Nu te supăra frate (B) C+S

Perju Mircea-Ștefan (2A4)

Universitatea Alexandru Ioan Cuza, Facultatea de Informatică, Iași, 700483

Abstract. În acest document se prezintă în mare parte ideile și resursele utilizate în realizarea proiectului. Proiectul constă în jocul "Nu te supăra frate" și este implementat cu ajutorul unui server TCP concurent. Proiectul este compus atât din componenta de server, cât și cea de client. O instanță a serverului permite un singur joc între maxim 4 jucători. Pentru a începe jocul trebuie să se conecteze cei 4 jucători, însă dacă se deconectează vreunul între timp va fi eliminat din joc. În document sunt prezentate regulile, funcționalitățile cât și anumite îmbunătățiri posibile. Logica este realizată în server, fiind trimise comenzi de către client și verificate mai apoi ca fiind sau nu valide, primind un răspuns aferent. Componenta client dispune de o interfață grafică atractivă. Comunicarea este continuă pe tot parcursul jocului până când clientul închide fereastra, caz în care este deconectat. Pentru a reține pozițiile pionilor fiecărui jucător se utilizează o matrice de dimensiune 11x11. Deoarece jocul este turn-based, fiecare jucător trebuie să își respecte rândul, iar dacă sunt comenzi trimise când nu este cazul, ele nu vor fi efectuate. În document se prezintă mai pe larg aceste aspecte. Biblioteca grafică utilizată este SFML, iar link-ul către documentația acesteia a fost atașat în bibliografie. Restul surselor bibliografice constă în cursurile universității la materiile "Rețele de calculatoare" și "Sisteme de operare".

Keywords: SFML · TCP concurent · Sincronizare.

1 Introducere

Proiectul "Nu te supăra frate" reprezintă un joc multiplayer care poate fi jucat în maximum 4 persoane. Este un turn-based board game în care clienții (jucătorii) își așteaptă rândul și vor da cu zarul, iar în funcție de numărul nimerit și de stateul jocului vor putea scoate(din casă)/muta pionii lor. Câștigă primul jucător care reușește să își ducă toți pionii la finish sau ultimul jucător rămas în joc (în cazul în care toti ceilalți se deconectează).

Fiecare jucător poate da cu zarul o singură dată când este rândul său. Nu se pot afla doi pioni pe aceeași poziție pe ecran simultan. Dacă pionul unui jucător se mută pe poziția unui pion al altui jucător, respectivul pion care se afla deja pe poziția respectivă este scos și se întoarce în punctul de start (în casă). Pentru a scoate pionii din casă jucătorul trebuie să dea 6 cu zarul. În cazul în care mutarea unui pion al unui jucător ar duce la suprapunerea a doi pioni ai săi, mutarea este invalidă și nu se va executa (datorită regulii 2). Mutările valide sunt cele în care jucătorul a dat 6 cu zarul si are un pion pe care dorește să-l

scoată din casă atunci când căsuța de pe care pornește nu este ocupata de alt pion de al sau sau cele în care jucătorul se deplasează pe tabla cu n poziții (n fiind numarul dat cu zarul) atunci când pe poziția pe care nimerește nu este deja alt pion de al sau. De asemenea există un număr maxim de mutări (mai exact 44 pentru a ajunge de la poziția de start până la ultima căsuță posibilă).

Interfața grafică este user-friendly și este adaptată pentru clienți în funcție de player id-ul lor. Asta înseamnă că mereu culoarea lor se va afla în colț stânga jos.

În momentul în care un client se deconectează piesele sale sunt eliminate de pe tabla de joc, iar acesta nu se mai poate reconecta în meciul curent, indiferent că jocul a început sau nu (astfel, acest lucru se aplică și în perioada în care se așteaptă conectarea jucătorilor pentru a inițializa variabila turn). Cu toate că jocul este inițializat doar când se conectează 4 persoane, acesta poate fi jucat chiar și în 2 sau 3, acest lucru fiind posibil prin instanțierea a una/două instanțe de client care vor fi închise pentru a ramâne doar două/trei deschise dacă utilizatorii doresc asta.

