

# Semestrální práce KIV/UPS

## Implementace server-klient aplikace: Wargame

## 1 Zadání

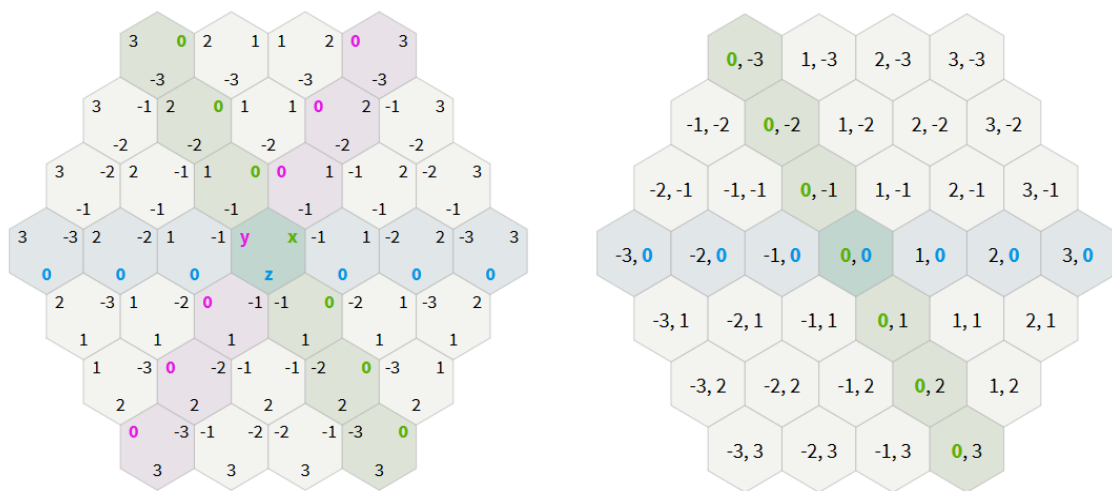
Cílem této práce je vytvoření aplikace serveru a klienta, které dohromady budou sloužit pro hraní hry Wargame přes počítačovou síť. Hra Wargame je jednoduchá tahová hra kde proti sobě 2 hráči zápasí o kontrolu bodů na hrací ploše. Každý z hráčů má přiděleno několik jednotek, které ovládá, a pomocí kterých se snaží již zmíněné body obsadit. Jednotky se mohou navzájem ničit. Každé kolo získá hráč ovládající kontrolní bod jeden bod strategických surovin. Hráč s největším množstvím strategických surovin po 5 kolech vyhrává hru.

Při implementaci serveru je třeba se zaměřit na jeho výkon a robustnost. Server nesmí ukončit svojí činnost z důsledku připojení nebo útoku nevalidních klientů.

## 2 Analýza

### 2.1 Práce s hexagony

Jeden z hlavních znaků hry Wargame je herní plocha složená ze šestiúhelníků (dále hexagon). Na rozdíl od čtverců se hexagony nemohou adresovat jednoduchým indexováním (jako např. dvou-rozměrným polem). Takové adresování by ztěžovalo práci s herním polem. Každý hexagon má tedy 3 koordináty (X, Y, Z). Součet těchto koordinát se pro každý hexagon rovná vždy nule. Z toho vychází, že není třeba si pamatovat všechny 3 koordináty, ale stačí si pamatovat pouze 2 a poslední odvodit ze součtu uložených koordinátů.



Obrázek 1: Koordináty políček[1]

Tomuto adresování se říká axiální. Dvourozměrná pole používají adresování čtvercové, které je pouze speciálním případem axiálního (osy jsou kolmé). Je tedy možné transformovat koordináty z jednoho do druhého a naopak. To dovoluje uložení herní plochy do dvourozměrného pole. Je ale třeba implementovat speciální metody pro přístup do takového pole.

$$\begin{aligned} \text{Index } I &= X + \frac{(Z + 1)}{2} \\ \text{Index } J &= Z \end{aligned}$$

Velká výhoda hexagonové plochy na rozdíl od čtvercové je možnost pohybu šesti směry, což přináší o hodně víc strategických možností při hraní hry. Dále přináší možnost blíže simulovat chování z reálného světa jako například zjišťování vzdáleností 2 bodů. Tyto výhody jsou vykoupeny pouze lehce složitějším zacházením s datovým polem, ve kterém jsou jednotlivá políčka uložena.

## 2.2 Datový model hry

Pro zaznamenání stavu hry na serveru i na klientovi je třeba vytvořit datové struktury. Budou v sobě mít uložený stav herního pole, pořadí hraní jednotek a stav jednotlivých jednotek na herní ploše. Tyto struktury musí být datově ekvivalentní na klientovi a serveru, ale mohou se lišit samotnou implementací.

