Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Návrh a implementace deskové hry pro kognitivní trénink

Plzeň, 2017 Michal Horký

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 4. května 2017

Michal Horký

Poděkování

Děkuji panu Ing. Pavlu Mautnerovi, Ph.D. za vedení, obětavou pomoc a věcné připomínky, které mi během bakalářské práce poskytl.

Abstrakt

Bakalářská práce popisuje výběr, návrh a implementaci deskové hry pro kognitivní trénink. Vybranou hrou jsou Lodě. Hra je pro dva hráče, kteří se snaží zničit jako první flotilu toho druhého.

Hra je upravená tak, aby hráče co nejvíce bavila a aby byla schopna využít čelenku NeuroSky MindWave Mobile, která snímá soustředění a meditaci uživatele. Snímaná data z čelenky jsou využívána ve hře, kdy při určitém soustředění uživatel může ovlivnit průběh hry. Při opakovaném hraní slouží aplikace pro trénink schopnosti soustředění a meditace.

Dále je přidělán modul pro průzkum mozkové aktivity uživatelů. Ten umožňuje kromě sběru dat i zobrazení výsledků měření.

Ke hře je vytvořen server, který zprostředkovává komunikaci mezi klienty a uchovává jejich konfigurace. Jak hra, tak server jsou vyvíjeny v programovacím jazyce C++ s využitím Qt frameworku, který zajišťuje vytvoření multiplatformního softwaru.

Abstract

The bachelor thesis describes choice, design and implementation of the board game for a cognitive training. Selected game is Naval Battles. The game is for two players, who try to destroy other player's fleet first.

The game is modified to be more enjoyable and to use headset NeuroSky MindWave Mobile. The headset gets values of player's attention and meditation. This data are used in the game. Player can influence a game situation with a certain level of attention or meditation. The game serves as a training for player's attention and meditation, when is repeatedly played.

There is a module for a brain activity research that besides a data collection shows a statistic results.

The game has server, which was created for communication with clients and preservation of their game configurations. Both programs are implemented in language C++ with Qt framework, which ensures to create multiplatform software.

Obsah

1 Úvod

Cílem práce bylo navržení nebo vylepšení již existující deskové hry, která by mohla, s využitím EEG čelenky MindWave Mobile, sloužit pro kognitivní trénink uživatele.

Součástí práce bylo prostudování problematiky měření mozkové aktivity, implementace algoritmu pro získání EEG dat z čelenky, konkrétně úrovně soustředění a meditace, a implementace vybrané deskové hry. Implementovaná hra měla být otestována. Dále se měla provést série měření na různých hráčích. Na základě domluvy se zadavatelem byl do aplikace přidán modul pro průzkum mozkové aktivity s možností zhlédnutí naměřených výsledků.

Jelikož se jedná o hru pro dva hráče, bylo zapotřebí naimplementovat server, který bude zprostředkovávat komunikaci mezi klienty a uchovávat jejich konfiguraci pro případ náhlého odpojení a ztráty dat.

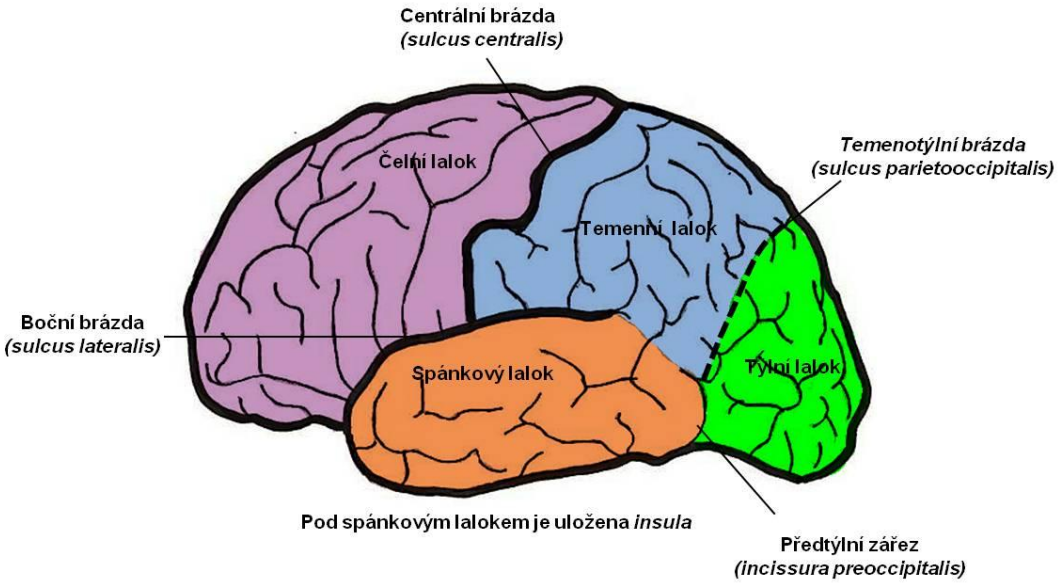
Na základě požadavku na multiplatformní hru byl zvolen jazyk C++ s využitím Qt frameworku.

Po přečtení práce budete seznámeni s tím, co přesně znamená mozková aktivita, jak se zjišťuje a měří. Dále budete seznámeni s čelenkou MindWave Mobile od firmy NeuroSky. Dozvíte se, jak pracuje a jak vytvořit aplikaci, která využívá její funkčnost. Seznámíte se s vývojem aplikací v jazyce C++ s využitím Qt frameworku a nahlédnete do programového řešení klienta a serveru.

Práce zahrnuje komunikační protokol pro síťovou komunikaci a uživatelskou dokumentaci.

2 Mozková aktivita

2.1 Lidský mozek



1. *Lidský mozek*

Lidský mozek je řídící centrum nervové soustavy člověka. Váží přibližně `1.4kg`, což jsou přibližně pouhá `2%` celkové váhy člověka. Řídí funkčnost všech svalů a vnitřních orgánů, zpracovává vjemy ze smyslové soustavy, zapamatovává si informace, apod. Mozek se dělí na několik částí (každá z nich plní často rozdílné funkce), které spolu spolupracují.

2.1.1 Mozkový kmen

Mozkový kmen řídí autonomní procesy člověka. Jedná se o činnosti, které jsou základem života, a člověk je vědomě nemůže příliš ovlivnit. Jde o srdeční činnost, dýchání, činnost močového měchýře a pocit rovnováhy.

2.1.2 Limbický systém

Často je nazýván jako emocionální mozek. Obsahuje hypotalamus, thalamus a amygdalu. Ovlivňuje chování při různých emocích, do kterých patří například úzkost, strach, vzrušení, radost, atd.

2.1.3 Mozeček

Mozeček je odpovědný za jemné pohyby, držení a rovnováhy těla. Přijímá signály z míchy a dalších částí mozku, které pak využívá k ladění pohybové aktivity. Zajímavostí je, že obsahuje přibližně `80%` veškerých mozkových neuronů.

2.1.4 Mozková kůra

Je největší částí lidského mozku. Je spojována s vědomým myšlením a se schopností rozhodování. Skládá se z levé a pravé hemisféry, které nejsou přímo spojeny. Jednotlivé hemisféry se pak dělí na dalších několik laloků. Konkrétně se jedná o čtyři, které jsou popsány níže. Hemisféry mezi sebou komunikují přes dlouhá spojení prostřednictvím thalamu a dalších částí lidského mozku.

2.1.5 Týlní lalok (okcipitální lalok)

Týlní lalok je centrem vizuálního zpracování. Vše, co vidíme, se zpracovává zde. Má na starosti orientaci v prostoru, vnímání barev a pohybu. Okcipitální léze mají za následek vizuální halucinace, barvoslepost, špatné vnímání pohybu nebo dokonce slepotu.

2.1.6 Spánkový lalok

Spánkový lalok je zodpovědný za dlouhodobou paměť. Kromě toho je důležitý pro lidskou řeč, pro porozumění a zpracování psaného textu.

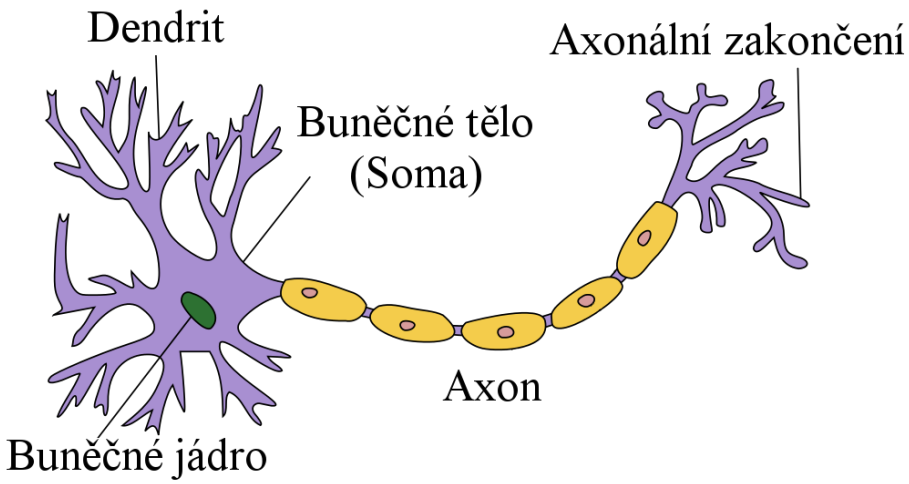
2.1.7 Temenní lalok

Kůra temenního laloku zpracovává tvar, velikost a orientaci objektů v našem okolí. Tyhle informace spojuje se signály od kosterních svalů končetin, hlavy, očí, apod. Na základě těchto informací jsme schopni prostorové orientace. Úkoly, které vyžadují pohyby očí a rukou by byly bez temenního laloku nemožné.

2.1.8 Čelní lalok

Jedná se o oblast, kde probíhá většina našich vědomých myšlenek a rozhodnutí. Jsou zde pohybová centra, která řídí například pohyby očí. Obsahuje většinu neuronů, které jsou citlivé na dopamin. Dopaminový systém je velmi důležitý pro kognitivní funkce. Kognitivní funkce slouží k vnímání okolního světa. Patří k nim paměť, pozornost, myšlení, atd. Díky čelnímu laloku si dokážeme uvědomit, co může způsobit naše chování. Jsme schopni rozeznat dobré a špatné a to špatné potlačit.

2.2 Neuron



2. Neuron

Lidský mozek obsahuje 50–100 miliard nervových buněk (neuronů). Mezi hlavní části každého neuronu patří buněčné tělo (soma), jeden a více dendritů a axon. Dendrity jsou malé výběžky neuronu, kterých může být až několik tisíc. Jejich velikost obvykle nepřekročí 2mm. Každý dendrit končí synapsí. Synapse je spojení dvou neuronů, které slouží k předávání vzruchů (impulsů). Axon je obvykle jedna větev, která vysílá výstupní signál neuronu do různých částí nervového systému. Jeho velikost je od 1mm do 1m. Delší axony jsou ty, které běží od míchy až k nohám.

V synapsích se neurony přímo nedotýkají, nachází se mezi nimi synaptická štěrbina o šířce asi 20nm. Jakmile přijde nějaký signál k nervovému zakončení, z nervového zakončení se vyloučí chemická látka (neurotransmiter – dopamin, epinefrin, acetylcholin, apod.), která způsobí vznik synaptického potenciálu na druhém neuronu (postsynaptický). Probíhá tak přeměna elektrického signálu v presynaptickém neuronu na chemický. Synaptická aktivita vytváří velmi malé elektrické pole (postsynaptický potenciál). Ten trvá desítky až stovky milisekund. Jelikož je postsynaptický potenciál příliš malý, u jednoho neuronu bychom ho nemohli zaznamenat. Neurony ovšem komunikují mezi sebou a už u malé skupiny neuronů se výsledné pole stává mnohem silnější.

Jakmile neuron přijme signál z jednoho či více dendritů a vypočítá výstupní hodnotu, signál jde, skrz axon až k zakončení, čistě elektrickou cestou. U tohoto způsobu nepotřebujeme receptory. Stejně jako každá buňka v těle, mají neurony uvnitř vysokou koncentraci iontů draslíku a chloridu, a z venku si udržují ionty sodíku a vápníku. Tohle uspořádání vytváří malou baterii, která udržuje napěťový rozdíl −60mV mimo buněčnou stěnu. Membrána neuronů obsahuje malé póry, kanály, kterými se ionty mohou pohybovat tam a zpátky. Neurony umí tyhle kanály velmi rychle otevírat a zavírat. Tím mění tok iontů a vytváří napěťové rozdíly na membráně. Potenciál se

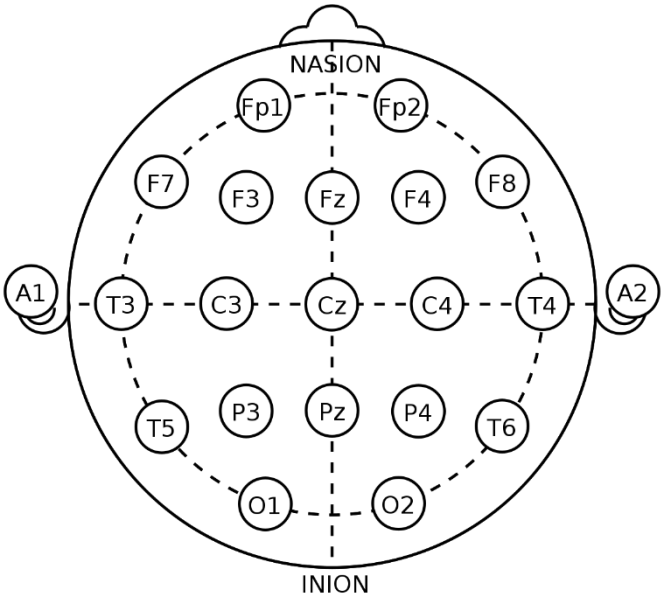
zvyšuje, když je membrána polarizována negativním nábojem, který se nachází uvnitř buňky.

2.3 EEG a mozkové vlny

Elektroencefalografie, zkráceně EEG, je diagnostická metoda, která se používá pro záznam aktivity mozku. Pořizuje se elektroencefalografem, který zaznamenává časové změny elektrického potenciálu způsobeného mozkovou aktivitou. Mozková aktivita není nic jiného než přesun elektrického signálu z jednoho neuronu na druhý. Používá se pro diagnostikování různých poruch, jako jsou třeba epileptické záchvaty či poruchy spánku.

Elektrické signály, snímané elektrodami, jsou velmi malé (desítky mikrovoltů), proto se napětí musí nejprve zesílit v zesilovači, aby bylo možné snímané údaje dále zpracovávat. Kromě zesilovače hraje také důležitou roli dobré umístění elektrod a konstrukce přístroje kvůli elektromagnetickému rušení, které je v dnešní době, obzvláště v nemocničním prostředí, všude. EEG používá elektrody, které se přikládají k pokožce hlavy. Aby byla data co nejkvalitnější, musí se mezi pokožku hlavy a danou elektrodu nanést vodivý gel, pasta či krém. Důležitá pro EEG je kombinace vhodného kovu elektrod (`Ag/AgCl` – ze stříbra s tenkou vrstvou chloridu stříbrného) a vodivého gelu. Existují i elektrody, které nepotřebují gel. Jsou rychlejší ale náchylnější k rušení.

Elektrody jsou obvykle v čepičkách či senzorových proužcích, což urychluje nastavení. Na umístění elektrod a jejich počet existují různé systémy. Systémy EEG obvykle nabízejí ukazatel kvality, kde je zobrazována impedance jednotlivých elektrod. Na pokožce hlavy se totiž hromadí staré kožní buňky a pot, což zvyšuje odpor. Jakmile je získána hodnota napětí, musí být analogový signál zesílen a digitalizován. Systémy EEG berou pouze snímky nepřetržitého signálu (stejný princip jako videokamera). Jednotlivé systémy se mezi sebou liší akorát vzorkovací rychlostí (počet vzorků za sekundu).



3. Rozmístění elektrod systému 10-20

Během snímání dat může dojít k různým artefaktům. Jedná se například o svalové pohyby, proto by měřený člověk měl být co nejvíce v klidu. Dokonce i pohyby očí ovlivňují naměřené změny napětí. Vertikální pohyby očí jsou na grafu podobné funkci sinus, zatímco horizontální pohyby mají obdélníkový průběh. Tyhle pohyby se mohou odchytávat jinými senzory, pomocí kterých se pak rušení odstraňuje.

EEG signál je výsledkem váženého součtu aktivit několika miliard neuronů, takže nelze odlišit jednotlivé akční potenciály, jako například u EMG. Proto je typický průběh EEG na první pohled hodně nepravidelný. Na základě frekvencí se pak rozlišují jednotlivé druhy mozkových vln na delta, théta, alfa, beta a gama vlny.

2.3.1 Delta vlny

Rozsah frekvence je `1-4Hz`. Jedná se o nejpomalejší vlny s nejvyšší amplitudou. Delta vlny jsou přítomny pouze ve stavu hlubokého spánku známého jako slow-wave sleep (pomalá vlna spánku, dále jen SWS). Tyto vlny doprovázejí procesy tělesné regenerace. Jsou důležité pro formování a uspořádávání paměti. Zdroj těchto vln je v thalamu. Silnější jsou v pravé hemisféře. Pití velkého množství alkoholu snižuje SWS. Má to na ní dlouhodobý vliv. Nepomáhá ani dlouhodobá abstinence.

2.3.2 Théta vlny

Rozsah frekvence je `4-8Hz`. Théta vlny převládají při zaměřené pozornosti, zjišťování informací, učení nebo vzpomínání. Objevuje se i během ospalosti. Proto jsou théta vlny spojovány s duševní pracovní zátěží. Dále se vyskytují při navigačních úlohách, při řízení dopravních prostředků, aj.

2.3.3 Alfa vlny

Rozsah frekvence je `8-12Hz`. Alfa vlny jsou generovány převážně v zadních částech mozku. Alfa vlny jsou zvýšené při fyzickém a psychickém uvolnění se

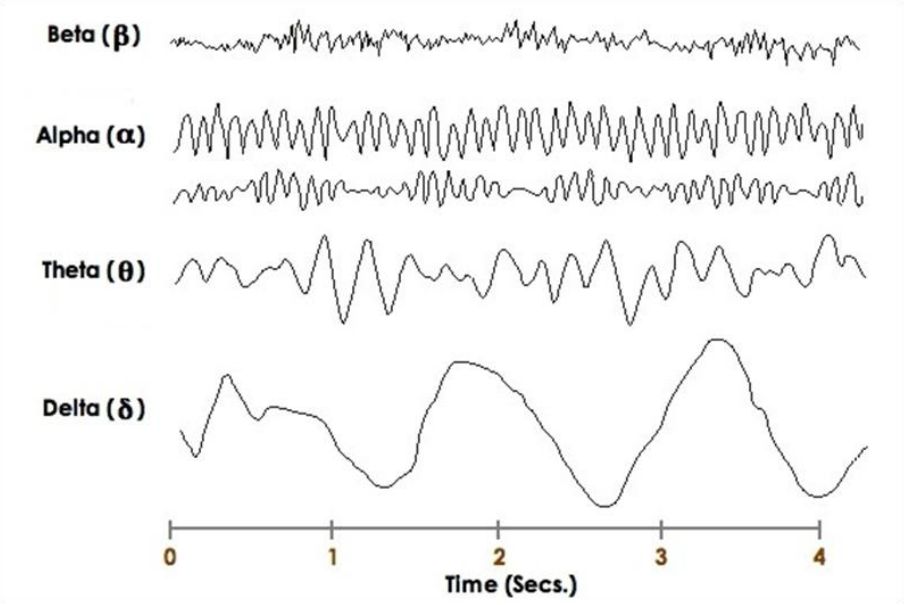
zavřenými oči. V normálním bdělém stavu při nějaké činnosti jsou alfa vlny nižší. Jakmile dojde ke snížení alfa vlny, znamená to, že se mozek připravuje zpracovat informace z různých smyslů. Alfa vlny se používají k tréninku Biofeedback, během kterého se snímá uvolnění. Vyšší hodnoty jsou brány jako hlubší relaxace.

2.3.4 Beta vlny

Rozsah frekvence je `13-25Hz`. Tahle frekvence je generována jak v zadní, tak v přední části mozku. Vyšší hodnoty nacházíme ve stavech zaneprázdnění, úzkosti, myšlení. Zajímavostí je, že vyšší hodnoty jsou i u lidí, kteří pozorují pohyby ostatních. Mozek nejspíše zdánlivě napodobuje pohyby ostatních. Vyšší hodnoty jsou také při pokusech, kdy člověk má uchopit nějaký objekt ve virtuální realitě. Dále jsou sledovány při různých světelných či zvukových podnětech.

2.3.5 Gama vlny

Jako gama vlny bereme vlny s frekvencí od `25Hz` výš. O gama vlnách se toho zatím moc neví. Někteří vědci tvrdí, že jsou podobné Théta vlnám, jiní, že gama vlny jsou vedlejším produktem neurálních procesů (pohyby očí, atd.).



4. Ukázka mozkových vln

3 NeuroSky MindWave Mobile

NeuroSky je americká společnost, která vyvíjí inovativní biosenzory a biometrické algoritmy za účelem vedení zdravějšího a kvalitnějšího života.

