

|  |
| --- |
| **Hra „Človeče nehnevaj sa!“** |
| SEMESTRÁLNA PRÁCA |

Vypracovali: **Adrián Ihring, Viliam Lapčák**

Študijná skupina: **5ZYR33**

Predmet: **Princípy operačných systémov**

Cvičiaci: *Ing. Peter Sedláček, PhD.*

**OBSAH**

[Úvod 3](#_Toc124104565)

[Používateľská príručka 4](#_Toc124104566)

[Základné pravidlá hry 4](#_Toc124104567)

[Spúšťanie hry 4](#_Toc124104568)

[Ovládanie hry 4](#_Toc124104569)

[Programátorská príručka 5](#_Toc124104570)

[Štruktúra projektu 5](#_Toc124104571)

[Trieda Kocka 5](#_Toc124104572)

[Trieda Figúrka 5](#_Toc124104573)

[Trieda Hráč 5](#_Toc124104574)

[Trieda Hracia plocha 5](#_Toc124104575)

[Trieda Manažér 5](#_Toc124104576)

[Makefile 6](#_Toc124104577)

[Popis synchronizačného problému 6](#_Toc124104578)

[Využitie vlákien a soketov 6](#_Toc124104579)

Úvod

Ako tému našej semestrálnej práce na predmete Princípy operačných systémov sme si vybrali hru „Človeče nehnevaj sa!“ naprogramovanú v jazyku C++. Hru je možné hrať na serveri. V jednotlivých kapitolách v dokumentácií vysvetľujeme základné pravidlá hry a jej samotnú logiku, účinkovanie na serveri, popis využitia soketov, vlákien a synchronizačného problému.

Používateľská príručka

Používateľská príručka obsahuje základné pravidlá hry, ovládanie hry a jednotlivých častí aplikácie.

Základné pravidlá hry

Hra „Človeče nehnevaj sa!“ je určená pre 2-4 hráčov, pričom na začiatku hry sa čaká dokým sa na server nepripoja všetci štyria hráči, následne hra môže začať. Hráči sa v priebehu hry môžu odpojiť zo serveru, pričom hra bude ďalej pokračovať aj bez ich prítomnosti.

Cieľom hry je dostať všetky štyri figúrky zo základne do domčeka. Po hodení šestky môže figúrka odštartovať. Hráčove figúrky musia prejsť dookola cez všetky políčka na hracej ploche, bez toho, aby ich súper nevyhodil. Počet prejdených polí sa určí hodom očíslovanej kocky. Vyhráva ten hráč, kto ako prvý dostane všetky svoje figúrky do domčeka.

Spúšťanie hry

Pre korektné spustenie hry je pre server potrebné zadať číslo portu a názov servera (napr. *12345 server*). Na klientskej časti je potrebné zadať hostname, číslo portu a názov klienta. (napr. *localhost 12345 klient*). Pre spúšťanie hry na Linuxe je nutné spustiť ./server a ./klient a do parametrov zadať to isté ako bolo spomenuté vyššie (napr. *./klient localhost 12345 klient*).

Ovládanie hry

Po spustení samotnej hry sa vykreslí hracia plocha, na ktorej sa nachádzajú 4 rôzne farby figúrok. Každý hráč má svoje vlastné 4 figúrky rovnakej farby. Farby figúrok sú reprezentované znakom (modrá *‘M’*, zelená *‘Z’*, červená *‘C’*, biela *‘B’*). Index figúrok je reprezentovaný číslom 1 až 4. Napríklad druhá modrá figúrka je vykresľovaná na hracej ploche ako *‘(M2)’*.

Na začiatku hry sa vypíše informácia o hráčovi, ktorý je aktuálne na ťahu. Následne musí hráč hodiť kockou (napísaním do konzoly slovo *“hod”*) a vybrať figúrku, s ktorou sa chce pohnúť o daný počet políčok (napísaním do konzoly číslo 1 až 4). V prípade, že hráč napíše zlý vstup, (napr. nesprávne hodí kockou, vyberie neexistujúcu figúrku, alebo zvolí figúrku, s ktorou nie je možné ťah vykonať) hra upozorní hráča vypísaním vhodného výpisu a hráč musí tak opakovať ťah dovtedy pokiaľ nie je vykonaný. Môže nastať situácia, že hráč po hodení kockou sa nemôže pohnúť so žiadnou figúrkou. V takomto prípade je hráč informovaný o nemožnosti ťahu a na ťahu je ďalší hráč.

Na jednotlivých políčkach sa v jednom okamihu môže nachádzať práve len jedna figúrka. Ak sa má hráčova figúrka pohnúť na políčko na ktorom sa nachádza už jeho iná figúrka, ťah sa nevykoná a hráč si tak vyberie inú figúrku, s ktorou sa pohne. V prípade, že sa na očakávanom políčku nachádza súperova figúrka, hráč sa so svojou figúrkou pohne na zvolenú pozíciu, vyhodí súperovu figúrku a vráti ju do základne.

Ak hráč nemá na hracej ploche žiadnu zo svojich figúrok, hádže dovtedy dokým nehodí 6, maximálne však 3-krát (potom nasleduje ďalší hráč). Ak má hráč na hracej ploche aspoň jednu figúrku hádže už iba 1-krát. Ak padne šestka, hráč môže opakovať hod a posunúť sa o súčet padnutých hodov na kocke.

