viditeľné. Experiment je zameraný na pozorovanie orientácie ľudí vo väčšom priestore bez výraznejších orientačných bodov. Predpokladá sa, že vzhľadom na rozmer miestnosti s bludiskom a jeho jednoduchosť nebude tento experiment náročný.

Návrh experimentu:

Na obrázkoch 36 a 37 na nasledujúcej strane je možné vidieť náhľad na bludisko vytvorené pre tento experiment. Na prvom obrázku je pohľad zhora v UnrealEd editore, na druhom z perspektívy avatara v hre UT2004. Aby sme mohli sledovať pohyb človeka v 2D mape, je bludisko umiestnené na veľkú kruhovú plošinu. Mohli by sme zvoliť aj štvorcový tvar (plošiny a prostredia), ale v tom prípade by som odporúčala nechať viac mesta na okrajoch (presnejšie povedané v rohoch) štvorcovej plošiny. Je to z dôvodu, že pri škálovaní obrázku do 2D mapy sa využíva polomer plošiny. V prípade štvorcu alebo iného tvaru by sa v závislosti od toho, ktorá hodnota (tj. hodnota od určitého okraju po stred objektu) by bola považovaná za polomer mohlo stať, že menšia alebo väčšia časť prostredia by na obrázku chýbala. Preto napríklad, ak by bolo potrebné vytvoriť obdĺžnikové bludisko, je vždy vhodnejšie umiestniť ho na štvorcovú, prípadne kruhovú väčšiu plošinu.
Vráťme sa teraz späť k popisu bludiska. Ako oddeľovacie steny boli zvolené objekty *Mark,* z rovnakého dôvodu ako v predchádzajúcom prípade. Teda, aby boli viditeľné v 2D mape. Okrem toho prostredie obsahuje 10 zelených a 1 červenú cieľovú oblasť(*Preference*). Mohli by byť zvolené aj objekty typu *Mark* alebo *Avoidance*. Nás ale bude zaujímať, ako dlho trvalo testovanému človeku prejsť od jednej oblasti k druhej, preto typ *Preference* najvhodnejší. Oblasti sú na obrázku 36 očíslované v takom poradí , v akom chceme, aby sa počas experimentu zobrazovali. Keď testovaná osoba príde k poslednej oblasti označenej červenou farbou je to pre ňu znamenie, že sa má otočiť a pokúsiť sa vrátiť rovnakou cestou, akou prišiel. Popísaná mapa sa nachádza na priloženom CD pod názvom TEST-Labyrint.



Obr. 36: Pohľad zhora v UnrealEd editore na schému rozmiestnenia objektov v mape TEST-Labyrint. Modrou farbou sú vyznačené značky, v tomto prípade použité ako steny bludiska. Zelenou a červenou sú znázornené cieľové oblasti určujúce dráhu, ktorú má človek prejsť . Oblasti sa zobrazujú v očíslovanom poradí.



Obr. 37: Pohľad na časť bludiska a jednu cieľovú oblasť v mape TEST-Labyrint z perspektívy testovaného človeka.

Od popisu prostredia prejdeme k popisu konfiguračného skriptu. Pri štarte experimentu nastavíme ako aktívnu a viditeľnú oblasť s menom „1“. Za zmienku stojí, že v skripte je použité meno bez úvodzoviek, teda ako číslo. Môžeme si to dovoliť, pretože ScriptEngine toto číslo pretypuje na *String*. Takýto zápis umožňuje postupne inkrementovať mená oblastí v cykle, čo zjednoduší zápis skriptu. Vstup avatara do oblasti spôsobí, že sa oblasť deaktivuje a skryje, pričom sa zároveň aktivuje a zviditeľní nasledujúca v poradí. Pri vstupe do poslednej oblasti aktivujeme všetky, pre prípad, keby sme chceli sledovať, či subjekt do nich cestou späť vstupoval. Keď bude mať človek pocit, že už prešiel celú trasu, stlačí kláves „q“, čím sa experiment ukončí.

