FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

Semestrální práce KIV/UPS

Síťová hra Onitama

Lukáš Varga

07. 01. 2023

Obsah

1 Pravidla hry

Onitama je japonská tahová a zároveň karetní hra, která svou filozofií vychází z bojových umění. Na první pohled připomíná šachy – je to hra pro dva hráče, kde se oba střídají v tazích svých figurek po herním poli. Oba hráči mají k dispozici čtyři figurky studentů (pěšák) a jednu figurku mistra (král). Na začátku hry je ze všech 16 karet balíčku náhodně vybráno 5 herních karet a to následovně: dvě pro černého hráče, dvě pro bílého hráče a jedna volná karta. Dále hra pokračuje pouze s těmito 5 kartami. Při této hře nevadí, že oba hráči vidí karty toho druhého, protože se po celou dobu hraje s nezakrytými kartami.

Bílý hráč začíná hru tím, že si ze svých dvou karet vybere jednu a v kombinaci s některou figurkou táhne kamkoliv po herním poli. Pokud nemá žádné pole, kam by mohl táhnout, může rovněž vybrat kartu a přeskočit svůj tah. Zahraná karta je nyní vyměněna s prostřední volnou kartou a jeho tah tímto končí. Dále pokračuje stejným způsobem černý hráč atd. Hra končí v momentě, kdy jeden z hráčů vezme oponentovi figurku mistra. Hra také končí, pokud se jednomu hráči povede dostat figurku mistra na startovní pozici oponentova mistra.

Obr. č. 1: Ukázka grafické implementace

2 Popis protokolu

V rámci protokolu jsou jednotlivé klientské i serverové zprávy oddělené znakem „\0“. Jako oddělovač parametrů ve zprávě je použit znak pipy „|“. U příkazů ve zprávách jsou použity kapitálky. Níže je ukázka struktury zprávy (kvůli čitelnosti textu jsou u pipy mezery a na konci chybí znak konce zprávy).

COMMAND | *arg1* | *arg2* | ... | *argN*

2.1 Přihlášení

LOGIN | *uživatelské jméno*
Uživatel po navázání spojení vyplní své uživatelské jméno a pošle na server požadavek o přihlášení. Pokud se uživatel odpojí od rozehrané hry, může se tímto příkazem vrátit ke hře pod jeho jménem. Jméno nesmí být prázdné ani větší než 10. Musí také být unikátní.

WAITING | uživatelské jméno
Validně přihlášený uživatel obdrží tuto zprávu a následně čeká, než se připojí oponent. Pokud se některý jiný hráč také připojí, hra může začít.

*FAILED | *detailnější chybové hlášení**
Při přihlašování může nastat spoustu variant, proč je uživatelské jméno označeno za nevalidní. Je-li například prázdné, obsahuje-li neplatné symboly či jedná-li se o již použité jméno. Při takové situaci obdrží uživatel tuto zprávu spolu s popisem, co přesně se při přihlašování pokazilo a může to zkusit znovu.

2.2 Začátek a konec hry

START | jméno černého hráče | jméno bílého hráče | B1 | B2 | SP | W1 | W2
Server pošle oběma klientům uživatelská jména, která budou ve hře. Podle pořadí ve zprávě se určí kdo má černé a kdo bílé figurky. Hráč s bílými figurkami začíná hru. B1 a W1 ve zprávě představují názvy vylosovaných karet černého a bílého hráče (detailněji popsané v sekci *Tahy*). SP je volná karta uprostřed hrací plochy, která není přiřazena nikomu (spare card).

GAME_OVER | jméno hráče, který vyhrál
Informace pro oba hráče, kdo vyhrál aktuální hru. Ta končí v momentě, kdy jeden z hráčů vezme oponentovi figurku mistra nebo pokud se jednomu hráči povede dostat figurku mistra na startovní pozici oponentova mistra. Po obdržení této zprávy hra končí a hráč si ještě může prohlédnout herní plochu. Při odchodu je znovu umístěn do čekací fronty, dokud se nepřipojí oponent.