### 2.2.1 Herní pole

Návrh struktury herního pole:

```
char[][] terrain;    //terén
unit[] units;        //seznam jednotek
int rows;            //počet řádků pole
int columns;         //počet sloupců pole
int on_turn;         //index jednotky na tahu
int attacking;       //jednotka útočí
int score_one;
int score_two;
```

### 2.2.2 Jednotka

Návrh struktury jednotky:

```
short ID;
unitType type;       //typ jednotky: INFANTRY = 'I', TANK = 'T', SPG = 'S', FLAG = 'F'
allegiance al;       //komu patří: BLU = 'B', RED = 'R', NEUTRAL = 'N'
short health;
short damage;
short move_range;
short attack_range;
int dead;
int coord_x;
int coord_z;
```

## 2.3 Datový model serveru

Aplikace serveru musí mít pro svojí správnou funkci vytvořené datové struktury pro klienta, místnost a server samotný. Tyto struktury jsou vytvořeny pro uložení všech důležitých informací jako je např. file descriptor klienta, file descriptor serveru, socket, ID klienta atd. Slouží také k uložení současného stavu těchto entit.

### 2.3.1 Server

Návrh struktury pro uložení stavu serveru:

```
int max_clients;
int max_lobbies;
int client_count;           //počet uložených klientů
int active_clients;
int active_lobbies;
int port;
client_data clients[];      //pole klientů
lobby lobbies[];           //pole místností
int running;
int server_socket;         //fd na kterém server přijímá spojení
```

### 2.3.2 Klient

Návrh struktury klienta:

```
int fd;                     //file descriptor klienta
int id_key;                 //identifikační klíč klienta
char player_name[NAME_LENGTH];
char message_buffer[];     //vstupní buffer zpráv
int read;                  //index čtení bufferu
int active;                //klient je aktivní na serveru
pthread_t client_thread;   //vlákno obsluhující klienta
int running;
```

### 2.3.3 Místnost

Návrh struktury pro uložení stavu místnosti:

```
client_data player_one;
client_data player_two;
int ready_one;             //první hráč připraven
int ready_two;            //druhý hráč připraven
int game_in_progress;     //probíhá hra
playfield pf;             //data hry
char lobby_name[NAME_LENGTH];
```

## 2.4 Návrh protokolu

Pro komunikaci mezi klientem a serverem byl navržen komunikační protokol. Jedná se o textový protokol s předepsaným formátem zprávy:

*ID/TYP/DÉLKA/DATA/ID\n*

Každá zpráva musí být ukončena znakem nového řádku a musí mít velikost větší než 24 znaků (vyplývá to z minimálního počtu znaků prázdné zprávy). K oddělení položek se používá znak „|“. Následuje vysvětlení jednotlivých položek:

- *ID* – Identifikátor klienta reprezentující 4B celé číslo. Je posíláno v hexadecimálním tvaru na začátku a konci zprávy. Díky tomu se dá zpráva lehce najít a přeložit. Toto číslo je unikátní pro každého klienta.
- *TYP* - Typ zprávy, který určuje jakým způsobem se bude se zprávou zacházet. Určená jedním velkým písmenem. Typy zpráv a různé zacházení je popsáno v následující podkapitole.
- *DÉLKA* – Celé 2B číslo určující počet argumentů v datech. V hexadecimálním tvaru. Slouží k ulehčení zpracování dat ve zprávě a kontrole správného přenosu.
- *DATA* – Data přenášená ve zprávě. Formou řetězce znaků. Každá položka dat je oddělena rozdělovacím znakem „|“. Bližší popis v následujících podkapitolách.

K escapování funkčních znaků se používá znak s ASCII hodnotou 126 - ~. Každá zpráva musí splňovat tento formát. Popisované argumenty v následujících kapitolách jsou pouze obsahem položky *DATA*.

### 2.4.1 Zpráva ACK/NACK

Tato zpráva je reakcí serveru na určitou akci klienta, která může být platná nebo neplatná. Reakce na platnou akci je zpráva typu ACK – písmeno „X“, a neplatná reakce je NACK – písmeno „Y“. Zpráva neobsahuje žádná data.

### 2.4.2 Zpráva Connect

Označena písmenem „C“. Slouží k autorizaci klienta na serveru. Server pošle v reakci odpověď ACK/NACK. To určuje zda-li proběhla autorizace v pořádku, nebo nastala chyba. Zpráva neobsahuje žádná data.

### 2.4.3 Zpráva Disconnect

Označena písmenem „D“. Slouží k oznámení klientovi, že jeho oponent v aktivní hře opustil hru. Server na tuto zprávu neočekává odpověď. Zpráva neobsahuje žádná data.

### 2.4.4 Zpráva Poke

Označena písmenem „P“. Slouží ke kontrole spojení klienta se serverem. Server odpoví klientovi stejnou zprávou. Zpráva neobsahuje žádná data.