V nasledujúcom odstavci je kód popísaného konfiguračného skriptu s názvom Labyrint.js.

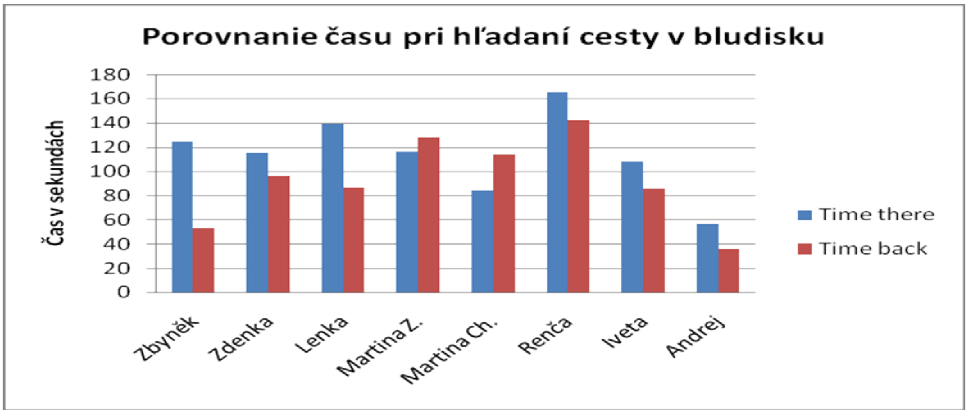
```
var i = 1;
function init() {
	experiment.setMap("TEST-Labyrint");
}

function run() {
	if (experiment.isStarted()){
		preference.get(1).setVisible(true);
		preference.get(1).setActive(true);
	}
	if ( i < 11 && preference.get(i).entered()){
		preference.get(i).setVisible(false);
		preference.get(i).setActive(false);
		i++;
		preference.get(i).setVisible(true);
		preference.get(i).setActive(true);
	}
	if (preference.get(11).entered()){
		preference.get(11).setVisible(false);
		for(i=1;i<=11;i++){
			preference.get(i).setActive(true);
		}
	}

	if (key.pressed("q")){
		experiment.setStop();
	}
}
```

Výsledné dáta:

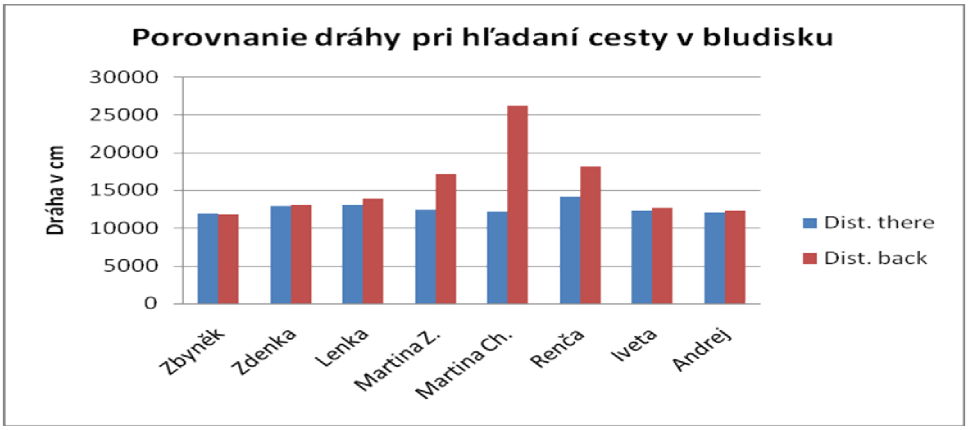
Pri experimente s bludiskom je najdôležitejším výstupom informácia, či testovaná osoba bola schopná zapamätať si cestu späť. Ďalším zaujímavým skúmaným údajom je čas a dráha prejdená po viditeľných značkách a pri hľadaní cesty späť. Tieto dáta sú zaznamenané v tabuľke 6 a zobrazené na grafoch na obrázkoch 38 a 39. Z grafov je vidieť, že väčšina ľudí venovala viac času prechodu bludiskom s viditeľnými značkami (teda smerom „tam“), ako hľadaniu cestu späť. Dôvodom je pravdepodobne ich snaha zapamätať si trasu a aj to, že trávili viac času hľadaním značiek.

