Semestrální práce z KIV/UPS

Dots and Boxes

Vladan Trhlík

5. ledna 2025

1 Popis hry

Hra Dots and Boxes je strategická tahová hra pro dva hráče. Herní pole tvoří čtvercová mřížka teček (v tomto případě 4×4), a cílem hry je propojit sousední tečky tak, aby vznikl čtverec. Hráč, který uzavře čtverec, získá bod a pokračuje dalším tahem. Hra končí, jakmile jsou všechny čtverce uzavřeny, a vítězem se stává hráč, který získal více bodů.

2 Protokol

2.1 Základní popis

Komunikace mezi klientem a serverem je zajištěna pomocí TCP protokolu. Při posílání zpráv mezi klienty a serverem se jednotlivé části zpráv oddělují znakem '`|`' a ukončovací znak je '`\n`'. Jména hráčů a názvy her jsou omezena na velká a malá písmena a znak '`_`'. Pokud není zpráva ve správném formátu, nebo zadané parametry nedávají v aktuálním kontextu smysl, odpovědí je zpráva `ERR<n>`, kde `n` je číslo jedné z chybových zpráv:

- `1` – invalid
- `2` – name already exists
- `3` – player not on turn
- `4` – player not in game
- `5` – already in game
- `6` – max limit exceeded

2.2 Zprávy

Přihlášení

- **Client:** `LOGIN|<name>`
- **Server:** `OK / ERR<1/2/6>`
- `<name>`: přihlašovací jméno hráče

Načtení všech her

- **Client:** `LOAD`
- **Server:** `OK|<game1-name>|<game2-name>|...|<gameN-name>`
- `<game-name>`: název I-té hry

Vytvoření hry

- **Client:** `CREATE|<game-name>`
- **Server:** `OK / ERR<1/2/6>`
- `<game-name>`: název hry

Připojení do hry

- **Client:** `JOIN|<game-name>`
- **Server:** `OK|<op-name> / ERR<1/5/6>`
- `<game-name>`: název hry, `<op-name>`: jméno oponenta

Oponent se připojil

- **Server:** `OP_JOIN|<op-name>`
- **Client:** `OK / ERR<4/5>`
- `<op-name>`: jméno oponenta

Oponent se odpojil

- **Server:** `OP_LEAVE`
- **Client:** `OK / ERR<4>`

Odpojení od hry

- **Client:** `LEAVE`
- **Server:** `OK / ERR<4>`

Tah

- **Server, Client:** `TURN|<X>|<Y>`
- **Client, Server:** `OK / ERR<1/3>`
- `<X>`, `<Y>`: pozice tahu
- Pokud přijde zpráva ze severu, jedná se o tah oponenta, pokud od klienta, posílají se data jeho oponentovi.

Obsazení čtverce

- **Server:** `(OP_)ACQ|<X>|<Y>[|<X2>|<Y2>]`
- **Client:** `OK`
- `<X>`, `<Y>`, `<X2>`, `<Y2>`: pozice čtverce
- `OP_ACQ` informuje o obsazení oponentem, `ACQ` hráčem, který zprávu přijímá.
- Při propojení dvou teček může dojít k obsazení jednoho nebo dvou čtverců – podle toho se pošle počet pozic čtverců.

