Online Chess

22.12.2021

Îndrumător:

dr. ing. Daniel Morariu

Student:

Mihăilescu Anduțu-Petronel

224/2

Istoric Versiuni

Data	Versiune	Descriere	Autor
17 noiembrie 2021	1.1	Generarea tablei de joc, adăugarea unui event handler pentru fiecare user control	Ștefan Precup
28 noiembrie 2021	1.2	Finalizarea claselor	
30 noiembrie 2021	1.7	Jocul este funcțional	
1 decembrie 2021	2.3	Adăugarea componentei multiplayer	Modelul de pe classroom
14 decembrie 2021	3.0	Adăugare șah mat	
15 decembrie 2021	Beta 1.4	Adăugare promovare pioni	
17 decembrie 2021	Beta 1.6	Optimizări	

Cuprins

1 Specificarea cerințelor software

1.1 Introducere

Aplicația prezentată reprezintă un joc clasic de șah, multiplayer. Piesele pot fi mutate după regulile originale, iar câștigătorul este primul care reușește să dea șah mat oponentului.

1.1.1 Obiective

Primul obiectiv pentru realizarea acestei aplicații a fost desenarea tablei de joc și a pieselor. Al doilea obiectiv a fost crearea claselor. Următorul obiectiv a fost posibilitatea de mutare a pieselor și abilitatea de a lua piesele adversarului. De asemenea, fiind un joc multiplayer, unul dintre obiective a fost implementarea componentei de rețea. Ca jocul să fie complet, am ales să adaug și mutările speciale, precum promovarea pionilor, rocada, En Passant. În final, ca jocul să se poată finaliza, am ales să implementez și posibilitatea de a da șah mat oponentului.

1.1.2 Definiții, Acronime și Abrevieri

Backup Move – este diferită de Possible Move; Reprezintă o poziție unde nu poate fi mutat regele pentru a nu intra în șah.

1.1.3 Tehnologiile utilizate

Pentru realizarea proiectului, a fost folosit limbajul de programare C#, iar ca mediu de programare, a fost folosit Visual Studio. Piesele de șah utilizate au fost descărcate de pe un site de game assets. Documentația proiectului a fost realizată în Microsoft Word. Pentru generarea tablei, am folosit un User Control realizat de mine.

1.2 Cerințe specifice

- Generarea tablei de joc și poziționarea pieselor
- Mutarea pieselor pe tablă și abilitatea de a lua piesele adversarului.
- Implementarea componentei de rețea (trimiterea unui mesaj codificat spre cealaltă aplicație, decodificarea acestui mesaj și executarea instrucțiunilor din mesaj)
- Mutare specială: Promovarea pionilor (când un pion ajunge la capătul opus al tablei de joc, utilizatorul are opțiunea de a schimba acel pion cu orice altă piesă, în afară de rege)
- Posibilitatea de șah mat (atunci când regele se află în șah, nu are mutări disponibile și nici nu poate fi apărat)

2 Mutarea pieselor pe tablă și abilitatea de a lua piesele adversarului

2.1 Descriere

Mutarea pieselor de șah pe tablă, precum și abilitatea de a lua piesele adversarului, reprezintă baza acestui joc. Ambii jucători trebuie să mute când le vine rândul, iar dacă o piesă este mutată pe poziția piesei adversarului, acesta ia piesa piesa adversarului, eliminând-o, astfel, din joc.

2.2 Fluxul de evenimente

2.2.1 Fluxul de bază

”Funcția main” a acestui program este reprezentată de metoda PieceClick, care este un event handler. Atunci când utilizatorul dă click pe o căsuță, se apelează această metodă. Ca programul să recunoască ce căsuță a fost apăsată, metoda utilizează parametrul sender. Am ales să folosesc o variabilă de tip PieceUC care reține o referință a căsuței apăsate.

Metoda este împărțită în 3 zone: Zona de validare a mutării, zona de verificare a componenței căsuței, zona de verificare dacă utilizatorul poate selecta piese.