	Zbyněk	Zdenka	Lenka	Martina Z.	Martina Ch.	Renča	Iveta	Andrej
Time there	123,953	114,984	139,156	116,171	84,109	165,015	107,562	56,531
Time back	53,172	95,906	86,359	128,172	113,625	142,281	85,906	35,906
Dist. There	11888,8	12927,9	13012,8	12402,1	12088,4	14119,8	12253,3	12080,9
Dist. Back	11764,7	12993,8	13905,7	17151,8	26150,5	18078,6	12669,5	12258,3
Correct	Yes	No	No	No	No	No	Yes	No

Tab. 6: Tabuľka zobrazujúca prejdený čas a dráhu pri experimente číslo 3. V stĺpcoch sú zaznamenané dáta pre každú testovanú osobu. V riadkoch je nasledovné: Time there – čas (v sekundách) od začiatku experimentu po vstup do červenej (poslednej) oblasti, Time back – čas (v sekundách) od vstupu do červenej oblasti po koniec experimentu, Dist. there – dráha (v cm) prejdená od začiatku experimentu po vstup do červenej oblasti, Dist. back – dráha (v cm) prejdená od vstupu do červenej oblasti po koniec experimentu, Correct – Yes, pokiaľ osoba našla správnu cestu späť.

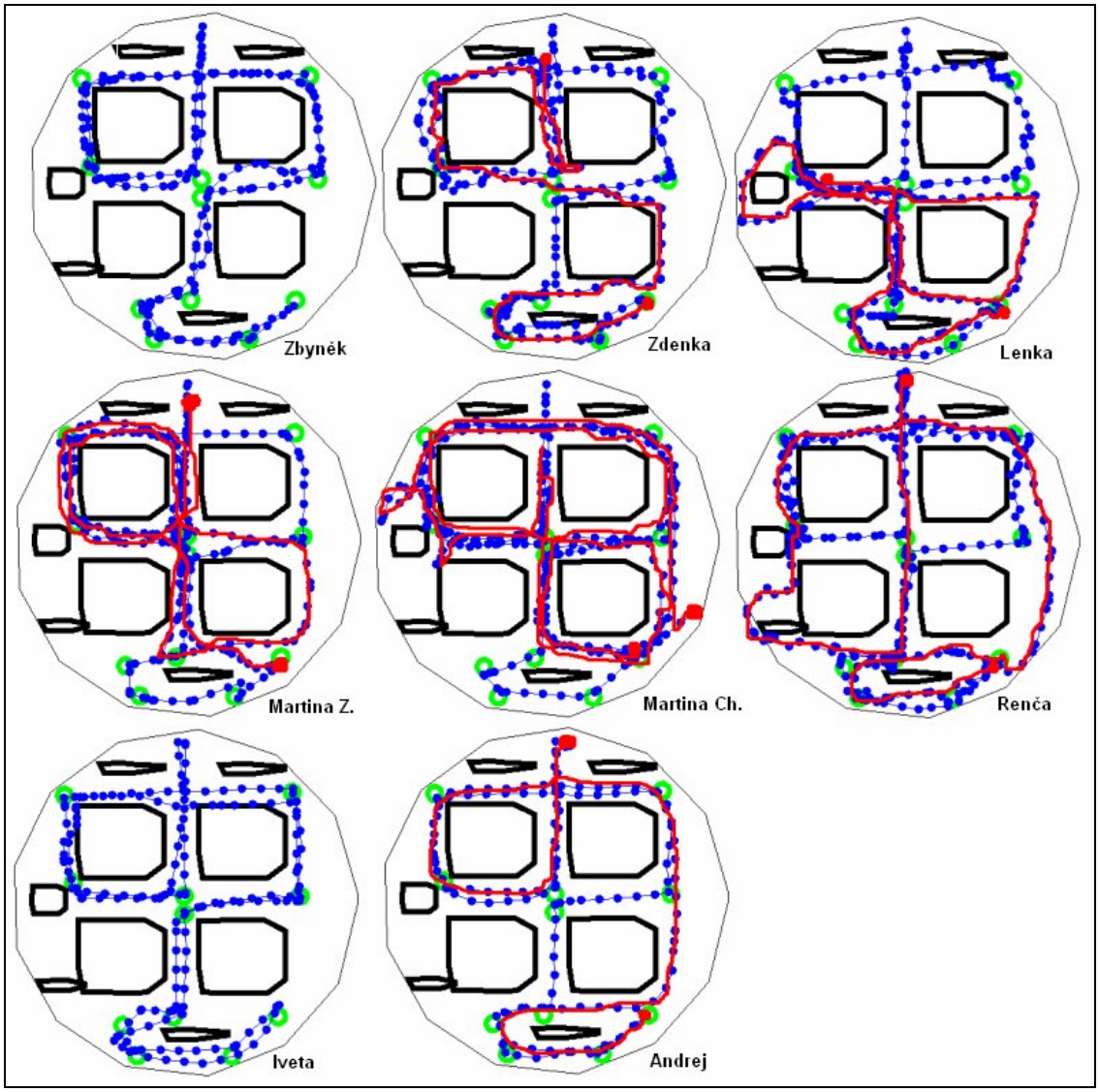


Obr. 38: Porovnanie času pri hľadaní cesty v bludisku. Na vodorovnej osi sú testované osoby. Na zvislej osi je čas v sekundách. Modrý stĺpec znázorňuje čas od začiatku experimentu po vstup do červenej (poslednej) oblasti. Červený stĺpec znázorňuje čas od vstupu do červenej oblasti po koniec experimentu.



Obr. 39: Porovnanie dráhy pri hľadaní cesty v bludisku. Na vodorovnej osi sú testované osoby. Na zvislej osi je dráha v centimetroch. Modrý stĺpec znázorňuje dráhu prejdenú od začiatku experimentu po vstup do červenej (poslednej) oblasti. Červený stĺpec znázorňuje dráhu prejdenú od vstupu do červenej oblasti po koniec experimentu.

