
Mravenci
Projektová dokumentace

12. dubna 2021

1 Zadání

Naším úkolem bylo replikovat kultovní hru **Mravenci**. Cílem bylo, aby hru bylo možné hrát ve dvou lidech po síti přes webové uživatelské rozhraní. Dále bylo potřeba nabídnout API pro možnost hry bez uživatelského rozhraní (např. pro použití v kombinaci se strojovým učením) a ukládat odehrané hry a nabídnout možnost získat tato data (např. pro trénovaní neuronové sítě).

2 Pravidla hry

Cílem hry je zničit hrad protihráče anebo postavit svůj hrad o výšce 100 a více jednotek. Každý hráč má 3 typy surovin, které může používat - cihly, zbraně a krystaly. Všechny suroviny hráči narůstají vždy na začátku tahu podle toho, kolik má pracovníků v dané oblasti (stavitelů, vojáků a mágů). Hráč má v ruce 8 karet a každé kolo bud' zahráje jednu kartu anebo některou zahodí. Zahraná / zahozená karta je na konci tahu nahrazena novou z balíčku.

3 API

Klienti se serverem komunikují pomocí jednoduchého protokolu, jehož zprávy si posílají pomocí websoketů. Zprávy jsou ve formátu JSON. Každá zpráva obsahuje tři položky: kód zprávy, obsah zprávy a ID hry. Významy kódů zpráv jsou uvedeny v tabulce 1.

kód zprávy	typ zprávy
0	create_game
1	join_game
2	game_id
3	state
4	action
5	end
6	throw_away
7	material_update

Tabulka 1: Příklady použití jednotlivých typů zpráv jsou na obrázku 1.

ID hry se nachází ve všech typech zpráv kromě zprávy *create_game*, kde je ID hry *null*, protože se jedná o zprávy, kterou se o ID hry žádá.

Obsahem zpráv *create_game* a *join_game* jsou jména hráčů. Obsahem zprávy *game_id* je ID hry. Obsahem zpráv *state*, *end* a *material_update* je stav hry, ukázka obsahu

těchto zpráv je na obrázku 2. Rozdíl mezi zprávou *state* a zprávou *end* je ten, že zpráva *end* ukončuje spojení skrz websocket a posílá se v okamžiku, kdy některý z hráčů zvítězil. Zpráva *material_update* se odesílá na začátku tahu hráče a obsahuje aktualizovaný stav hry obohacený o automatický přírůstek surovin. Obsahem zpráv *action* a *throw_away* je index karty hráče, kterou chce zahrát nebo odhodit.

Nyní popíšeme strukturu stavu hry zasílaného zprávami *state*, *end*, *material_update*, viz obrázek 2. Položka *game_status* obsahuje jméno hráče, který je na tahu. Položka *last_played_card* obsahuje jméno poslední karty na odhazovacím balíčku a to, zdali byla karta zahrana (*played*) nebo odhozena (*thrown_away*). Položky *player_1_status* a *player_2_status* obsahují statistiky hráčů ve stejné struktuře, která se liší jedním detailem. Seznam karet, které hráč drží v ruce, je vždy zaslán jen hráči, který je opravdu v ruce drží, seznam karet oponenta zpráva neobsahuje (posílá se prázdné pole). Seznam karet je reprezentován dvojicemi (název karty - informace, zdali je možné kartu zahrát). Kromě položky *hand* (pole karet v ruce) obsahují statistiky hráče informace o jeho jméně a o surovinách, které vlastní, viz obrázek 2.

4 Data pro analýzu a strojové učení

Díky API je možné k serveru přistupovat i jinými způsoby než jen grafickým uživatelským rozhraním. Jednak lze vytvořit boty, kteří můžou hrát, jak proti lidským hráčům, tak mezi sebou. A za druhé je možné stáhnout si data odehraných her z databáze, kam se ukládají všechny tahy a pomocí nich poté vytvářet boty.

Databáze typu SQL Lite obsahuje jednu tabulkou s názvem *state*, která obsahuje sloupce s názvy *id* (primární klíč), *game_id* (id hry), *status* (hráč, který vykonal akci), *player_1_status* (statistiky hráče 1), *player_2_status* (statistiky hráče 2), *type_of_action* (typ akce - jednalo se o zahrání karty (0) nebo odhození(1)) a *action* (jméno zahráne/odhozené karty).

Ve chvíli, kdy je spuštěný server, lze data z databáze stáhnout pomocí cesty *address : port / <from – to>*. Příkladem je adresa <http://127.0.0.1:5000/0-5>, která stáhne nultý až pátý stav z databáze a vrátí ho uživateli v JSON struktuře.