2.3 Tahy

MAKE_MOVE | zahraná karta | počáteční pozice figurky | konečná pozice figurky
Uživatel si vybere jednu ze dvou jeho karet a pozici figurky dle svého uvážení. S figurkou pak dle možností vybrané karty táhne na dostupné herní pole (není mimo hrací plochu a není na něm přátelská figurka). Zahraná karta je zadávána jejím názvem s malými písmeny (například *tiger*, *ox* nebo *dragon*). Každá karta má definované možné tahy figurky z výchozí/relativní pozice (kartézský systém [x, y]). Černé figurky mají hodnoty invertované o 180°. Dále je popsán seznam všech možných karet ve hře a jejich pohybů:

```
tiger:      [(0, 2), (0, -1)],
dragon:     [(-2, 1), (2, 1), (-1, -1), (1, -1)],
frog:       [(-1, 1), (-2, 0), (1, -1)],
rabbit:     [(1, 1), (2, 0), (-1, -1)],
crab:       [(0, 1), (-2, 0), (2, 0)],
elephant:   [(-1, 1), (1, 1), (-1, 0), (1, 0)],
goose:      [(-1, 1), (-1, 0), (1, 0), (1, -1)],
rooster:    [(1, 1), (-1, 0), (1, 0), (-1, -1)],
monkey:     [(-1, 1), (1, 1), (-1, -1), (1, -1)],
mantis:     [(-1, 1), (1, 1), (0, -1)],
horse:      [(0, 1), (-1, 0), (0, -1)],
ox:         [(0, 1), (1, 0), (0, -1)],
crane:      [(0, 1), (-1, -1), (1, -1)],
boar:       [(0, 1), (-1, 0), (1, 0)],
eel:        [(-1, 1), (1, 0), (-1, -1)],
cobra:      [(1, 1), (-1, 0), (1, -1)],
```

Pozice figurek je dvojice čísel oddělená pipou „|“. První číslo udává řádek hracího pole a druhé udává sloupec. Obě čísla leží v intervalu <0, 4>. Možná ukázka tahu figurky z pozice řádku 4 a sloupce 2 (bílý král) na pozici řádku 3 a sloupce 3 za použití karty *cobra (1, 1)*:

MAKE_MOVE | cobra | 4 | 2 | 3 | 3

MOVE_WAS_MADE | zahraná karta | počáteční pozice figurky | konečná pozice figurky
Po tom, co první hráč zahraje serverem ověřený tah zprávou *MAKE_MOVE*, je třeba informovat oba hráče o tomto tahu, aby se jim updatovala herní plocha. Zahraná karta je vyměněna s volnou kartou a hraje druhý hráč. Pro aktuálního hráče slouží tato zpráva jako pozitivního ověření jeho tahu.

MAKE_PASS | zahraná karta
V momentě, kdy není žádný pohyb možný, hráč si vybere jednu kartu a pošle žádost o přeskočení tahu. Jako při tahu si rovněž vymění svou vybranou kartu s prostřední volnou kartou.

PASS_WAS_MADE | zahraná karta
Pokud hráč nemá žádné pole, kam by mohl táhnout, může rovněž vybrat kartu a přeskočit svůj tah pomocí zprávy *MAKE_PASS*. Tento způsob tahu může nastat pouze, pokud není žádná jiná možnost. To je ověřeno serverem po analýze možných tahů.

*INVALID_MOVE | *detailnější chybové hlášení**
Při neúspěšném pokusu o tah (*MAKE_MOVE* či *MAKE_PASS*) je danému hráči poslána tato zpráva jako negativní potvrzení. Jím poslaný tah buďto obsahuje nevalidní data nebo je jiným způsobem v rozporu s pravidly hry, například tah mimo hrací plochu či tah na pole vlastní figurky. Po hráči je poté znovu vyžadován validní tah.