### 2.4.5 Zpráva Get Server

Označena písmenem „G“. Slouží k poslání informací o serveru. Obsahuje informace o místnostech. Každá místnost je určena řetězcem formátu:

*INDEX|JMÉNO|AKTIVNÍ|HRÁČ1| HRÁČ1RDY | HRÁČ2 | HRÁČ2RDY*

- *INDEX* – Index místnosti v dekadickém tvaru

- *JMÉNO* – Řetězec znaků určující jméno místnosti
- *AKTIVNÍ* – Písmeno „T“ nebo „F“ určující zda-li se v místnosti právě hraje
- *HRÁČ* - Řetězec znaků určující jméno hráče
- *HRÁČRDY* - Písmeno „T“ nebo „F“ určující zda-li je hráč připraven ke hře

#### 2.4.6 Zpráva Create Lobby

Označena písmenem „L“. Slouží k vytvoření místnosti na serveru. Server pošle v reakci odpověď ACK/NACK. To určuje zda-li proběhlo vytvoření v pořádku, nebo je již server plný. Zpráva obsahuje řetězec znaků určující jméno místnosti. Server dále odpoví zprávou *Get Server* všem klientům.

#### 2.4.7 Zpráva Join Lobby

Označena písmenem „J“. Slouží k připojení klienta do místnosti. Server pošle v reakci odpověď ACK/NACK. To určuje zda-li proběhlo přidání v pořádku, nebo je místnost už plná. Zpráva obsahuje číslo v dekadickém tvaru určující index místnosti.

Server po přijmutí této zprávy pošle zprávu stejného typu všem klientům v místnosti s daty stejnými jako ve zprávě *Get Server*, s tím rozdílem, že pošle informace pouze o lobby, do kterého se klient připojil.

#### 2.4.8 Zpráva Leave Lobby

Označena písmenem „V“. Slouží k opuštění místnosti klientem. Klient na tuto zprávu neočekává odpověď. Zpráva neobsahuje žádná data.

Server po přijmutí této zprávy pošle zprávu *Join Lobby* všem klientům v místnosti s daty stejnými jako ve zprávě *Get Server*, s tím rozdílem, že pošle informace pouze o lobby, do kterého se klient připojil.

#### 2.4.9 Zpráva Toggle Ready

Označena písmenem „T“. Slouží ke změně stavu připravenosti ke hře. Klient na tuto zprávu neočekává odpověď. Zpráva neobsahuje žádná data.

Server po přijmutí této zprávy pošle zprávu *Join Lobby* všem klientům v místnosti s daty stejnými jako ve zprávě *Get Server*, s tím rozdílem, že pošle informace pouze o lobby, do kterého se klient připojil.

#### 2.4.10 Zpráva Start

Pokud jsou všichni hráči v místnosti připraveni ke hře, server všem hráčům pošle serii zpráv, které slouží k připravení hře na klientovi. Tato zpráva je první v této serii.

Označena písmenem „S“. Slouží k poslání informací o herním poli. Pošle herní pole po řádcích ve formě textových řetězců. Server na tuto zprávu neočekává odpověď. Pole je určeno řetězcem formátu:

*ŘÁDEK1/ ŘÁDEK2/.../ŘÁDEKN*

#### 2.4.11 Zpráva Units

Označena písmenem „I“. Slouží k poslání informací o všech jednotkách. Druhá zpráva v serii pro začátek hry viz. předchozí kapitola. Server na tuto zprávu neočekává odpověď. Každá jednotka je určena řetězcem formátu:

*ID/TYP/STRANA/ŽIVOT/ZRANĚNÍ/DSHPOHYBU/DSHÚTOKU/MRTEV/KOORDX/KOORDZ*

- *ID* – Identifikátor místnosti v dekadickém tvaru
- *TYP* – Typ jednotky určený jedním písmenem viz. kapitola 2.2.2
- *STRANA* – Určuje vlastníka jednotky jedním písmenem viz. kapitola 2.2.2
- *ŽIVOT* – Počet životů jednotky určené číslem v dekadickém tvaru
- *ZRANĚNÍ* – Síla útoku jednotky určená číslem v dekadickém tvaru
- *DSHPOHYBU/DSHÚTOKU* – maximální dosah pohybu/útoku určené číslem v dekadickém tvaru
- *MRTEV* - Písmeno „T“ nebo „F“ určující zda-li je jednotka mrtvá
- *KOORDX/KOORDZ* – Axiální koordináty jednotky určené číslem v dekadickém tvaru

#### 2.4.12 Zpráva Update

Označena písmenem „U“. Slouží k poslání informací o tahu. Je také poslední zprávou v serii pro začátek hry. Server na tuto zprávu neočekává odpověď. Posílá informace v následujícím formátu:

*HRÁČ1 | HRÁČ1SKORE| HRÁČ2 | HRÁČ2SKORE/NATAHU/ÚTOČÍ/STRANA*

- *HRÁČ* - Řetězec znaků určující jméno hráče
- *HRÁČSKORE* – Skóre hráče určené číslem v dekadickém tvaru
- *NATAHU* – Index jednoty na tahu určený číslem v dekadickém tvaru
- *ÚTOČÍ* - Písmeno „T“ nebo „F“ určující zda-li jednotka na tahu útočí
- *STRANA* – Písmeno určující stranu, za kterou hráč hraje viz. kapitola 2.2.2

Tato zpráva je poslána oboum klientům v místnosti, ale každému s jinou stranou.