5 Implementace

Backend aplikace jsme implementovali v jazyce Python 3. Frontend aplikace jsme implementovali v jazyce javascript pomocí JS knihovny React. Frontend a backend mezi sebou komunikují pomocí websocketů. Websockety jsme použili, protože poskytují oboustranný komunikační kanál přes jedno TCP spojení.

Jazyk Python 3 jsme pro implementaci backendu zvolili kvůli tomu, že poskytuje příjemné řešení komunikace přes websockety, implementaci mechaniky hry a napojení na databázi obsahující zahráné stavy. V backendu využíváme několik důležitých tříd. Třída *Server* zajišťuje komunikaci backendu s frontendem před websockety. V této třídě se nachází objekt *GameManager*. Jedná se o třídu, která zajišťuje logiku provádění akcí v jednotlivých hrách na základě přijatých zpráv třídou *Server*. Tedy obsahuje mimo jiné slovník objektů *Game*, což je třída starající se o akce v konkrétní hře. Tato třída pak obsahuje dva objekty hráčů, které jsou implementované pomocí třídy *PlayerStatus*. Projekt pak obsahuje ještě třídu *Message* pro obecnou zprávu.

React jsme zvolili proto, že nabízí příjemnou dekompozici pomocí znovupoužitelných komponent, je jednoduchý a přirozený pro učení se tzv. *od nuly* a umožňuje rychlé změny prostředí pomocí virtuálního DOM a stavů komponent. Uživatelské rozhraní se skládá z jednotlivých komponent pro herní statistiky obou hráčů, vizuální stav hradů a jejich obranných zdí, karet, které hráč drží v ruce anebo odhazovacího balíčku. Rozhraní reaguje na přijaté zprávy ze serveru pomocí naslouchačů (angl. *listeners*), pokaždé, když dojde nějaká zpráva, *listener* zavolá funkci která upraví rozhraní dle přijatých dat a stavu hry. Při hře udržuje hráčům pojem o situaci pár animací hraných karet, barevné odlišení karet, které může zahrát a které může odhodit, a také sada zvukových efektů z původní hry, které doprovází jednotlivé akce, které se dějí ve hře. V souboru *src/config.js* je nastaveno, na jaké adresu a portu se klientské rozhraní má připojit na server poskytující veškeré informace potřebné pro zahájení a průběh hry. Na obrázku 3 je vidět uživatelské rozhraní v počátečním stavu hry z pohledu hráče na tuhu. Uživatel v okně vždy vidí svůj stav hry na levé polovině hracího pole.

5.1 Spuštení aplikace

Backend aplikace se spouští příkazem *python server.py*. Jeho konfiguraci je možné nastavit v souboru *config.ini*. Server, který umožňuje stahování dat z databáze se spouští pomocí dvou příkazů *export FLASK_APP = database_server.py* (ve Windows je místo *export* klíčové slovo *set*) a příkazu *flask run*.

Pro frontend aplikace je potřeba nainstalovat nainstalovat npm závislosti pomocí *npm install*. V případě úspěšné instalace stačí použít příkaz *npm start* a následně je možné se pomocí webového prohlížeče připojit na *http://localhost:3000/*.

