Semestrální práce z KIV/UPS

Dots and Boxes

Vladan Trhlík

22. 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

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

Načtení všech her

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

Vytvoření hry

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

Připojení do hry

- **Klient:** `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>`
- **Klient:** `OK / ERR<4/5>`
- `<op-name>`: jméno oponenta

Oponent se odpojil

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

Odpojení od hry

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

Tah

- **Server, Klient:** `TURN|<X>|<Y>`
- **Klient, 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>]`
- **Klient:** `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`
- **Klient:** `OK`

Oponent je na tahu

- **Server:** `OP_TURN`
- **Klient:** `OK`

Konec hry

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

Ping

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

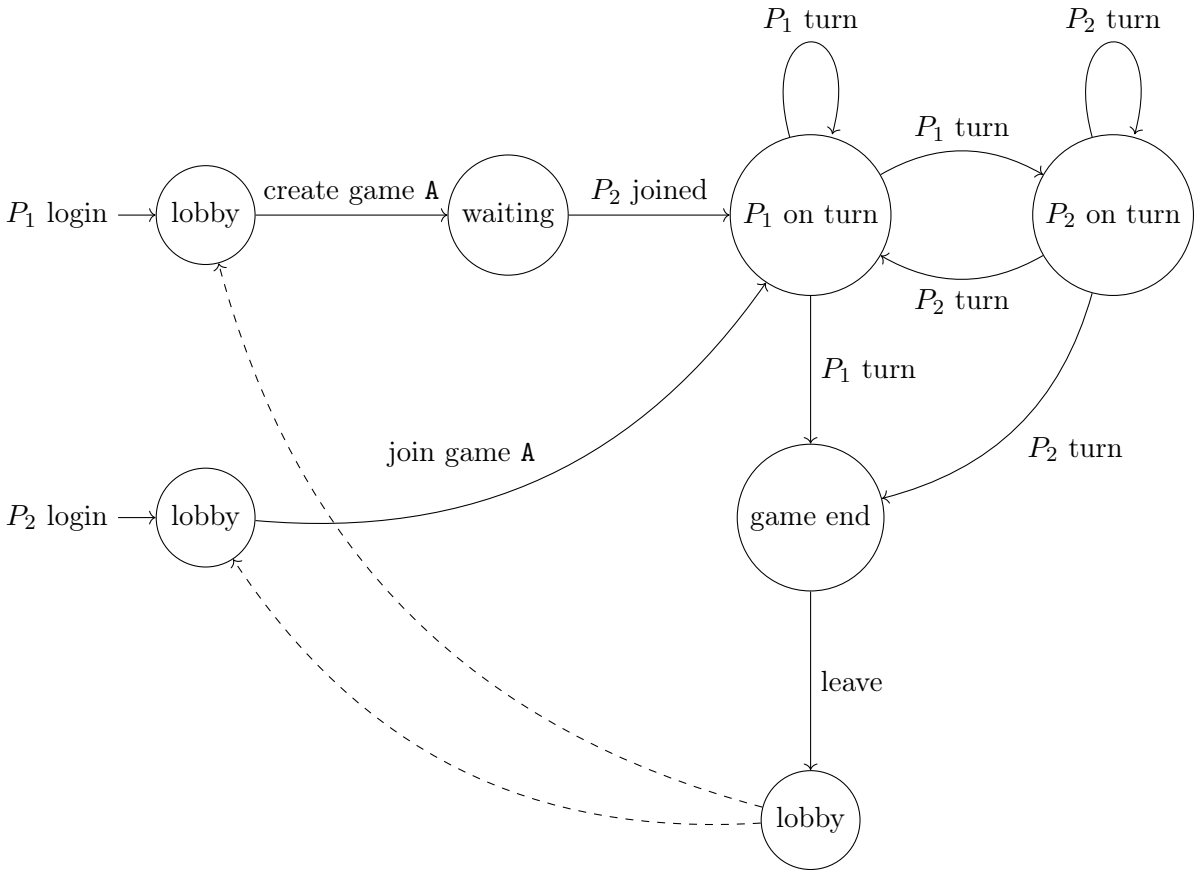
Synchronizace stavu hry

- **Klient:** `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

- **Klient:** `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

Při implementaci bylo nutné sjednotit reprezentaci herního pole serveru a klienta. Pro uložení stavu hry bylo potřeba ukládat propojení teček a obsazení čtverců + kterému hráči dané pole patří. Reprezentace čtverců byla jednodušší, protože se dá jednoduše uložit na dvourozměrné pole o velikosti 3×3.