#### 2.4.13 Zpráva End

Označena písmenem „E“. Slouží k ukončení hry klientem. Klient na tuto zprávu neočekává odpověď. Zpráva neobsahuje žádná data.

Server po přijmutí této zprávy pošle zprávu stejného typu všem klientům v místnosti a přidá k ní výherce hry daného textovým řetězcem.

#### 2.4.14 Zpráva Move

Označena písmenem „O“. Slouží k pohybu jednotky klientem. Klient na tuto zprávu očekává odpověď ACK/NACK. Ta určuje zda-li je tah platný. Zpráva obsahuje 3 položky dat o pohybu. Jedná se o 3 celá čísla v dekadickém tvaru určující index jednotky, koordinát X a koordinát Z v tomto pořadí.

Server po přijmutí této zprávy pošle zprávu stejného typu všem klientům v místnosti. Pouze pokud je tah platný.

#### 2.4.15 Zpráva Attack

Označena písmenem „A“. Slouží k útoku jednotky klientem. Klient na tuto zprávu očekává odpověď ACK/NACK. Ta určuje zda-li je tah platný. Zpráva obsahuje 2 položky dat o útoku. Jedná se o 2 celá čísla v dekadickém tvaru určující index jednotky útočníka a index jednotky, na kterou je útočeno v tomto pořadí.

Server po přijmutí této zprávy pošle zprávu stejného typu všem klientům v místnosti. Pouze pokud je tah platný.

#### 2.4.16 Zpráva Capture

Označena písmenem „Z“. Slouží k obsazení jednotky klientem. Server na tuto zprávu neočekává odpověď. Zpráva obsahuje 2 položky dat o obsazení. Jedná se o 2 celá čísla v dekadickém tvaru určující index jednotky obsazujícího a index jednotky obsazeného v tomto pořadí.

Server tuto zprávu posílá po pohybu, který zároveň obsazuje jednotku všem klientům v místnosti.

#### 2.4.17 Zpráva Skip

Označena písmenem „K“. Slouží k ukončení tahu klientem. Klient na tuto zprávu neočekává odpověď. Zpráva neobsahuje žádná data.

Server po přijmutí této zprávy pošle zprávu typu *Update* všem klientům v místnosti.

#### 2.4.18 Zpráva Reconnect

Označena písmenem R. Posílá server klientovi ve chvíli kdy je možné se znovu připojit do rozehrané hry. Obsahuje celé číslo v dekadické podobě určující index místnosti se hrou. Klient potom může odpovědět stejnou zprávou v případě, že se chce znovu připojit. Po odeslání zprávy o znovupřipojení na server odešle server klientovi všechny informace pro připojení do hry viz. Start, Units a Update. Odpověď není povinná.

## 3 Implementace

### 3.1 Server

Aplikace serveru je napsána jazykem C standardu c99 a s využitím knihovny POSIX Threads. Je to vícevláknová aplikace kde jedno vlákno obsluhuje akceptování a autorizaci klientů. Součástí autorizace je i vytvoření vlastního vlákna pro klienta. Toto vlákno obsluhuje potom všechny požadavky klienta.

Počet vláken pro klienty je omezený maximálním počtem aktivních klientů, který je zadán argumentem při spuštění. Klientská vlákna končí svojí činnost ve chvíli kdy je spojení s klientem ztraceno.

Datové struktury jsou implementovány podle návrhu v kapitole 2.2 a 2.3. Byli pouze přidány zámky pro zajištění bezpečnosti paralelního běhu vláken.

#### 3.1.1 Moduly.

- *server.c* - Modul *server.c* obsahuje všechny funkce potřebné pro běh serveru na více vláknech a obsluhu klientských požadavků. Nejdůležitější funkce jsou *authenticate\_client* a *execute\_command*. Ty jsou podle potřeby předávány funkci



*read\_input*, která čte data z bufferu. Probíhá zde také kontrola správnosti typu příkazů od klienta. K obsluze připojení se používá systémové volání *select*.

- *net\_interface.c* – Server pro čtení zpráv ze sítě využívá buffer znaků. Tento buffer je pole znaků, ke kterému se ale přistupuje pouze přes metody z tohoto modulu. To zajišťuje správné a bezpečné čtení dat ze sítě. Buffer má definovanou svojí velikost a počet znaků, které může maximálně najednou přečíst. Po přidání znaů do bufferu se pokusí přečíst zprávu. Pokud zprávu nenajde tak posune index čtení a načte další znaky. Ve chvíli kdy najde validní zprávu, nebo už nemá místo k zápisu se buffer vyprázdní. Pokud zprávu v bufferu nenajde pak nevyprázdní celý buffer, ale jen část a „sesype“ buffer na začátek.
- *parser.c* – Zajišťuje překlad zpráv na vstupu i na výstupu. Rozděluje příchozí zprávy na položky a řeší escapování znaků. Překlad začíná hledáním znaku konce zprávy. Od konce zprávy se pokusí načíst 8 znaků a přebvést je z hexadecimálního tvaru. Následně se pokusí najít stejné číslo někde v předchozích znacích čímž najde začátek zprávy. Pak proběhnou kontroly o správné délce a tvaru.
- *hex.c* – Implementace samotné hry. Obsahuje všechny funkce pro přístup k hernímu poli, tahy jednotkami a správu herního pole viz. kapitola 2.1.