3.1 Čelenka MindWave Mobile

Čelenka MindWave Mobile měří mozkovou aktivitu, monitoruje úroveň pozornosti a sleduje mrkání uživatele. Kromě toho měří jednotlivé mozkové vlny, ale hodnoty, které z čelenky vycházejí, slouží pouze k porovnání vzrůstů a poklesů jednotlivých druhů vln (měření těchto vln není moc přesné). Čelenka komunikuje s počítačem pomocí technologie bluetooth.

5. MindWave Mobile čelenka

Bluetooth je technologie sloužící pro bezdrátovou komunikaci, během které se propojují dvě a více zařízení. Příkladem může být mobilní telefon a bezdrátová sluchátka. Technologie bluetooth je definovaná standardem `IEEE 802.15.1` a spadá do kategorie malých osobních počítačových sítí s názvem PAN (Personal Area Network). Čelenka využívá bluetooth verze `2.1`. Hlavní výhodou téhle verze je bezpečné a jednoduché párování.

Čelenka se skládá z jedné elektrody a ušního klipu. Ušní klip slouží k uzemnění. Elektroda se uživateli přiloží na čelo a pomocí ní čelenka snímá aktivitu frontálního laloku, kde se právě zjišťuje úroveň soustředění.

Existuje celá řada aplikací, které využívají právě čelenku NeuroSky MindWave Mobile. Například v základním balíčku čelenky je CD s aplikacemi SpeedMath a BrainWave Visualizer. Všechny aplikace využívající tuhle čelenku většinou slouží k zobrazení mozkové aktivity uživatele při nějaké činnosti a umožňují následný kognitivní trénink, například schopnosti soustředění či meditace.

3.2 ThinkGear technologie

ThinkGear je technologie od firmy NeuroSky, která umožňuje zařízení zpracovávat mozkové vlny nositele. Zahrnuje senzor s EEG elektrodami, který se dotýká čela, kontaktní body uložené v ušním klipu a čip, který zpracovává všechna data. Výstupními hodnotami pak jsou: hrubý signál (vzorkovaný `128Hz – 512Hz`), signál nekvalitní metriky (jak dobře je uživatel k čelence připojen), eSence hodnoty

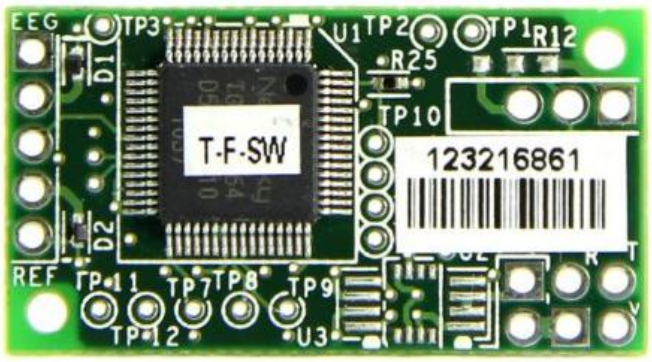
soustředění a meditace, EEG hodnoty pro jednotlivé mozkové vlny a v některých hardwarových modelech i další hodnoty.

3.3 eSence

eSence je algoritmus firmy NeuroSky, který slouží pro výpočet úrovně soustředění a meditace. Než přijde na řadu eSence, musí technologie ThinkGear zesílit naměřený hrubý signál a odstranit z něj rušení a pohyb svalů. Na takto upravený signál se aplikuje eSence algoritmus, který zjišťuje rozsah aktivity. eSence měřič nám říká, jak efektivně se uživatel dokáže soustředit (úroveň soustředění) nebo naopak uvolnit (úroveň meditace).

Úroveň soustředění uživatel zvýší, když se vizuálně soustředí na nějaký objekt nebo když nad něčím přemýšlí. Například počítání různých příkladů nebo přeříkávání si slov nějaké písničky. Úroveň meditace bude vyšší při odpočinku. Zvýší se při uklidnění, při relaxaci, když uživatel zavře oči a „vyčistí si hlavu“. Zajímavé je sledovat vzrůsty a poklesy při určitých činnostech. Například, když člověk pracuje, soustředění je vyšší a uvolnění nižší. Jakmile ovšem člověk zavře oči, dojde k poklesu soustředění a k navýšení uvolnění.

4 ThinkGear



6. ThinkGear AM

4.1 ThinkGear hodnoty

4.1.1 POOR_SIGNAL Quality

Toto kladné celé číslo o délce jednoho bytu popisuje, jak moc je měřený signál špatný. Nabývá hodnot `0-255`. Každá nenulová hodnota znamená, že je detekován nějaký šum. Čím vyšší číslo, tím je šum větší. Hodnota `200` je speciální případ a většinou znamená, že elektrody nejsou v kontaktu s kůží. Tato hodnota je na výstupu obvykle každou sekundu. Špatný signál může být dále způsoben tím, že uživatel na sobě nemá čelenku, je špatný kontakt elektrody (nebo ušního klipu) s kůží, uživatel se nadměrně pohybuje nebo je v okolí elektrostatické rušení.

4.1.2 eSense(tm) Meters

Hodnoty pro všechny typy eSense (tj. soustředění a meditace) jsou od `0-100`. Na tomto měřítku je hodnota mezi `40` až `60` považována za „neutrální“. Tyto hodnoty se dají brát jako „základní linie“, které jsou stanovené v běžných EEG měřících technikách (základní linie ThinkGear a běžného EEG se ovšem mohou lišit). Hodnoty od `60` do `80` se berou jako „mírně zvýšené“. Hodnoty od `80` do `100` jsou považovány za „zvýšené“. Stejně tak naopak. Hodnoty v rozmezí `20` až `40` se označují jako „snížené“ a hodnoty v rozmezí `1` až `20` jsou označovány jako „silně snížené“. Důvodem velkých rozsahů pro jednotlivé druhy je, že některé části eSense algoritmu se dynamicky učí a někdy se přizpůsobují přirozenému kolísání a trendům každého uživatele.

Speciálním případem je hodnota `0`, která značí, že ThinkGear není schopen vypočítat hladiny eSense s rozumnou spolehlivostí (např. kvůli nadměrnému šumu).

4.1.3 ATTENTION eSense

Jedná se o kladnou hodnotu o velikosti jednoho bytu, která hlásí aktuální eSense soustředěnost uživatele. Hodnota se pohybuje v rozmezí `0` až `100`. Ve výchozím nastavení je tento výstup povolen a obvykle probíhá jednou za sekundu.

4.1.4 MEDITATION eSense

Opět se jedná o kladnou hodnotu v rozsahu `0` až `100`. Důležitá je tady mentální relaxace, ne fyzická. Na druhou stranu i fyzická relaxace, v některých případech, přispívá k mentálnímu uvolnění. Relaxace je spojena se snížením aktivity mentálních procesů v mozku. Ve výchozím nastavení je tento výstup povolen a obvykle probíhá jednou za sekundu.

4.1.5 8BIT_RAW Wave Value

Hodnota je ekvivalentní k `RAW Wave Value (16b)` s tím rozdílem, že je upravena tak, aby byla bez znaménka, a na výstupu je pouze osm nejvýznamnějších bitů. Díky tomu dosáhneme rychlejšího přenosu, ale také nižší přesnosti. Ve výchozím nastavení je výstup zakázán. K dispozici je pouze v ThinkGear modulech. Není k dispozici v ThinkGear ASIC.

4.1.6 RAW_MARKER Section Start

Nejedná se o reálnou hodnotu a primárně se využívá pro ladění časování a synchronizace hrubého signálu. Hodnota je vždy `0x00`. Ve výchozím nastavení je tento výstup zakázán (jinak probíhá jednou za sekundu).

4.1.7 RAW Wave Value (16b)

Tato hodnota se skládá ze dvou bytů a představuje jeden vzorek hrubého signálu. Nabývá celočíselných hodnot v rozmezí `-32768` až `32767`. První byte představuje bity vyššího řádu, druhý představuje bity nižšího řádu. Rekonstrukce hodnoty probíhá následovně:

```
short raw = Value[0]*256 + Value[1];
if( raw >= 32768 ) raw = raw - 65536;
```

Každý model ThinkGear hlásí informace o hrubém signálu jen v určitých oblastech plného rozsahu. Ve výchozím nastavení je výstup tohoto údaje zakázán. Pokud je povolen, probíhá jednou za `2ms`.

4.1.8 EEG_POWER

Tato hodnota představuje aktuální velikost osmi běžně uznávaných typů EEG frekvenčních pásem (mozkových vln). Skládá se z osmi čtyř-bytových čísel s plovoucí desetinnou čárkou v následujícím pořadí: delta (`0.5 – 2.75Hz`), théta (`3.5 – 6.75Hz`), nízká alfa (`7.5 – 9.25Hz`), vyšší alfa (`10 – 11.75Hz`), nízká beta (`13 – 16.75Hz`),

vysoká beta (18 – 29.75Hz), nízká gama (31 – 39.75Hz) a střední gama (41 – 49.75Hz).

Nemají žádné jednotky, proto mají význam pouze při porovnávání mezi sebou (posuzování množství a výkyvů). Ve výchozím nastavení je výstup tohoto údaje zakázán. Pokud je povolen, probíhá jednou za sekundu. Tato verze je k dispozici pouze v ThinkGear modulech. Pro ASIC viz ASIC_EEG_POWER_INT.

4.1.9 ASIC_EEG_POWER_INT

Jde o ASIC ekvivalent EEG_POWER. Mezi nimi je pouze jeden rozdíl a to, že se tahle datová struktura uloží jako série osmi tří-bytových celých čísel bez znaménka (ve formátu big-endian). Ve výchozím nastavení je výstup povolen a typicky probíhá jednou za sekundu.

4.1.10 Eye Blink Strength

Tato hodnota je v současné době k dispozici pouze přes ThinkGear Serial Stream API. Není k dispozici jako výstup ThinkGear hardwaru.

4.1.11 Mind-wandering Level

Jde o kladnou celočíselnou hodnotu o velikosti jednoho bytu. Hlásí intenzitu Mind-wandering úrovně. Hodnota je v rozmezí 0 až 10. Hodnota 0 znamená, že je úroveň nedostupná. Hodnota 1 až 10 je konkrétní úroveň (čím vyšší hodnoty, tím vyšší úroveň).

4.2 ThinkGear pakety

ThinkGear posílá data prostřednictvím ThinkGear paketů (dále jen paketů). Transportní médium může být UART, serial COM, USB, bluetooth, soubor nebo jakýkoli jiný mechanismus, který může přenášet byty. Pakety jsou odesílány jako asynchronní sériový proud bytů. Abychom dokázali správně extrahovat posílané hodnoty, potřebujeme znát strukturu paketu. Každý paket je rozdělen na tři části: hlavička (header), tělo paketu (payload) a kontrolní součet těla paketu (payload checksum). Formát paketů je navržen tak, aby posílání dat bylo spolehlivé.

```
[SYNC] [SYNC] [PLENGTH]  [PAYLOAD...]     [CHKSUM]
________Header_________  __Payload__  __Checksum__
```

Hlavička paketu obsahuje tři byty. Dva synchronizační, které se používají k signalizaci příchodu nového paketu. Jsou to byty s hodnotou 0xAA (desítkově 170). Třetí byte v hlavičce udává délku, v bytech, těla paketu. Může nabývat hodnot od 0 do 169. Jakákoli vyšší hodnota signalizuje chybu. Celková délka paketu bude vždy [PLENGTH] + 4.

Checksum musí být použit k ověření integrity paketu. Checksum zjistíme tak, že sečteme všechny byty těla paketu, ze součtu se vezme pouze osm nejnižších bitů a na nich je provedena bitová inverze. Jestliže se námi vypočtená hodnota neshoduje s hodnotou checksumu, paket je chybný. Jestliže kontrolní součty souhlasí, můžeme přejít na rozdělování obsahu těla paketu.

4.2.1 Struktura těla paketu

```
([EXCODE]...) [CODE]   ([VLENGTH])   [VALUE...]
____ Value Type ____   __Length__   ___Value___
```

Tělo paketu je rozděleno na skupiny, které se nazývají `DataRow`. Každý `DataRow` obsahuje informace o tom, co data reprezentují, délku dat a konkrétní data. Je potřeba analyzovat každý `DataRow`, abychom se ujistili, že je tělo paketu v pořádku.

Každý `DataRow` může začínat žádným nebo více `[EXCODE]` byty, které obsahují hodnotu `0x55`. Počet těchto bytů označuje `Extended Code Level`. `Extended Code Level`, podle pořadí, se používá ve spojení s `[CODE]` bytem pro určení, který typ dat `DataRow` obsahuje. Například, při `Extended Code Level` na úrovni `0` a `[CODE]` bytem s hodnotou `0x04`, znamená, že `DataRow` obsahuje úroveň soustředění.

Pokud je `[CODE]` byte mezi `0x00` a `0x07`, pak `[VALUE]` byte je o velikosti `1`. V tomto případě není žádný `[VLENGTH]` byte. Je-li však `[CODE]` větší než `0x07`, pak `[VLENGTH]` byte obsahuje číslo, představující počet bytů v `[VALUE…]`.

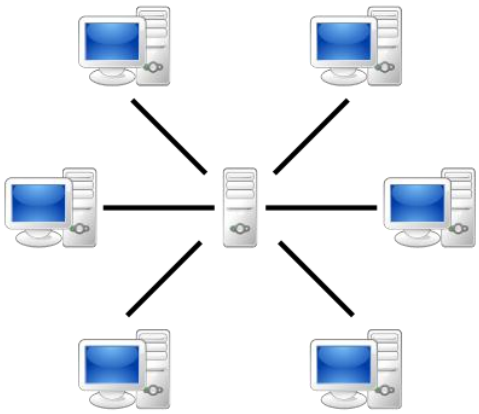
Takhle složitě je definován `DataRow` jen kvůli tomu, aby řádně vytvořené programy, v případě, že budou přidány nové druhy `[CODE]` bytů, neměly v budoucnu problém s rozdělováním těla paketu.

Tabulka hodnot:

Extended Code Level	[CODE]	(Byte) [LENGTH]	Data Value Meaning
0		0x02	- POOR_SIGNAL Quality (0-255)
0	0x03	-	HEART_RATE (0-255)
0	0x04	-	ATTENTION eSense (0 to 100)
0	0x05	-	MEDITATION eSense (0 to 100)
0	0x06	-	8BIT_RAW Wave Value (0-255)
0	0x07	-	RAW_MARKER Section Start (0)
0	0x80	2	RAW Wave Value (16-bit)
0	0x81	32	EEG_POWER
0	0x83	24	ASIC_EEG_POWER
0	0x86	2	RRINTERVAL
Any	0x55	-	NEVER USED (reserved for [EXCODE])
Any	0xAA	-	NEVER USED (reserved for [SYNC]

5 Síťová komunikace

5.1 Klient-server



7. Architektura klient-server

Klient-server je síťová architektura, která rozděluje úlohu mezi klienta a server. Tato architektura pracuje na aplikační vrstvě. Využívá se zde protokolů předchozí úrovně, protokolu TCP a protokolu UDP. Aplikace, které jsou založené na architektuře klient-server, obsahují jak program klienta, tak program serveru. Klient a server spolu komunikují přes počítačovou síť. Klient má ve většině případů, v dnešní době, nějaké grafické uživatelské rozhraní. Komunikace mezi klienty nemusí být bezpečná, proto je vhodné doplnit program klienta a serveru o šifrování odesílaných zpráv a paketů.

Architektura klient-server popisuje chování mezi dvěma procesy, kdy jeden proces (klient) posílá požadavek druhému procesu (server). Klient pouze zobrazuje vhodným způsobem data, která se pro zpracování, musí nejdříve odeslat na server, poté buď čeká na odpověď, nebo pokračuje dál ve své práci. Server požadavek zpracuje a někdy klientovi zpětně posílá odpověď, pokud je požadována (odpověď na dotaz nebo výsledek zpracování). Klient-server je alternativou k architektuře peer-to-peer (P2P = rovný s rovným). Jednotliví klienti spolu komunikují přímo a nepotřebují k tomu server. V P2P každý klient může fungovat jako klient i jako server.

Výhodou architektury klient-server je, že se klient nemusí starat o vyřízení svého požadavku, stačí, když porozumí odpovědi od serveru. Další výhodou je lepší údržba serveru, protože klient a server na sobě nejsou závislí. Aktualizování údajů je mnohem jednodušší než u P2P, protože k aktualizaci dochází pouze na jednom místě, a to na serveru.

Nevýhodou architektury klient-server je přetěžování sítě a malá robustnost. Se serverem totiž může najednou komunikovat několik klientů, díky tomu může lehce dojít k přetížení serveru. Navíc, pokud dojde k výpadku serveru, požadavky jednotlivých klientů nemohou být vyřízeny a klienti často čekají, než se server opět spustí.

5.2 Protokoly transportní vrstvy

Poskytují komunikační službu mezi vrstvou síťovou a aplikační. Existují dva protokoly na transportní vrstvě a to protokol TCP a UDP.

5.2.1 Protokol TCP

Jedná se o spolehlivý protokol, který zaručuje bezchybnost přenosu paketů, včetně jejich správného pořadí. Jednotkou je zde paket. Využívá opětovné posílání dat, potvrzování o přijetí a překročení časového limitu. Během přenosu nejsou žádná data ztracena. Jestliže ovšem několikrát za sebou vyprší časový limit, spojení je ukončeno. Jeho použití je vhodné pro aplikace, které potřebují, aby data přišla v pořádku, a na rychlosti jim tolik nezáleží.

5.2.2 Protokol UDP

UDP protokol je označován jako nespolehlivý, protože nezaručuje bezchybnost přenosu dat. Jednotkou, se kterou se zde pracuje, je datagram. Jedná se o paket, jednotku zprávy, nespolehlivého přenosu.

Bezchybným přenosem je myšleno správné pořadí přijatých datagramů, stejný obsah odeslaného a přijatého datagramu a to, že datagram nebude přijat vícekrát. V případě, že dojde k chybě a záleží na správnosti doručení datagramů, musí problém vyřešit aplikační vrstva. Ošetřit chybové stavy lze pomocí přidání dat do odesílaného paketu. Konkrétně jde o označení id a kontrolní součet.

UDP je jednodušší protokol než protokol TCP. UDP je vhodné použít v systémech, ve kterých nevadí, že se nějaký datagram občas ztratí. V takovýchto systémech je UDP velice vhodným protokolem, protože neztrácí čas opětovným posíláním chybných datagramů nebo navazováním spojení, apod. Příkladem použití může být živý přenos nějakého zápasu nebo některá z online her. Nevadí nám, že se obraz občas zasekne nebo lehce zdeformuje.

Implementace hry vyžaduje návrh a implementaci serveru, který bude zpracovávat data od klientů a bude uchovávat konfigurace rozehraných her kvůli možným výpadkům. Jeho komunikační protokol přikládám v příloze.

Rozhodl jsem se dělat server s využitím TCP protokolu. Tenhle protokol jsem si vybral, protože je pro tenhle druh aplikací vhodnější. Navíc je s ním i méně práce, právě díky tomu, že se TCP protokol o vše stará na pozadí.

6 Knihovna Qt

Hra vyžaduje grafické uživatelské rozhraní. V požadavcích bylo, že se má jednat o multiplatformní aplikaci, která je napsaná v jazyce C/C++. Rozhodl jsem se tedy použít Qt framework, který mi ulehčil práci a zajistil přenositelnost na různé platformy. Kromě grafického uživatelského rozhraní, dále jen GUI, jsem knihovnu použil i pro zprovoznění komunikace mezi čelenkou a počítačem.

Programovací jazyk C++ vznikl rozšířením jazyka C. Nejde o čistě objektově programovací jazyk. Podporuje více programovacích stylů. Proto dokáže být velmi složitý a nepřehledný. Knihovna Qt hodně ulehčuje a zpřehledňuje celý kód. Jednou z největších výhod je přítomnost signálů a slotů pro komunikaci mezi objekty.

Qt je nejznámější multiplatformní knihovna pro jazyk C++. Nejčastěji se používá pro vytváření programů s grafickým uživatelským rozhraním. Qt existuje, kromě jazyka C++, i pro řadu dalších programovacích jazyků. Pomocí téhle knihovny byly vytvořeny programy Skype, Opera, VirtualBox a další. Celý framework je zdarma dostupný pod licencí GPL a LGPL nebo za poplatek pro komerční použití.

Součástí Qt SDK je Qt Creator. Jedná se o multiplatformní IDE, které kromě zvýrazňování syntaxe, funkce debugování a spousty dalších funkcí, ulehčuje vytváření GUI aplikací díky integrovanému Qt Designeru.