Hráč je na tahu

- **Server:** `ON_TURN`
- **Client:** `OK`

Oponent je na tahu

- **Server:** `OP_TURN`
- **Client:** `OK`

Konec hry

- **Server:** `END|<WIN/LOSE>`
- **Client:** `OK`

Ping

- **Client:** `PING`, `PONG`
- **Server:** `PONG`, `PING`

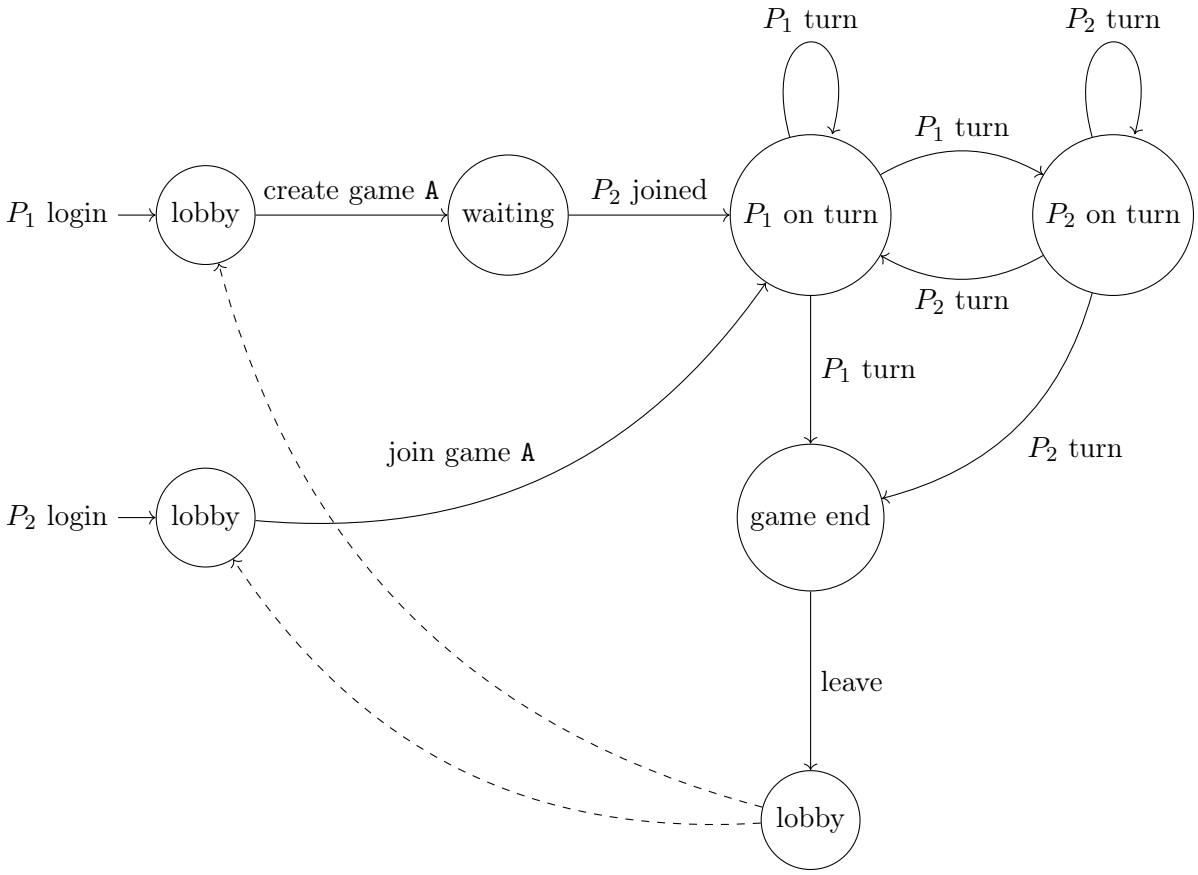
Synchronizace stavu hry

- **Client:** `SYNC`
- **Server:** `OK|<w>|<h>|<stick-data>|<square-data> / ERR<4>`
- `<w>`, `<h>`: velikost hracího pole, `<stick-data>`: data o propojených tečkách, `<square-data>`: data o obsazených čtvercích
- data o propojených tečkách a obsazených čtvercích jsou řetězce z čísel 0,1,2
  - 0 = nepropojeno / neobsazeno
  - 1 = hráč který žádá o `SYNC`
  - 2 = oponent

Připojení po výpadku spojení

- **Client:** `RECONNECT|<name>|<game-name>`
- **Server:** `OK|<scene-code>`
- `<name>`: jméno hráče (nepovinné), `<game-name>`: název hry (nepovinné)
- `<scene-code>`: kód akce pro klienta (0 = login, 1 = lobby, 2 = game)
- Server se na základě údajů `<name>` a `<game-name>` rozhodne, zda-li se hráč připojí zpět do hry (2), nebo zůstane v lobby (1). Při neznámém jménu hráče je přesunut na nový login (0). Pokud byl hráč ve hře a připojí se zpět po více než 30 s, je automaticky přesunt do lobby.

3 Diagram



Obrázek 1: Stavový automat představující cyklus hráče.

4 Popis implementace

4.1 Server

Server je napsán jazyce C za použití základních knihoven. Je rozdělen do několika modulů:

- `main` – vstupní bod programu
- `server` – funkce pro spuštění serveru a obsluhy klientů
- `handlers` – obslužné funkce pro jednotlivé typy zpráv
- `game` – funkce pro vytváření hry, ověřování správnosti tahů apod.
- `utils` – užitečné funkce pro odesílání zpráv a ověřování přechodů mezi stavy klientů

Všechny struktury jsou definovány ve `structs.h` – obsahují strukturu `Game`, `Server` a `Player`, které slouží k uložení dat o jednlivých hráčích a hrách. Kromě toho se zde nachází i typy zpráv, výčty možných stavů a událostí, které jsou využity v přechodovém automatu (v `utils.c`).

4.1.1 `main.c`

Jako vstupní bod programu slouží `main.c`. Na začátku vytváří strukturu `Server`, která připravena a naplněna daty pomocí funkce `server_create()` a v nekonečném cyklu volá `server_handle()`.

4.1.2 `server.c`

Tento soubor obsahuje všechny potřebné funkce pro spravování připojení serveru k jednotlivým klientům a správnou manipulaci s daty. Patří mezi ně:

- `int server_create(Server *s, char *config_file)`

  Nejprve načte konfigurační soubor se jménem `config_file`, který obsahuje IP adresu a port serveru a další hodnoty jako maximální počet připojených hráčů a vytvořených her v jeden moment. Dále vytvoří sever socket a připraví potřebné struktury pro server.