Culoarea care începe întotdeauna prima este culoarea alb. Dacă este rândul utilizatorului, iar piesa selectată îi aparține, se colorează pe tablă toate mutările posibile. Pentru acest lucru, ne folosim de o listă de mutări temporare care reține mutările posibile ale piesei selectate. Dacă nu este rândul utilizatorului, sau piesa selectată nu îi aparține, lista de mutări temporare este resetată, la fel și culorile tablei de joc.

După ce a fost selectată o piesa potrivită la momentul potrivit, utilizatorul trebuie să realizeze o mutare. Dacă a fost selectată căsuța ce aparține unei mutări posibile, se va realiza acea mutare. În caz contrar, lista temporară de mutări posibile se resetează, la fel și culorile tablei. Pentru mutări, folosim metoda MovePiece. În cazul în care pe căsuța unde dorim să mutăm piesa noastră se află o piesă ce aparține adversarului, îi luăm piesa, eliminând-o astfel din joc.

Pașii enumerați mai sus sunt realizați doar pentru culoarea alba de la aplicația de client și pentru culoarea neagră de la aplicația de server. Pentru culorile negru, de la client, respective alb, de la server, sunt realizate doar mutările. Condițiile nu mai sunt verificate, deoarece au fost deja verificate în cadru celeilalte aplicații. Mutările sunt realizate utilizând metoda ExecuteInstructions.

2.2.2 Fluxuri alternative

2.2.2.1 Primul flux alternativ

Pașii enumerați mai sus sunt realizați doar pentru culoarea alba de la aplicația de client și pentru culoarea neagră de la aplicația de server. Pentru culorile negru, de la client, respective alb, de la server, sunt realizate doar mutările. Condițiile nu mai sunt verificate, deoarece au fost deja verificate în cadru celeilalte aplicații. Mutările sunt realizate utilizând metoda ExecuteInstructions, care decodifică o variabilă de tip String, codificată în zona de validare a metodei PieceClick. Variabilele clientInstructions, respective serverInstructions, conțin 2 coordonate ale piesei care trebuie mutate, 2 cordonate ale locației unde trebuie mutată și alte câteva valori (Mai multe detalii la capitolul 3 și 4).

2.2.2.2 Al doilea flux alternativ

Un alt mod de a utiliza această funcționalitate este atunci când verificăm dacă regele poate fi apărat (Mai multe detalii la capitolul 4).

2.2.3 Pre-condiții

Ca și pre-condiții, utilizatorul trebuie doar să se conecteze la aplicația server și să respecte regulile acestui joc.

2.2.4 Post-condiții

Utilizatorul va avea experiența obișnuită a unui joc de șah multiplayer.

3 Implementarea componentei de rețea

3.1 Descriere

Această funcționalitate este esențială pentru a putea juca acest joc între două calculatoare diferite. Când unul dintre utilizatori va face o mutare, acea mutare se va sincroniza și pe celălalt calculator și vice versa.

3.2 Fluxul de evenimente

3.2.1 Fluxul de bază

Pentru a stabili o conexiune între cele 2 aplicații, am utilizat, pentru aplicația client metodele - Asculta_client, SendInstructions și BtnConect_Click, iar pentru aplicația Server - Asculta_Server și SendInstructions.

Pentru a conecta aplicația client la aplicația server, utilizatorul trebuie să introduce adresa IP a server-ului, iar apoi să apese pe butonul de conectare, cel cu simbolul de antenă. În acel moment, aplicația apelează metoda BtnConect_Click, care se conectează la server utilizând thread-ul 8098.

După acest pas, meciul poate începe. Pe parcursul rulării aplicațiilor, metodele Asculta_client, respective Asculta_Server, așteaptă input din partea celeilalte aplicații. Input-ul este transmis de ambele aplicații utilizând metodele SendInstructions. Acestea trimit o variabilă de tip String care conține un set de instrucțiuni. Când este apelată metoda SendInstructions din aplicația Client, este executat automat și codul din metoda Asculta_Server. Același principiu se aplică și pentru aplicația server.