Vo väčšine prípadov osoby prešli približne rovnako dlhú trasu na ceste „tam“ aj na ceste „späť“. Súvisí to pravdepodobne s ich odhadom toho, že aká dlhá bola cesta „tam“, taká dlhá by mala byť aj späť. Na obrázku 40 je možné si všimnúť, že jediní, kto našiel správnu cestu späť, boli Zbyněk a Iveta. Štyria ľudia sa vrátili na štartovaciu pozíciu, ale zlou cestou a dvaja ju nenašli.



Obr. 40: Výsledné trasy testovaných osôb pri experimente číslo 3. Modrou farbou je znázornené kadiaľ sa človek pohyboval. Ak sa pri ceste od poslednej značky späť vracal nesprávnou trasou, je táto trasa vyznačená červenou farbou. Pri každom obrázku je vpravo dole meno osoby, ku ktorej sa vzťahuje.

Každý z týchto troch experimentov bol zameraný na testovanie orientácie v priestore, ale každý iným spôsobom. Prvý skúmal schopnosť oddeliť točiacu sa plošinu od statickej miestnosti, druhý preveroval odhad polohy vzdialeného objektu a v treťom bolo úlohou zapamätať si cestu bludiskom. Ukázalo sa, že každému človeku išla jedna úloha lepšie, iná horšie, a že nikto nevynikal vo všetkých troch testoch zároveň. U ľudí však bolo možné pozorovať tendenciu mať sklon k lepším alebo k horším výsledkom. Výsledky takýchto testov môžu byť ovplyvnené osobným prístupom testovanej osoby, jej nadšením, motiváciou, náladou alebo aj skúsenosťou s hraním počítačových hier. Preto by bolo dobré, pri testovaní so skutočným pacientom vykonávať viac testov podobného zamerania v rôzne dni a sledovať prípadné zlepšovanie alebo zhoršovanie.

