Dáma pro dva hráče

Semestrální práce z předmětu
KIV/UPS

David Markov

Fakulta aplikovaných věd
Západočeská univerzita v Plzni
20. 1. 2021

Obsah

Kapitola 1

Zadání

Cílem této semestrální práce je vytvořit síťovou hru pro dva a více hráčů. Mělo by se jednat hru tahovou, například prší, šachy nebo dáma. Další možností jsou real-time nebo pseudo-real-time hry. Server by měl být implementován v jazyce C nebo C++. Klient by měl být napsán ve vysokoúrovňovém jazyce, jako je Java, C# nebo Kotlin. Protokol musí být textový nebo nešifrovaný. Na každý požadavek musí přijít od serveru odpověď.

Kapitola 2

Hra a její pravidla

Jako řešení byla zvolena Dáma. Z logiky hry vyplývá, že je hra vytvořena pro dva hráče. Herní plátno je pro zjednodušení a zrychlení hry pouze 6x6 polí, místo obvyklých 8x8 polí. Pole se střídají tmavá a světlá. Na horní a spodní polovině plátna jsou na tmavých polích rozmístěny žetony hráčů. Žetony se mohou pohybovat pouze po tmavých polích, tedy pouze diagonálně, a pouze ve směru vpřed. Horní hráč se tedy smí posouvat pouze dolů a spodní hráč pouze nahoru. Žetony mají dovoleno se posouvat pouze o jedno pole. Pokud je v diagonálně sousedním dostupném poli soupeřův žeton a další pole je volné hráč může žetonem pohnout o dvě pole, přemístit se na volné pole za soupeřovým žetonem a ten tímto způsobem vyřadit. Pokud se žeton dostane na opačný konec hracího pole, tedy na začátek hracího pole u soupeře, stává se z žetonu takzvaná Dáma. Dámy se mohou pohybovat opět pouze diagonálně, ale nyní nejsou limitovány počtem polí. Mohou se tedy diagonálně pohybovat o libovolnou vzdálenost. Oproti klasické dámě, zde mohou Dámy přeskakovat i vlastní žetony a svým pohybem vyřadí soupeřovy všechny žetony, které se nacházejí mezi počátečním a konečným bodem pohybu Dámy. Již tedy není nutné dodržet podmínku toho, že bude za soupeřovým žetonem jedno pole volné. Oproti klasické dámě, pokud lze soupeřův žeton vyřadit, není tento pohyb vynucený. Hra končí ve chvíli, kdy jeden z hráčů nemá žádné zbývající žetony na hracím plátně.

Kapitola 3

Popis implementace

Hra je rozdělena na klientskou a serverovou část. Klient odesílá požadavky na server přes BSD Sockety a server na ně odpovídá odpověďmi definovanými v protokolu níže.

Server

Server je napsaný v jazyce `C++`. Lze ho zkompilovat pomocí nástrojů `CMake` a `make` příkazy `cmake .` a `make` v kořenovém adresáři projektu. Pro úspěšný překlad je třeba kompilátor s podporou standardu `C++17` a knihovna `filesystem`. Po překladu je v kořenovém adresáři vytvořen spustitelný soubor `dama-server`. Server pro obsluhu jednotlivých připojení využívá pseudo-paralelní přístup pomocí funkce `select`.

Při startu serveru se validují dva vstupní parametry - IP adresa, na které má server naslouchat a port. Parametry jsou předávány z příkazové řádky ve zmíněném pořadí. Server naslouchá na socketu spjatým s předanou adresou a portem a validuje příchozí zprávy klientských připojení. Pokud je zpráva validní, server vykoná akci odpovídající žádosti klienta a odešle potvrzující odpověď v předem definovaném formátu. Pokud zpráva není validní, server klientovi odešle odpověď signalizující nevalidní zprávu. Všechna klientská připojení jsou uchovávána a mají počítadlo nevalidních dotazů. Pokud klient dosáhne maximálního počtu nevalidních zpráv, ve výchozím stavu 3, je mu odeslána odpověď o jeho odpojení a je automaticky odpojen.

Hráči jsou po připojení přesunuti do lobby, kde vidí jednotlivé hry dostupné k připojení. Pokud se hráč nechce připojit k žádnému z čekajících protihráčů, může si vytvořit hru vlastní a počkat na připojení protihráče. Po ukončení hry, jsou oba hráči přesunuti zpět do lobby. Pokud jeden z hráčů hru úmyslně opustí, hra končí. Pokud hráč hru opustí neúmyslně, například kvůli chybě připojení, hra je pozastavena a čeká se na jeho opětovné připojení. Pokud hráč, čekající

na opětovné připojení protihráče, hru opustí, hra je ukončena a oba hráči jsou přesunuti do lobby.