Pro vytvoření projektu a jeho přeložení musí být na daném počítači nainstalováno Qt společně s překladačem MinGW (překladač není nutně zapotřebí, pokud budete soubory překládat jinak). Vytvoření a překlad zdrojových souborů bez IDE, probíhá následovně. Zdrojový soubor lze vytvořit v obyčejném textovém editoru. Výsledný soubor musí obsahovat příponu podle zvoleného programovacího jazyka. Zdrojové soubory musí být ve složce se stejným názvem, jako je název daného projektu. Nyní stačí zapnout příkazovou řádku a přepnout se do daného adresáře, ve kterém jsou dané zdrojové soubory. V tomhle adresáři pak musíme použít sekvenci následujících příkazů:

1) `qmake -project`
   - tento příkaz vytvoří soubor s projektem (soubor s příponou `.pro`). Do něho se zapisuje, které části Qt chceme v projektu využívat, jak se má jmenovat spustitelný binární soubor, jestli se jedná o aplikaci s GUI nebo pouze o konzolovou aplikaci (či obojí), kde se nacházejí hlavičkové a zdrojové soubory vzhledem k souboru s příponou .pro, a další informace jako používané knihovny, flagy, atd.
2) `qmake „název projektu".pro`
   - tento příkaz slouží pro vytvoření `makefilu` a potřebné adresářové struktury. Název projektu bude stejný jako název složky, ve které se nacházíme.

3) `mingw32-make (pro Windows), make (pro Linux)`
   - tento příkaz slouží pro přeložení celého projektu do spustitelné podoby. Využijí se k tomu `makefily` vytvořené v předchozím kroku. Spustitelný soubor se po vykonání tohoto příkazu bude nacházet ve složce `release`, která byla vytvořena po druhém příkazu.

6.1 Signály a sloty

Slot je funkce, která je zavolaná při vyslání konkrétního signálu. Propojení signálu a slotu zajišťuje metoda `QObject::connect()`. Vyslání signálu se provádí pomocí klíčového slova `emit`. Každá třída, která chce využívat signály a sloty musí zadat při své deklaraci `Q_OBJECT` macro. Dále musí dědit od třídy `QObject`. Při překladu se pak projdou všechny hlavičkové soubory a v případě, že se narazí na `Q_OBJECT` macro, pro danou třídu se vytvoří zdrojový `moc_` soubor, který se přeloží a přidá k ostatním objektovým souborům. V `moc_` jsou důležité informace pro fungování nadefinovaných signálů a slotů. Po přidání třídy s `Q_OBJECT` macrem musíme provést nad celým projektem příkaz `qmake`, kvůli vytvoření (či úpravě) daného `moc_` souboru.

```
class Priklad: public QObject
{
  Q_OBJECT

  public:
    Priklad(): QObject()
    {
     QObject::connect(this, SIGNAL(mySignal()), this, SLOT(mySlot()));
     emit mySignal();
    };

  public slots:
    void mySlot()
    {
     printf("Hello World!\n");
    };

  signals:
    void mySignal();
};
```

6.2 QWidget

`QWidget` je hlavní třídou pokud chcete pracovat s GUI. Dědí od ní veškeré grafické komponenty. Jestliže chcete vytvořit svojí vlastní komponentu, stačí vytvořit třídu, která bude dědit od třídy `QWidget`. Já jsem si vytvořil svojí vlastní komponentu `Graph`, která zobrazuje přijatá data z čelenky MindWave Mobile v podobě křivek.

7 Výběr deskové hry

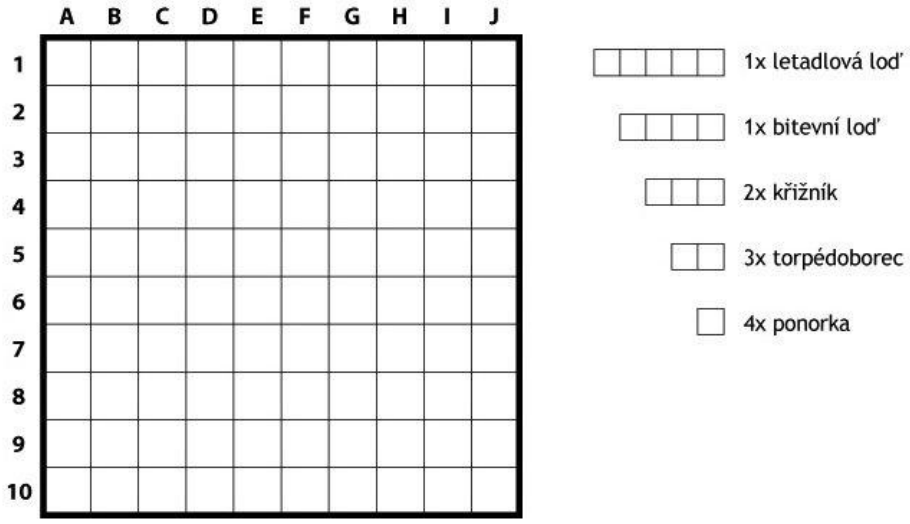
Desková hra je druh hry, jejíž průběh je realizován na herním plátnu. Obvykle jsou deskové hry pro více hráčů. Herní plátno je místo, kam hráči umisťují figury, žetony, atd. a poté s nimi pohybují. Mezi nejznámější deskové hry patří Šachy, Go, Člověče nezlob se, Monopoly, Dáma, Mlýn, Lodě, Scrabble a další.

Jedním z úkolů mé bakalářské práce bylo navrhnout deskovou hru pro dva hráče, která by užitečným způsobem mohla využívat čelenku MindWave Mobile a byla by tím pádem vhodná pro kognitivní trénink. Kognitivním tréninkem je myšleno cvičení kognitivních funkcí, mezi které patří pozornost, paměť, logika, řeč a prostorová orientace. Mého zadání se ovšem týká hlavně schopnost soustředění a meditace.

Chtěl jsem si zvolit známou, pro mě zábavnou, deskovou hru, která by mohla být rozšířena o spoustu funkcí, které by udělaly zvolenou hru o hodně zábavnější. Takovou hru, která by hráče zaujala a zároveň je trénovala. U všech her, které mě z počátku napadly, kromě jedné, jsem nepřišel na nějaké větší úpravy, které by hru udělaly zábavnější a zároveň by mohly sloužit pro kognitivní trénink. Například u her šachy a dáma bych mohl udělat akorát ochránění políček, nebo figurek, na základě soustředěnosti. A i to by bylo problematické z hlediska provedení a následných pravidel pro oba hráče.

Nejvíce ze všech se mi zalíbila hra Lodě. Četl jsem, že se jedná o jednu z nejznámějších her na světě. Mě osobně přijde, že tahle hra už dávno není tak rozšířená jako kdysi. Přitom se jedná o hru, která je určena pro všechny věkové kategorie. Bral jsem to tedy i jako výzvu, která by mohla lidi vrátit k této hře.

8 Hra Lodě



8. Ukázka plátna a nabídky lodí

Původně jste ke hře potřebovali pouze papír (nejčastěji čtverečkovaný) a tužku. Původně píšu, protože v dnešní době existuje spousta způsobů jak si hru Lodě zahrát. Existují různá plastová vydání nebo si naopak hru Lodě můžete zahrát na počítači v podobě nějaké online hry. Já se budu zabývat popisem původní verze.

Hra probíhá na dvou plátnech. Na prvním si hráč zvolí umístění svých lodí, druhé slouží pro zápis informací o stavu plátna soupeře. Hráči volí taktiku, se kterou vkládají lodě na vlastní plátno a taktiku, kterou vybírají pole u soupeře. Cílem hry je zničit celou soupeřovu flotilu dříve, než soupeř zničí flotilu hráče.

Hráči si na začátku zvolí velikost plátna, typ a počet lodí. Poté si načrtnou prostor pro své plátno a pro plátno soupeře. Plátna mají stejný počet řádků a sloupců a obsahují souřadný systém, kdy sloupce představují písmena a řádky čísla. Zvolením písmena a čísla vznikne souřadnice, která představuje konkrétní políčko na plátně. Každý hráč si rozmístí všechny lodě na své plátno. Jednotlivé lodě se nesmí dotýkat hranou a mohou se libovolně otáčet. Jakmile oba hráči rozmístí na plátno všechny lodě, začíná hra. První hráč ohlásí zvolené pole soupeře, na které namířil. Odpověď soupeře může být „voda“, „zásah“ nebo „potopená“. Když hráč netrefí pole, na jehož souřadnici má soupeř nějakou loď, odpověď zní „voda“. Jestliže se hráč trefí na pole, ve kterém má soupeř loď, odpovědí je „zásah“ nebo „potopená“, pokud se hráč trefil do posledního zbývajícího políčka, na kterém se loď nacházela.

Hráči si mohou označovat pole, na která neúspěšně „vystřelili“, třeba pomocí tečky, aby si udrželi přehled o již zkoušených polích. Jakmile dojde k potopení nějaké lodě, automaticky mohou tečkou označit okolí, do kterého už nemá cenu „střílet“, díky

tomu, že se jednotlivé lodě nesmí dotýkat hranou. Průběh hry může mít dvojí provedení. Buď se hráči střídají po jednom výstřelu, nebo hrají, dokud neminou. Záleží na jejich předchozí domluvě.

9 Naval Battles

Jedná se o mnou upravenou hru Lodě, kterou popisuji výše. Základní pravidla byla ponechána. Kromě jiného byla přidána možnost výběru map. Každá mapa má spousty možností, kterými si hráči mohou hru zpříjemnit.

Nemusí hrát jenom námořní bitvy, ale mohou si zvolit jiný herní mód. Ve výběru herního módu je, kromě námořních bitev, bitva na pobřeží a bitva na pevnině. Jediný rozdíl mezi módy jsou plátna, na kterých se hraje. Námořní bitva probíhá celá na vodě, pozemní bitva probíhá celá na souši a pobřežní bitva probíhá z části na pláži a z části na vodě (útočící hráč je na vodě a snaží se dobít území soupeře).

Hráči si v mapě mohou dále nastavit, jak má probíhat střídání hráčů (po jednom tahu nebo dokud neminou), čas pro celou hru a čas pro jeden tah. Kdyby totiž hra trvala moc dlouho, hráče by to spíše vyčerpávalo, než bavilo. Pokud nikdo nevyhraje do stanoveného limitu, vyhraje ten, kdo napáchal největší škody soupeři.

Kromě toho byla přidána možnost úpravy plátna v průběhu hry. To, kolikrát lze úpravu využít, si hráči opět nastavují v dané mapě. Konkrétně se jedná o to, že si hráč může přesunout lodě, které ještě nebyly ani z části odkryty. Takže v případě, že si v průběhu hry rozmyslí taktiku, může změnit vzhled plátna (včetně chráněných políček a vlajkové lodi, viz dále). Kromě přesunů lodí se celé plátno vyčistí od zásahů mimo lodě. Hráč může dále využít změnu zbylých chráněných políček a může přendat vlajku na jinou loď. Hráč musí stihnout úpravu do daného časového limitu jednoho tahu a soupeř je informován o tom, že hráč tuhle akci využil. Po změně plátna je automaticky na řadě soupeř.

Kromě těchto drobností si hráči mohou zvolit tvar a velikost plátna, na kterém chtějí hrát. Dále si nastavují seznamy lodí, jejich velikosti i tvar. Díky tomu, že se jedná o aplikaci, hráči mohou vytvářet lodě, které se skládají z částí, to by na papíře nešlo moc dobře. Navíc, každý hráč může mít jiný seznam lodí.

Poslední důležitou úpravou bylo přidání vlajkové lodě. Než hra započne, hráči musí zvolit jednu loď, která ponese vlajku. Jakmile bude daná loď zničena, hráč prohrává, ať je hra v jakémkoli stavu.

Všechny tyhle úpravy byly provedeny proto, aby si hráči mohli vytvořit různé bitvy, které jsou třeba inspirovány skutečnými. Na souš si mohou umístit děla, Trebuchety, obrané stěny, apod. Mohou tedy pokaždé hrát jinou hru a naplno zapojit fantazii.

9.1 Vylepšení s použitím MindWave Mobile

Z čelenky MindWave Mobile můžu získávat hrubý signál, soustředění a meditaci daného hráče. Na hrubém signálu je dobře vidět mrkání uživatele a rušení, které může být způsobeno špatným nastavením čelenky. Nicméně, pro tuhle hru nelze tyhle hodnoty nějak využívat. Rozhodl jsem se tedy využívat pouze soustředění a meditaci, a hodnoty hrubého signálu uživateli alespoň zobrazovat (pro případné ladění, apod.).

Abych tedy mohl využívat soustředění a meditaci hráčů, přidal jsem do hry další funkcionalitu. Hráči si mohou ochránit části svých lodí. Konkrétně to znamená, že pokud soupeř zaútočí na chráněné políčko, vznikne tzv. duel.

9.1.1 Duel

Duel je stav hry, při kterém se určitou dobu snímá buď soustředění, nebo meditace obou hráčů. Po uplynutí určité doby je určen lepší z nich. Lepším hráčem je ten, který byl lepší po delší dobu. Pokud byl lepší útočník, pole, na které střílel, se zničí. Jestliže byl lepší obránce, pole si zachránil a navíc zůstává nadále chráněné. Hráč si totiž může ochránit i políčko na vodě, aby soupeře zmátl a získal tím potřebný čas k zničení jeho flotily. Kdybych políčko po duelu zpřístupnil, trošku by ztrácelo význam chránění políček. Nejde ani tak o vysoké hodnoty jako spíš o výdrž. Pokud se jednalo o vyrovnaný souboj, určí se průměrná hodnota a lehce se zvýhodní útočník. Pokud totiž i průměrná hodnota bude stejná, vyhrává útočící hráč.

Aby čelenka nesloužila pouze k duelům, přidal jsem do hry ještě dvě herní akce. Jedná se o opravu a bombardování. Tak se budou hráči věnovat kognitivnímu tréninku skoro neustále.

9.1.2 Bombardování

Bombardování je herní akce, kterou může hráč využít pouze při dosažení určité úrovně soustředění, kterou udrží na stejné nebo vyšší úrovni po nějakou dobu. Hráč pak určí, do jaké oblasti chce provést bombardování a zvolená oblast bude kompletně zničena včetně všech chráněných políček. Po využití téhle akce je na řadě soupeř.

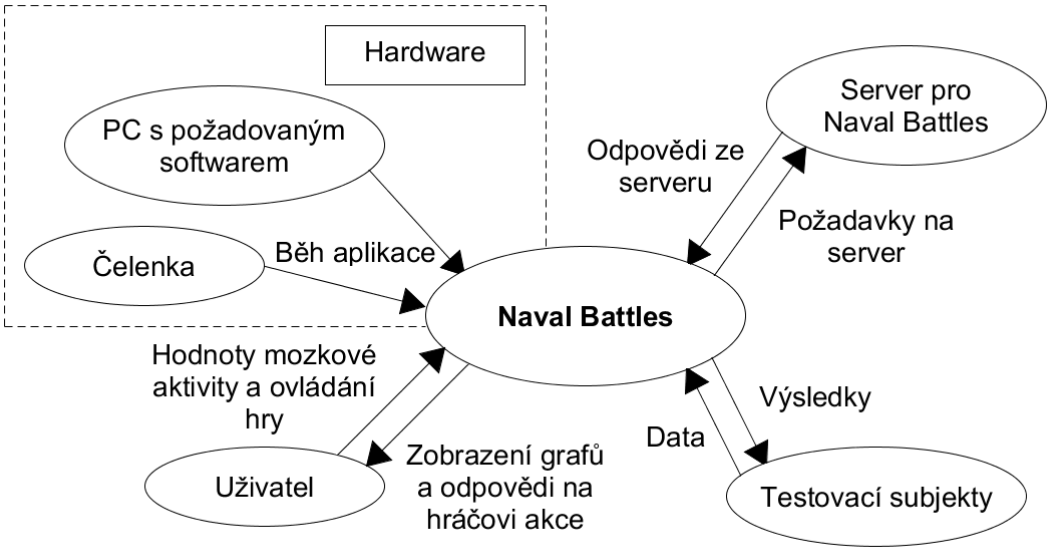
9.1.3 Oprava

Oprava je herní akce, kterou hráč může využít pouze při dosažení určité úrovně meditace a zároveň ji udrží na stejné nebo vyšší úrovni po nějakou dobu. Hráč si poté zvolí pouze jednu loď, kterou chce opravit. Při téhle herní akci má možnost navíc využít přemístění lodí, změnu zbylých chráněných políček a změnu vlajkové lodě. Po využití opravy je na řadě soupeř.

Na základě dalších modifikací byly mapy rozšířeny o další nastavení. Hráči si mohou určit počet chráněných políček, kdo chráněná políčka zadává (mohou být

chráněna všechna políčka, na kterých jsou lodě, nebo se políčka vyberou náhodně). Dále si určují, jak dlouho bude trvat případný duel, kolikrát lze využít bombardování a opravu, co se snímá při duelu (může se snímat soustředění, meditace, může se to určovat náhodně, nebo se může výběr střídat), A na závěr si určí rozsah vybombardované oblasti při využití bombardování, při jakých eSense hodnotách lze využít bombardování a opravu a jak dlouho musí hráči vydržet, aby se jim daná herní akce zpřístupnila (bombardování a oprava).

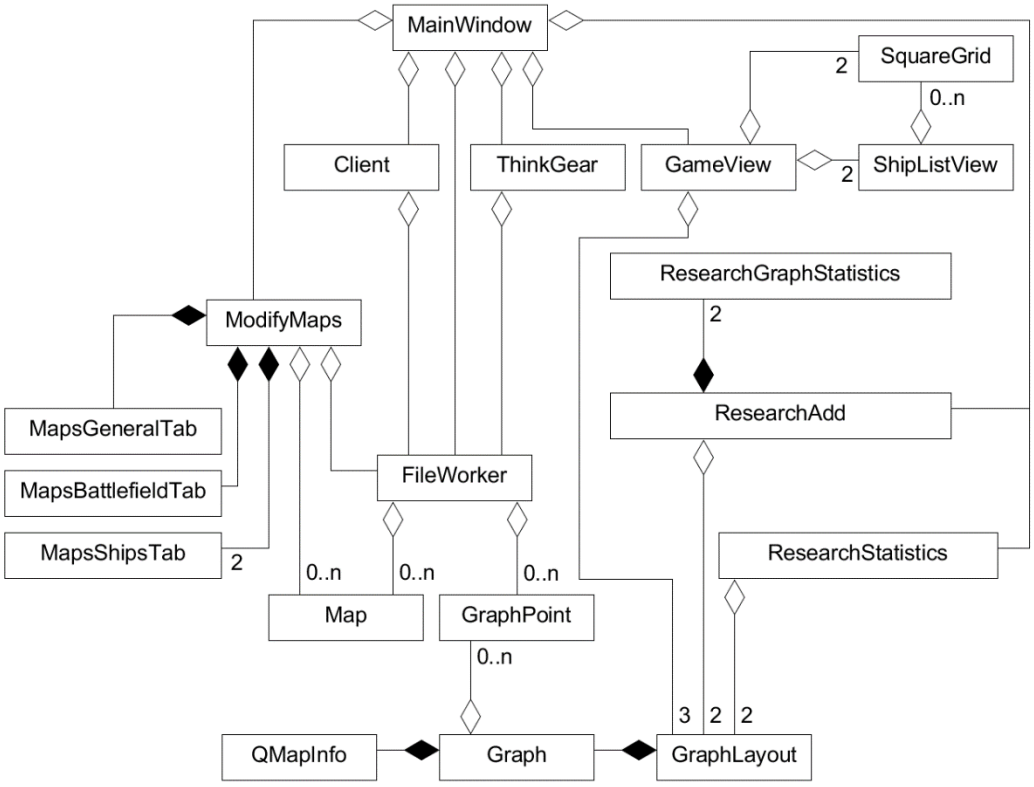
10 Implementace



9. Kontextový diagram

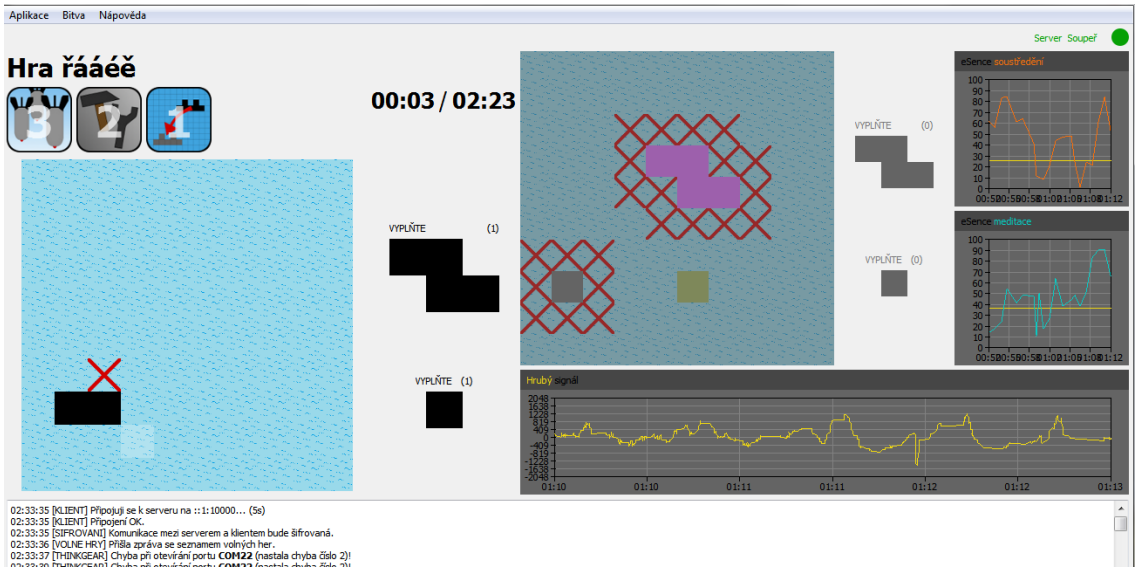
Celá práce se skládá ze dvou aplikací, klienta a serveru. Ty pro svůj běh potřebují počítač s nainstalovaným požadovaným softwarem (více informací se dočtete v `10.4 Dosažené výsledky`). Server nepotřebuje žádnou aplikaci pro svůj běh. Klient, pro svojí smysluplnou činnost, potřebuje mít spojení se zapnout aplikací serveru. Klient, kromě počítače, potřebuje pro svojí úplnou funkčnost i připojenou čelenku od firmy NeuroSky. Díky modulu pro průzkum mozkové aktivity uživatelů lze porovnávat naměřené hodnoty soustředění a meditace různých věkových kategorií a určovat vhodné soupeře.