Kvôli neúplnosti výstupných dát z Eyelinku bolo pri analýze očných pohybov použitých pri prvom experimente 6 výsledkov a pri druhom a treťom experimente 5 výsledkov z 8. Zvyšné údaje boli nahradené pozorovaním uhlu výhľadu človeka. Neúplnosť dát bola spôsobená nedostatočným hardwarovým vybavením. Pri experimentoch sa ukázalo, že použitý počítač (Intel Pentium M, 800 MHz, 504 MB RAM) nie je pre takéto testy dostatočne výkonný a je pod spodnou hranicu odporúčaného vybavenia. Pri použití lepšieho počítača (Intel Core 2 Duo, 2.4 GHz, 4GB RAM) problém s neúplnosťou dát nenastával, avšak pre tieto experimenty nebol k dispozícii.

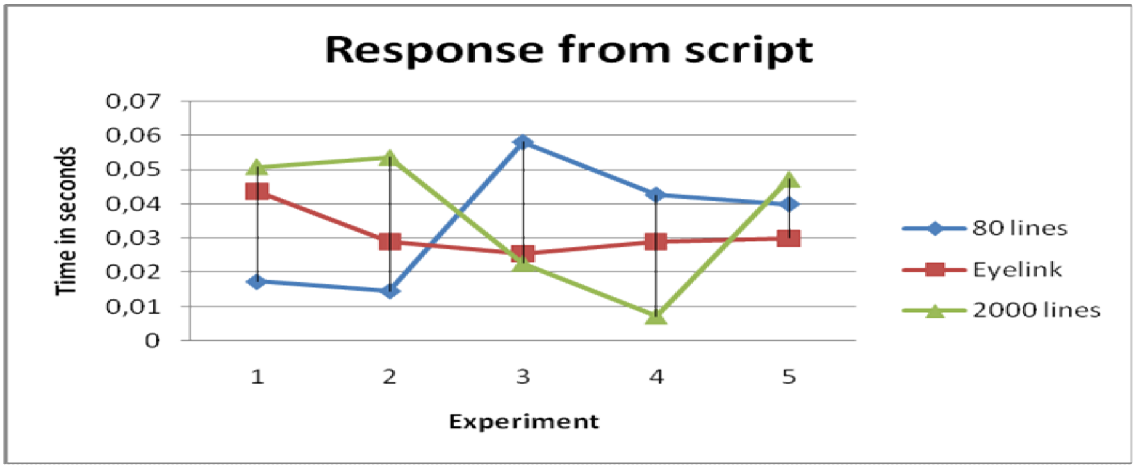
5.4 Meranie rýchlosti komunikácie

V tejto časti je spracované meranie rýchlosti odpovedí z aplikácie na administráciu experimentov smerovaných grafickej aplikácii. Odpovede sú vo väčšine prípadov reakcie na udalosti vo virtuálnom svete a sú definované v konfiguračnom skripte. Čo znamená, že ich oneskorenie by malo byť čo najmenšie. Použitý počítač mal ako v predchádzajúcom prípade nasledujúce parametre: Intel Pentium M, 800 MHz, 504 MB RAM. Merania boli prevedené počas už popísaného experimentu číslo 1 (tento krát bez skupiny testovaných osôb). Meral sa časový rozdiel, od momentu kedy avatar vstúpil do oblasti po príchod správy, že sa má daná oblasť deaktivovať. Správa o deaktivácii je teda odpoveď na vstup do oblasti. Indikácia o vstupe do oblasti je odoslaná do aplikácie na administáciu, tá ju predá skriptu, ktorý vyhodnotí, že sa má táto oblasť deaktivovať a aktivovať ďalšia. Bolo prevedených 5