2.4 Odpojení a připojení

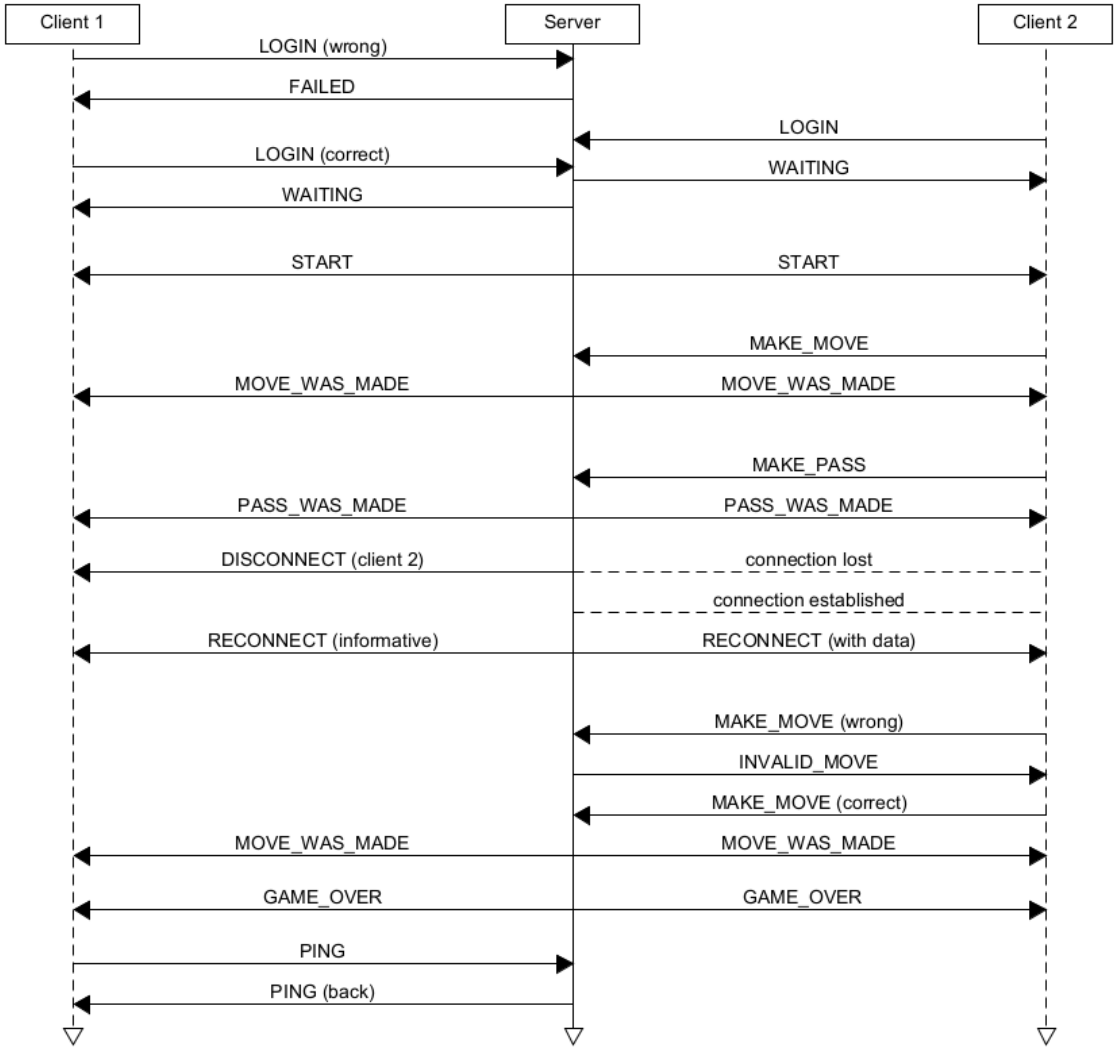
DISCONNECT | jméno odpojeného hráče
Pokud hráč ukončí klienta uprostřed rozehrané hry nebo pokud klient po nějakou dobu nekomunikuje se serverem (ať už herními zprávami nebo pingem), je oponent informován o výpadku/odpojení a hra je pozastavena. Odpojený hráč se může připojit zpátky při použití stejného jména. Pokud tak hráč neučiní, tak je po určitém čase hra ukončena a vítězí oponent.

RECONNECT | jméno černého | jméno bílého | 5x karty | je-li hráč bílý | je-li bílý na tahu | herní plocha s figurkami
RECONNECT | jméno znovu připojeného hráče
Hráč, který se odpojí ze hry, se může zpátky připojit při použití stejného uživatelského jména (nezávisle na čísle socketu). V momentě, kdy se připojí zpátky, jsou klientovi předány data o probíhající hře, aby mohl pokračovat. Zpráva je také informativně poslána druhému klientovi, aby věděl, že se první hráč vrátil. Avšak obsahuje pouze jméno hráče, co se připojil zpět (přetížená zpráva).

PING | PING
Speciální zpráva sloužící pro identifikaci mrtvého spojení. Zpráva je prvně poslána klientem na server (posílá se v určitém intervalu). Server získá informaci o tom, že hráč má aktivní spojení. Následně je klientovi poslána stejná zpráva jako ping back. Klient si uchovává čas poslední zprávy, aby věděl, že je připojen k serveru. Čas přijetí zprávy si server taktéž uloží ve formě časového razítka k danému hráči. Pokud tento údaj není určitou dobu updatován, je hráč nejprve odpojen z aktuální hry, která je pozastavena (v případě, že hráč ještě není ve hře, je ihned odebrán). Pokud hráč opět nereaguje delší čas, je kompletně

odebrán ze hry a danou hru vyhrává čekající oponent. Tato serverová logika je použita v metodě *keep_alive()* souboru *main.cpp*.