10.1 Klient



10. UML diagram klienta

Hlavní okno představuje instance třídy `MainWindow`. Ta vytváří instance tříd `Client`, `ThinkGear`, `GameView`. Dále načítá herní mapy a informace o poslední nedohrané hře pomocí funkcí `loadGameMaps()` a `loadPreviousGame()` ve třídě `FileWorker`. Vytváří menu aplikace a upravuje průběžně stavový řádek, který se nachází v dolní části hlavního okna.



11. Průběh hry

O veškeré změny v aplikačním okně se stará třída `GameView`. Třída `GameView` si hlavní layout nastaví už při vytváření její instance. Poté pouze skrývá či zobrazuje určité grafické komponenty. Po zapnutí aplikace klienta uživatel vidí pouze tři grafy. To má na starosti funkce `zeroLayoutScenario()`. Po vytvoření hry (nebo po připojení do hry) vidí své plátno a musí si do něj rozmístit všechny lodě. Tohle zobrazení zajišťuje funkce `firstLayoutScenario()`. Když má vše správně nastaveno a klikne na tlačítko `Start`, zobrazí se mu hlavní herní plocha. To zajišťuje funkce `secondLayoutScenario()`. Herní plocha se skládá z levé, pravé, spodní a horní části. V levé části je plátno soupeře se seznamem jeho lodí, které ještě nejsou zničeny. Nad ním jsou umístěna tlačítka představující herní akce „bombardování“, „opravu lodě a přenastavení plátna“ a „přenastavení plátna“. K využití poslední zmiňované akce se hráč nemusí vůbec soustředit ani meditovat. Vedle herních akcí se nachází časomíry pro aktuální tah a pro celou hru. V pravé části je plátno se seznamem lodí hráče, které je z počátku nepřístupné. U něho jsou tři grafy zobrazující hrubý signál z EEG čelenky, soustředění a meditaci hráče. V dolní části je textové pole pro výpis informativních hlášení. V horní části (napravo) jsou kontrolky, představující spojení se serverem, soupeřem (samozřejmě přes server) a s čelenkou.

V případě duelu nastává poslední možné zobrazení a to má na starosti funkce `duelLayoutScenario()`. Vše skryje a zobrazí pouze popisek, který se nachází nad eSense grafy. K němu nechá zobrazený pouze požadovaný graf pro duel.

Plátna hráčů jsou instance třídy `SquareGrid`. Instancemi třídy `SquareGird` jsou ale i zobrazované lodě. Jejich seznam je ovšem instancí třídy `ShipListView`. U plátna jsou důležité dvě funkce ve třídě `SquareGrid` a těmi jsou `restoreBattlefield()` a `updateBattlefield()`. Druhá funkce slouží pouze k nastavení vzhledu daného tlačítka. První funkce vymaže, až na určitá políčka (chráněná políčka, zasažené a potopené části lodí), celé plátno a pomocí seznamu vytvořených lodí ho obnoví. Obnova plátna je využívána v případě, kdy se nějaká loď odebírá, nebo se ruší chráněné políčko, apod. Prostě ve stavech, kdy nejsme schopni jednoduše zjistit, jak má vypadat plátno po dané akci.

Při umisťování lodí je v téhle třídě důležitá funkce `drawShip()`, která buď zobrazí stín lodi, nebo vybranou loď umístí do plátna.

Třída `ShipListView` představuje rolovací oblast, do které se vkládají instance třídy `SquareGrid`, které představují jednotlivé lodě. Její šířka se nastavuje manuálně, neboť docházelo při jejím zobrazování k chybám. K manuálnímu nastavení šířky slouží funkce `setMinSize()`.

10.1.1 Komunikace s čelenkou

Tuhle činnost má na starosti třída `ThinkGear`. Ta navazuje a udržuje komunikaci s čelenkou NeuroSky MindWave Mobile. Jedná se o vlákno, které běží na

pozadí po celou dobu běhu aplikace. K jeho ukončení dojde až při ukončování celé aplikace.

Na začátku objekt téhle třídy čeká, než se zvolí port, který má otevřít. Poslední nastavený port je uložen v textovém souboru, ze kterého se název daného portu načítá při zapnutí aplikace. Jakmile daný port otevře a nepřichází po určitou dobu nějaká data, nejspíše byl zvolený špatný port, proto dojde k uzavření sériového portu a k opětovným pokusům o otevření, vždy po určitém časovém intervalu. Je to tak dělané z toho důvodu, že čelenka se může v průběhu hry vybít, vypnout, apod. Kdyby se vlákno automaticky neodpojilo, uživatel by musel zpátky do nastavení a změnit vybraný port, aby vlákno daný port uzavřelo a otevřelo jiný.

Jakmile vlákno otevře jakýkoli port, čeká na data z čelenky. Případ, kdy nic nepřijde po určitou dobu, byl již rozebrán výše. Pokud ale přijdou nějaká data, zavolá se funkce `readInput()`, která z portu přečte veškeré byty a uloží je do pomocného bufferu `QByteArray data`. Může být přijato několik paketů, poslední z nich nemusí být celý. V téhle funkci probíhá v cyklu jejich kontrola. Jestliže je v paketu chyba, tak se zahodí. Z pomocného bufferu se odstraní přečtené byty a pokračuje se další iterací cyklu. Pokud jsou pakety celé a v pořádku, zavolá se funkce `parsePayload()`, která získá konkrétní hodnoty z těla paketu a podle typu hodnoty (soustředění, meditace, hrubý signál či signál kvality) vyšle signál, ve kterém získanou hodnotu pošle do napojeného slotu. V případě soustředění, meditace a hrubého signálu se jedná o slot ve třídě `GraphLayout`, který data uloží do grafu a obnoví jeho vzhled. Pokud se v průběhu kontroly zjistí, že poslední přijatý paket není celý, cyklus se ukončí a opět se čeká, dokud na portu nebudou připravena nějaká data k přečtení. Algoritmem pro zkontrolování a přečtení hodnot jsem se již zabýval v kapitole `4.2 ThinkGear pakety`.

10.1.2 Zobrazení získaných dat

K tomuhle byla vytvořena třída `Graph`, která dědí od třídy `QWidget`. Jedná se tedy o grafickou komponentu. Referenci na ní obsahuje pouze třída `GraphLayout`, která ke grafu přidává další grafické komponenty, jako například popisek, a která spravuje uložená data v téhle třídě. Třída `Graph` má na starosti vykreslování přijatých dat z čelenky v podobě křivek do grafu. Kromě křivek vykresluje i souřadnicový systém. Naměřená data z čelenky se ukládají do atributu `QMap<int,QVector<GraphPoint *> > curves`. Jedná se o mapu, kde klíčem je id dané křivky a hodnotou je dynamické pole (dále jen vektor) konkrétních bodů. Dynamické pole je zde v podobě třídy `QVector`. Jedná se o třídu knihovny Qt. Do jednoho grafu se totiž může vykreslovat více křivek. Například při hře, pokud mají oba hráči nasazené čelenky, se do grafu pro soustředění zaznamenávají křivky obou hráčů. Stejně tak u grafu pro meditaci. Je pravda, že budou v aktuální verzi hry zobrazovány maximálně dvě křivky, ale pokud by se v dalších plánovaných úpravách hry rozhodlo, že se budou zobrazovat hodnoty například mozkových vln do jednoho grafu, bude tahle

třída upravována jen minimálně. Navíc, u každé křivky vždy známe její id, proto jsem se rozhodl zvolit mapu vektorů.

Jeden bod je tvořen objektem `GraphPoint`, jedná se o návrhový vzor přepravky. Obsahuje souřadnici bodu v grafu, která je dána naměřenou hodnotou a časem získání téhle hodnoty. Jednotlivé grafy musíme časově synchronizovat, aby zobrazovali ten samý časový úsek. Pro skupinu grafů (instance třídy `GraphLayout`) zjistíme počet milisekund od počátku (v IT se jedná o datum `1.1.1970`). Hodnotu pak uložíme do všech `GraphLayout` v dané skupině do atributu `unsigned long long start`. Objekt třídy `GraphLayout` před vložením naměřené hodnoty do grafu, sníží čas získání o hodnotu atributu `start`. Tímhle postupem je zajištěno, že se nám zobrazuje rozumný čas a navíc u všech grafů dané skupiny stejný.

Před vykreslováním si nejdříve musíme projít celou mapu vektorů (všechny křivky) a zjistit nejnižší a nejvyšší čas pořízení. K tomu slouží funkce `QMapInfo *getMinMaxTime()` ve třídě `Graph`. Rozdíl nejvyššího a nejnižšího času pak představuje časový rozsah na ose `x`. Pro osu y známe rozsah od vytvoření grafu. Ten se nemění. V jednom grafu chceme zobrazovat pořád jeden a ten samý druh hodnot.

Třída `Graph` nabízí ještě jednu funkcionalitu. Buď zobrazuje maximálně n nejnovějších hodnot, nebo zobrazuje všechny pořízené body. K tomuto rozlišení slouží atribut `int removing` ve třídě `Graph`.

10.1.3 Komunikace se serverem

Komunikaci se serverem zajišťuje třída `Client`. Jedná se o vlákno, které běží na pozadí po celou dobu běhu aplikace. Jeho běh se zastaví pouze při ukončení celé aplikace.

Na začátku si vytvoří soket, který se pokusí připojit k serveru. Jestliže se nepodaří navázat spojení, uživateli se zobrazí chybová hláška díky vyslání signálu `void message()` a pokus o připojení se opakuje. Po zdařilém spojení se serverem začne probíhat síťová komunikace. Vlákno umí zareagovat na jakoukoli, i nesmyslnou, zprávu ze serveru. V případě, že dojde k přerušení spojení, zkouší se, vždy po určitém časovém intervalu, opět připojit k serveru. Při běhu aplikace lze měnit adresu serveru, ke kterému se má klient připojit. Veškeré zprávy se ukládají do pomocného vektoru `vector<PreparedToSend *> toSend` a odstraňují se až v případě správného odeslání. Zde je dynamické pole v podobě třídy `vector`, která je součástí jazyka C++.

Třída `Client` obsahuje funkce, které nastavují, co se má poslat na server a funkce, které mají za úkol zpracovat přijatou zprávu a vyslat požadovaný signál, který je zpracován ve většině případů při probíhající hře ve třídě `GameView`. Ta se stará o zobrazení průběhu hry.

Více o komunikaci se serverem a o třídě `Client` se dočtete v kapitole `10.3.1 Klient`.

10.1.4 Vytváření map

Hlavní třídou tohoto modulu je `ModifyMaps`. Ta si v konstruktoru vytvoří kopii seznamu map, kterou upravuje. Jedná se o atribut `vector<Map *> maps`. Hlavním layoutem je `QTabWidget`, který zobrazí vždy pouze určitou část nastavení mapy. Buď obecné informace, plátno, seznam lodí pro prvního hráče nebo seznam lodí pro druhého hráče. Obecné informace zobrazuje a umožňuje upravovat třída `MapsGeneralTab`. U velikosti a tvaru bitevního pole jde o třídu `MapsBattlefieldTab` a u seznamu lodí jde o `MapsShipsTab`.

Mapy mohou být měněny i v textovém editoru. Stačí se přepnout do složky data ve složce s programem a otevřít soubor `maps.txt`.

Na první řádku souboru se zapisují hlavní informace o dané mapě. Na `N` dalších řádcích jsou informace o lodích z prvního i druhého seznamu. Jednotlivé mapy jsou odděleny prázdnou řádkou. Hodnoty jsou odděleny tabulátorem. Pokud uživatel upravuje nebo vytváří mapu přes grafické rozhraní, nemusí se o nic starat. Jestliže bude mapa upravována v textovém editoru a dojde ke špatnému upravení, aplikace to při načítání map zjistí a nenačte ani jednu z map. Je tedy třeba dávat pozor na pravidla, která jsou definována ve třídě `Map`.

Hlavní informace o mapě

Hlavní informace jsou uloženy do prvního řádku. První hodnotou je název mapy. Název musí obsahovat alespoň jeden znak. Jeho největší možná velikost je pak `20bytů` (jednoduchých znaků). V případě multicharakterních znaků je třeba pamatovat na to, že se s každým takovým znakem snižuje maximální velikost názvu o jedna. Druhá hodnota představuje velikost herního plátna. Tvar každého plátna se vykresluje do čtverce, který představuje `NxN` tlačítek. Minimální velikost je `10`, maximální `25` čtverečků. Třetí hodnotou je konkrétní tvar herního plátna. Jedná se buď o znak '`2`', pokud má herní plátno klasický čtvercový tvar. Jestliže je plátno tvarově upravené, třetí hodnotou je řetězec N^2 znaků. Znak '`1`' znamená prázdné pozadí, které během hry není vidět. Znak '`2`' představuje pole, na které pak hráč může umístit své lodě.

Další hodnotou je herní mód. Jedná se buď o znak '`N`', '`L`' a nebo '`C`'. Znak '`N`' znamená, že se daná bitva bude odehrávat pouze na vodě. Znak '`L`' představuje bitvu na souši a poslední znak představuje pobřežní bitvu. Jedná se o speciální druh bitvy, která probíhá z části na vodě a z části na souši. Útočník se snaží z moře dobít území druhého hráče, obránce, který se nachází na pláži.

Následuje znak '`F`' nebo '`O`'. Jedná se o střídání hráčů. Jestliže se mají hráči střídat po jednom tahu, znakem bude '`O`'. Jestliže se mají vystřídat až potom, co jeden z nich mine, znakem bude '`F`'.

Dále se na řádce nachází číslo a znak. Číslo představuje počet chráněných políček, znak potom toho, kdo chráněná políčka určuje. Mohou být určena hráčem ‘`H`’, náhodně ’`R`’ nebo mohou být chráněna všechna políčka, na kterých se nachází loď ’`A`’.

Dalšími hodnotami jsou čísla, představující, jak dlouho má trvat duel, kolikrát lze využít herní akci bombardování, změnu plátna a opravu lodě, čas pro celou hru a čas pro jeden tah.

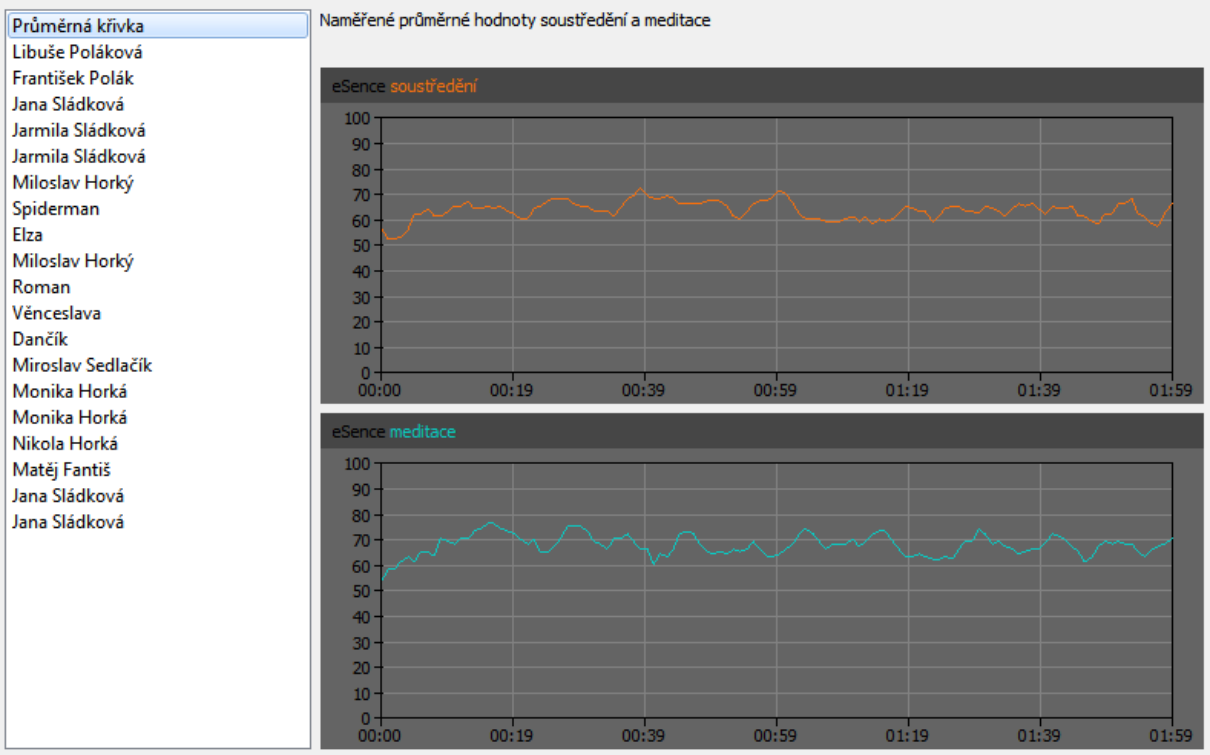
Následuje znak, představující to, co se má snímat v probíhajícím duelu. Může se jednat buď o soustředění ‘`A`’, meditaci ‘`M`’, náhodné určení ‘`R`’, nebo o střídání snímání soustředění a meditace ‘`C`’.

Poté je opět skupina číselných hodnot, které představují rozsah bombardování, při jakých hodnotách soustředění lze využít herní akci bombardování, při jakých hodnotách meditace lze využít opravu lodě, jak dlouho musí mít hodnotu stejnou či vyšší, aby se jim zpřístupnila daná herní akce, počet lodí v seznamu pro prvního hráče, který zakládá hru a počet lodí v seznamu pro druhého hráče, který se do hry připojuje. Jestliže jsou oba seznamy lodí shodné, místo počtu lodí v seznamu druhého hráče je `-1`.

Informace k lodím

Každý řádek se skládá ze šesti částí a představuje jeden druh lodě. První část představuje jméno lodě (stejná pravidla, jako u názvu mapy), druhá, délku lodě (horizontální rozměr), třetí, šířku lodě (vertikální rozměr), čtvrtá, jestli se loď dá otáčet (pokud ano, napsaným číslem bude 1, jinak 0), pátá, kolik lodí tohohle typu se musí vložit do herního plátna, a šestá, řetězec představující podobu lodě (stejná pravidla, jako u tvaru plátna v prvním řádku).

10.1.5 Statistiky průzkumu mozkové aktivity



12. Statistiky

V tomto okně je seznam již naměřených osob (subjektů). K němu je přidána `Průměrná křivka`, na kterou když uživatel klikne, zobrazí se mu průměrné hodnoty soustředění a meditace od všech měřených subjektů. Subjekty prozatím nejdou smazat. Okno slouží pouze pro informativní výpis hodnot, které se pak mohou použít k výběru vhodných soupeřů.