### 3.2 Klient

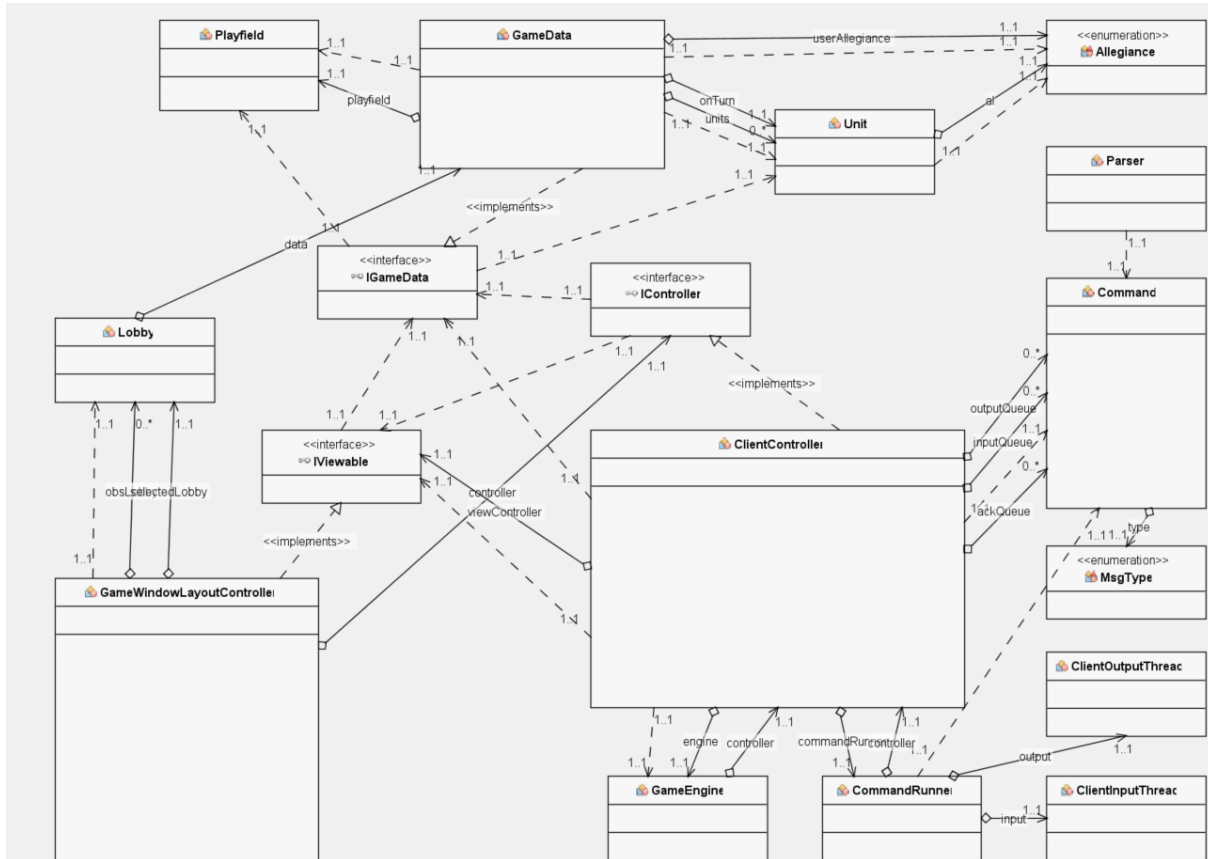
Aplikace klienta je napsána jazykem Java s využitím JavaFX. Je to 3-vláknová aplikace. První vlákno je vlákno GUI. Stará se o interakci s uživatelem a zobrazování všech důležitých informací o hře.

Druhé vlákno se stará o přijímání zpráv od serveru a posílá je poslednímu vláknu pro zpracování.

Třetí, nejdůležitější, vlákno je engine, který se stará o síťovou komunikaci, obsluhu uživatelských požadavků, správu dat hry a informací o serveru, a zajištění správného průběhu podle komunikačního protokolu. Metoda běhu tohoto vlákna se nachází v modulu *Engine.java*. Jedná se o smyčku, která se opakuje buď při příchodu nové zprávy a nebo po uplynutí určité doby. Pokud nepříde po tuto dobu žádná zpráva ze serveru tak se pokusí poslat zprávu typu *Poke* a znovu se uspí.

Je zde implementován systém zpráv mezi vlákny, pomocí kterého spolu vlákna mohou komunikovat. Vlákno GUI i vlákno pro příjem zpráv podávají vláknu enginu zprávy a tím ho budí. Engine se potom pokusí příkazy vykonat. Zprávy se ukládají do 3 různých front – vstupní, výstupní a kontrolní. Ve vstupní a výstupní frontě se nacházejí příkazy ze vstupu a výstupu. Kontrolní fronta slouží pro kontrolu odpovědi ze serveru.

