FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

Semestral Work Documentation

KIV/UPS

Nikol Caltová
A21B0091P
ncaltova@students.kiv.cz
27. února 2024

1 Popis hry

Za zadání jsem zvolila karetní hru prší, která se drží oficiálních pravidel.

1.1 Pravidla

- Počet hráčů: 2 - 4
- Cílem hry je co nejrychleji nemít žádné karty
- Na začátku hry, se všem hráčům rozdají čtyři karty z balíčku a jedna karta se vyloží na stůl
- První vyložená karta po rozdání nemá žádný efekt, tedy eso by neznamenalo, že první hrající stojí
- Dále platí, že karty na sebe lze pokládat, pokud mají stejný druh či hodnotu, nebo pokud se jedná o svrška, a také nelze přebíjet

2 Protokol

2.1 handshake

- Slouží pro prvotní navázání spojení se serverem, tedy připojí uživatele do lobby, odkud se může dále připojit do fronty na hru.
- Request:

```
request_type=handshake\n
```

- Response:

```
response_type=handshake&status_code=200\n
```

2.2 join_game

- Slouží pro připojení hráče do fronty na hru, nebo pro zpětné připojení do aktivní hry, pokud je zadán parametr player_id. Pokud se hráč připojuje do herní fronty, dostane s odpovědí player_id.

- Request:

```
request_type=join_game\n
request_type=join_game?player_id=INT\n
```

- Response:

```
response_type=join_game&status_code=200&player_id=INT\n
```

2.3 leave_game

- Slouží pro odpojení z fronty na hru, nebo aktivní hry. Po odpojení je hráč připojen zpět do lobby.
- Request:

```
request_type=leave_game\n
```

- Response:

```
response_type=leave_game&status_code=200\n
```

2.4 exit

- Slouží pro odpojení z herního lobby, tedy odpojení ze serveru.
- Request:

```
request_type=exit\n
```

- Response:

```
response_type=exit&status_code=200\n
```

2.5 is_alive

- Slouží pro pingování serveru, zda je stále naživu.
- Request:

```
request_type=is_alive\n
```

- Response:

```
response_type=is_alive&status_code=200\n
```

2.6 move_finish

- Slouží pro potvrzení dokončení tahu, musí být odeslán i pokud je svrchní karta stopka či sedmička. Odpovědí server informuje hráče o tahu, který provedl, tedy i pokud se hráč pokouší položit kartu, když stojí, tak server pokus ignoruje a odešle odpověď, která klientovi říká, že stojí. Parametr change požadavku má význam druhu karet, na který se mění, pokud je použit svršek. Pokud uživatel není na tahu, ale odešle tento požadavek, server odpoví chybovou odpovědí.
- Request:

```
request_type=move_finish?player_id=INT&
card=[card_type=STR&card_value=STR]&change=STR
```

- Response:

```
response_type=move_finish&status_code=200&cards=
[card_type=STR&card_value=STR]&type=STR

response_type=error&status_code=400&active=true&game_end=false
```

2.7 game_state

- Slouží pro zjištění aktuálního stavu hry až na karty, které aktuálně uživatel drží v ruce. Pokud je požadavek odeslán ve frontě na hru nebo lobby, hra odpoví errorovou odpovědí s kódem 300, tedy lze podle odpovědi poznat, zda už byl klient připojen do herní místnosti. Odpověď obsahuje ID aktuálního hráče na tahu, pole informací o oponentech, tedy jejich ID, počet karet v ruce a zda jsou aktivní, aktuální kartu vyloženou na stůl a current_type, který je vyplněn, pokud je použit svršek a změněna barva, tedy obsahuje aktuální typ karet, které lze vyložit.

- Request:

```
request_type=game_state\n
```

- Response:

```
response_type=game_state&status_code=200&current_player_id=INT&
players=[player_id=INT&count=INT&is_active=BOOL; ...]&top_card=
[card_type=STR&card_value=STR]&current_type=STR

response_type=error&status_code=300
```

2.8 player_deck

- Slouží pro zjištění aktuálního stavu karet, které uživatel drží v ruce. Pokud je požadavek odeslán ve frontě na hru nebo lobby, hra odpoví errorovou odpovědí s kódem 300, tedy lze podle odpovědi poznat, zda už byl klient připojen do herní místnosti. Odpověď obsahuje pole karet, které uživatel právě drží v ruce.

- Request:

```
request_type=player_deck\n
```

- Response:

```
response_type=player_deck&status_code=200&cards=
[card_type=STR&card_value=STR; ...]

response_type=error&status_code=300
```

2.9 game_end

- Slouží pro potvrzení konce hry, hra čeká dokud uživatel o toto potvrzení nezažádá, až pak hráče přepojí do lobby. Pole winner odpovědi obsahuje ID vítěze hry.

- Request:

```
request_type=game_end\n
```

- Response:

```
response_type=game_end&status_code=200&winner=INT
```

3 Popis implementace

3.1 Server

3.1.1 Využité technologie

Server je psán v jazyce go, za využití standardní knihovny pro práci se sockety, členěn je do dvou částí, obsluhy lobby a čekací místnosti a hry, kdy jednotlivé hry jsou vlastní vlákna. Aplikace se přeloží do spustitelného balíčku za využití go modulů a příkazu `go build`

3.1.2 main.go

Jedná se o hlavní soubor serveru, stará se o obluhu nových připojení a kontrolu lobby a čekací místnosti, jednotlivé místnosti jsou pak reprezentovány jako pole se sockety, nebo obalující strukturou Player. O obsluhu každé místnosti se stará odpovídající funkce `checkLobby()` a `checkWaitRoom()`.