5.2 Použité knihovny

5.2.1 Backend

- json

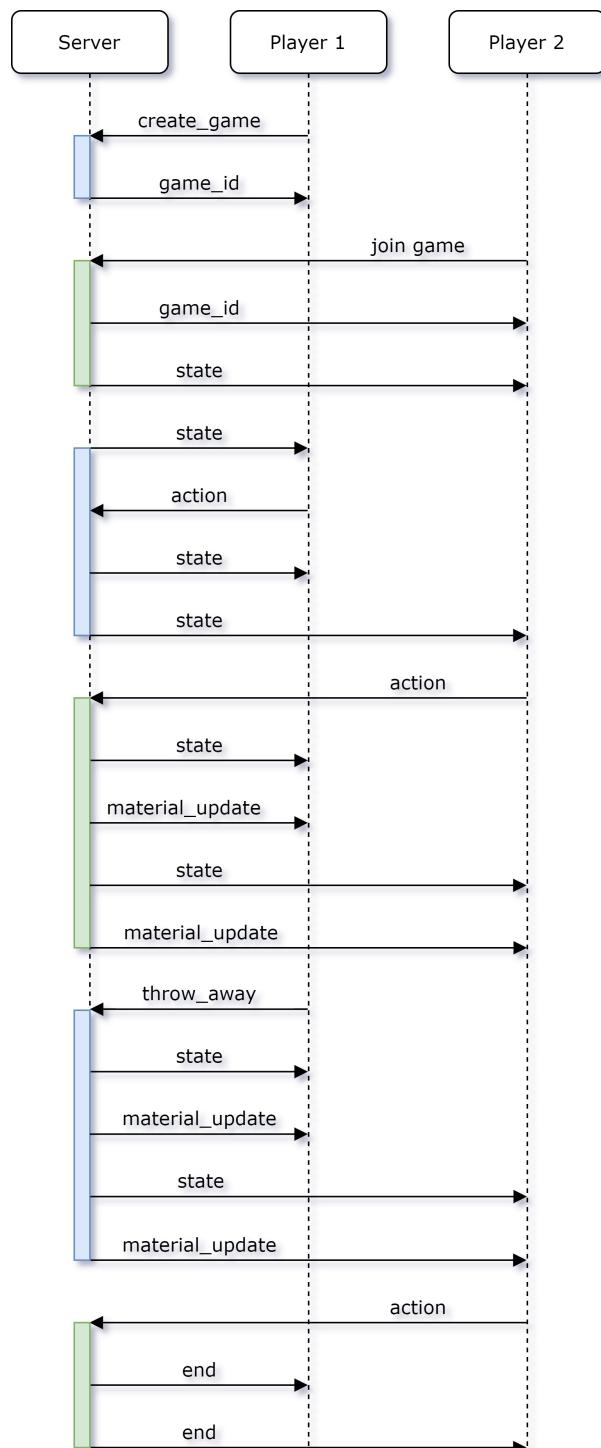
-
- enum
 - typing
 - random
 - configparser
 - flask
 - asyncio
 - websockets
 - sqlalchemy

5.2.2 Frontend

- react
- react-dom
- react-router-dom
- react-scripts

7 Závěr

Volba zvolených technologií pro vývoj této hry se nám osvědčila po všech stránkách, výhodná pro nás byla dekompozice uživatelského rozhraní pomocí komponent, práce s jejich stavů a směrování knihovny React, na backendu pak jednoduchost knihoven jazyka Python pro práci s daty v JSON formátu, tvorbu serveru a práci s databází a protokolem websocket. Nejvíce zkoumání, zkoušení, hledání a učení jsme museli investovat do práce s protokolem websocket. V rámci budoucího vývoje by bylo vhodné rozšířit hru o možnost hry jednoho hráče.



Obrázek 1: Ukázka komunikace mezi klienty a serverem. Barvy symbolizují jednotlivé procesy.

```
{  
    "type": 3,  
    "content": {  
        "game_status": "smrkev",  
        "last_played_card": [  
            "zakladny",  
            "played"  
        ],  
        "player_1_status": {  
            "name": "LachubCz",  
            "stats": {  
                "building": {  
                    "workers": 2,  
                    "material": 4  
                },  
                "army": {  
                    "workers": 2,  
                    "material": 5  
                },  
                "magic": {  
                    "workers": 2,  
                    "material": 5  
                },  
                "estate": {  
                    "castle": 32,  
                    "wall": 10  
                }  
            },  
            "hand": []  
        },  
        "player_2_status": {  
            "name": "smrkev",  
            "stats": {  
                "building": {  
                    "workers": 2,  
                    "material": 4  
                },  
                "army": {  
                    "workers": 2,  
                    "material": 5  
                },  
                "magic": {  
                    "workers": 3,  
                    "material": 1  
                },  
                "estate": {  
                    "castle": 32,  
                    "wall": 10  
                }  
            },  
            "hand": [  
                [  
                    "nabor",  
                    false  
                ],  
                [  
                    "zed",  
                    true  
                ],  
                [  
                    "znic_krystaly",  
                    false  
                ],  
                [  
                    "ceta",  
                    true  
                ],  
                [  
                    "znic_krystaly",  
                    false  
                ],  
                [  
                    "strelec",  
                    true  
                ],  
                [  
                    "skola",  
                    false  
                ],  
                [  
                    "pevnost",  
                    false  
                ]  
            ]  
        }  
    },  
    "game_id": "LachubCz7531"  
}
```

Obrázek 2: Zpráva *state* ve formátu JSON popisující aktuální stav hry.



Obrázek 3: Uživatelské rozhraní v počátečním stavu hry z pohledu hráče na tahu.