### 3.2.1 Moduly



### Obrázek 2: Zjednodušený diagram tříd

#### 3.2.1.1 ClientController

Tato třída reprezentuje nejdůležitější část aplikace. Je to modul který spojuje všechny ostatní moduly. Je podle návrhového typu jedináček, což znamená že může vždy existovat pouze jedna instance. Toho je dosaženo statickým blokem inicializace.

Obsahuje odkazy na všechny ostatní hlavní moduly (*GameWindowLayoutController* a *GameData*) a všechny fronty příkazů. Drží také ID klienta, které se posílá na server k autentizaci.

### 3.2.1.2 GameWindowLayoutController

Jedná se o JavaFX kontrolní třídu, která se stará o správný běh GUI. Posílá výstupní příkazy do *ClientController* a překresluje herní plochu podle vstupních příkazů. Reakce klienta, které se posílají na server začínají tady.

### 3.2.1.3 GameData

Třída pro udržování stavu hry. Obsahuje instance dalších tříd, které reprezentují herní plochu, jednotky a stav hry. *GameEngine* při každé smyčce vybírá nová data z této třídy a v případě, že nějaká jsou se pokusí herní plochu překreslit na GUI.

#### 3.2.1.4 GameEngine

Zde se nachází metoda, která definuje běh vlákna co zpracovává zprávy. To provádí pomocí třídy *Command*, kde je uložený typ a data příkazu, a *CommandRunner*, který příslušný příkaz provede podle jeho typu. Tento modul také kontroluje timeout.

## 4 Uživatelská příručka

### 4.1 Server

Apliakci serveru je možné spustit pouze pod operačním systémem Linux. Před spuštěním je třeba aplikaci sestavit příkazem *Make*. Příslušný makefile je přiložený ve složce s aplikací serveru. Je podmínkou na zařízení mít nainstalovanou aplikaci *Make*, která je např. součástí balíčku *build-essentials*. Ten je možný nainstalovat příkazem:

```
apt-get install build-essential
```

Spuštění samotného serveru se pak provede přechodem do složky *dist/release/GNU-Linux/* a spuštěním aplikace *ups\_wargame\_server* zadáním následujícího příkazu do terminálu v této složce:

```
./ups_wargame_server $pocetKlientu $port
```

Server požaduje ke spuštění 2 vstupní parametry. První parametr *\$pocetKlientu* nastavuje maximální počet aktivních klientů, který může server v jednu chvíli obsluhovat. Toto číslo by nemělo překročit hodnotu 100. Pro větší množství neproběhlo testování. Druhý parametr *\$port* určuje port na kterém bude server přijímat a vysílat data. Toto číslo musí být větší než 0 a menší než 65356. Port nesmí být vyžíván jinou aplikací. Doporučuji tedy vybrat port z rozmezí 49152-65535.

Během chodu je možné server ovládat psaním ovládacích znaků do terminálu:

- h – vypíše nápovědu do konzole
- w – vypíše do konzole status klientů a místností
- q – zahájí ukončování serveru, tato akce může trvat několik vteřin

V případě chyby při připojování serveru k síti se do terminálu vypíše chybová hláška a je třeba server manuálně vypnout.

### 4.2 Klient

Aplikace klienta funguje pod jakýmkoliv zařízením, které má nainstalované Java Runtime Enviroment v1.8.0\_101 a vyšší. Zařízení musí také podporovat JavaFX aplikace. Sestavení klienta je možné pomocí souboru *build.xml* ve složce s klientem. Je k tomu ale třeba mít nainstalovaný libovolný sestavovací nástroj pro Javu (Maven, Ant). Ve složce *dist* se již nachází sestavená spustitelná aplikace.

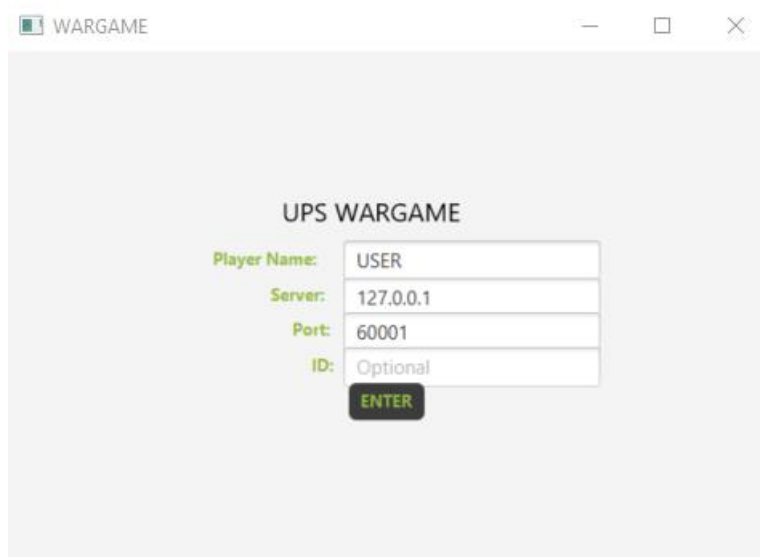
Aplikace se dá spustit pouze dvojitým kliknutím, nebo zadáním příkazu do terminálu:

```
java -jar UPS_wargame_client.jar
```

Spuštění z terminálu přináší uživateli bližší pohled do běhu aplikace.