Struktura

Server je postaven na zjednodušené třívrstvé architektuře.

SocketListener, ConnectionService

Třídy `SocketListener` a `ConnectionService` tvoří komunikační vrstvu aplikace. `SocketListener` naslouchá na socketu svázaném s adresou a portem a přijímá nová připojení pomocí funkce `select`. Přijaté zprávy posílá dále do aplikační vrstvy k vyhodnocení a odpovědi aplikační vrstvy předává k odeslání klientovi do `ConnectionService`.

`ConnectionService` obstarává odesílání zpráv na jednotlivé klientské sockety a ukládání příchozích připojení do datové vrstvy. Pokud je klient odpojen, třída obsluhuje vymazání a změnu náležitých datových objektů.

ResponseHandler, PlayerService, GameService

Třídy `ReponsoHandler`, `PlayerService` a `GameService` tvoří aplikační vrstvu serveru. `ResponseHandler` přebírá přečtené dotazy od klienta a vyhodnocuje je. Pokud jsou dotazy ve validním formátu, dále komunikuje s `PlayerService` a `GameService` pro manipulaci s vhodnými datovými objekty. Pokud je nutno odeslat více klientům informaci o stavu aplikace, komunikuje s `ConnectionService` pro odeslání patřičných informací klientům.

PlayerRepository, GameRepository, ConnectionRepository

Datovou vrstvu aplikace tvoří třídy `PlayerRepository`, `GameRepository` a `ConnectionRepository`, které ukládají jednotlivé datové objekty a poskytují rozhraní pro jejich snadné vyhledávání podle vstupních parametrů.

Player, Game, Connection

- **Player** je datový objekt reprezentující přihlášeného hráče a obsahuje důležité informace o jeho stavu ve hře a identifikaci
  - `int userId` - jednoznačný identifikátor hráče
  - `std::string nickname` - hráčem zvolená přezdívka
  - `PlayerState` - stav hráče (v lobby, ve hře čeká, ve hře na tahu, odpojen)
- **Game** je datový objekt reprezentující založenou hru, obsahuje informace o aktuálním stavu hry
  - `int gameId` - jednoznačný identifikátor hry

  - `Player* playerOne` - hráč, který hru založil
  - `Player* playerTwo` - hráč, který se do hry připojil
  - `GameState state` - aktuální stav hry (čeká na připojení protihráče, v průběhu, pozastaveno)
  - `std::vector<GameToken> playerOneDraughts` - žetony hráče 1, které jsou přeměněny v Dámu
  - `std::vector<GameToken> playerTwoDraughts` - žetony hráče 2, které jsou přeměněny v Dámu
  - `int board[ROW_COUNT][ROW_COUNT]` - matice reprezentující hrací plochu, nabývá hodnot 0, `playerOne.userId`, `playerTwo.userId`
  - `bool isOver` - indikátor stavu hry
- **Connection** reprezentuje jednotlivá klientská připojení
  - `int socket` - file descriptor socketu, na který je klient připojen
  - `int userId` - přiřazený identifikátor hráči, slouží jako odkaz na objekt `Player` po přihlášení

Klient

Klientská část je implementována v jazyce `Java` verze 11 pomocí grafické knihovny `JavaFX`. Projekt lze zkompilovat pomocí nástroje `Maven` příkazem `mvn package` v kořenovém adresáři projektu. Po úspěšném přeložení je vytvořena podsložka `target`, ve které se nachází soubor `dama-client-1.0-SNAPSHOT.jar`, spustitelný z příkazové řádky pomocí příkazu `java -jar dama-client-1.0-SNAPSHOT.jar`.

Struktura

Jednotlivé obrazovky jsou spravovány svými přidruženými `Controllery`.

Komunikace se serverem je zprostředkována pomocí třídy `Connector`, která odesílá zprávy na server a zprávy přijímá.

Vykreslování jednotlivých oken zprostředkovává třída `Renderer` s pomocí jednotlivých kontrolerů.

Uživatelské rozhraní běží v samostatném vlákně, ve kterém jsou odesílány zprávy na server. Zprávy ze serveru jsou přijímány v druhém samostatném vlákně. Ve třetím vlákně je periodicky odesílám dotaz `PING` pro zjišťování stavu serveru.