K propojení teček by se dal použít například graf, ale ten by byl i při propojení všech sousedních teček velice řídký. Nakonec se i propojení teček reprezentuje dvourozměrných polem, ale počet prvků v jeho řádcích je proměnlivý. Pole má 7 řádků, kde sudé řádky mají 3 prvky (vodorovné spojnice) a liché 4 (svislé spojnice). Tato struktura je zachována i při synchronizaci herního pole pomocí `SYNC`, kde jsou přenášena data ve stejném pořadí jako jsou uložena v poli.

4.1 Server

Server je napsán jazyce C za použití standardních 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`).

Obsluha klientů běží jako jeden proces, který pomocí `select()` vybírá klienty, u kterých došlo ke změně nebo od nich přišla zpráva. Pokud od nějakého klienta nepřijde po nějakém časovém kvantu žádná zpráva, je jeho připojení ověřováno pomocí zprávy `PING`. Aby mohl jeden proces ověřovat připojení a obsluhovat klienty, je funkci `select()` nastaven timeout tak, aby se v rozumných intervalech kontrolovala doba od poslední zprávy jednotlivých klientů.

Pro každý typ zprávy je v modulu `handlers` obslužná funkce. Všechny obslužné funkce mají stejnou hlavičku, což umožňuje vytvořit pole těchto funkcí a pole řetězců definující jednotlivé typy zpráv. Díky tomu pak stačí ve smyčce projet všechny možné zprávy a při shodě zavolat příslušnou funkci.

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`.

4.2 Klient

Klient je napsán v jazyce Python za použití základních knihoven, grafické knihovny `pygame` a GUI knihovny `pygame_gui`. Program je rozdělen do několika modulů:

- `main` – vstupní bod programu
- `user` – třída pro ukládání dat o uživateli
- `scene`, `scene_manager` – základní třída scény + obsluha scén
- `mysocket` – třída pro zajištění komunikace se serverem
- `login` – scéna přihlášení
- `lobby` – scéna lobby
- `game`, `game_data`, `game_view` – scéna hry

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`. Data o uživateli si jednotlivé scény předávají v instanci třídy `User`, která obsahuje všechna potřebné informace.

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.

5 Sestavení a spuštění

5.1 Server

Pro sestavení serveru je potřeba překladač `gcc` a nástroj k sestavení `make`. Samotný překlad je zahájen příkazen `make` v adresáři obsahující soubor `Makefile`. Po úspěšném dokončení by měl vzniknout spustitelný soubor `server`.

Pro nastavení serveru lze využít konfigurační soubor `config.txt`, který obsahuje 4 předdefinované proměnné a jejich hodnoty (odděleny mezerou, každá na jeden řádek):

proměnná	popis	příklad
`maxPlayers`	maximální počet přihlášených hráčů	10
`maxGames`	maximální počet her	4
`port`	port, na kterém server naslouchá	10000
`ip`	ip adresa, na kterém server naslouchá	192.168.1.16

Tabulka 1: Proměnné konfiguračního souboru.

5.2 Klient

Pro spuštění klienta je potřeba mít nainstalovaný `Python` a knihovny `pygame` a `pygame_gui`. Klient ma při spuštění dva nepovinné parametry pro nastavení ip adresy a portu serveru, a spustí se takto:

```
python main.py [ip] [port]
```

Pokud nejsou parametry zadány, jsou využity výchozí hodnoty (port = 10000, ip = localhost).

6 Závěr

S výsledkem semestrální práce jsem velmi spokojen. Tato práce byla mou první zkušeností s tvorbou programu využívajícího síťovou komunikaci, což mi umožnilo naučit se řadu nových věcí a získat cenné praktické zkušenosti.

Pro vývoj serveru jsem zvolil programovací jazyk `C`, protože mě tento jazyk baví a cítím se v něm velmi komfortně. S implementací serveru jsem spokojen, protože je navržen tak, aby byl snadno rozšiřitelný a dokázal obsloužit až desítky klientů současně (větší zátěž nebyla testována). Jedním z možných zlepšení by mohl být podrobnější výpis stavu serveru, který by v přehledné tabulce zobrazoval informace o připojených hráčích a probíhajících hrách.

Pro klienta jsem zvolil `Python`, díky jeho jednoduchosti, která usnadňuje rychlé prototypování, a také díky svým předchozím zkušenostem s knihovnou `pygame`. Aplikace klienta je navržena tak, aby poskytovala veškerou potřebnou funkcionalitu v co nejjednodušším a přehledném rozhraní. K dispozici je lišta, která vždy zobrazuje informace o připojení k serveru a jménu uživatele. Možným vylepšením by bylo zlepšení uživatelského rozhraní v lobby, kde by bylo vhodné lépe odlišit možnosti připojení k existující hře od vytvoření nové.