2.5 Diagram komunikace



Obr. č. 2: Ukázka komunikace serveru se dvěma klienty

3 Programátorská dokumentace

3.1 Server

Server hry je napsán převážně objektovým přístupem v jazyce C++. Hlavní vstupní bod programu se nachází v metodě *main()* souboru *server.cpp*. Metoda nejprve nakonfiguruje IP, port a další proměnné a začne naslouchat. Poté v nekonečné smyčce iteruje přes všechny file deskriptory a reaguje na požadavky klientů. K tomu je využita konstrukce *select*, která zároveň umožňuje paralelizaci aplikace a server tudíž může komunikovat s více klienty bez použití vláken nebo procesů. U *selectu* je použit timeout, který odpojuje

neaktivní klienty. Dále je serverová aplikace dekomponována do 7 zdrojových *.cpp* a 7 hlavičkových *.h* souborů.

Soubor *Command.cpp* spravuje veškeré příkazy zpráv, které jsou použity v aplikační implementaci TCP protokolu. Příkazy jsou vytvořeny pomocí enumů, ale mají i textový ekvivalent pro zjednodušení práce s nimi. Příkazy jsou pomyslně rozděleny na klientské a serverové.

Soubor *Parser.cpp* řeší překlad příchozích zpráv do formátu aplikace. Je ošetřen i na nevalidní vstupy, aby v programu nedošlo k nečekaným výpadkům. Zároveň loguje do konzole veškerou příchozí i odchozí komunikaci kvůli čitelnosti a ladění.

Soubor *Card.cpp* modeluje použité karty ve hře pomocí kombinace mapy, vektorů a tuplů. Ke každému názvu karty je přiřazeno pole možných tahů z počáteční pozice ve formě (x, y) souřadnicového systému.

Soubor *Player.cpp* modeluje připojeného hráče. Obsahuje řadu informací o konkrétním hráči jako třeba jméno, socket či další parametry a stavy.

Důležitým souborem je *Game.cpp*, který reprezentuje hru dvou hráčů a zároveň slouží jako čekací místnost. Každá hra obsahuje kromě různých ukazatelů a proměnných také hrací pole reprezentované jako 2D vektor. Hrací pole nepoužívá (x, y) souřadnice, ale indexy řádků a sloupců. Třída dále obsahuje několik metod, které popisují životní cyklus jedné hry - od startu hry, přes řešení herních situací a validaci data, až po ukončení hry.

Dalším stěžejním souborem je *Lobby.cpp*. Je úzce svázán s hlavní metodou programu *main()*. Ve své podstatě převážně reaguje na klientské zprávy. Řeší především logiku přihlašování, znovu připojení do hry či odpojování klientů - ať už ruční, časové nebo odpojení po několika neúspěšných pokusech.

Soubor *State.cpp* obsahuje matici stavů a popisuje jednotlivé stavy a možné události.

3.2 Klient

Klientská část hry je napsaná v jazyce Python. Pro grafickou implementaci hry jsem použil knihovnu *Pygame*. Přihlašovací okno je vytvořeno pomocí defaultní knihovny *Tkinter*. Hlavním vstupním bodem klienta je metoda *main()* v souboru *main.py*. Klient se zde připojí k serveru a spustí se přihlašovací okno. Po návratu z tohoto okna (ať už chybovém nebo standartním) se řádně ukončí spojení se serverem a program skončí. Dále je aplikace členěna do dalších 6 souborů.

Soubor *parser.py* obsahuje příkazy TCP protokolu, použité při komunikaci. Ty mají opět formu enumů. Provádí se zde také překlad příchozích zpráv do formátu aplikace a odchozí zprávy ze zde serializují.

Konfigurační soubor *components.py* definuje různé systémové konstanty jako použité barvy nebo font písma. Dále jsou tu uvedeny herní karty pomocí mapy, kde klíčem je název karty a hodnotou je pole možných tahů z výchozí pozice (souřadnicový systém x, y).

V souboru *network.py* se nachází stejnojmenná třída, co řeší síťovou komunikaci se serverem, ať už se jedná o navázání a ukončení spojení, posílání a příjem dat (pomocí bufferování v *selectu*) nebo pingování serveru.