Po spuštění aplikace uživatel uvidí okno, do kterého může napsat údaje o sobě (jméno) a údaje serveru, na který se hodlá připojit (adresa a port). Je nutné předem znát adresu a port serveru. Uživatel má ještě možnost napsat svoje klientské identifikační číslo. Tento parametr není povinný a v případě, že ho uživatel nezadá je náhodně vygenerováno aplikací. Tento parametr slouží jako způsob připojení zpět do hry ve chvíli kdy bylo

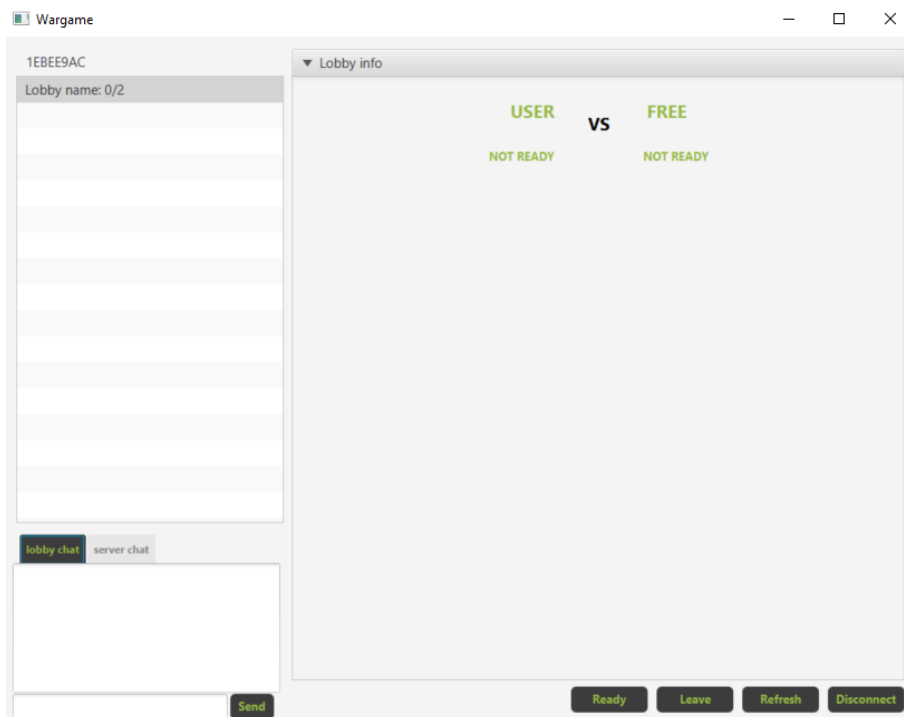
přerušené spojení nebo nastala nějaká jiná chyba. Připojení na server proběhne po stisknutí tlačítka *Enter*.



The screenshot shows a window titled "WARGAME" with a light gray background. In the center, the text "UPS WARGAME" is displayed. Below it, there are four input fields with labels: "Player Name:" (containing "USER"), "Server:" (containing "127.0.0.1"), "Port:" (containing "60001"), and "ID:" (containing "Optional"). Below these fields is a dark green button with the text "ENTER" in yellow.