10.1.6 Přidání dat k průzkumu mozkové aktivity

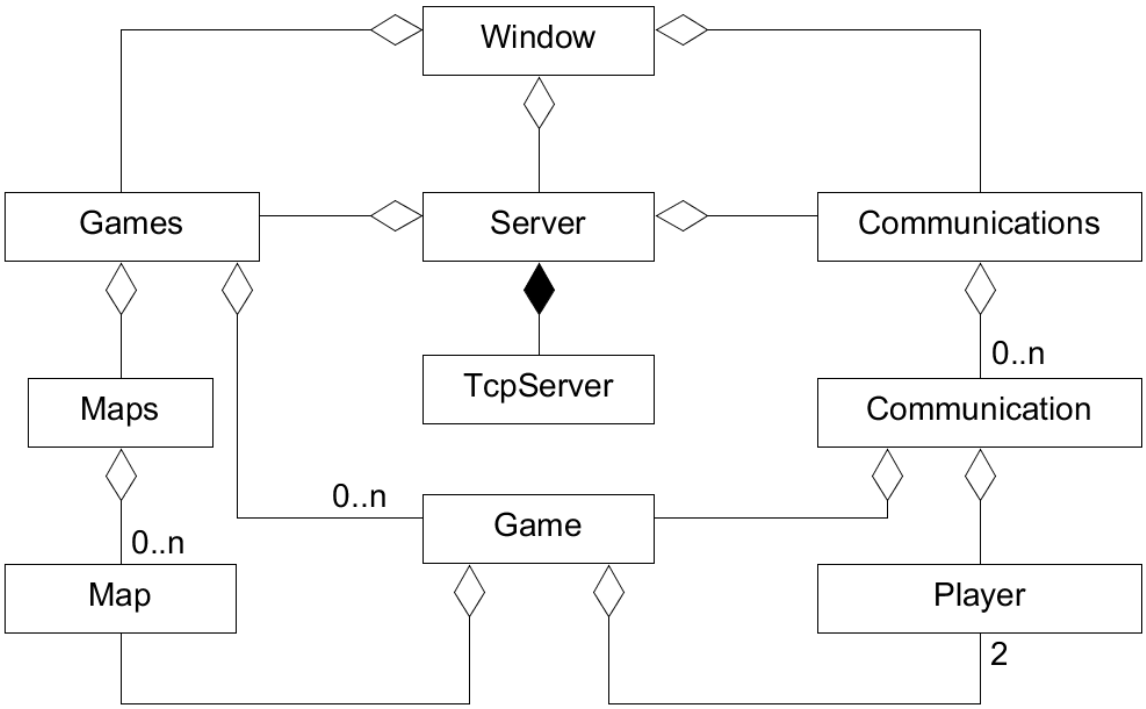
Zde je opět zobrazen graf pro soustředění a pro meditaci. Nejsou přímo napojeny na grafy v hlavním okně. Začnou zaznamenávat hodnoty až po spuštění testu. Celá funkcionalita tohohle modulu je ve třídě `ResearchAdd`. Ta přijímá signály ze třídy `ThinkGear`, které vysílají získaná data z čelenky. Jestliže měření není zapnuté, se získanými daty nic nedělá (na rozdíl od grafů v hlavním okně, které přijímají a zobrazují každou hodnotu). V opačném případě ukládá získané hodnoty do konkrétního grafu. Kromě grafů a jednoduchých komponent zobrazuje komponenty pro přehled základních údajů z daného měření. Ke každému grafu je jedna taková komponenta, která se nachází v pravé části. Komponentou je třída `ResearchGraphStatistics`. Z právního hlediska je pro jistotu přidán checkbox, jehož zaškrtnutím uživatel souhlasí se zpracováním naměřených i osobních údajů. Kontrola vyplněného jména, stav měření a zaškrtnutí checkboxu je ve funkci `controlSaveEnable()`, která v případě splnění všech podmínek, zpřístupní tlačítko pro uložení naměřených údajů.

Do textového souboru se na první řádku zapíše jméno, na druhou věk a na třetí řádku se poté ukládají naměřené hodnoty soustředění ve formátu:

<CAS><MEZERA><PRVNI_HODNOTA><TABULATOR><CAS><MEZERA><DRUHA_HODNOTA>, atd.

Při ukládání hodnot meditace jde o úplně stejný princip, akorát se hodnoty ukládají na čtvrtém řádku. Výsledky dalšího testovaného subjektu jsou odděleny prázdnou řádkou.

10.2 Server



13. UML diagram serveru

Server slouží ke zprostředkování komunikace mezi klienty a k uložení stavu jejich her. Dále zde probíhá řízení hry. Zařizuje střídání a zpracování akcí od hráčů, jejichž výsledky pak posílá na soupeře. Může na něm běžet několik her. To, kolik her jde spustit, je dáno konstantou `MAX_PENDING_CONNECTIONS` ve třídě `Server`. Všechny přijaté a odeslané zprávy zapisuje do logu, který se nachází ve složce `logs`. Soubor se jmenuje `log.txt` a aplikace jeho obsah nikdy nemaže (pouze přidává data na konec souboru).

Při zapnutí a ukončení aplikace se do souboru `log.txt` vždy zapíše:

```
START SERVER: <DATUM> <CAS>
END APPLICATION: <DATUM> <CAS>
```

Přijaté a odeslané zprávy poté zapisuje v nezašifrované podobě ve formátu:

```
Received message from <ADRESA>:<PORT> <ZPRAVA>
Posted message to <ADRESA>:<PORT> <ZPRAVA>
```

Hlavní třídou je třída `Window`. Ta na začátku vytvoří instance tříd `Statistics`, `Games` a `Communications`.

Objekt třídy `Statistics` slouží pro uchovávání statistik serveru a práci s nimi. Určuje, jak se má zobrazit čas běhu serveru, počet všech a aktuálně držených spojení, počet přijatých a odeslaných zpráv, velikosti zpráv, apod. Výsledné řetězce vysílá

v signálech, které si odchytává objekt třídy `Window` a upravuje obsah konkrétních labelů. Jedná se o skupinu popisků, které se nacházejí v levé dolní části hlavního okna. Jelikož se ke statistikám přistupuje z více komunikačních vláken, je třeba řešit synchronizaci, ke které zde využívám semafor. Synchronizace probíhá v každém systému trochu jinak a každý systém obsahuje své vlastní knihovny pro synchronizaci. Aby tedy aplikace splňovala podmínku multiplatformního softwaru, využívám zde synchronizační třídu knihovny Qt, `QSemaphore`.

Objekt třídy `Games` slouží pro uložení a práci s veškerými vytvořenými hrami. Hry jsou ukládány do jednorozměrného dynamického pole. Pomocnými atributy pro manipulaci s polem jsou `unsigned int allocated` (velikost pole) a `unsigned int occupied` (počet obsazených pozic). Datová struktura pole je zde použita z toho důvodu, že se nepočítá s tím, že by na serveru běželo velké množství her a také kvůli tomu, že každá hra má přiděleno unikátní ID, pomocí kterého se jednoznačně přistupuje k indexu v poli, na kterém hra leží.

Kromě toho vytváří strukturu `MAPS`. V ní jsou uloženy všechny mapy běžících her. Mapy jsou uloženy ve vektoru (dynamické pole). Aby zde nedocházelo k plýtvání pamětí, je každá mapa reprezentována unikátním řetězcem, který je reprezentován atributem `char *settings`. Žádná mapa se stejnou konfigurací tedy neexistuje dvakrát.

Při vytváření nové hry se nejdříve musí vytvořit řetězec z přijaté zprávy (viz `Příloha C, Komunikační protokol`) představující identifikátor herní mapy.

Ten je složen z uživatelem nastavených hodnot pro konkrétní mapu. Daný identifikátor se pak odesílá při připojení soupeře do jím spuštěné aplikace, která si identifikátor zpracuje a vytvoří si podobu mapy na své straně.

Následně se projde seznam map. Pokud existuje mapa se stejným nastavením, zvýší se u ní atribut `int number`, který představuje počet her, které ji využívají. Jestliže mapa se stejným nastavením neexistuje, dojde k přidání nové herní mapy do vektoru. Dále se musí najít volný index v poli her, na který se může nově vytvořená hra uložit. Jestliže je pole plné, dojde k jeho zvětšení. Na základě zvoleného indexu se určí herní ID. To je vypočteno jako `index + ID_FIRST_GAME`. Vytvoření hráči pak mají ID `(index + ID_FIRST_GAME) * 10 + 1` a `(index + ID_FIRST_GAME) * 10 + 2`. Jedná se o velmi jednoduché vytvoření jednoznačných identifikátorů.

Pokud hra doběhla dokonce nebo je jeden z hráčů nepřítomný příliš dlouho, dojde k jejímu odstranění z pole. Kvůli herním identifikátorům a jednoznačnému přístupu k daným indexům v poli, musí hry zůstat na stejných pozicích. Tím pádem se pole her může pouze ořezávat a to v případě, že došlo k odstranění hry na posledním indexu. To ovšem tolik nevadí, protože namísto odstraněných her se vkládá hodnota `NULL` a při vkládání nových her se nejdříve obsazují dříve uvolněné pozice.

K atributům třídy `Games`, konkrétně k dynamickému poli, se přistupuje z komunikačních vláken. Stejně jako u třídy `Statistics`, řeším synchronizaci pomocí semaforu.

Objekt třídy `Communications` obsahuje pouze vektor vytvořených objektů třídy `Communication` a funkce pro práci s ním. Objekt třídy `Communication` pak představuje jednu komunikaci s klientem. Jedná se o vlákno, které běží na pozadí po celou dobu komunikace. Jakmile dojde k ukončení komunikace, vlákno uvolní všechny jím alokované objekty na haldě a ukončí se. Aby mohla komunikace probíhat v jiném vlákně, musí se soket vytvořit v metodě `run()`. Soketu se poté musí nastavit popisek, který je vyslán signálem ze třídy `Server`.

Hlavní funkcionalita serverové části se nachází právě ve třídě `Server`. Jedná se opět o vlákno, které ale nemusí běžet po celou dobu běhu aplikace. Buď se může vlákno ukončit samo, hned na začátku vykonávání své činnosti, pokud došlo k nějaké chybě, nebo ho můžou ukončit uživatelské události z objektu třídy `Window`. Po ukončení tohoto vlákna je uživateli dána možnost spustit nové vlákno, třeba s jiným nastavením adresy a portu pro naslouchání. Aby mohl server vytvářet nová spojení, je zapotřebí vytvořit objekt třídy `QTcpServer`.

Jelikož jsem v této části nechtěl využívat signály tříd `QTcpServer`, `QTcpSocket` a rozhodl se pro vytvoření vlastních vláken, musel jsem si vytvořit pomocnou třídu `TcpServer`, která dědí od `QTcpServer` a definuje virtuální funkci `incomingConnection(qintptr)`. Ta v případě vytváření nového spojení vyšle signál `connection(qintptr)`, ve kterém pošle popisek soketu. Ve třídě `Server` je tento signál odchycen a zpracován ve funkci `newConnection(qintptr)`. Zde proběhne vytvoření nového komunikačního vlákna, kterému se právě předá získaný popisek pro vytvoření soketu. Více o komunikaci s klientem a o třídě `Server` se dočtete v kapitole `10.3.2 Server`.

Poslední velmi důležitou částí serveru je kontrola stavu her (včetně připojení hráčů), herních map a jednotlivých komunikací.

Každá hra má nějaký životní cyklus. Vzniká na požadavek hráče. Po jejím dokončení může dojít buď k opakování, nebo zániku hry. Pokud ale nesplňuje určitá kritéria, je předčasně ukončena. Kritériem je, že nesmí být oběma hráči ukončena. Dále to, že hráč nesmí být odpojený příliš dlouho. To, jak dlouho může být hráč nepřítomný, udává konstanta `TOLER_TIME`. Tahle konstanta je zavedena, protože hráči mají možnost návratu do hry. Může se totiž v průběhu hry stát, že dojde k nechtěnému odpojení hráče, například kvůli problémům na síti, výpadku elektřiny, apod. Ten pak má daný čas na to, aby se do hry připojil zpět. Pokud se tak nestane, je určen vítězem hry druhý hráč a hra je ukončena bez možnosti opakování.

Herní mapy jsou kontrolovány již při vytváření. Ovšem její odstranění jsem se rozhodl oddálit. Může se stát, že danou herní mapu už žádná hra nevyužívá (všechny

byly ukončeny). Po nějaké chvíli se ale může založit nová hra, která bude mít stejné nastavení. Pak by se zbytečně vytvářela nová mapa a celé zakládání hry by trvalo o to déle.

Komunikace mezi serverem a klientem je odstraněna po krátké chvíli, co přestane být využívána (dojde k odpojení hráče). Mohla by se sice odebrat rovnou po ukončení, do budoucna ale plánuji kontrolu komunikace a její případné zastavení. Proto jsem odstraňování komunikace vložil ke kontrole a odstranění her a herních map.

Kontrola a odstraňování probíhá ve třídě `Control`. Jedná se o vlákno, které běží po celou dobu běhu serveru. To v nekonečné smyčce, jednou za určitý čas, zavolá funkci `remove_unused_maps()`, `removeCompletedCommunications()` a `removeCompletedGames()`. Dobu, po které se provádí kontrola, udává konstanta `SLEEP_CONSTANT`.

10.3 Implementace síťové komunikace

10.3.1 Klient

Klient, kromě přijímání zpráv, může zprávy také odesílat. Přijímání zpráv ze serveru probíhá jednoduše. Jakmile nějaká zpráva přijde, projde základní kontrolou, a pokud splní podmínky konzistence, zpráva je zpracována a je vyslán signál se získanými daty. Ten si odchytí konkrétní objekt a zpracuje získaná data. Odesílání dat na server je už ale složitější. Jelikož si soket pro komunikaci vytvořilo vlákno klienta, objekt třídy `Client`, nelze k němu přistupovat z jiných vláken. Při uživatelských událostech (jedná se o jiné vlákno) tedy nelze jednoduše využít referenci na soket a poslat zprávu na server. Proto, se zprávy musí ukládat do pomocného seznamu, který se vláknem klienta vždy po krátké době vybere. Vybraná data jsou pak odeslána na server. V době odesílání dat na server klient nepřijímá žádné zprávy, protože čeká, dokud data nejsou v pořádku odeslána (jinak je ze seznamu nesmaže). Tím je zajištěno, že se žádná ze zpráv, určená pro server, neztratí. Pokud odesílání trvá příliš dlouho, nejspíše došlo k problémům na síti, takže nevadí, že klient nepřijímá zprávy (neměl by co přijímat). Čekání na zprávu ze serveru, odeslání připravených dat a kontrola změn informací pro připojení k serveru probíhá ve funkci `waiting()` ve třídě `Client`. Její obsah je vložen, i s komentáři, níže.

```
// Pokud dojde k ukonceni spojeni.
int bytes = SOCKET_END;
// Dokud je vlakno klienta aktivni a probiha spojeni se serverem.
while (active && socket->state() == QAbstractSocket::ConnectedState)
{
    // Odeslat pripravene zpravy na server.
    sendPreparedMessages();
    // Zjisteni, jestli uzivatel nezmenil adresu a port serveru.
    if (dataChanged())
    {
        // Pokud doslo ke zmenam, vyskocit ze smycky a vytvorit novy
        // soket.
        bytes = CON_DATA_CHANGED;
        break;
    }
    // Pokud nic nemenil, cekat urcitou dobu na zpravu ze serveru.
    else if (socket->waitForReadyRead(WAIT_FOR_MESSAGE))
    {
        // Pokud jsou nejake zpravy k precteni, vyskocit ze smycky a
        // vratit pocet dostupnych bytu.
        bytes = socket->bytesAvailable();
        break;
    }
}
return bytes;
```

Kromě toho řeší klient případ, kdy dojde k přečtení jen části zprávy. Vždy tedy přečte pomocí funkce `QTcpSocket::readAll()` veškeré možné znaky ze soketu,

které uloží do pomocného bufferu `QByteArray msgBufferu`. Zprávy se poté rozdělují pomocí znaku '`\n`' a rovnou se vyřizují. Pokud znak v buffer není, znamená to, že přišla jen část zprávy a zbytek teprve dojde. Zpracování dané zprávy se tedy o nějakou dobu odloží. Pokud ovšem, při dalším nahlédnutí do soketu, nejsou k dispozici nějaké znaky k přečtení, jednalo se nejspíše o špatnou zprávu, která se v takovém případě vyhodí z bufferu.

10.3.2 Server

Zde se nachází obdobný problém jako na klientovi (přístup k soketu z více vláken). Každé komunikační vlákno si vytvoří svůj soket. Jakmile se hráč připojí do hry, nastane potřeba odesílání dat na soupeře. To je vyřešeno pomocí dvou pomocných bufferů, které se nacházejí v dané hře, objektu třídy `Game`. Jedná se o `QString bufferForFirst` a `QString bufferForSecond`. Při určování hry a hráče pro dané komunikační vlákno (určování probíhá u vytvoření hry nebo připojení do hry), si vlákno určí referenci na svůj buffer, který zaplňuje soupeř (jeho komunikační vlákno), a soupeřův buffer, který má zaplňovat výsledky z hráčových akcí. Synchronizace je zde provedena pomocí semaforu `QSemaphore semaphore` ve třídě `Game`. Data od soupeře jsou odesílána v průběhu čekání na zprávu od klienta. Pokud soupeřovo komunikační vlákno přidá nějakou zprávu do bufferu v době, kdy hráčovo vlákno zpracovává nějaký požadavek, zpráva od soupeře se odešle na hráče s případnou odpovědí na požadavek. Níže je obsah funkce `Communication::waiting()`. Ta, kromě jiného, volá funkci, která odešle data od soupeře a čeká na zprávy z klienta. Dále je uveden kus funkce `Communication::sendData(bool)`, ve které se vytváří celková zpráva, která se posílá na daného klienta. Buď se na klienta nepošle nic, nebo se pošlou pouze data od soupeře, nebo se pošle pouze odpověď na daný požadavek, nebo se k odpovědi na požadavek přidají data od soupeře.

Obsah funkce `Communication::waiting()`:

```
int bytes = -1;
while (active && socket->state() == QAbstractSocket::ConnectedState)
{
    // Odeslat data od soupere.
    sendData(false);
    // Cekat na data.
    if (socket->waitForReadyRead(RECEIVE_WAITING))
    {
        bytes = socket->bytesAvailable();
        break;
    }
}
return bytes;
```

Začátek funkce `Communication::sendData(bool)`:

```
if (strlen(result) > 0)
{
    // Pridani vysledku hracovi akce. Odpoved na pozadavek.
    data.append(result);
    data.append(END_LINE);
    memset(result, '\0', MAX_LENGTH_MESSAGE + 1);
}
// Zabranit tomu, aby vlakno pro kontrolu danou hru vymazalo pri praci
// s ni a osetrit pristup k bufferu se zpravami od soupere.
games->semaphore.acquire();
if (gameExist())
{
    // Pridani zprav od soupere. V kazde casti uz je \n.
    game->semaphore.acquire();
    data.append(bufferFromOpp->toUtf8().data());
    bufferFromOpp->clear();
    game->semaphore.release();
}
games->semaphore.release();
```

Na serveru se pak řeší problém s přijetím jen části zprávy, stejně tak, jako u klienta. Řešení probíhá úplně stejným způsobem, jen se změnil název atributu, představující buffer pro přečtené byty, na `QByteArray mainBuffer`.

10.3.3 Komunikace

Jelikož se mezi serverem a klientem posílá text, který může obsahovat i mezery, třeba jména map a lodí, zvolil jsem jako oddělovač dat ve zprávě tabulátor. Každá zpráva je pak ukončena znakem '`\n`'.

Při vytvoření spojení mezi serverem a klientem se musí odeslat na klienta informace o tom, jestli bude obsah komunikace šifrován. Výměna téhle informace je zatím nezabezpečena. Je to z toho důvodu, že se nejedná o informaci, která by se dala nějakým způsobem zneužít. To, jestli má být komunikace zabezpečena, si uživatel nastavuje v aplikaci serveru. Z časového důvodu a z důvodu, že se nejedná o přenos důležitých informací, jsem se rozhodl, pro šifrování komunikace, použít jednu z jednodušších šifer. Vybral jsem Albertiho šifru. Jedná se o polyalfabetickou substituční šifru, která pro šifrování používá dvě šifrovací abecedy. Ty se při šifrování střídají. Útočníkovi by tedy nestačilo použít jednoduchou frekvenční analýzu k dešifrování komunikace. Do budoucna bych rád komunikaci šifroval složitějším algoritmem. Pro šifrování používám následující dvě abecedy.

```
YW2aiITPvo360pr_H9wJUKbxcCy71d5sthDES8NgGumfLnBel4qAFzOjkMVZX-RQ
ZmhMU_LJ7NR9wFKlHCqVtiETSdsG3kjYe5QgWuzo01AbI2n6pac8DrPvOf-BXy4x
```

Ty zahrnují čísla `0-9`, znaky mezera a pomlčka, velká a malá písmena anglické abecedy. Všechny ostatní znaky ve zprávě zůstávají v nezměněné podobě. To může útočníka zmást a ztížit mu dešifrování.

Po odeslání informace o tom, jestli bude komunikace šifrovaná, může hráč na server poslat jednu ze zpráv, které se týkají všech existujících her na serveru. Konkrétně jde o:

- Zjištění všech volných her (`FREE_GAMES`).
- Informace o nové hře, kterou chce klient vytvořit (`CREATE_GAME`).
- Název hry, ke které se chce klient připojit (`CONNECT_TO_GAME`).
- Informace o hře, ze které se hráč odpojil a ke které se chce zpět připojit (`BACK_TO_GAME`).

Jakmile klient pošle nějakou jinou zprávu, server mu na ní neodpojí a zahodí ji. Po vytvoření nebo připojení do hry, se určí reference na hru, ke které se bude přistupovat po celou dobu herní komunikace. Jelikož by mohla být hra vymazána kontrolním vláknem a nahrazena jinou, musí se, kromě reference, uložit i index do pole, na kterém se daná hra nachází a čas, kdy byla hra vytvořena. Vždy, než se přistoupí ke konkrétní hře, zablokuje se seznam her pomocí semaforu, provede se kontrola existence hry, a až poté se vykoná daná herní akce. Na server konkrétně mohou chodit následující zprávy:

- eSense hodnoty hráče (`E_SENSE`).
- Nastavené plátno a lodě hráče – začátek hry (`SET_DATA`).
- Přenastavené plátno a nové nastavení lodí – průběh hry, využití herní akce změna nebo oprava (`CHANGE_DATA`).
- Střela hráče (`SHOOT`).
- Hráč se vzdává (`GIVE_UP`).
- Hráč se rozhodl, jestli chce opakovat hru (`REPEAT`).

