Sake

Samuel Krajči

1 Úvod

Základné pokyny pre užívateľa sa nachádzajú v `README.md` v git repozitári:

Sake

A project for programming class (year 1, term 2) at Charles University, Prague.

Sake is an arcade-like multiplayer game, that uses monogame C# library. Last snake slithering wins.

How to run the game

In order to run the game, server must be also running.

Server

To run server, you have to build and run *server* project. There are 3 commands in the *server* program:

- **lobby**: puts server in a lobby state, enabling users to connect. Max 4 players are supported.
- **game**: starts the game
- **quit**: terminates the program

Game

First you have to configure IP address of the server in the *constants* library in the *Sake* project (use `"localhost"` for localhost) To run the game, build and run *Sake* project. Then wait for the server to star the game. **Do not terminate the program until the game ends, the server will crash** (a feature, not a bug...).
Use right and left arrow to turn.

It is not recommended to publish the game yet, any sort of configuration is missing, so in order to change server IP address or game paramateres, you have to change the code

2 Viac o hre

Hra je inšpirovaná klasickou hrou *Snake* a online hrou *Curve Fever*. Každý hráč má svojho hada ktorého vie ovládať, cieľom je ostať posledný v hre. Hady sa pohybujú po štvorčekovej sieti na tóruse (vedia prechádzať cez okraje), vedia sa pohybovať iba do štyroch smerov. Hra prebieha v *kolách*, teda diskrétne, nie spojito, za sekundu sa štandardne udeje 10 kôl.V hre sa nachádzjú iba dva typy objektov. Hady a *powerupy*.

2.1 Powerupy

Hady vedia zbierať powerupy tak, že cez ne prejdú. Powerupy môžu dať hadu *vlastnosť*, alebo vykonajú *okamžitú akciu*. Powerupy sa objavujú v náhodných časoch a na náhodných miestach, každý s inou pravdepodobnosťou. V hre je 6 powerupov:

- **food** *(jablko)*: ako v klasickom *snake*, had narastie o 1 políčko.
- **mega food** *(hamburger)*: podoblné ako *food*, ale had narastie o 5 políčok.
- **invincibility** *(zlatá huba)*: spraví hada nesmrteľným na 30 kôl.
- **slow** *(slimák)*: spraví hada dva krát pomalším, hýbe sa len každé druhé kolo po dobu 40 kôl.
- **reverse** *(zrkadlo)*: obráti hada. Hlavu má tam, kde mal chvost a naopak.
- **stone** *(kameň)*: tento powerup sa nedá zobrať, je to iba prekážka, do ktorej ak had narazí, zomrie.

2.2 Smrť

Had vie zomrieť iba dvoma spôsobmi:

- nabúraním do kameňa (*stone*),
- alebo nabúraním do hociakého hada, vrátane seba.

Ak je had nesmrteľný (po vzatí powerupu *invincibility*), nevie zomrieť ani jedným z týchto spôsobov.

3 Štruktúra programu

Program pooužíva knižnicu *MonoGame*.
Hra je zložená z dvoch projektov (*server* a *klient* (*Sake*)) a štyroch knižníc (*constants*, *game library*, *protocol library* a *client library*).

3.1 Knižnice

3.1.1 Constants

Tu nájdeme všetky konštanty ako napríklad *rozmery mapy*, alebo *trvania powerupov*.

3.1.2 Game library

Obsahuje viditeľné objekty v hre:

Snake

Táto trieda predstavuje hada. Má všetky vlastnosti popisujúce hada ako *dĺžka*, *poloha*, *smer* alebo jeho stav (ktorý je ovplyvňovaný *powerupmi*). Taktiež má metódy, ktoré s ním hýbu, alebo ho vykresľujú.

Má podtriedu *SnakeUser*, ktorá obsahuje navyše iba nasledujúci smer hada.

Powerup

Jednoduchá trieda, objekt triedy predstavuje powerup, ktorý má priradený *typ*, *polohu* a *textúru*. Má iba metódu na vykreslenie.