Obrázek 3: Připojovací okno

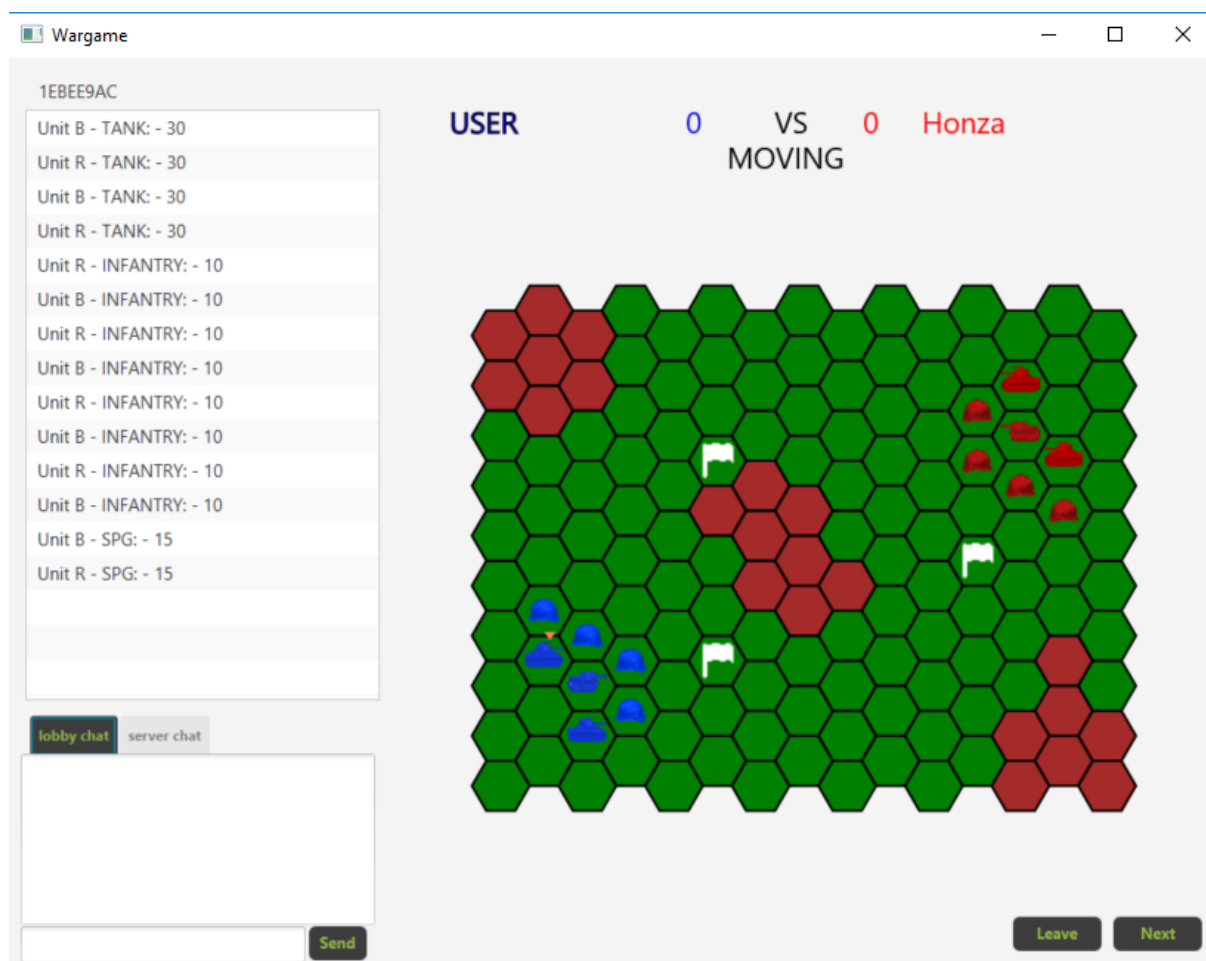
Pokud spojení proběhne úspěšně, okno připojení zmizí a místo něj se objeví okno pro interakci se serverem a následnou hrou. Na levé straně okna uživatel uvidí všechny dostupné místnosti na serveru. V případě, kdy žádné místnosti nejsou volné, se může uživatel pokusit vytvořit další místnost pomocí tlačítka *Lobby*. Po vybrání místnosti musí uživatel pro připojení stisknout tlačítko *Connect*. Uživatel může server opustit tlačítkem *Disconnect*. V levém horním rohu je viditelné ID uživatele, které je potřeba zadat do pole pro ID ve chvíli kdy se uživatel chce znovu připojit do neukončené hry.



The screenshot shows a window titled "Wargame" with a light gray background. On the left side, there is a panel with the user ID "1EBEE9AC" and a "Lobby name: 0/2" label. Below this is a list of lobby slots, each with a name and a status. At the bottom of this panel are tabs for "lobby chat" and "server chat", and a "Send" button. On the right side, there is a "Lobby info" panel showing a match between "USER" and "FREE", both with a status of "NOT READY". At the bottom of the window are four buttons: "Ready", "Leave", "Refresh", and "Disconnect".

Obrázek 4: Okno připojení do místnosti

Po připojení do místnosti uvidí uživatel všechny ostatní připojené hráče a jejich status připravenosti na hru. Pro změnu připravenosti musí uživatel kliknout na tlačítko *Ready*. Hra začne automaticky ve chvíli kdy jsou všichni hráči připraveni na hru. Pro opuštění místnosti musí uživatel stisknout tlačítko *Leave*.



Obrázek 5: Pohled na hru

Hra se ovládá pouze pomocí klikání na hrací plochu a po odehrání kola stisknutí tlačítka *Next*. Na levé straně okna je výpis jednotek, kde je vidět počet životů všech jednotek. Pravidla hry jsou popsána v první kapitole. Hráč může kdykoliv ukončit hru stisknutím tlačítka *Leave*.

## 5 Závěr

Aplikace serveru i klienta funguje a je možné si pomocí této dvojice zahrát hru Wargame přes počítačovou síť. Během vývoje jsem se seznámil s problematikou vytvoření komunikačního protokolu a implementace aplikací, které pomocí daného protokolu komunikují. Nejtěžší částí práce bylo vytvoření vícevláknové aplikace s blokujícími operacemi čtení z datových streamů. Další překážkou bylo vytvoření responzivní aplikace klienta a práce s frontami příkazů. Obě aplikace byly otestovány. Server jsem testoval pomocí aplikací netcat a valgrind. Všechny kritické chyby byly odstraněny.

### 5.1 Citace

[1] **Red Blob Games**. Hexagonal Grids [Online] 2016. <http://www.redblobgames.com/grids/hexagons/>.