Na klienta mohou během celé komunikace, kromě informace o šifrování, přijít následující zprávy (pokud hra nebude nalezena, odešle se pouze `GAME_ERROR`):

- Seznam volných her (`FREE_GAMES`).
- Informace o vytvořené hře (`CREATE_GAME`).
- Informace o hře, do které se hráč připojil (`CONNECT_TO_GAME`).
- Informace o posledním stavu hry, ze kterého se hráč odpojil (`BACK_TO_GAME`).
- eSense hodnoty soupeře (`E_SENSE`).
- Soupeř se odpojil/připojil (`OPP_CONNECTION`).
- Soupeř využil herní akci změna nebo oprava (`CHANGE_DATA`).
- Informace o tom, že je hráč na řadě (`YOU_PLAY`).
- Výsledek hráčovi střely (`SHOOT`).
- Výsledek soupeřovi střely (`SHOOT_RESULT`).
- Informace k právě probíhajícímu duelu (`DUEL`).
- Soupeř se vzdal (`GIVE_UP`).
- Jestli se hra opakuje nebo ne (`REPEAT`).

Více informací naleznete v `Příloha C, Komunikační protokol`.

10.4 Dosažené výsledky

Hra byla otestována tak, aby se zjistilo, jestli je opravdu multiplatformní a jestli neobsahuje chyby. Aplikace klienta a serveru byly přeloženy a spuštěny jak na systému `Windows 7 64b`, tak na systému `GNU Linux, Debian 64b`. Aplikace, včetně komunikace mezi klientem a serverem, byla testována na několika hrách různých hráčů. Během testování se přišlo na pár chyb, které byly opraveny. Jednalo se o stavy, kdy se na server poslalo více zpráv za sebou. Zprávy se na soketu řadily do fronty. Prvních `N` zpráv se načetlo rychle, další se ale načítaly až poté, co byla z klienta odeslána další zpráva. Chyba byla v použití špatné funkce pro načítání dat ze soketu.

Dále bylo zjištěno, že pokud je aplikace zapnuta a počítač se dá do stavu hibernace, tak po obnovení systému dojde při vypnutí aplikace ke stavu, ve kterém neodpovídá, a k tomu, že konzole, spuštěná s aplikací, nejde ukončit (na `Windows 7`). Tento stav se mi ale nepodařilo opravit. Nejspíše se bude jednat o vnitřní problém Qt. Tento problém ovšem nebrání používání aplikace.

Aplikaci serveru trvá spuštění pár sekund. Jedná se o dobu, během které se zavádějí dynamické knihovny, vytvářejí se počáteční objekty a načítá se obrázek pro ikonu programu. Vypnutí aplikace serveru může trvat až `5` sekund. U serveru je vytvořeno vlákno třídy `End`, které veškeré uvolňování paměti a zastavování činností provádí na pozadí. Během této činnosti se do textového pole uživateli zobrazují kroky ukončování. Vypnutí trvá tak dlouho, protože se čeká na ukončení na pozadí spuštěného kontrolního vlákna, které se pokaždé uspí na `5` sekund. Aplikace serveru nepožaduje speciální výkon počítače. Podmínky spuštění viz `10.4.1 Podmínky překladu a spuštění aplikací`.

Doba spuštění aplikace klienta je podobná. Může trvat o trochu déle, protože se zde na začátku načítají různé textové a binární soubory, vytvářejí se složitější objekty, apod. Během hry, když dochází ke střídání hráčů kvůli vypršení časového limitu pro daný tah, může dojít k sekundovému zpoždění na straně druhého klienta. Je to kvůli spuštěnému vláknu na serveru, které zařizuje střídání a které se uspává na několik set milisekund (při dvou hrách běží dvě vlákna, takže, pokud by vlákno čekalo kratší dobu a na serveru běželo několik desítek her, docházelo by k velkému zatěžování procesoru a k větším požadavkům na výkon počítače). Vypínání může trvat podobnou dobu jako je tomu u serveru. Není zde sice kontrolní vlákno, ale musí se ukončit vlákno pro spojení se serverem a vlákno pro spojení s čelenkou. Jelikož jde o složitější aplikaci, která provádí spoustu činností na pozadí v několika vláknech, komunikuje s externím hardwarovým zařízením (čtení přibližně `600` zpráv za sekundu) a v určitých stavech hry překresluje `5` grafů celkově `170` krát za sekundu, doporučuji mít počítač alespoň s dvou-jádrovým procesorem o frekvenci `2,0GHz` a s RAM `2GB`. Další informace se dočtete v `10.4.1 Podmínky překladu a spuštění`.

10.4.1 Podmínky překladu a spuštění aplikací

Pro překlad aplikací musí být na počítači nainstalována knihovna Qt. Aplikace byla implementována pro verzi `5.5`. Volbou jiné verze může dojít k tomu, že aplikace nepůjde přeložit, nebo k tomu, že určité části aplikace nebudou fungovat.

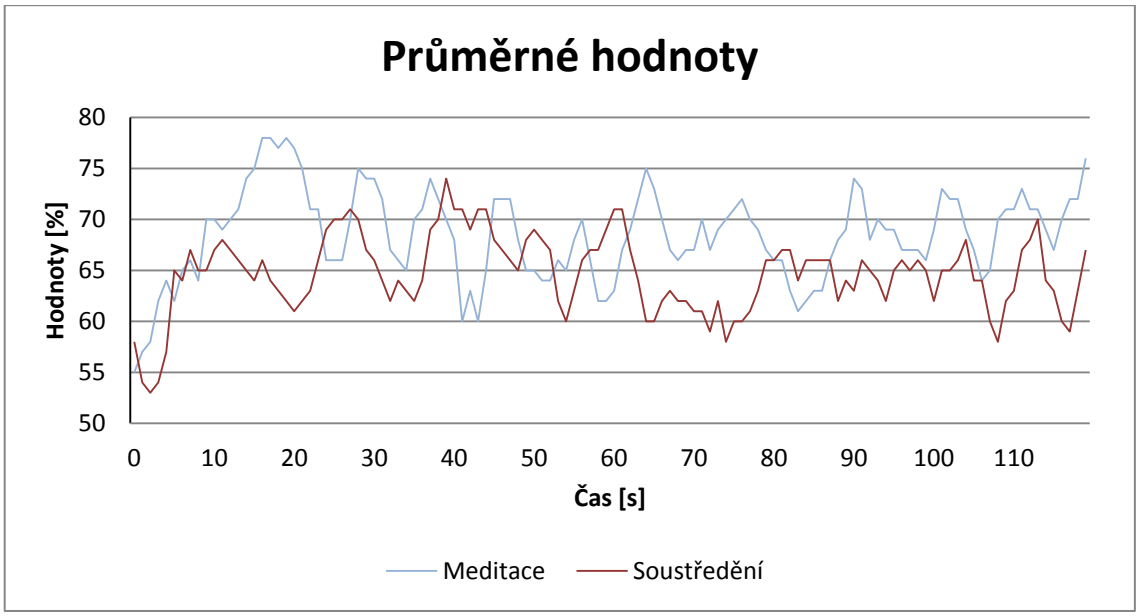
S prací jsou dodány přeložené binární soubory pro `Windows` i `GNU Linux`. Binární soubory byly přeloženy na systémech `64b`. K binárním souborům jsou přiloženy dynamicky linkované knihovny, díky kterým si uživatel nemusí stahovat a instalovat Qt. Pokud si ovšem uživatel bude chtít aplikaci přeložit pro jiný systém (případně jinou bitovou verzi), musí si projít instalací Qt frameworku, která trvá několik hodin, nebo si aplikaci přeložit na počítači s již nainstalovaným Qt frameworkem.

Pozor ovšem na již nainstalovanou nebo právě instalovanou verzi. Volbou jiné verze než je `5.5`, může dojít ke stavům, ve kterých buď aplikace nepůjde přeložit, nebo nebude fungovat. Qt se totiž neustále vyvíjí a s každou novou verzí mohou přijít i menší změny. Pro Qt verze `5` byly, kromě jiných změn, provedeny i zásadní změny v `Qt Network`. Změnil se například parametr virtuální funkce `incomingConnection()` z datového typu `int` na nový typ `qintptr`. Díky tomu aplikace vyvíjené pro starší verze, které využívaly tuhle funkci, nemohly navázat spojení mezi serverem a klientem.

11 Závěr

Všechny body zadání byly splněny. Byla prostudována problematika, týkající se mozkové aktivity a jejího měření pomocí čelenek NeuroSky MindWave Mobile. Byla navržena a implementována desková hra pro dva hráče, ve které se k ovládání využívá jejich soustředění a meditace. Na závěr bylo provedeno otestování aplikace a měření na různých hráčích.

Série měření byla provedena pomocí přidaného modulu pro průzkum mozkové aktivity. Každý testovaný subjekt byl několikrát změřen a uloženo bylo pouze měření, které bylo, z pohledu testovaného subjektu, nejúspěšnější. Na základě těchto dat od všech subjektů byl vytvořen graf průměrné hodnoty soustředění a meditace, viz níže.



Na základě přidaného modulu pro průzkum mozkové aktivity, by šla udělat verze hry proti počítači. Rád bych ji naimplementoval v budoucnu. Hráč si na začátku vybere věkovou kategorii, která bude představovat konkrétní obtížnost. Z průzkumu se vezmou výsledky vybrané věkové kategorie a vypočítá se z nich průměrná křivka soustředění a meditace, kterou bude počítač v průběhu hry napodobovat. Při hře bude vybírat pole na principu půlení intervalů. Jakmile zasáhne nějakou loď, vybere si lodě, o které by se mohlo jednat. Samozřejmě vezme v potaz i jejich natočení. Poté bude zkoušet střílet do tvarů těch nejmenších lodí. Jakmile bude mít možnost bombardování, využije ji. Opravu si buď nechá pro vlajkovou loď, nebo pro ty největší lodě. Změnu plátna využije pouze v případě, kdy soupeř začne ničit území blízko vlajkové lodě. Konkrétní detaily budou ještě doladěny (například schopnost učení, apod.)

Dalším rozšířením by mohlo být umožnění náhledu stavu her v aplikaci serveru. Uživatel, by pak viděl soustředění a meditaci jednotlivých hráčů a stavy jejich lodí.

Do budoucna by šla práce rozšířit o získávání dat ze senzorů pohybu, tím by mohla být vhodná i k pohybovým rehabilitacím.

Přínosem pro mě bylo, že jsem se naučil vyvíjet aplikace v jazyce C++ s využitím Qt frameworku. Dále, že jsem zjistil spoustu zajímavých informací o lidském mozku a o technologii, která snímá mozkovou aktivitu člověka.

Použité zdroje

Literatura

[1] BUZSÁKI, G. *Rhythms of the brain*. New York: Oxford University Press, 2006. ISBN 9780199828234.

[2] SÖRNMO, Leif. a Pablo LAGUNA. *Bioelectrical signal processing in cardiac and neurological applications*. Boston: Elsevier Academic Press, c2005. ISBN 0124375529.

[3] SANEI, Saeid a Jonathon A. CHAMBERS. *EEG signal processing*. Hoboken, NJ: John Wiley, 2007. ISBN 0-470-02581-6.

[4] TANENBAUM, Andrew S. a WETHERALL, D. *Computer networks*. 5th ed. Harlow: Pearson Education, ©2014. ii, 803 s. Pearson custom library. ISBN 978-1-292-02422-6.

[5] CHROBOCZEK, Martin. *Grafická uživatelská rozhraní v Qt a C++*: [plně kompatibilní s Qt 5]. 1. vyd. Brno: Computer Press, 2013, 392 s. ISBN 978-80-251-4124-3.

Internetové zdroje

[1] EEG (Electroencephalography): The Definitive Pocket Guide. *iMotions: Human Behavior Research Software, Simplified* [online]. Copyright © Copyright 2005 [cit. 02.05.2017]. Dostupné z: https://imotions.com/blog/eeg/

[2] *Developer Toolkit | Learn | Exchange | Publish | NeuroSky Developer Program* [online]. Copyright © 2017 NeuroSky [cit. 02.05.2017]. Dostupné z: http://developer.neurosky.com/docs/doku.php?id=thinkgear_communications_protocol

[3] Qt Documentation. *Qt Documentation* [online]. Copyright © 2017 The Qt Company [cit. 02.05.2017]. Dostupné z: http://doc.qt.io/

[4] Reference - C++ Reference. *cplusplus.com - The C++ Resources Network* [online]. Copyright © cplusplus.com, 2000 [cit. 02.05.2017]. Dostupné z: http://www.cplusplus.com/reference/

[5] Qt | Cross-platform software development for embedded & desktop. *Qt | Cross-platform software development for embedded & desktop* [online]. Copyright © 2017 The Qt Company [cit. 02.05.2017]. Dostupné z: https://www.qt.io/

Obrázky

[1] **Lidský mozek**
Stavba a funkce lidského mozku « E-learningová podpora mezioborové integrace výuky tématu vědomí na UP Olomouc. [online]. Copyright © UP Olomouc 2012 [cit. 02.05.2017]. Dostupné z: http://pfyziollfup.upol.cz/castwiki/?p=3265

[2] **Neuron**
[online]. Copyright © [cit. 02.05.2017]. Dostupné z: https://upload.wikimedia.org/wikipedia/commons/thumb/b/bc/Neuron_Hand-tuned.svg/2000px-Neuron_Hand-tuned.svg.png

[3] **Rozmístění elektrod v systému**
[online]. Copyright © [cit. 02.05.2017]. Dostupné z: https://upload.wikimedia.org/wikipedia/commons/thumb/7/70/21_electrodes_of_International_10-20_system_for_EEG.svg/1200px-21_electrodes_of_International_10-20_system_for_EEG.svg.png

[4] **Ukázka mozkových vln**
403 *Forbidden* [online]. Copyright © [cit.02.05.2017]. Dostupné z: http://files.silent-voice.webnode.cz/200000014-538ac54040/mozkov%C3%A9%20vlny.jpg

[5] **MindWave Mobile čelenka**
[online]. Copyright © [cit. 02.05.2017]. Dostupné z: http://cdn.shopify.com/s/files/1/0031/6882/products/MWM_WhiteBG_1_large.jpeg?v=1424448067

[6] **ThinkGear AM**
[online]. Copyright © [cit. 02.05.2017]. Dostupné z: https://statics3.seeedstudio.com/product/Brainwave%20Sensor_01.jpg

[7] **Architektura klient-server**
[online]. Copyright © [cit. 02.05.2017]. Dostupné z: https://upload.wikimedia.org/wikipedia/commons/thumb/f/fb/Server-based-network.svg/500px-Server-based-network.svg.png

[8] **Ukázka plátna a nabídka lodí**
Twitter. *Welcome to Twitter* [online]. Copyright © [02.05.2017]. Dostupné z: https://mobile.twitter.com/gbgprague/status/741294068563628032/photo/1

Seznam zkratek

API – aplikační programové rozhraní.
ASIC – aplikační integrovaný obvod.
COM – hardwarové rozhraní.
EEG – elektroencefalografie.
EMG – elektromyografie.
GNU – svobodný operační systém.
GPL – všeobecná veřejná licence GNU.
GUI – grafické uživatelské rozhraní.
ID – jednoznačný identifikátor.
IDE – grafické rozhraní pro vývoj aplikací.
IT – informační technologie.
LGPL – licence svobodného softwaru.
Min GW – překladač zdrojových souborů.
Qt – knihovna pro tvorbu GUI.
Qt SDK – sada vývojových nástrojů od Qt.
RAM – operační paměť počítače.
RAW Wave Value – hodnota hrubého signálu z čelenky MindWave Mobile.
SDK – sada vývojových nástrojů.
SWS – pomalá vlna spánku.
TCP – spolehlivý protokol transportní vrstvy pro přenos informací.
UDP – nespolehlivý protokol transportní vrstvy pro přenos informací.
USB – univerzální sériová sběrnice.

Seznam příloh

- Příloha A – obsah přiloženého DVD
- Příloha B – uživatelská dokumentace
- Příloha C – komunikační protokol

Příloha A

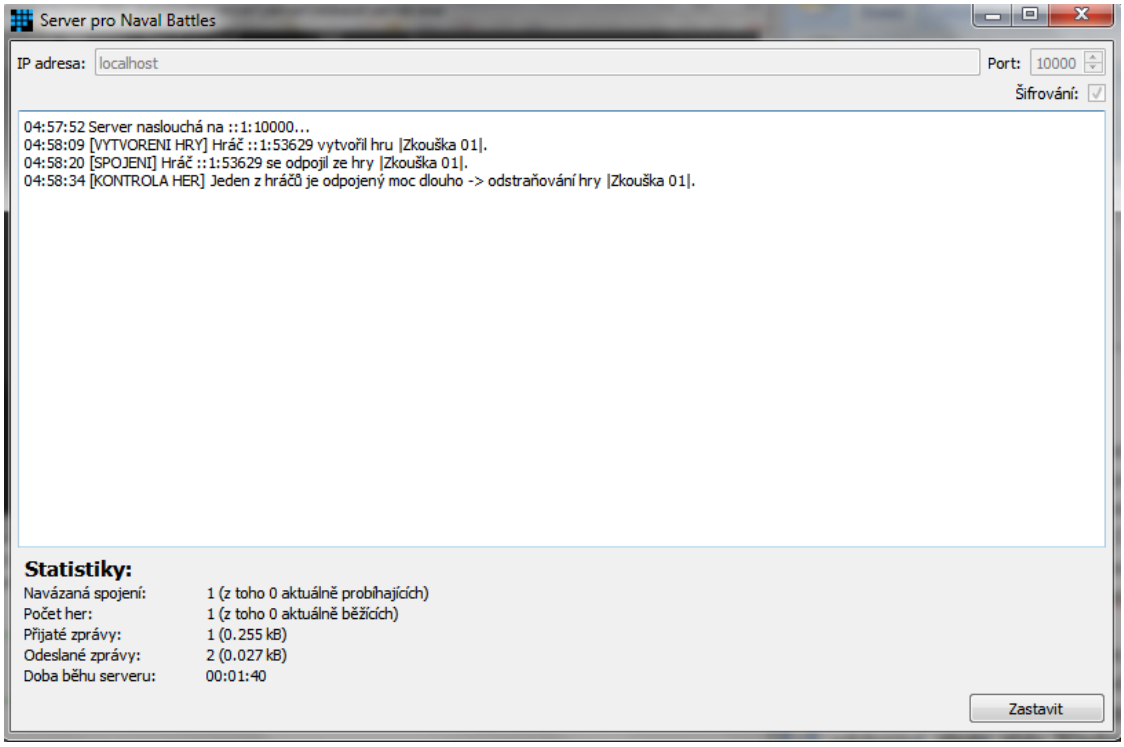
Obsah na přiloženém DVD

- **/NavalBattles** – veškeré soubory potřebné pro překlad a spuštění klienta. Je zde i soubor `READ_ME.txt` s veškerými instrukcemi.
- **/server** –veškeré soubory potřebné pro překlad a spuštění serveru. Je zde i soubor `READ_ME.txt` s veškerými instrukcemi.
- **/BP** – bakalářská práce ve formátu `DOCX` a `PDF`.

Příloha B

Uživatelská dokumentace

Server



14. Server pro Naval Battles

Překlad a spuštění

K překladu jsou ve složce `/server` dva soubory `build`. Pro `Windows` je soubor s příponou `.cmd`, pro `GNU Linux` pak s příponou `.sh`.

Po překladu můžete aplikaci serveru spustit dvěma způsoby. Ke spuštění jsou k dispozici ve složce `/server` dva soubory `run`. Pro `Windows` je soubor s příponou `.cmd`, pro `GNU Linux` pak s příponou `.sh`. Druhým způsobem je spustit rovnou binární soubor ve složce `release`. Ten má název `server`.

Po spuštění

Po zapnutí aplikace se uživateli zobrazí hlavní okno. Okno serveru je jednoduché. Skládá se z hlavičky, těla a patičky. Tělo se pak rozděluje na horní a dolní část. V horní části je textové pole, v dolní pak statistiky běhu serveru.

V hlavičce uživatel nastavuje adresu a port, na kterém má server naslouchat. Zaškrtnutím checkboxu v hlavičce uživatel nastavuje šifrovanou komunikaci. Do adresy

nemusí zadávat jen IP adresu. Může sem zadávat i hostname nebo řetězec „`ANY`“, pokud chce naslouchat všude.