experimentov, z každého z nich sa získalo 11 meraní. Priemerné hodnoty týchto meraní sú v tabuľke 7. Graficky znázornené sú na obrázku 41 na nasledujúcej strane. UT2004 aj EyelinkClient zaberajú vyše 90% CPU, a preto im bolo potrebné znížiť prioritu na *„Below normal“*. Experimenty boli vykonané tromi spôsobmi: s konfiguračným skriptom s dĺžkou 80 riadkov, s použitím simulácie Eyelinku, a so skriptom s dĺžkou 2000 riadkov. Predpokladalo sa, že veľká dĺžka skriptu alebo použitie prístroja Eyelink ovplyvnia rýchlosť odpovedí. Z výsledku sa však vo všeobecnosti nedá určiť, či tieto podmienky majú významnejší vplyv na odozvu zo skriptu. Spustenie Eyelinku a UT2004 zároveň zníži výkon oboch, čo bolo pri použití uvedeného počítaču v 3D aplikácii vizuálne znateľné .

Experiment	80 lines	Eyelink	2000 lines
1	0,01727273	0,04363636	0,05090909
2	0,01454545	0,02909091	0,05363636
3	0,05818182	0,02545455	0,02272727
4	0,04272727	0,02909091	0,00727273
5	0,04	0,03	0,04727273
Average:	**0,03454545**	**0,03145455**	**0,03636364**

Tab. 7: Meranie rýchlosti odpovedí z aplikácie na administráciu do grafickej aplikácie. Stĺpce po rade zľava obsahujú: číslo experimentu, výsledok merania so skriptom s 80-timi riadkami, výsledok merania s Eyelinkom, výsledok merania so skriptom s 2000 riadkami. Posledný riadok obsahuje priemernú hodnotu pre všetky experimenty daného typu.



Obr. 41: Graf znázorňuje rýchlosť odpovedí z aplikácie na administráciu na udalosti vzniknuté v grafickej aplikácii za rozličných podmienok. Podmienky boli nasledovné: použitie skriptu s 80-timi riadkami, použitie prístroju Eyelink, použitie skriptu s 2000 riadkami. Na zvislej osi je čas v sekundách. Na vodorovnej osi je päť pokusov meraní.

6 Záver

Hlavným cieľom tejto diplomovej práce bolo previesť do virtuálnej reality experimentálne reálne prostredie Blue Velvet Arena, slúžiace na výskum orientácie v priestore. To vyžadovalo navrhnúť a vytvoriť komplexný software, ktorý umožňuje vytvárať virtuálne prostredia, pomocou skriptovacieho jazyka konfigurovať priebeh experimentov a vykonávať a riadiť takéto experimenty vrátane zberu dát a merania očných pohybov.

Výsledný software je navrhnutý pre dva navzájom prepojené počítače, jeden pre testovaný subjekt a druhý pre osobu dohliadajúcu na experiment. Ako nástroj pre vytvorenie virtuálneho prostredia bol zvolený Unreal engine. Táto voľba obstála pri spätnom hodnotení, pretože poskytuje množstvo vlastností (editor prostredia, 3D náhľad, video záznam), na ktorých vytvorenie by bolo treba priveľa času. Rozhrania na administráciu experimentu a 2D náhľad sú napísane v jazyku Java. Jeho novšie verzie obsahujú *script engine*, ktorým možno prepojiť vnútorný kód aplikácie s textovým súborom napísaným v jazyku JavaScript. Tú vlastnosť využívame na parametrizáciu priebehu experimentov vo virtuálnom prostredí pomocou konfiguračného súboru. Systém Eyelink II je s grafickým prostredím prepojený pomocou klienta, ktorý sprostredkuje ich vzájomnú komunikáciu. Pomocou vytvoreného softwaru boli uskutočnené tri prototypové experimenty, demonštrujúce spôsob návrhu testovacích prostredí a ich konfigurácie.