Uživatelské rozhraní ošetřuje nevalidní vstupy, stejně tak jako server.

Kapitola 4

Protokol

Oddělovač parametrů zprávy - `|`

Znak konce zprávy - `\n`

Na začátku zprávy musí být vždy `userId`.

Formát zprávy - `<userId>|TYP_POZADAVKU|parametry\n`.

Uživatel je jednoznačně identifikován svým `userId` a zvolenou přezdívkou. Přezdívka může obsahovat pouze znaky `a-zA-Z0-9`.

Oddělovač parametrů se nesmí vyskytnout nikde v jiném významu než ve významu oddělení parametrů.

Obecné odpovědi

`410\n` - nevalidní ID

`420\n` - nevalidní stav hráče/hry, nelze provést operaci

`450\n` - obecná chyba protokolu, nerozeznán dotaz

Přihlášení

- Dotaz - `<userId>|CONNECT|<prezdivka>\n`
  - uživatel zatím nemá `userId`, musí být 0
- Možné odpovědi:
  - `200|<userId>\n` - přihlášení ok, přišlo přiřazené ID
  - `201|<userId>\n` - reconnect ok, přišlo nové ID

  - `400\n` - nevalidní přezdívka
  - `401\n` - přezdívka obsazena
  - `410\n` - posláno nevalidní ID

Odhlášení

- Dotaz - `<userId>|LOGOUT\n`
- Možné odpovědi:
  - `202\n` - odhlášení ok
  - `410\n` - nevalidní ID
  - `420\n` - nevalidní stav, je třeba být v lobby

Get games

Vrátí seznam všech her čekajících v lobby na připojení protihráče.

- Dotaz - `<userId>|GET_GAMES\n`
- Možné odpovědi:
  - `203|<prezdivka1>|<prezdivka2>|...\n` - přezdívky jsou přezdívky hráčů, kteří čekají na připojení protihráče
  - `410\n` - nevalidní ID

Vytvoření hry

Vytvoří hru a bude čekat na protihráče, než se připojí.

- Dotaz - `userId|CREATE_GAME\n`
- Možné odpovědi:
  - `204\n` - hra vytvořena, čekám na protihráče
  - `410\n` - nevalidní ID
  - `420\n` - nevalidní stav, je třeba být v lobby

Všichni hráči v lobby dostanou zprávu `250|<prezdivka>\n` s přezdívkou hráče, který hru založil.

Opuštění hry

Pokud je hráč ve hře nebo čeká na připojení protihráče, hra bude opuštěna a smazána. Protihráč dostane zprávu `280\n` signalizující. že je konec hry.

- Dotaz - `userId|CREATE_GAME\n`
- Odpovědi:
  - `205\n` - ok, hra opuštěna, návrat do lobby
  - `410\n` - nevalidní ID
  - `420\n` - nevalidní stav, hráč není ve hře

Pokud hráč ztratí spojení, protihráč dostane zprávu `290\n` a hra se pozastaví. Pokud se hráč připojí zpět, protihráč dostane zprávu `295\n` a hra se opět spustí.

Připojení do existující hry v lobby

- Dotaz - `<userId>|JOIN_GAME|<prezdivka>\n` - přezdívka je nick hráče, který ve hře čeká na připojení.
- Odpovědi:
  - `206\n` - OK, jste ve hře
  - `410\n` - nevalidní ID
  - `420\n` - nevalidní stav, hráč není v lobby
  - `402\n` - hráč se zadanou přezdívkou nečeká na připojení nebo neexistuje

Dotaz na aktuální stav hry

- Dotaz - `<userId>|GET_GAME_STATE\n`
- Odpovědi:
  - `207|<prezdivkaHrac1>,<zeton1X>,<zeton1Y>,<zeton2X>,<zeton2Y>,...|<poziceDamy1X>,<po`
    - stav hráče je vždy přezdívka, souřadnice žetonů oddělené čárkou a souřadnice Dám oddělené čárkou
  - `410\n` - nevalidní ID
  - `420\n` - nevalidní stav, hráč není ve hře