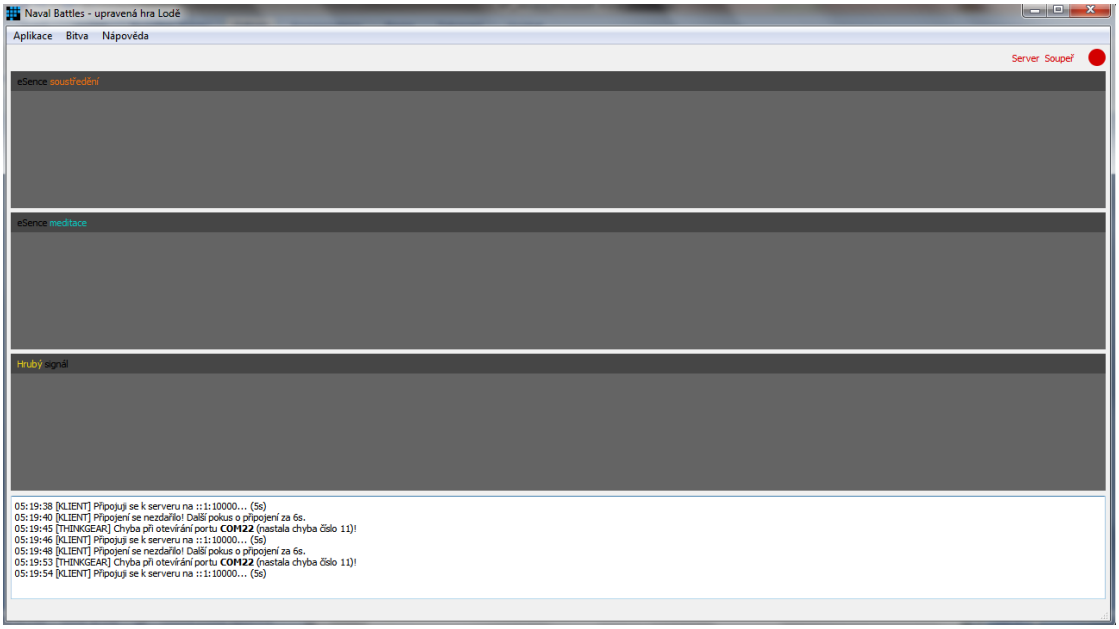
V patičce je pouze jedno tlačítko, kterým uživatel spouští nebo zastavuje server. Pokud server spustí a rozhodne se změnit adresu, port nebo šifrovanou komunikaci, musí server zastavit a poté opět spustit.

V textovém poli se zobrazují informace o aktivitě serveru. Kromě výpisů výsledků požadavků se vypisují i informace o startu a zastavení serveru. Na závěr se vypisují informace spojené s ukončováním serveru (dochází k ukončování vláken a čištění paměti).

Statistiky obsahují informace o tom, kolik bylo navázaných spojení, včetně toho, kolik z nich je aktuálních. Dále počet celkově vytvořených a právě probíhajících her, počet a velikost přijatých a odeslaných zpráv a na závěr dobu běhu serveru.

Pokud chce uživatel ukončit aplikaci serveru, musí kliknout na křížek v pravé horní části aplikačního okna a potvrdit dialog, který mu vyběhne.

Klient



15. Hlavní okno po spuštění

Překlad a spuštění

K překladu jsou ve složce `/NavalBattles` dva soubory `build`. Pro `Windows` je soubor s příponou `.cmd`, pro `GNU Linux` pak s příponou `.sh`.

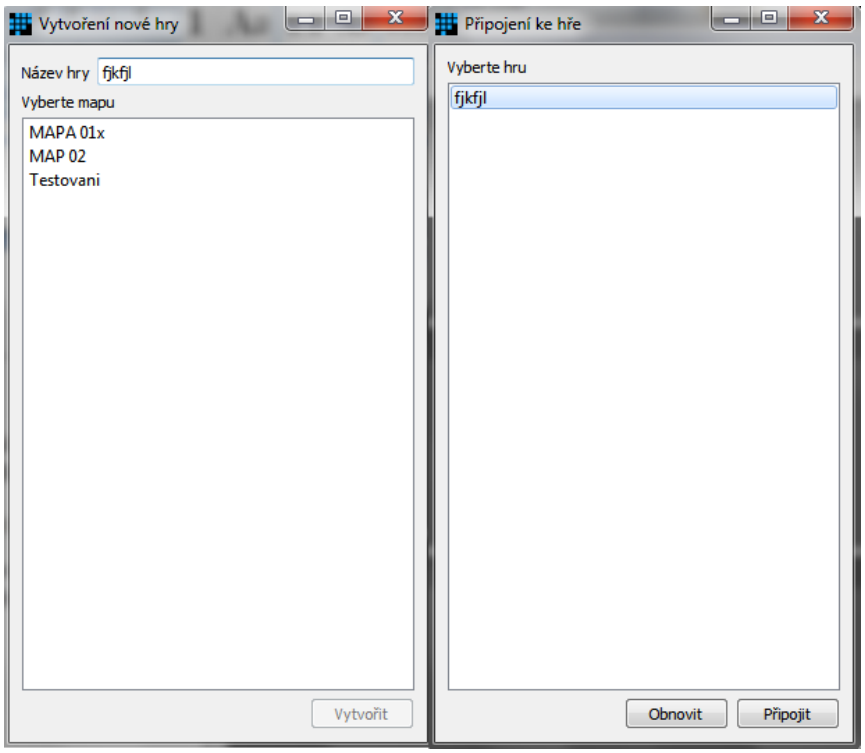
Po překladu můžete aplikaci klienta spustit dvěma způsoby. Ke spuštění jsou k dispozici ve složce `/NavalBattles` dva soubory `run`. Pro `Windows` je soubor s příponou `.cmd`, pro `GNU Linux` pak s příponou `.sh`. Druhým způsobem je spustit rovnou binární soubor ve složce `release`. Ten má název `NavalBattles`.

Po spuštění

Po zapnutí klienta se uživateli zobrazí hlavní okno. V horní části je menu, které je rozděleno na tři části: `Aplikace`, `Bitva` a `Nápověda`. Všechny položky v menu jsou přístupné také pomocí klávesových zkratek.

V části `Nápověda` jsou pouze informace a nápověda ke hře NavalBattles.

V části `Bitva` je pak možnost zhlédnutí statistik her, vytvoření nové hry, připojení k nové (či nedohrané) hře a v průběhu hry jsou zde políčka pro ukončení hry.

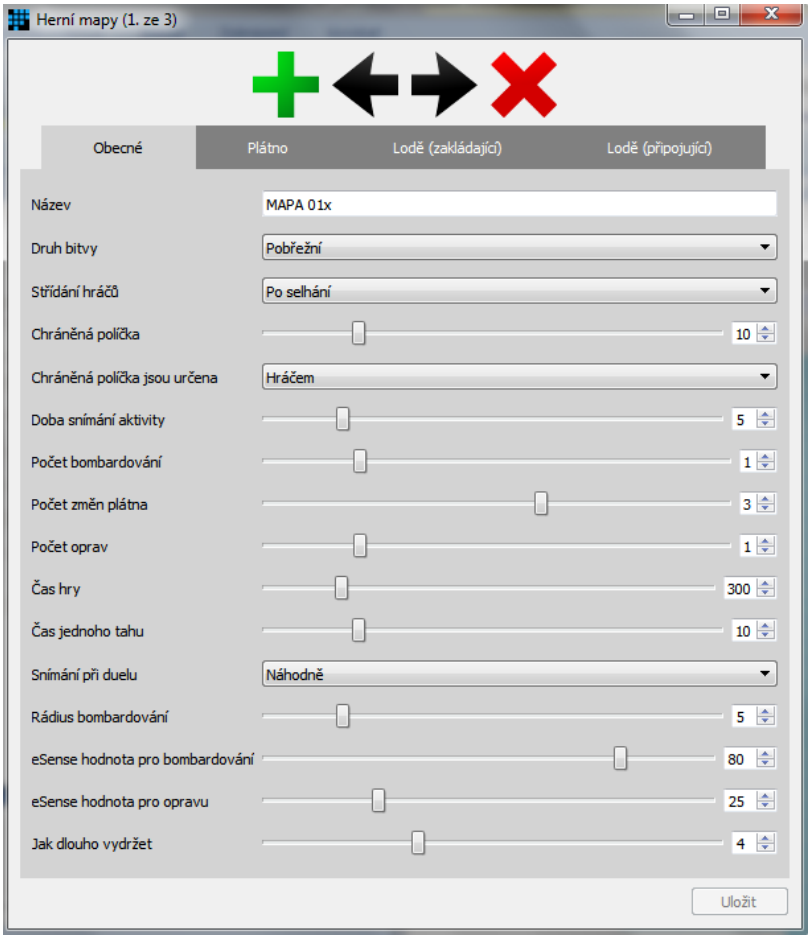


16. Okno pro vytvoření nové hry a okno pro připojení k nové hře

Při vytváření nové hry uživatel musí zadat název hry (ten uvidí soupeř, který se bude připojovat) a vybrat mapu, kterou chce hrát. Poté stačí kliknout na tlačítko `Vytvořit`. V případě připojování k nové hře, uživatel musí vybrat název hry, kterou chce hrát a kliknout na tlačítko `Připojit`. Je zde i tlačítko `Obnovit`, které obnoví seznam volných her na serveru. V případě připojování k nedohrané hře, žádné okno uživateli nevybíhá. Pokud chce uživatel v průběhu hry danou hru ukončit, má na výběr mezi dvěma akcemi. Buď může ukončit hru bez možnosti opakování kliknutím na políčko `Ukončit hru`, nebo může hru ukončit s možností opakování kliknutím na políčko `Vzdát se`.

V části `Aplikace` pak uživatel může spustit úpravu herních map, zhlédnout statistiky výzkumu, přidat měření testovacího subjektu, může hru zobrazit přes celé okno, spustit okno pro nastavení aplikace a ukončit aplikaci.

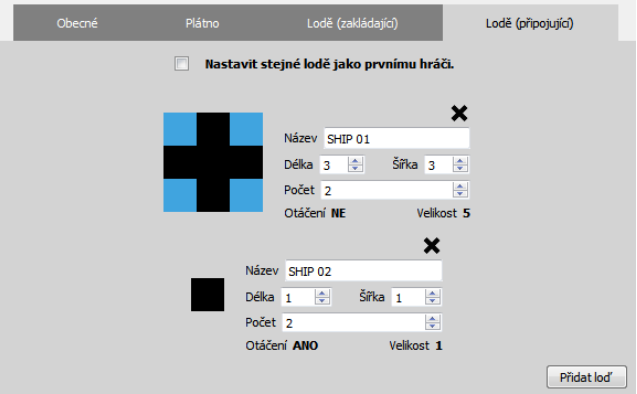
Vytváření map



17. Okno pro vytváření a úpravu map

Okno je rozděleno na tři části: hlavičku, tělo a patičku. V hlavičce jsou tlačítka pro vytvoření a odebrání mapy a pro pohyb mezi mapami. V těle jsou pak čtyři panely pro nastavení konkrétní mapy a v patičce je pouze tlačítko pro uložení provedených změn. Tlačítko se zobrazí pouze v případě provedení nějaké změny.

V prvním panelu „`Obecné`“ uživatel nastavuje obecné informace k mapě (jméno mapy, druh bitvy, apod.). V druhém panelu „`Plátno`“ uživatel nastavuje velikost a vzhled plátna (šedá políčka jsou ta, která nebudou během hry vidět). Ve třetím nastavuje lodě pro prvního hráče (hráč, který zakládá hru, je brán jako útočník) a ve čtvrtém pak lodě pro druhého hráče (hráč, který se připojuje do hry, je brán jako obránce).



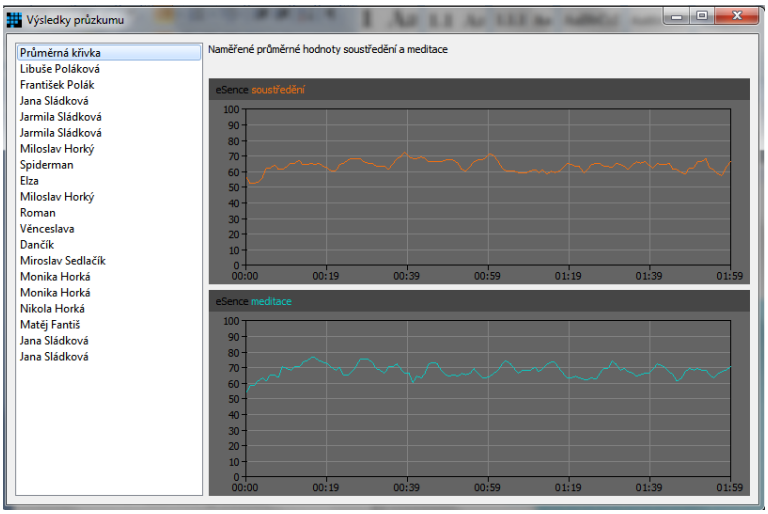
18. Seznam lodí pro druhého hráče

V každém seznamu je tlačítko `Přidat loď`. Každá loď má svůj název, délku, šířku a počet, který znamená, kolikrát se loď musí vložit do plátna, než započne hra. Druhý seznam se od prvního liší v jediné komponentně a to v checkboxu, který se nachází v horní části seznamu. Jeho zaškrtnutím uživatel říká, že chce, aby měli oba hráči totožné seznamy. Černá barva políček u lodí znamená vzhled lodi. Modrá pak prostory, které v průběhu hry nebudou vidět.

Průzkum

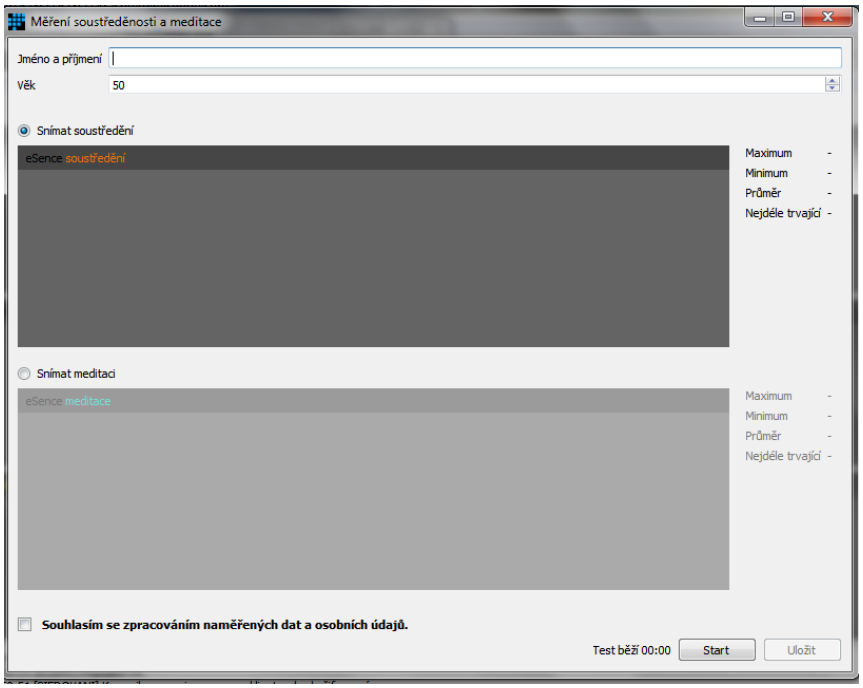
Modul pro průzkum je rozdělen do dvou částí. Zhlednutí statistik průzkumu a přidání dat do průzkumu.

- Zhlédnutí průzkumu – zde jsou pouze měřené subjekty a jejich celková průměrná křivka. Subjekty zatím nejdou odstraňovat.



19. Zobrazení výsledků průzkumu

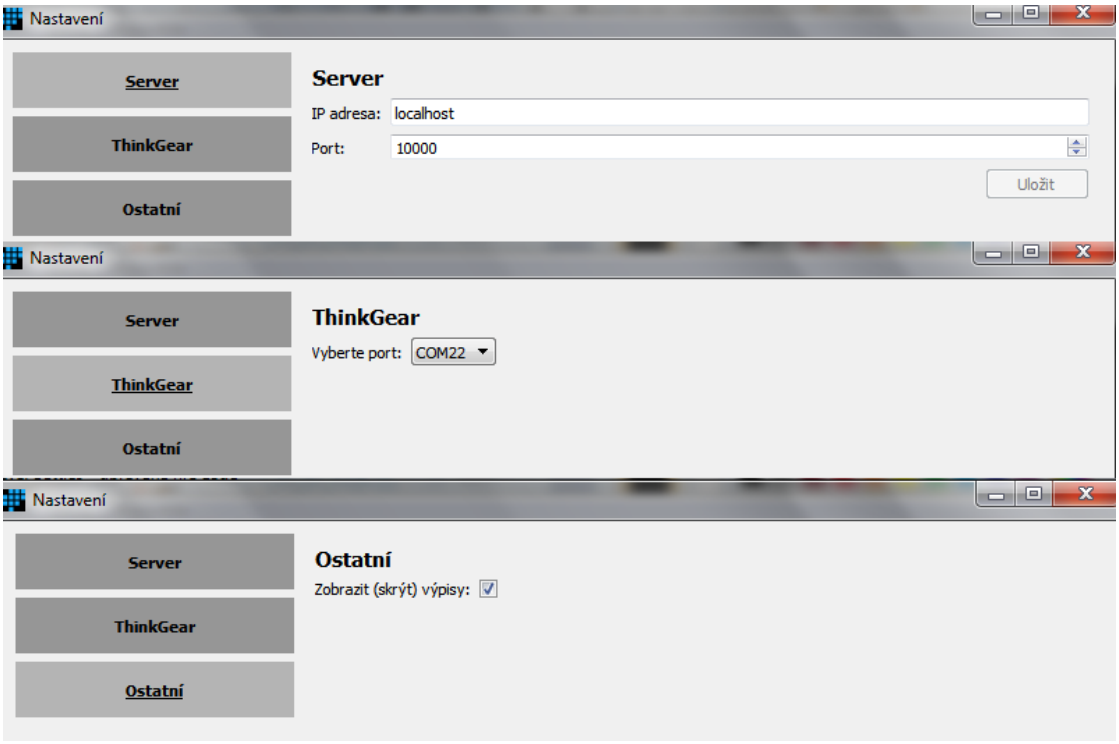
- Přidání dat



20. Měření nového subjektu

V případě, že chce uživatel provést měření nějakého subjektu, musí vyplnit jeho jméno, zadat věk a měřený subjekt musí souhlasit se zpracováním naměřených i osobních údajů. Poté zvolí, co bude snímáno jako první (zvolením daného radiobuttonu) a klikne na tlačítko `Start`. Po dokončení testu vyběhne dialog, který musí uživatel potvrdit. Následně musí zvolit druhý graf (překliknutím na druhý radiobutton) a opět kliknout na tlačítko `Start`. Jakmile budou dokončeny oba testy, zpřístupní se tlačítko `Uložit`, pomocí kterého se měřený subjekt uloží.

Nastavení

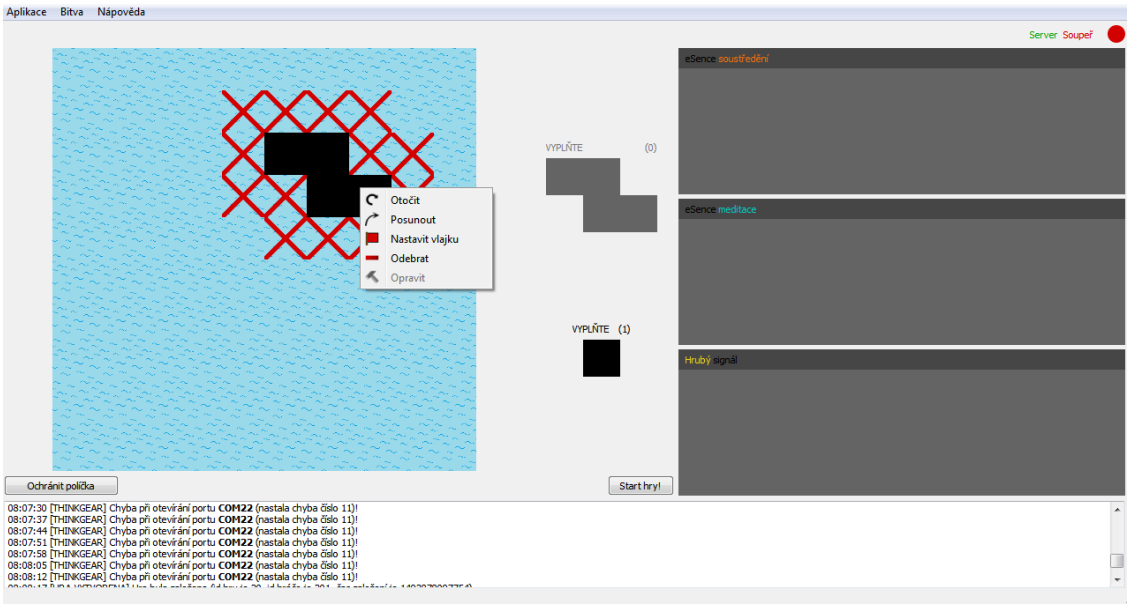


21. Nastavení

Okno nastavení je rozdělené na dvě části. V té levé je menu v podobě tří tlačítek. Po kliknutí na první tlačítko, uživatel mění IP adresu, na které se nachází server a port, na kterém server poslouchá. Změny potvrdí kliknutím na tlačítko `Uložit`. Po kliknutí na druhé tlačítko, uživatel mění port, na který čelenka NeuroSky MindWave Mobile posílá naměřená data. Po kliknutí na poslední tlačítko v menu, uživatel akorát nastavuje, zda se má zobrazovat textové pole s informačními výpisy v hlavním okně.