Po provedení handshake, je klient přesunut do lobby, pokud uživatel v tuto chvíli odešle požadavek na připojení do hry, je odpovídající socket zabalen do struktury Player a přesunut do čekací místnosti. Pokud se s požadavkem na připojení do hry odešle i ID hráče, server se pokusí hráče nabídnout jedné z aktuálně běžících herních místností.

Pokud server zjistí, že v čekací místnosti je více než dvě připojení, tak je odebere z fronty a spustí pro ně herní místnost. Připojení jsou v tuto chvíli už zabalené ve struktuře Player. Při přesunu se pak připojení mažou z pole reprezentující čekací místnost.

3.1.3 game.go

Jedná se o kód samotné herní místnosti, celý chod hry zařizuje funkce

```
gameService(),
```

která po skončení hry vrátí ID výherce, pokud ovšem narazí na situaci, kdy jsou všichni klienti nektivní vrací error. Po skončení, v situaci, kdy vrátí ID výherce bez erroru se přejde k obsluze posledních requestů, kdy pokud si uživatel zažádá o game_end je přepojen zpět do lobby. Herní místnost se ukončí ve chvíli, kdy jsou všichni klienti neaktivní, nebo úspěšně obdrželi game_end. Ve chvíli, kdy vrátí error je místnost automaticky ukončena a nečeká se na opětovné připojení.

Na začátku hry se inicializují karty pro každého hráče, než se přejde k samotné herní smyčce. V herní smyčce se pak postupně zkontrolují opětovná připojení

```
checkReconnect(game *Game),
```

zda nejsou hráči neaktivní

```
checkResponsive(game Game),
```

zda není potřeba doplnit balíček pro lízání karet

```
checkDecks(game *Game),
```

zda-li hra už nemá výherce

```
checkWinner(game *Game)
```

a obsloužení požadavku od uživatele

```
handleRequest(player *Player, game *Game, index int, data string,
lobby *[]net.Conn)
```

.

3.1.4 structures.go

Obsahuje pomocné struktury využívaná na serveru, které jsou Player, který slouží jako obalová třída na klientské připojení ve hře

```
    type Player struct {
conn      net.Conn
playerId  int
cardCount int
cards     []Card
active    bool
    },
```

Game, která nese informaci o aktuální hře

```
    type Game struct {
        players            []Player
        currentPlayer      *Player
        currentPlayerIndex int
        topCard            Card
        throwAwayDeck      []Card
        gameDeck           []Card
        currentCardType    string
    },
```

Card, která nese informaci o dané kartě

```
    type Card struct {
cardType    string
cardValue   string
hasBeenUsed bool
    },
```

Reconnect, která slouží ke znovu připojení hráče do hry

```
    type Reconnect struct {
request string
conn    net.Conn
    }
```

3.1.5 constants.go

Obsahuje konstanty využívané na serveru a funkci pro inicializaci herního balíčku karet.

3.2 Klient

3.2.1 Využité technologie

Klient je psán v jazyce Python za využití herního engine pygame a vytvoření spustitelného souboru pomocí pyinstaller. Pro síťovou komunikaci je využita standardní knihovna. Aplikace je rozdělena na část síťovou, modul connection, a část aplikační, modul game. Každý modul je pak členěn na komponenty a samotný výkonný kód.

3.2.2 Modul connection

Dodržuje strukturu popsanou výše, v části s komponentami se nacházejí:

- Třída `Player`, která reprezentuje aktuálního klienta a drží si informace o uživatelském jménu, id hráče, držených kartách a aktivitě hráče
- Třída `Opponent`, která reprezentuje jednoho protihráče a drží si informace o id hráče, počtu držených karet a aktivitě hráče
- Třída `Card`, která reprezentuje herní kartu a drží si informace o typu a hodnotě karty
- Třída `GameState`, která reprezentuje aktuální stav hry a obsahuje aktuálního hráče na tahu, pole všech hráčů ve hře, aktuální kartu na vrchu a aktuálně hraný druh karet
- Třída `GameInfo`, která sdružuje aktuální karty držené aktuálním klientem a aktuální stav hry

Výkonný kód pro komunikaci se serverem se pak nachází v souboru `connection.py`, kde je každý typ requestu rozdělen do vlastní metody.

3.2.3 Modul game

Obsahuje komponenty členěné na základní komponenty (karta, text, ...), které se nacházejí v souboru `components.py` a jednotlivé složené herní pohledy (úvod, herní místnost, ...), které se nacházejí každý ve vlastním souboru, celý výkonný kód hry se pak nachází v souboru `game.py` s hlavní herní smyčkou v metodě `game_loop`.

4 Uživatelská příručka

Klient se sestaví z přiloženého makefile pomocí zavolání příkazu make v rootu repositáře, to samé pak platí i pro server.

Spuštění serveru má možnost volby portu a ip a je následovné: `ups.exe <ip> <port>`, pokud nejsou parametry zadané, tak se server spustí s portem 3333.

Spuštění klienta má možnost volby adresy serveru a uživatelského jména také pomocí parametrů a je následovné: `ups.exe <adresa, formát ip:port> <username>`, pokud nejsou zadané, tak se adresa a uživatelské jméno volí až později pomocí gui klienta.

5 Závěr

Téma, které jsem si zvolila mě bavilo jak na straně serveru, tak ale především na straně klientské aplikace, která nabízela spoustu možností jak kreativně reprezentovat informace ze serveru. Myslím si, že by práce šla i dále zdokonalovat už například zvolením lepších technologií pro klientskou aplikaci. Práce mi přinesla spoustu zkušeností i díky tomu, že jsem si mohla vyzkoušet psaní vlastního serveru a navržení protokolu, tak aby se mohl dále používat.