2 Tehnologii Aplicate

Pentru a realiza conexiunea client-server aplicația se folosește de protocolul TCP concurent. Acest fapt se datorează comunicării continue între clienți și server. Aplicația servește clienții în ordinea în care aceștia trimit request-urile, nu în ordinea în care s-au conectat. Cu toate acestea, executarea comenzilor de către server se efectuează doar în cazul în care clientul își respectă rândul. Fiecare client comunică constant cu un copil al serverului (fork()). Am ales să utilizez TCP concurent deoarece oferă un avantaj de siguranță a livrării datelor între client și server comparativ cu protocolul UDP, iar varianta iterativa nu ar putea satisface cerința. În server utilizez variabile mapate în memorie pentru a sincroniza datele între procesele copil (fork) care sunt trimise mai departe spre clienți, astfel mereu când este efectuată vreo modificare aceasta va fi transmisă tuturor. Pentru a rezolva problema sectiunii critice utilizez semafoare.

3 Structura Aplicației

Pentru interfața grafică utilizez biblioteca open-source SFML (C++) care oferă suport și sistemelor UNIX, cât și celor care utilizează Windows. Biblioteca este beginner-friendly și utilizează clase pentru a trata funcțiile pentru respective obiecte precum cercuri, text, imagini etc. Site-ul developerilor SFML oferă detalii și îndrumare atât în instalare, cât și în utilizare, fiind prezentate toate clasele, respectiv funcțiile pentru acestea cu exemple.

Pentru sincronizarea datelor între clienți utilizez un semafor și variabile mapate în memorie. Pentru a rezolva acest task am revizuit problema secțiunii critice. Ideea inițială a fost să utilizez pipe-uri, însă întâmpinând dificultăți, am ajuns la o soluție care este în opinia mea mai ușoara și mai stabilă, astfel am decis să rămân la aceasta.

Atunci când se deschide un client acesta încearcă să se conecteze la server, iar în funcție de starea serverului conectarea va fi acceptată sau va eșua. Dacă sunt deja 4 jucători care s-au conectat în sesiunea curentă serverul returnează clientului o eroare în care este menționat acest fapt, astfel conexiunea are succes și legătura între server și clientul conectat se va realiza printr-un copil al procesului server, utilizând fork().

Jocul ia sfârșit atunci când a câstigat un jucător sau atunci când s-au a mai rămas maxim un jucător în joc. Se va trimite deschide un popup care să anunțe victoria jucătorului către toti clienții care sunt încă conectați,

Comunicarea între server și client constă în requesturi, iar logică este abordată de server după care clientului îi sunt trimise înapoi datele necesare pentru a putea efectua transformările necesare în cazul în care este rândul lui și are mutări valide.

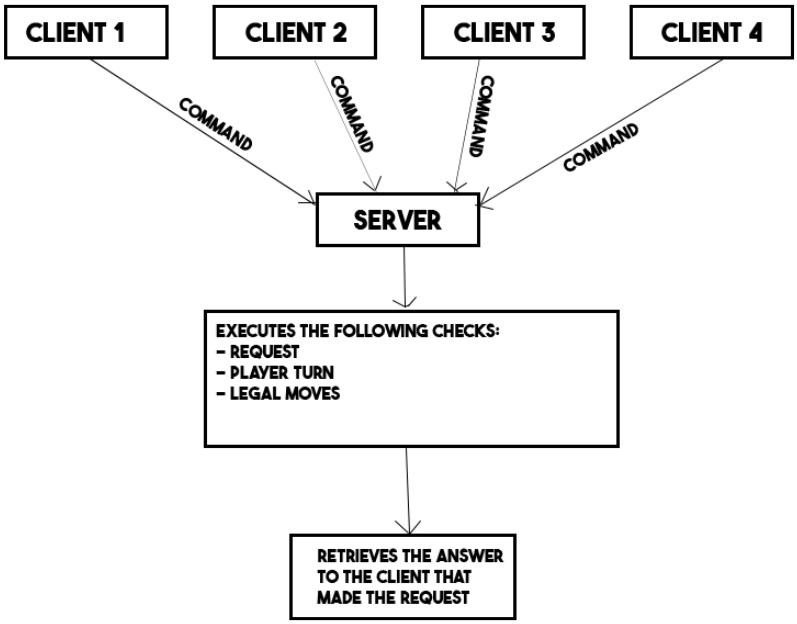
Tabla este reprezentată de o matrice 11x11, care este rotită în funcție de jucator pentru a putea păstra miscările la fel. De fiecare dată când este efectuată o operație pe matrice aceasta este rotită pentru a favoriza respectiva operație în funcție de id-ul jucătorului, fiind mai apoi rotită în poziția inițială.