Odată ce a fost trimisă o instrucțiune către cealaltă aplicație, este executată metoda ExecuteInstructions, care decodifică variabila de tip String și imită acțiunile celeilalte aplicații. Este utilizată variabila MethodInvoker deoarece nu putem avea simultan un thread pentru rețea și un thread pentru user interface. În cazul în care am fi adăugat, spre exemplu, linia de cod: clientForm.textBox1.Add(“ceva”); - am fi avut o excepție.

3.2.2 Pre-condiţii

Utilizatorul client trebuie să introducă o adresă IP validă, înainte de conectare, altfel programul va genera o excepție. De asemenea, ca jocul să ruleze corect, clientul trebuie să fie rămână la server. În cazul în care clientul se deconectează, programul va genera o excepție.

3.2.3 Post-condiţii

Utilizatorul va vedea piesele adversarului cum se deplasează singure pe tablă, având abilitatea de a duce jocul până la capăt. (Șah Mat)

4 Posibilitatea de șah mat

4.1 Descriere

Un meci de șah se termină într-una dintre următoarele situații posibile: șah mat sau remiză. Șah mat este atunci când unul dintre regi este atacat de o altă piesă, nu are unde să "fugă" și nici nu poate fi apărat de o piesă de aceeași culoare. Această funcționalitate a fost implementată prin metoda TestForCheckMate, care apelează câteva alte metode pentru a realiza mutări ipotetice și verificări dacă regele se află în șah.

4.2 Fluxul de evenimente

4.2.1 Fluxul de bază

Această funcționalitate a fost concepută să se verifice după fiecare mutare. După fiecare mutare pe care o realizează utilizatorul, se calculează toate mutările posibile dar și backup moves pentru fiecare piesă de pe tablă, apoi se verifică dacă regele se află în șah (se verifică atât mutările posibile ale adversarului cât și backup moves). După aceea se apelează metoda TestForCheckMate, care are 2 ramuri, pentru fiecare rege în parte.

În cazul în care regele se află în poziție de șah și nu are mutări posibile, se verifică dacă poate fi apărat de vreuna dintre piesele de aceeași culoare. Pentru acest lucru, se realizează ipotetic (lista de piese se resetează la fiecare ciclu), pentru fiecare piesă de aceeași culoare, fiecare mutare posibilă, urmând să se verifice dacă regele încă se află în șah. Dacă încă se află în șah, înseamnă că regele nu poate fi apărat de nicio piesă de aceeași culoare, deci regele se află în situația de șah mat. În caz contrar, rezultă că regele poate fi apărat, fiind nevoie ca utilizatorul să găsească singur una dintre mutările care îl apără pe rege.

În caz de șah mat, tabla se dezactivează pentru ambele aplicații (nu mai putem face click pe piese).

4.2.2 Fluxuri alternative

4.2.2.1 Primul flux alternativ

Verificare dacă regele se află în șah mat se realizează doar în cadrul aplicației unde se face mutarea. Cealaltă aplicație decide starea acestei situații prin decodificarea variabilei de tip String în cadrul metodei ExecuteInstructions. Penultima valoarea reprezintă dacă unul dintre regi se află în șah mat (1) sau nu se află în șah mat (0). Ultima valoare reprezintă care rege se află în șah mat (regele alb – 1, regele negru – 0).