Map

Táto trieda popisuje celú hraciu plochu (resp. *mapu*). Obsahuje *rozmery mapy*, *zoznam hadov* a *zoznam powerupov*, a metódu *updateFromMapUpdatePacket*, ktorá mení stav mapy, bližšie to popíšeme pri *klient - server komunikácii*.

Zložitejšia je ale podtrieda *MasterMap*, ktorá má (pre efektívu detekciu kolízií) navyše

- (pre efektívu detekciu kolízií) uložený obsah každého políčka na mape (v slovníku),
- metódu na *generovanie powerupov*,
- metódu na *vykreslenie*, celej mapy, zavolá iba vykreslenie všetkých objektov ktoré sa na nej nachádzajú,
- metódu *AutoUpdate*, ktorá sa volá v každom kole a *updatene stav celej mapy*,
- a metódu *createMapUpdatePacket*, ktorú bližšie popíšeme v *klient - server komunikácii*.

Program *klienta* používa popis stavu mapy triedu *Map* a *server* používa *MasterMap*, viac v *klient - server komunikácii*.

Rand

Statická trieda na generovanie náhodných čísel.

3.1.3 Client library

Tu sa nachádza iba jedna trieda *TcpClient*, ktorá obsahuje objekt striedy *Socket* z knižnice *System.Net.Sockets*. Objekt tejto triedy je v programe *klienta* a zabezpečuje komunikáciu so serverom cez *TCP protokol*. Má teda všetky potrebné metódy ako napríklad *connectToServer*, *disconnect*, *requestLoop*, ...

3.1.4 Protocol library

Tu nájdeme štyri triedy paketov, ktoré sa v komunikácii používajú. Každá trieda má niekoľko parametrov a navyše má aj jeden parameter typu string (*serialized*) v ktorom sú parametre objektu serializované a teda sú pripravené byť *payload* v *TCP pakete*.

- **PowerupInfo** - popisuje stav powerupu,
- **SnakeInfo** - popisuje stav hada,
- **MapInfo** - popisuje zmenu stavu mapy,
- **InitialInfo** popisuje počiatočný stav mapy.

Každá trieda má dva typy konštruktorov - *zo serializovaného stringu* a *klasický, explicitný* a teda vieme jednoducho informácie serializovať alebo rozbaliť.

3.2 Klient

Klient si sa skladá vpodstate iba z troch objektov - *TcpClient*, *Map* a *SnakeUser*. O vykresľovanie sa stará knižnica *MonoGame*.
Na klientovi bežia dva procesy paralelne - *Update*, ktorý vykresľuje mapu a paralelne *RunTaskAfterResponseLoopAsync*, funkcia objektu *TcpClient*, ktorá sa stará o komunikáciu so serverom.

3.3 Server

Server sa skladá vpodstate iba z dvoch objektov - *MasterMap* ktorý popisuje stav celej hry a *Socket*, objekt knižnice *System.Net.Sockets*, ktorý odosiela pakety klientom.

3.4 Klient - server komunikácia

Komunikácia prebieha cez *TCP protokol*. *Server* má všetky inforácie o hre. Keď je server v stave *lobby*, klienti sa vedia pripojiť. Po prejdení do stavu *game* server rozpošle paket popisujúci začiatočný stav hry (objekt triedy *InitialInfo*). Potom začne nasledovný loop:

Fáza	**Server**	**Klient**
1	pýta sa všetkých klientov na zmenu smeru (`"requesting move"`)	čaká na výzvu od serveru
2	čaká na odpovede od všetkých klientov	odpovedá serveru zmenou smeru (`"l"` / `"r"` / `"f"`)
3	updatne mapu a pošle zmenu stavu (objekt *MapInfo*, ktorý vráti metóda *createMapUpdatePacket* objektu *MasterMap*)	čaká na update mapy
4	čaká kým ubehne dĺžka jedného kola	updatne stav metódou *updateFromMapUpdatePacket* objektu *Map*

Server hru ukonči, keď ostanú menej ako dva hady nažive správou `"game over"` a následne ukonči všetky spojenia.