Hra

Jestliže chce uživatel hrát hru, musí ji buď založit, nebo se k ní připojit. Obě možnosti jsou popsány výše. Jakmile se spustí hra, zobrazí se uživateli, kromě grafů, i plátno se seznamem lodí. Do plátna musí vložit všechny lodě ze seznamu. Pokud nějaká nebude vložena, hráč se nedostane dál. Musí nastavit vlajkovou loď, a v případě, že se mají nastavovat nějaká chráněná políčka a že je má nastavovat hráč, musí kromě vlajkové lodě nastavit i chráněná políčka. Pokud něco z toho nenastaví, opět se nedostane dál.



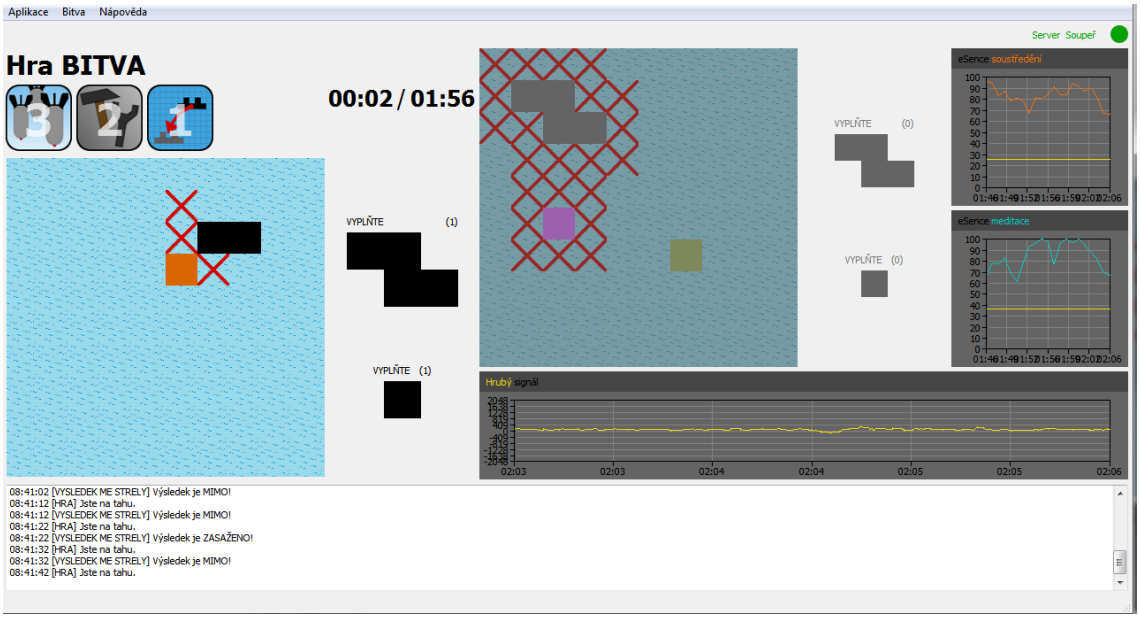
22. Nastavování herního plátna

Lodě se umisťují do plátna následujícím způsobem. Uživatel ji musí nejdříve vybrat. To provede kliknutím levým tlačítkem myši na danou loď v seznamu lodí napravo od plátna. Pokud chce vybrat jinou loď a nevložil všechny lodě vybraného modelu, musí provést odznačení (opět kliknutím na danou loď) a poté může vybrat jiný druh lodě. Do plátna ji umístí kliknutím na levé tlačítko myši. Jakmile je loď umístěna v plátně, uživatel s ní může manipulovat kliknutím pravého tlačítka myši přímo na danou loď (v plátně). Po kliknutí na pravé tlačítko myši vyběhne kontextové menu, které obsahuje pět položek.

- Položka `Otočit` otočí loď. Pokud loď nejde z nějakého důvodu otočit (byla by mimo plátno, zasahovala by do nějaké jiné lodě, případně daný druh lodí nejde otáčet), tato položka není zpřístupněna.
- Položka `Posunout` odebere loď z plátna, ale ponechá jí v paměti (daný druh lodě je jakoby znovu vybrán), takže jí uživatel může jednoduše rovnou vložit někam jinam (opět kliknutím na levé tlačítko myši).
- Položka `Nastavit vlajku` nastaví danou loď jako vlajkovou. Jejím zničením útočící hráč vyhrává. Vlajková loď má fialovou barvu.
- Položka `Odebrat` odebere loď z plátna.
- Položka `Opravit` funguje pouze při hře a to pouze ve stavu, kdy uživatel využije možnost opravy lodě a změnu plátna.

Určení chráněných políček probíhá následovně. Uživatel nejdříve musí kliknout na tlačítko `Ochránit políčka`. Tím se dostane do výběrového módu. V tomhle módu nemůže manipulovat s loděmi (resp. nelze je vkládat do plátna). Chráněná políčka vybírá kliknutím na levé tlačítko myši. Chráněná políčka mají žluto-zelenou barvu. Zvolená chráněná políčka jdou opět odebírat (stejným postupem jako se vkládají). Zatím zde není žádné políčko, které by ukazovalo, kolik chráněných políček musí uživatel ještě vložit. Výběrový mód ukončí kliknutím na tlačítko `Konec vybírání`.

Jakmile má vše správně nastaveno, začíná hra.



23. Průběh hry

V levé části je soupeřovo pole se seznamem lodí, které se musí ještě najít a zničit. Nad soupeřovým plátnem nalevo jsou tlačítka pro herní akce a napravo jsou časomíry. Levá časomíra je pro daný tah, pravá pak pro celou hru. Žlutá čára v grafech pro soustředění a meditaci značí, jakou úroveň musí hráč přesáhnout, aby mohl využít konkrétní herní akci. Herní akce „`bombardování`“ (první tlačítko) je zpřístupněná při určité úrovni soustředění. Herní akce „`oprava lodě a změna plátna`“ (druhé tlačítko) je zpřístupněná při určité úrovni meditace. Kromě toho, že hráč musí danou hodnotu přesáhnout, musí soustředění či meditaci držet nad danou úrovní určitou dobu (tu si nastavují v mapě v položce „`Jak dlouho držet`“). V pravé části před grafy je plátno hráče s jeho seznamem lodí, které ze začátku není zpřístupněné. Zpřístupní se až při výběru herní akce „`oprava lodě a změna plátna`“ nebo při výběru „`změna plátna`“ (třetí tlačítko). Po kliknutí na tlačítko jedné z dvou jmenovaných herních akcí, může dělat s plátnem stejné úpravy, jako před začátkem hry. Pokud zvolí herní akci „`oprava lodě a změna plátna`“, může navíc ještě jednu loď opravit. Stačí, když na vybranou loď klikne pravým tlačítkem a dá opravit. Úpravu plátna musí stihnout do konce časomíry pro daný tah (musí stihnout kliknout na tlačítko `Uložit změny`, které se mu, po výběru druhé nebo třetí herní akce, zobrazí pod seznamem jeho lodí). Navíc nemůže hýbat s loděmi, které již byly zasaženy. Jinak platí stejná pravidla, jako při nastavování plátna před začátkem hry. Musí být umístěné všechny lodě, musí být vybrána vlajková loď a musí být nastavena všechna chráněná políčka. Pořadí hráčů určuje server na základě nastavení v dané mapě. To znamená, že hráč nemusí, během svého tahu, udělat vůbec nic. Server automaticky předá slovo druhému hráči. Pokud se jeden z hráčů trefí do chráněného políčka, započne duel. Hráči se zobrazí graf soustředění nebo meditace přes celé hlavní okno. Nad grafem a do textového pole se mu vypíšou informace k duelu. Po uplynutí časového limitu se zobrazí výsledek a hra pokračuje dál. Jakmile hra skončí (je zničena vlajková loď, hráč přišel o celou flotilu,

jeden z hráčů ukončil hru nebo se ze hry odpojil na moc dlouho, vypršel čas pro jednu hru), poté se zobrazí výsledek a pokud hráč neodešel ze hry, zobrazí se dialog, ve kterém se hráči mohou rozhodnout, jestli chtějí opakovat hru. Pokud ano, hra jde znovu od nastavování plátna. Pokud ne, zobrazí se hlavní obrazovka, která se zobrazí při startu aplikace.

Příloha C

Komunikační protokol

Jedná se o textový protokol. Oddělovačem slov je pokaždé tabulátor.

Konstanta `GAME_ERROR`

Když na server přijde zpráva s herním požadavkem a server danou hru nenajde, pošle zpátky na klienta odpověď `GAME_ERROR`. Pokud je klientem přijata tahle hodnota, způsobí to okamžité ukončení probíhající hry a vypsání chybové hlášky. Konstanta `GAME_ERROR` je zastoupena hodnotou `-2`.

id_game (identifikátor hry)

Minimální id hry je `20`. Hodnoty od `0` do `20` jsou rezervovány pro požadavky (s určitou rezervou pro případ potřeby doplnění). Id hry se nastaví podle počtu dosud vytvořených her (k počtu se přičte `ID_FIRST_GAME = 20`).

id_player (identifikátor hráče)

Id prvního hráče (zakladatele) se vypočítá jako (`id_game * 10 + 1`). Výpočet id druhého hráče probíhá obdobně, akorát se přičítá k součinu číslo `2`.

Hlavní zprávy		Zprávy, týkající se jedné hry	
Konstanta, hodnota (`int`)		*Konstanta, hodnota (`int`)*	
ENCRYPTION	1	E_SENSE	6
FREE_GAMES	2	OPP_CONNECTION	7
CREATE_GAME	3	SET_DATA	8
CONNECT_TO_GAME	4	CHANGE_DATA	9
BACK_TO_GAME	5	GAME_STARTED	10
		YOU_PLAY	11
		SHOOT	12
		SHOOT_RESULT	13
		DUEL	14
		GIVE_UP	15
		REPEAT	16

Základní odpovědi na klienta	
Konstanta, hodnota (`int`)	
GAME_ERROR	-2
CHYBA	-1
SHOT_HIT	0
SHOT_OUT	1
SHOT_SINK	2
SHOT_END	3
SHOT_DUEL	4

`map`

Jedná se o řetězec, který se posílá v určitých zprávách buď na server, nebo na klienta. Jelikož se jedná o velice rozsáhlý řetězec, jeho struktura je uvedena mimo konkrétní zprávy.

Část zprávy	Datový typ	Vysvětlení
`name`	`string`	Název mapy.
`size`	`int`	Velikost plátna.
`battlefield`	`string`	Vzhled herního plátna.
`game_mode`	`char`	Herní mód.
`changing`	`char`	Kdy proběhne střídání hráčů.
`protected_sqaures`	`int`	Počet chráněných políček.
`protected_mode`	`char`	Jak se určují chráněná políčka.
`eSense_time`	`int`	Jak dlouho má být snímána aktivita při duelu.
`bombing`	`int`	Kolikrát lze využít bombardování.
`change`	`int`	Kolikrát lze využít změnu plátna.
`repair`	`int`	Kolikrát lze využít opravu lodě a následnou změnu plátna.
`game_time`	`int`	Čas pro celou hru.
`move_time`	`int`	Čas pro jeden tah.
`duel_mode`	`char`	Co je snímáno při duelu.
`bombing_radius`	`int`	Poloměr bombardování.
`eSense_BL`	`int`	Při jaké úrovni soustředění lze využít bombardování.
`eSense_RL`	`int`	Při jaké úrovni lze využít opravu a následnou změnu plátna.
`keep_it`	`int`	Jak dlouho má hráč držet pozornost.
`x_p`	`int`	Počet druhů lodí pro první seznam.
`y_p`	`int`	Celkový počet lodí pro první seznam.
`x_d`	`int`	Počet druhů lodí pro druhý seznam.
`y_d`	`int`	Celkový počet lodí pro druhý seznam.
`ships`	`string`	Dva seznamy lodí (pro zakládajícího a pro připojujícího hráče).

`ships`

Jedná se o pokračování předchozího řetězce **`map`**. Daná šestice hodnot je do řetězce vložena (`x_p + x_d`) – krát.

Část zprávy	Datový typ	Vysvětlení
`name`	`string`	Název lodě.
`length`	`int`	Délka lodě (horizontální rozměr).
`width`	`int`	Šířka lodě (vertikální rozměr).
`rotation`	`int`	Jestli se loď může otáčet.
`count`	`int`	Kolikrát musí uživatel vložit loď do plátna.
`view`	`string`	Vzhled lodě.

Šifrovaná komunikace – posílá se jako první zpráva ze serveru na klienta.

Odeslání na	Tělo zprávy	
Klient	`ENCRYPTION`	`0`
		`1`

`0 (int)` - komunikace není šifrovaná.
`1 (int)` - komunikace je šifrovaná.

Zjištění volných her

Odeslání na	Tělo zprávy		
Server	`FREE_GAMES`		
Klient	`FREE_GAMES`	`first_name`	`...`

`first_name (string)` - název první volné hry.
… - názvy dalších volných her (odděleny tabulátorem).

Založení nové hry

Odeslání na	Tělo zprávy		
Server	`CREATE_GAME`	`name`	`map`
Klient	`CREATE_GAME`	`id_game`	`id_player`	time
		`ERROR`	

`name (string)` - název hry, kterou chce hráč založit (max. `20` znaků).

`map (string)` - mapa dané hry (soupeř ji tak nemusí mít ve svém seznamu).
`time (unsigned long long)` - kdy byla hra vytvořena (kvůli návratu do hry).
`ERROR` - v tomhle případě znamená, že na serveru už hra s takovým názvem existuje (nebo chyba při alokaci).

Připojení druhého hráče

<table>
<tr><th>Odeslání na</th><th colspan="5">Tělo zprávy</th></tr>
<tr><td>Server</td><td>CONNECT_TO_GAME</td><td colspan="4">name</td></tr>
<tr><td rowspan="2">Klient</td><td rowspan="2">CONNECT_TO_GAME</td><td>id_game</td><td>id_player</td><td>time</td><td>map</td></tr>
<tr><td colspan="4">ERROR</td></tr>
</table>

`name (string)` - název hry, ke které se chce soupeř připojit.
`time (unsigned long long)` - kdy byla hra vytvořena.
`map (string)` - mapa dané hry (soupeř ji tak nemusí mít ve svém seznamu).
`ERROR` - v tomhle případě znamená, že hra již není volná.

Návrat do hry

<table>
<tr><th>Odeslání na</th><th colspan="7">Tělo zprávy</th></tr>
<tr><td>Server</td><td>BACK_TO_GAME</td><td colspan="2">id_game</td><td colspan="2">id_player</td><td colspan="2">time</td></tr>
<tr><td rowspan="2">Klient</td><td rowspan="2">BACK_TO_GAME</td><td>first</td><td>name</td><td>opp_bf</td><td>opp_ship</td><td>player_bf</td><td>player_ship</td></tr>
<tr><td colspan="6">GAME_ERROR</td></tr>
</table>

`time (unsigned long long)` - kdy byla nedohraná hra vytvořena.
`first (bool)` - jestli se jedná o zakladatele / připojeného (`0`/`1`).
`name (string)` - název dané hry.
`opp_bf (string)` - soupeřovo odkryté plátno (stejně jako platno v předchozím požadavku, kdy hodnoty '`0`' představují pozadí plátna, hodnoty '`1`' odkrytá pole lodí, hodnoty '`2`' odkrytou vodu a hodnoty '`4`' pole potopené lodi).
`opp_ship (string)` - počet nepotopených lodí soupeře (`7` druhů lodí => `7` hodnot oddělené mezerou představující počet zbylých lodí daného druhu).
`player_bf (string)` - hráčovo plátno (stejné jako `platno` v předchozím požadavku, navíc hodnoty '`2`' = odkrytá voda, '`3`' = zasáhnuté pole lodi, '`4`' = pole potopené lodi).
`player_ship (string)` - aktuální stav hráčovo lodí (stejné jako `lode` v předchozím požadavku).

eSence

Odeslání na	Tělo zprávy			
Server	E_SENCE	E_SENCE_M E_SENCE_A	time	value

`E_SENCE_M (char 'M')` - jedná se o hodnoty meditace.
`E_SENCE_A (char 'A')` - jedná se o hodnoty soustředění.
`time (unsigned long long)` - čas získání hodnoty.
`value (int)` - získaná hodnota z čelenky NeuroSky MindWave Mobile.

Spojení soupeře

Odeslání na	Tělo zprávy	
Klient	OPP_CONNECTION	0 1

`0 (int)` - soupeř se připojil do hry.
`1 (int)` - soupeř se odpojil ze hry.

Nastavení plátna a lodí – začátek bitvy

Odeslání na	Tělo zprávy		
Server	SET_DATA	battlefield	ships

`battlefield (string)` - nastavené plátno hráče (plátno `NxN` => řetězec N^2 znaků, kde znak '`0`' = voda, '`1`' = loď).
`ships (string)` - posílají se informace o `N` lodích. Posílá se `8` informací ke každé z nich. `ID (int)`, `X (int)`, `Y (int)`, `WIDTH (int)`, `LENGTH (int)`, `VIEW` (řetězec o délce `WIDTH * LENGTH`), `FLAGSHIP (bool)`, `ROTATION (int)`.

- `X` = index `i`.
- `Y` = index `j`.
- `VIEW` = podoba lodi (řetězec, znak '`1`' = LOD, '`0`' = prázdné pozadí).
- `FLAGSHIP` = jestli se jedná o vlajkovou loď (`1` ano, `0` ne).
- `ROTATION` = zvolené natočení lodě.

Nastavení plátna a lodí – v průběhu hry

Odeslání na	Tělo zprávy		
Server	CHANGE_DATA	battlefield	ships
Klient	CHANGE_DATA	m_battlefield	m_ships

`battlefield, ships` – stejné jako v předchozím požadavku.

`m_battlefield (string)` - upravený vzhled – pouze soupeřem odkrytá pole.
`m_ships (string)` - `n` hodnot představující počty lodí ke každému modelu lodi, které ještě nejsou potopeny.

Hra začala

Odeslání na	Tělo zprávy	
Klient	`GAME_STARTED`	`timeout`

`timeout (unsigned long long)` - čas kdy bude končit daná hra (milisekundy od roku `1970`).

Jsi na řadě

Odeslání na	Tělo zprávy	
Klient	`YOU_PLAY`	`timeout`

`timeout (unsigned long long)` - čas kdy bude končit daný tah (milisekundy od roku `1970`).

Střela na souřadnici [x, y]

Odeslání na	Tělo zprávy					
Server	`SHOOT`	`type`	`x`		`y`	
Klient	`SHOOT`	`type`	`x`	`y`	`result`	`id_ship`

`type (char 'N' nebo 'B')` - normální střela nebo bombardování.
`x (int)` - souřadnice x, do které hráč vystřelil.
`y (int)` - souřadnice y, do které hráč vystřelil.
`result (int)` - výsledek střely. Buď zásah, mimo, potopená nebo konec (viz konstanty `SHOT_*`).
`id_ship (int)` - id potopené lodě (`-1` v případě, že se jedná o trefu mimo loď).

Pokud se jedná o bombardování, hodnoty `x, y, result` a `id_ship` se posílají `n`-krát, kolikrát, to záleží na velikosti bombardované oblasti.

Výsledek střely

Odeslání na	Tělo zprávy					
Klient	`SHOOT_RESULT`	`type`	`x`	`y`	`result`	`id_ship`

`type (char 'N' nebo 'B')` - normální střela nebo bombardování.

`x (int)` - souřadnice x, do které soupeř vystřelil (první rozměr v poli).
`y (int)` - souřadnice y, do které soupeř vystřelil (druhý rozměr v poli).
`result (int)` - výsledek střely (`SHOT_HIT`, `SHOT_OUT`, `SHOT_SINK`, `SHOT_END`).
`id_ship (int)` - id zasáhnuté lodě (`-1` v případě, kdy se soupeř trefil do vody).

V případě bombardování se hodnoty `x, y, reset` a `id_ship` posílají `n`-krát. Kolikrát, záleží na velikosti vybombardované oblasti.

Hráč se vzdává

Odeslání na	**Tělo zprávy**	
Server	`GIVE_UP`	`end_game`
Klient	`GIVE_UP`	`end_game`

Na server posílá klient konstantu `GIVE_UP` v případě, že se rozhodl předčasně ukončit hru. Server pak pošle stejnou konstantu na klienta soupeře. Pro soupeře to znamená výhru a ukončení hry.
`end_game (bool)` – jestli uživatel klikl na `Konec hry` (`1`), nebo `Vzdát se` (`0`).

Opakování hry

Odeslání na	**Tělo zprávy**	
Server	`REPEAT`	`0`
		`1`
Klient	`REPEAT`	`0`
		`1`

`0 (int)` - posílá se na server při rozhodnutí o opakování.
`1 (int)` - server posílá, jakmile se rozhodnou oba hráči, na oba klienty.

Duel

Odeslání na	**Tělo zprávy**		
Server	`DUEL`	`E_SENCE_M`	`timeout`
		`E_SENCE_A`	

`E_SENCE_M (char 'M')` – hráči soupeří v meditaci.
`E_SENCE_A (char 'A')` – hráči soupeří v soustředění.
`timeout (unsigned long long)` - čas konce duelu.