4.2.3 Pre-condiții

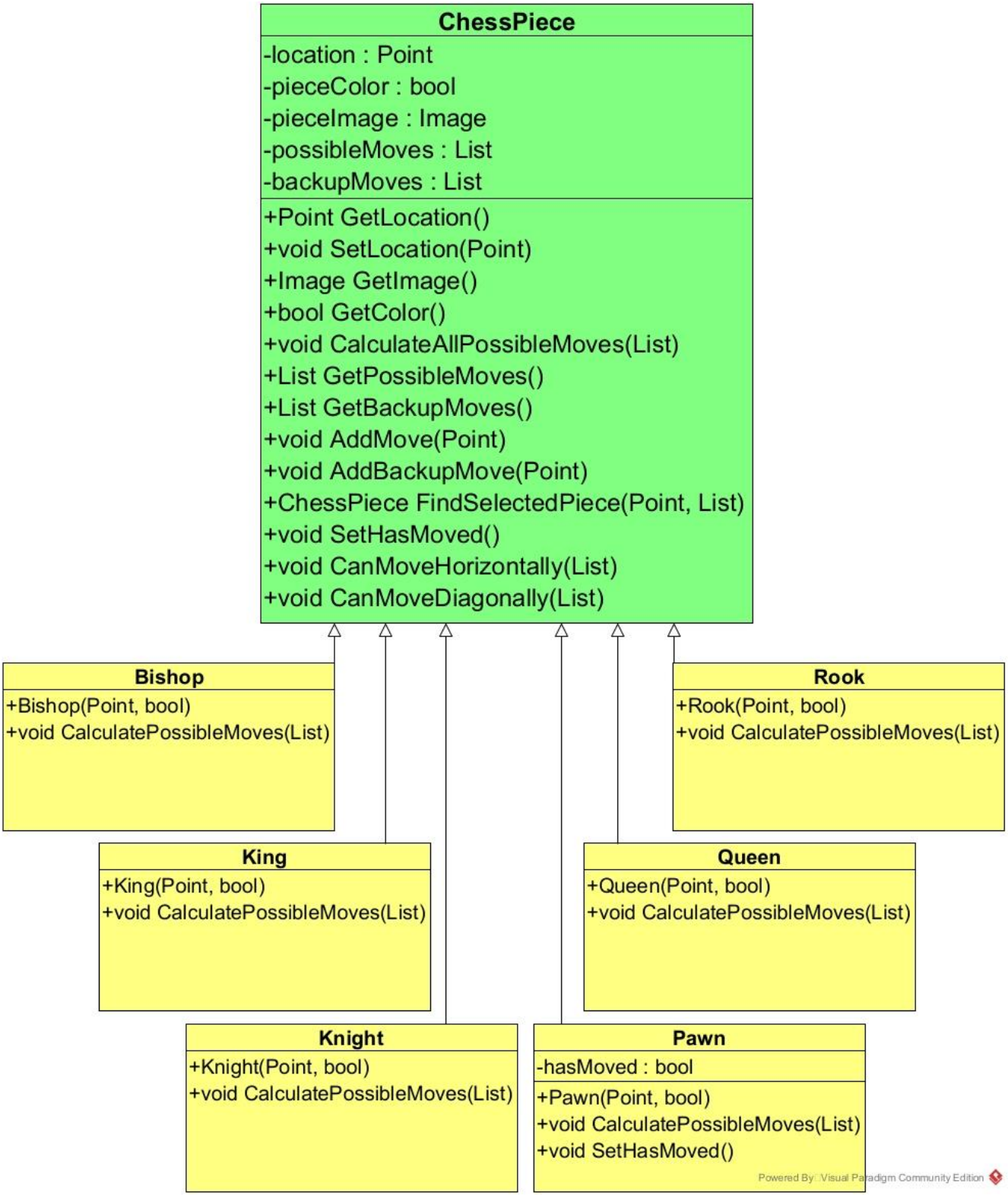
Pentru a porni corect această funcționalitate, utilizatorul trebuie decât să respecte regulile jocului de șah.