Hlavným prínosom tejto práce je, umožnenie takmer neobmedzenej parametrizácie prostredia, a to z hľadiska vizuálneho ako aj z hľadiska konfigurácie priebehu experimentov. Ostatné dostupné aplikácie podobného zamerania obsahujú len napevno vytvorené prostredia bez možnosti do nich zasahovať. Simuláciou navigačných testov vo virtuálnej realite tento software dovoľuje testovať pacientov v ktorejkoľvek ambulancii. Umožňuje použiť techniky (napríklad sledovanie očných pohybov alebo aktivity mozgu), ktoré pri pohybe testovaných osôb v reálnom prostredí nebolo možné používať.

Software je otvorený ďalšiemu vývoju, hlavne v oblasti zavedenia do praxe a v sprístupnení pre potreby lekárov na včasnú diagnostiku AD v bežných ambulanciách. Preto by bolo vhodné vytvoriť sadu štandardizovaných testovacích prostredí, ktoré by dokázali odhaliť poškodenú schopnosť orientácie v priestore. K týmto konkrétnym testovacím prostrediam by mohlo byť vytvorené automatické a prehľadné vyhodnotenie výstupov pre lekára tak, aby ich nemusel svojpomocne analyzovať.

7 Použitá literatúra

[1] O'Keefe, J., Dostrovsky J. (1971): The hippocampus as a spatial map. Preeliminary evidence from unit activity in the freely-moving rat. Brain Res. 34: 171-175

[2] Vlček K., Laczó J., Vajnerová O., Blahna K., Vyhnálek, M., Hort J. (2006): Impairement of episodic-like memory in different stages of Alzheimer's disease. 1-1

[3] Vlček K., Laczó J., Vajnerová O., Ort M., Kalina M., Blahna K., Vyhnálek, M., Hort J. (2006): Spatial navigation and episodic-memory tests in screening of dementia. Psychiatrie 10: 1-3

[4] Vlček K., Laczó J., Vajnerová O., Ort M., Vyhnálek, M., Hort J. (2007): Amnestic mild cognitive impairment and Alzheimer's disease is impaired in navigation in two dissociated spatial reference frames.1-1

[5] J. M. Cimadevilla, M. Wesierska, A. A. Fenton, and J. Bures (2000) : Inactivating one hippocampus impairs avoidance of a stable room-defined place during dissociation of arena cues from room cues by rotation of the arena. Proc. Natl. Acad. Sci. USA 98 (6):3531-6

[6] Vlček K.1, Laczo J.2, Blahna K.1, Hort J.2, Kalina, M.3, Bureš J. (2006): Impairment navigation in two dissociated spatial reference frames in different stages of Alzheimer's disease.1-1

[7] Mgr. Iveta Fajnerová, Bakalárska práca, PRF UK (2007): Neuronálne základy priestorovej orientácie u ľudí. 14-15

[8] Morris, R. G. M. (1981): Spatial localization does not require the presence of local cues. Learn Motiv 12: 239-261 (prevzaté z 7)

[9] Water maze
http://www.watermaze.org (15.7.2008)

[10] Maria T. Schultheis, Albert A. Rizzo (2001): The Application of Virtual Reality Technology in Rehabilitation.

[11] Riva G.(2003): Virtual enviroments in clinical psychology. Psychotherapy: Theory, Research, Practice, Training, Vol. 40, No. 1/2, 68–76

[12] Baumann S., Sayette M. (2006): Smoking Cues in a Virtual World Provoke Craving in Cigarette Smokers. Psychology of Addictive Behaviors Vol. 20, No. 4, 484–489

[13] Maltby N., Kirsch I., Mayers M., Allen G. (2002): Virtual Reality Exposure Therapy for the Treatment of Fear of Flying:A Controlled Investigation. Journal of Consulting and Clinical Psychology Vol. 70, No. 5, 1112–1118

[14] Glantz K., Rizzo A., Graap K. (2003): Virtual reality for psychoterapy: Current reality and future possibilities. Psychotherapy: Theory, Research, Practice, Training, Vol. 40, No. 1/2, 55–67

[15] Pravidelne aktualizovaná stránka o VR
http://vroot.org/ (12.9.2007)