Soubor *intro.py* je zodpovědný za přihlášení hráče pomocí uživatelského jména. V metodě *login()* máme nekonečný cyklus, který vykresluje okno a přijímá zprávy ze serveru *selectem*. Validita jména je zkontrolována až na serveru. Případné nesprávné pokusy vyhodí uživateli chybovou hlášku pomocí

vyskakovací okna. Toto okno také uživatele informuje, pokud je odpojen od serveru. Pokud uživatel zadal správné jméno, je mu zobrazeno nové okno informující ho o tom, že je v čekající herní frontě.

Soubor *game.py* při startu hry zobrazí herní okno, které po úvodní inicializaci herních zdrojů (načtení obrázků a vytvoření objektu hry) opět iteruje v nekonečné smyčce. Ve smyčce se nejprve odchytávají a řeší uživatelské vstupy (kliknutí myší), poté *selectem* přijímá příchozí komunikace a nakonec je celá herní plocha překreslena řadou grafických metod za použití aktuálních dat o hře. Levá část okna obsahuje herní pole s figurkami. V pravé části se nachází herní kary a interaktivní labely, které hráči předávají informace o stavu hry a jménech hráčů.

Soubor *engine.py* reprezentuje již zmíněný objekt hry pomocí třídy *GameState*. Máme zde 2D herní pole, vybrané karty a další potřebné proměnné. Třída obsahuje několik metod - od zobrazení a realizace tahů, přes řešení herních stavů, až po opětovné nahrání dat po znovu připojení. Dále zde máme třídu *Move*, která představuje jeden tah a je použita především pro ladění a výpisy.

4 Požadavky pro spuštění

4.1 Server

Serverová aplikace dodržuje verzi C++17 a je přeložitelná nástrojem *Makefile* (s kompilátorem g++). Stačí v linuxové konzoli zadat příkaz *make* a vygeneruje se spustitelný soubor *server*. Při spuštění tohoto souboru je potřeba zadat dva parametry, kde port je v intervalu <0, 65535> a max. počet hráčů v intervalu <2, 100>:

./server <port> <max. počet hráčů>

4.2 Klient

Klientská aplikace je psána v interpretovaném jazyce Python, který se nativně spouští bez překladu. Podmínkou je mít nainstalované prostředí (v tomto případě Python 3.10) a potřebné knihovny.

4.2.1 Linux

Pro instalaci všech potřebných nástrojů můžeme v linuxu využít následující příkazy ve složce *client*. První nainstaluje prostředí s knihovnou *Tkinter* a balíčkový systém *pip* pro závislosti. Druhý příkaz využije *pip* k instalaci python knihoven popsaných v textovém dokumentu *requirements.txt*.

sudo apt install python3 python3-tk pip

pip install -r ./requirements.txt –user

Dále můžeme přistoupit ke spuštění klienta, u kterého potřebujeme dva parametry. V konzoli přejdeme do složky *client/onitama* a zde spustíme v linuxové konzoli následující příkaz:

python3 ./main.py <ip adresa serveru> <port>

4.2.2 Windows

U operačního systému Windows je opět potřeba mít nainstalované prostředí python spolu s defaultní grafickou knihovnou *Tkinter* a balíčkovým závislostním systémem *pip*. Oba dva nástroje se většinou instalují rovnou spolu s prostředím. Dále je potřeba obdobným způsobem ve složce *client* spustit Windows konzoli a zadat instalaci potřebných knihoven:

pip install –r requirements.txt –user

Pro spuštěné hry nyní podobným způsobem ve složce *client/onitama* zapneme konzoli a zadáme příkaz s následujícími parametry, kde port je opět v intervalu <0, 65535>:

python main.py <ip adresa serveru> <port>

5 Závěr

Serverovou i klientskou aplikaci se mi podařilo úspěšně naimplementovat podle bodů zadání. Navrhnul jsem protokol, který umožňuje oběma aplikacím komunikaci mezi sebou. Při práci jsem se seznámil s implementací grafického rozhraní pomocí knihoven *Tkinter* a *Pygame*. Dále jsem využil jak *sockety* pro síťovou komunikaci, tak strukturu *select* pro paralelizaci serverové aplikace.

Možným vylepšením do budoucna je možnost připojení do hry skrze herní lobby. První hráč by nejprve vytvořil lobby pro dva hráče. Druhý hráč by si mohl po přihlášení vybrat dané lobby, připojit se k němu a začít hrát s prvním hráčem.