4.2.4 Post-condiții

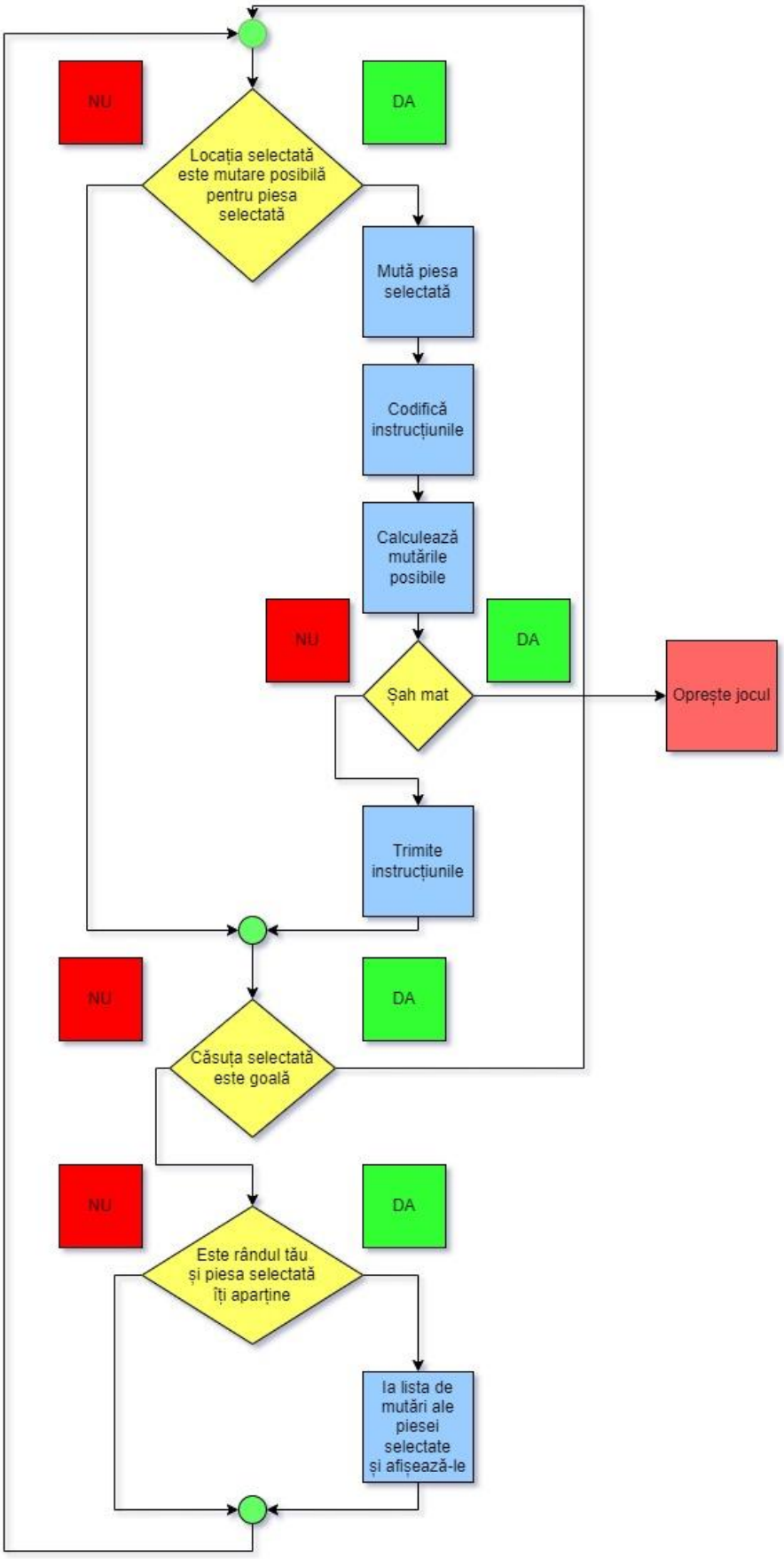
În caz de șah mat, utilizatorul va putea vedea în consolă mesajul: White/Black is in Check Mate.

5 Implementare

5.1 Diagrama de clase



5.2 Descriere detaliată



6 Bibliografie

1. Programare C# - https://www.geeksforgeeks.org/csharp-programming-language/

2. Rezolvare problem - https://stackoverflow.com

- https://docs.microsoft.com/en-us/dotnet/csharp/

3. Reguli șah - https://www.chess.com/learn-how-to-play-chess

4. Game assets - https://gamesupply.itch.io/basic-chess-asset