- `int server_handle(Server *s)`

  Toto je základní funkce, která běží v `main.c` v nekonečné smyčce a obstarává komunikaci s klienty na nejnižší úrovni. Pomocí `select()` přijímá nové klienty a zprávy, které dále předává funkci `handle_msg`. Dále také v intervalech posílá `PING` všem klientům a spravuje tak připojené a odpojené klienty.

- `int handle_msg(Server *s, SEvent type, int fd, char *msg)`

  K zpracování zpráv a událostí slouží funkce `handle_msg()`. Přijímá 3 různé typy událostí – `CONNECT`, `MSG` a `DISCONNECT`.

  Při připojení nového klienta je vytvořena nová struktura `Player`, která je přidána do pole `players[]` ve struktuře `Server`. Naopak při odpojení je z tohoto pole odstraněn, ale to jen v případě, že nebyl přihlášen pomocí `LOGIN`. Pokud se již hráč přihlásil, zůstává na serveru uložen pro budoucí přihlášení zpět pod stejným jménem.

  Pro zpracování jednotlivých zpráv jsou nadefinované pole, které obsahuje zprávu v podobě řetězce a ukazatel na funkci, která tuto zprávu obsluhuje. Díky tomu stačí zjistit, která z obslužná funkce patří k přijaté zprávě a následně ji zavolat. Zbytek přijaté zprávy je v každé obslužné funkci načítán pomocí `strtok()`, díky čemuž nemusí přebírat žádné parametry a všechny funkce mají stejný prototyp. Pokud je zpráva ve špatném formátu, zvedne se čítač nevalidních zpráv u daného hráče. Pokud překročí 10 nevalidních zpráv, je hráč odpojen.

- `int remove_player(Server *s, Player *p)`

  Tato funkce slouží k odstranění struktury hráče ze serveru a uvolnění z paměti.

- `int remove_game(Server *s, Game *g)`

  Tato funkce slouží k odstranění hry ze serveru a uvolnění z paměti.

4.1.3 `handlers.c`

Při zpracování zpráv všechny obslužné funkce kontrolují, jestli je hráč ve správném stavu stavového automatu z obrázku 1. Pokud je zpráva validní, je hráčův stav změněn na jiný dle předchozího stavu a typu události.

Pokud jsou předané argumenty zpráv chybné nebo se hráč nenachází ve správném stavu, je o tom informován chybovou zprávou `ERR`.

- `int login_handler(Server *s, Player *p)`

  Tato funkce slouží k přihlášení hráče na server. Načítá jeho jméno, u kterého ověřuje zda-li obsahuje pouze povolené znaky a jestli už na serveru neexistuje hráč se stejným jménem.

- `int create_handler(Server *s, Player *p)`

  Funkce `int create_handler()` slouží k vytvoření nové hry. Testuje, jestli název obsahuje pouze povolené znaky a jestli už na serveru neexistuje hra se stejným jménem. V případě úspěšného vytvoření hry je odesílatel zprávy přidán jako první hráč.

- `int join_handler(Server *s, Player *p)`

  Pro zpracování zprávy k připojení do hry je `int join_handler()`. Najde hru, jejíž název se shoduje s předaným názvem a zkontroluje zda-li hra není plná. Pokud vše sedí, hráč je připojen do hry.

- `int turn_handler(Server *s, Player *p)`

  Tato funkce slouží ke zpracování zprávy o tahu hráče. Kontroluje, jestli je hráč na tahu, jestli je jeho tah v mezích herního pole a také že na zadané pozici už nebyl proveden jeden z předchozích tahů.

- `int leave_handler(Server *s, Player *p)`
- `int load_handler(Server *s, Player *p)`
- `int sync_handler(Server *s, Player *p)`
- `int reconnect_handler(Server *s, Player *p)`

4.1.4 `game`

4.1.5 `utils`

4.2 Client

Client je napsán v jazyce Python za použití základních knihoven, grafické knihovny `pygame` a GUI knihovny `pygame_gui`. Program je rozdělen do scén, mezi kterými se přepíná. Při implemetaci byla snaha se držet MVC architektury, ale ta byla kvůli jednoduchosti většiny scén zachována jen v samotné scéně hry.

Každá scéna dědí třídu `Scene`, která obsahuje základní metody na zpracování vstupu od uživatele a správného vykreslení všech komponent. Měnění mezi scénenami zajišťuje jednoduchá třída `SceneManager`.

Pro komunikaci se serverem je zde třída `Socket`, která za využitím knihovny `socket` zajišťuje komunikaci se serverem, frontu přijatých zpráv a metody zajišťující opětovné připojení při přerušení spojení. Tato třída běži v druhém vlákně pro zajištění plynulého chodu uživatelského rozhraní.