Request-urile trimise din clienți sunt trimise în funcție de acțiunile acestuia (unde se efectuează click-urile), astfel mai întâi se verifică în client dacă click-urile sunt valide (dacă se selectează pionii proprii sau zarul, ordinea în care se dau click-urile). În cazul în care un jucător încearcă să efectueze o acțiune cât timp nu este rândul său aceasta nu se va efectua. Dacă este rândul jucătorului curent acesta trebuie sâ înceapă prin a da cu zarul ca mai apoi sa fie în funcție de rezultat (dacă rezultatul oferă opțiuni valide) să efectueze o mutare (ambele cazuri aduc o stare blocantă până jucătorul efectueaza acțiunea corectă).

În server sunt tratate atât modificările efectuate matricei cât și schimbarea rândului jucătorului după ce o mutare a fost efectuată cu succes (sau daca nu există mutări posibile). Rândul nu poate fi actualizat pe un client neconectat, acest caz fiind tratat separat, astfel dacă jucătorul 3 se deconectează (ca exemplu), turn va avea valorile din mulțimea 1, 2, 4.

Aplicația dispune de anumite posibile îmbunătățiri pe care le voi discuta la concluzii.

Pe pagina următoare este atașată diagrama proiectului în care se observă structura aplicației și logica utilizată:



4 Aspecte De Implementare

Aplicația funcționează pe baza unui server TCP concurent, iar comunicarea este continuă. Serverul acceptă conexiunile clienților si crează un copil pentru a menține comunicarea cu clientului respectiv în mod continuu. Astfel pentru 4 clienți vor fi 4 copii ai serverului. Utilizez fork(), iar sincronizarea între copii se efectuează cu mapare persistentă în memorie cu un semafor care asigură că nu modifică mai multi copii simultan datele sincronizate.

Aplicația este utilizată în scop de entertainment, fiind un joc. Serverul inițializează matricea care reprezintă tabla la începutul execuției, iar aceasta se va actualiza pe parcurs în funcție de mutările efectuate.

Funcționalități implementate:

1. Zar care constă în generarea unei valori la întamplare între 1 și 6. Fiecare client o să poată să dea cu zarul o singura dată când este rândul său iar în funcție de numărul de pe zar se pot efectua anumite acțiuni.

2. Actualizarea graficii în raport cu matricea ce constituie tabla. De asemenea adaptarea matricei în raport cu clientul specific (rotirea ei pentru fiecare P1 - 0°, P2 - 90°, P3 - 180°, P4 - 270° counter-clockwise).

3. Adăugarea de butoane și informații (warning-uri) pe partea grafica a clientului si a unui pop-up care anunță victoria unui jucător și în urma acestuia jocul este închis.

4. Verificare a condițiilor de sfârsit al jocului. Verificare de condiții la mutare. La cazul general verificat dacă ce dorește clientul să facă este conform regulilor.

5. Comunicarea client-server a datelor multiple precum player id-ul, rândul jucătorilor si datele din matrice.

6. Interfață grafică atractivă și tratarea eventurilor.

7. Captura de pioni ai inamicilor. În cazul în care jucătorul efectuează o mutare pe o căsuța unde alt jucător are un pion, acesta va trimite pionul inamicului înapoi în casă, celălalt jucător fiind nevoit să dea din nou 6 pentru a-l scoate.

8. Timer pentru warning-uri (2.5s) - anumite probleme, se resetează atunci când este trimis un alt warning, însă pentru a se actualiza interfața trebuie ca aceasta să primească un update. Acest lucru nu ține de timer, ci de abordarea eventurilor și voi încerca să rezolv în viitor.

5 Concluzii

Potențialele îmbunătătiri principale ale aplicației ar fi: gestionarea multiplelor gameroom-uri în loc de permiterea unui singur joc, detectarea mai usoară a deconectărilor și anumitor erori, tratarea cazurilor de eroare a conexiunii, eficientizarea în condiții de lag (posibile buguri), îmbunătățirea warning-urilor (lack of consistency), adăugarea unui timer care să restricționeze timpul de mutare a unui jucător, posibilitatea de reconectare a unui jucător.

font utilizat - DayPosterBlack

compilare client : g++ client.cpp -lsfml-graphics -lsfml-window -lsfml-system -o client (necesită instalarea librăriei în prealabil)

References

1. https://www.andreis.ro/teaching/computer-networks
2. https://profs.info.uaic.ro/ computernetworks/index.php
3. https://profs.info.uaic.ro/ vidrascu/SO.html
4. https://www.sfml-dev.org/learn.php