Po tom, ako hráč vykoná ťah, vykreslí sa hracia plocha a vypíše sa informácia o ďalšom hráčovi, ktorý je na rade. Následne sa celá logika opakuje dovtedy pokiaľ nejaký z hráčov nevyhrá. Ak sa tak stane, vypíše sa informácia o výhercovi a hra končí.

Programátorská príručka

Štruktúra projektu

Projekt je naprogramovaný v jazyku C++. Pre vrátenie sa k jednotlivým verziám projektu sme využili Git a zdrojové kódy sme uverejnili na GitHub. V projekte sa nachádza 7 súborov typu .h a 9 súborov typu .cpp.

Trieda Kocka

Trieda kocky obsahuje jedinú funkciu, ktorá slúži pre generovanie náhodného čísla hodeného na kocke.

Trieda Figúrka

Trieda figúrky pozostáva z viacerých funkcií, slúži prevažne na nastavenie pozície, farby, identifikačného čísla figúrky. Ďalej obsahuje funkcie pre posun figúrky, odpočítavania krokov a samotné gettre o počte krokov, ktoré figúrka prešla na hracej ploche, identifikačnom čísle figúrky, a pozícii, kde sa figúrka nachádza.

Trieda Hráč

Trieda hráč reprezentuje hráča ako takého. Hráč vykonáva ťahy jednotlivými figúrkami. Pri vykonaní samotného ťahu, hráč hodí kockou, vyberie figúrku s ktorou sa chce pohnúť, skontroluje, či sa dá s danou figúrkou pohnúť, overuje ich pozície a následne ťah vykoná pokiaľ je to možné.

Pri vytváraní samotného hráča sa mu inicializuje jeho farba, počet figúrok, počet figúrok v domčeku a samotné figúrky.

Trieda Hracia plocha

Táto trieda je určená pre vytvorenie hracej plochy, možnosti jej nastavenia a vykreslenia. Pri vytvorení sa inicializuje základná plocha, nastavia sa štartovacie pozície pre jednotlivé farby hráčov a do základne sa nastavia jednotlivé hráčske figúrky.

Trieda Manažér

Manažérska trieda slúži na vytvorenie a obsluhovanie samotnej hry. Pozostáva z hracej plochy a hráčov ktorý hrajú danú hru. Manažér vypisuje informáciu o hráčovi na ťahu, informuje o skončení hry, kontroluje figúrky pri vyhadzovaní, nastavuje vyhodené figúrky na základňu.

Makefile

Postupne sme si vytvorili aj Makefile, ktorý sme priebežne aktualizovali. Makefile sme vytvárali nasledovne. Využili sme kompilátor *g++*, vzhľadom na použitie *c++* v projekte. Tento kompilátor, skompiloval *.cpp* súbory do *.o* súborov, aby ich nebolo možné spustiť samostatne. Tieto objektové súbory *(.o)* sme použili následne v spustiteľných súboroch, ako je klient, alebo server (bez koncovky). V *all:* sme zavolali server a klienta a v *clean:* vieme cez príkaz *"make clean"* vymazať všetky *.o* súbory plus server a klienta. Pomocou Makefile sme tak mohli otestovať našu hru aj na školskom linux serveri (frios2.fri.uniza.sk).

Popis synchronizačného problému

Analógia problému je založená na 4 hráčoch, ktorý vykonávajú ťahy na hracej ploche, pričom každý z hráčov musí čakať na poradie, kedy môže vykonať ťah. Pri každom ťahu figúrkou musí hráč napísať najmenej 2 vstupy, jeden pri hode kockou a druhý pri výbere figúrky, s ktorou sa chcel daný hráč pohnúť. Po každom ťahu bolo potrebné poslať štruktúru dát pre vykonanie ťahu nasledujúcemu hráčovi. To bolo nutné preto, aby nasledujúci hráč mohol vykonať ťah na tej istej hracej ploche ako predošlý hráč. Po každom vykonanom ťahu je nutné zmeniť hráča na ťahu.

Využitie vlákien a soketov

Sokety sme používali na komunikáciu medzi serverom a klientom. Konkrétne sme ich využívali pri zapisovaní a čítaní na serveri aj klientovi. Cez sokety sme si posielali štruktúru pre zdieľané dáta obsahujúcu inicializáciu mutexu, a podmienok pre server a klienta. Funguje to tak, že po tom ako server vykoná ťah, server nastaví dáta, pošle ich klientovi, ktorý si aktualizuje dáta, vykoná ťah a následne nastaví dáta a pošle ich naspäť serveru. Server si dáta aktualizuje a komunikácia medzi serverom a klientom pokračuje.

Pre server a klienta sme priradili vlákno pre vykonanie funkcie *„void\* hrajServer(void\* args)“* a „*void\* hrajKlient(void\* args)*“. Funkcia pre vlákno serveru čakala na vykonanie ťahu klientom, a potom ako klient vykonal ťah, mohol vykonať ťah aj samotný server. Pri funkcií klienta to fungovalo rovnako, ale na opačnom princípe. Vlákno muselo čakať na vykonanie ťahu na serveri, a potom ako server vykonal ťah, mohol vykonať ťah aj klient.