[16] Stang B. (2003): Game Engines features and Possibilities. IMM DTU, 7-11

[17] Lewis, Jacobson (January 2002): Game engines in scientific research. Vol. 45, No. 1 Communications of the ACM, 27-31

[18] LudoCraft, ELIAS-project (2005): Open Source & Low Cost Game Engines, http:\\ludocraft.oulu.fi/elias/dokumentit/open_source_game_engines.pdf (13.4.2007)

[19] 3DGameStudio
www.3dgamestudio.com (10.6.2007)

[20] Blitz3D
www.blitzbasic.com (10.6.2007)
[21] Crystal space
www.crystalspace3d.org (10.6.2007)
[22] Delta 3D
www.delta3d.com (15.6.2007)
[23] Irrlicht
http://irrlicht.sourceforge.net (15.6.2007)
[24] Panda3D
http://www.panda3d.org (15.6.2007)
[25] Reality engine
www.artificialstudios.com (16.6.2007)
[26] Torque
www.garagegames.com/pg/product/view.php?id=1 (3.7.2007)
[27] Nebula
www.nebuladevice.cubik.org (3.7.2007)
[28] Unreal
http://udn.epicgames.com/Two/UnrealEngine2Runtime (13.7.2008)
[29] Quake III
www.idsoftware.com (7.10.2008)
[30] OGRE
www.ogre3D.org (10.6.2007)
[31] OpenSceneGraph
www.openscenegraph.org (10.6.2007)
[32] Blender
www.blender.org (11.7.2007)
[33] Systém triggerov Unreal Engine
http://wiki.beyondunreal.com/wiki/Trigger_Systems (20.10.2008)
[34] Mozilla Rhino engine
http://www.mozilla.org/rhino/ (4.5.2008)
[35] Špecifikácia JSR-233
http://www.jcp.org/en/jsr/detail?id=223 (4.5.2008)
[36] The Mustang Meets the Rhino: Scripting in Java 6, John Ferguson Smart, 2006 http://www.onjava.com/pub/a/onjava/2006/04/26/mustang-meets-rhino-java-se-6-scripting.htm (4.5.2008)
[37] SR Research Ltd.
www.sr-research.com (9.10.2008)
[38] SR Research Ltd.(2006): EyeLink Programmer's Guide,Version 3.0
[39] Lukavský, J. Sledování očních pohybů. Bakalárska práca, MFF UK Praha, 2005
[40] Projekt Pogamut
http://artemis.ms.mff.cuni.cz/pogamut_wiki/ (10.11.2007)
[41] Tarquin brush builder pack
http://tarquin.planetunreal.gamespy.com/pack.tarq.html (2.1.2008)
[42] Zoznam jazykov použiteľných so Scipt enginom
https://scripting.dev.java.net/ (4.5.2008)
[43] Wiki pre modifikácie UT2004
http://wiki.beyondunreal.com/ (1.3.2009)

8 Príloha A – Obsah priloženého CD

Priložené CD obsahuje:

- thesis
  - thesis.pdf – text diplomovej práce
- doc
  - user_doc.pdf - užívateľská dokumentácia
  - prog_doc.pdf – programátorská dokumentácia
- src
  - SpaNav
    - SpaNav – zdrojový kód
    - SpaNav.jar – spustiteľný súbor aplikácie na administráciu
    - script-examples – testovacie konfiguračné skripty
    - javadoc – dokumentácia
  - SpaNavUT
    - UT2004
      - SpaNavUT – zdrojový kód
      - System – skompilovaný projekt
      - Maps – testovacie mapy
      - StaticMeshes – testovacie mriežky
    - tarquin – inštalátor pre tarquin brush builder
    - smith – human skin avatara
  - EyelinkClientUT
    - install – súbory potrebné k spusteniu Eyelink klienta
    - EyelinkClientUT – zdrojový kód
    - EyelinkClient.exe – spustiteľný súbor
- outputs – výstupné dáta z experimentov

Popis inštalácie sa nachádza v užívateľskej dokumentácii na priloženom na CD.