Tah ve hře

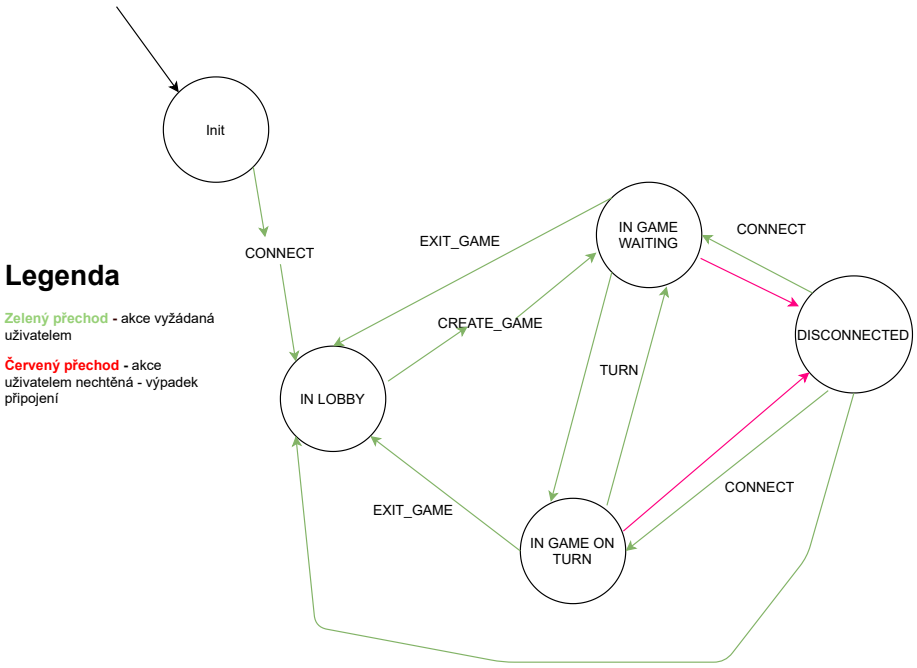
- Dotaz - `<userId>|TURN|<fromX>,<fromY>|<toX>,<toY>\n` - dotaz na tah z XY na XY
- Odpovědi:
  - `208|<GAME_STATE>` - OK, vrátí aktuální stav hry podle vzoru z odpovědi na dotaz `GET_GAME_STATE`
  - `403\n` - nevalidní tah, špatné souřadnice - souřadnice nejsou v hracím poli/na zdrojové souřadnici není hráčův žeton/na cílovou souřadnici se nelze přesunout
  - `410\n` - nevalidní ID
  - `420\n` - nevalidní stav, hráč není v běžící hře nebo není na tahu

Ping

Informativní dotaz na server, zda běží

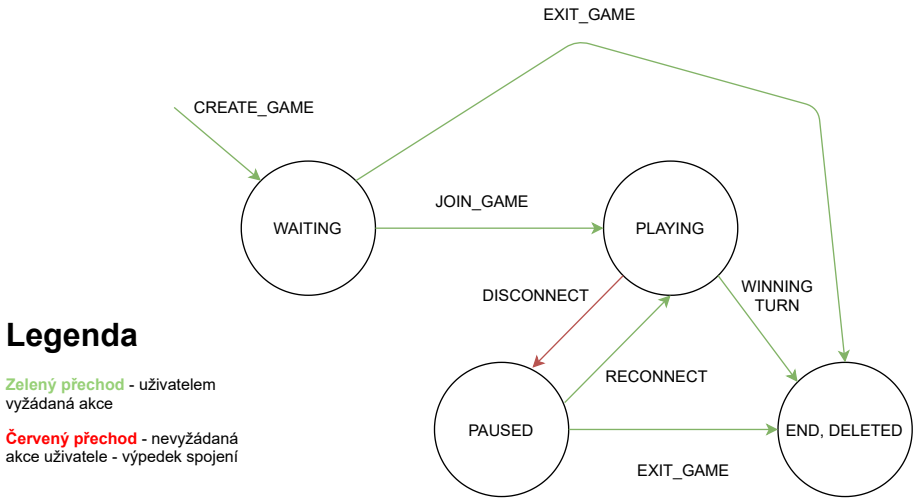
- Dotaz - `<userId>|PING\n` - user ID může být libovolné
- Odpověď - `100\n` - OK

Stavový diagram hráče



Obrázek 4.1: Stavový diagram hráče

Stavový diagram hry



Obrázek 4.2: Stavový diagram hry

Kapitola 5

Závěr

Server i klientská část se podařilo implementovat podle požadavků v zadání a obě části jsou plně funkční. Server dokáže obsluhovat vícero klientů a her najednou a reaguje správně na validní i nevalidní vstupy. Po maximálním množství nevalidních vstupů server klienty odpojuje a nastavuje je do korektního stavu. Klientská část nabízí přívětivé a jednoduché uživatelské rozhraní a správně zobrazuje aktuální stav hry a připojení k serveru. K validaci vstupů dochází i na klientské části