Șah atomic cu inteligență artificială

Proiect

Inteligență artificială

Student: Giubega Larisa-Nicoleta

Semigrupa: 232/2

Sesiunea: iunie-iulie 2021

Cuprins:

1. Descrierea jocului ales. Obiective
2. Diagrama claselor
3. Structura și logica aplicației
4. Bibliografie
5. Descrierea jocului ales. Obiective

Proiectul prezent propune o implementare în C# a unei variante a jocului de șah, anume jocul de șah atomic. Introdus publicului in 2000, acest joc este asemănător jocului clasic de șah, singura diferență constând în ce se întâmplă după ce o piesă a fost capturată. În această variantă a jocului, după captura piesei are loc o “explozie” care înconjoară piesa la distanță de un pătrat, iar această explozie “ucide” toate piesele în raza de acțiune a exploziei cu excepția pionilor. Considerăm că fiecare captură este sinucigașă, deoarece inclusiv piesa care face captura va exploda.

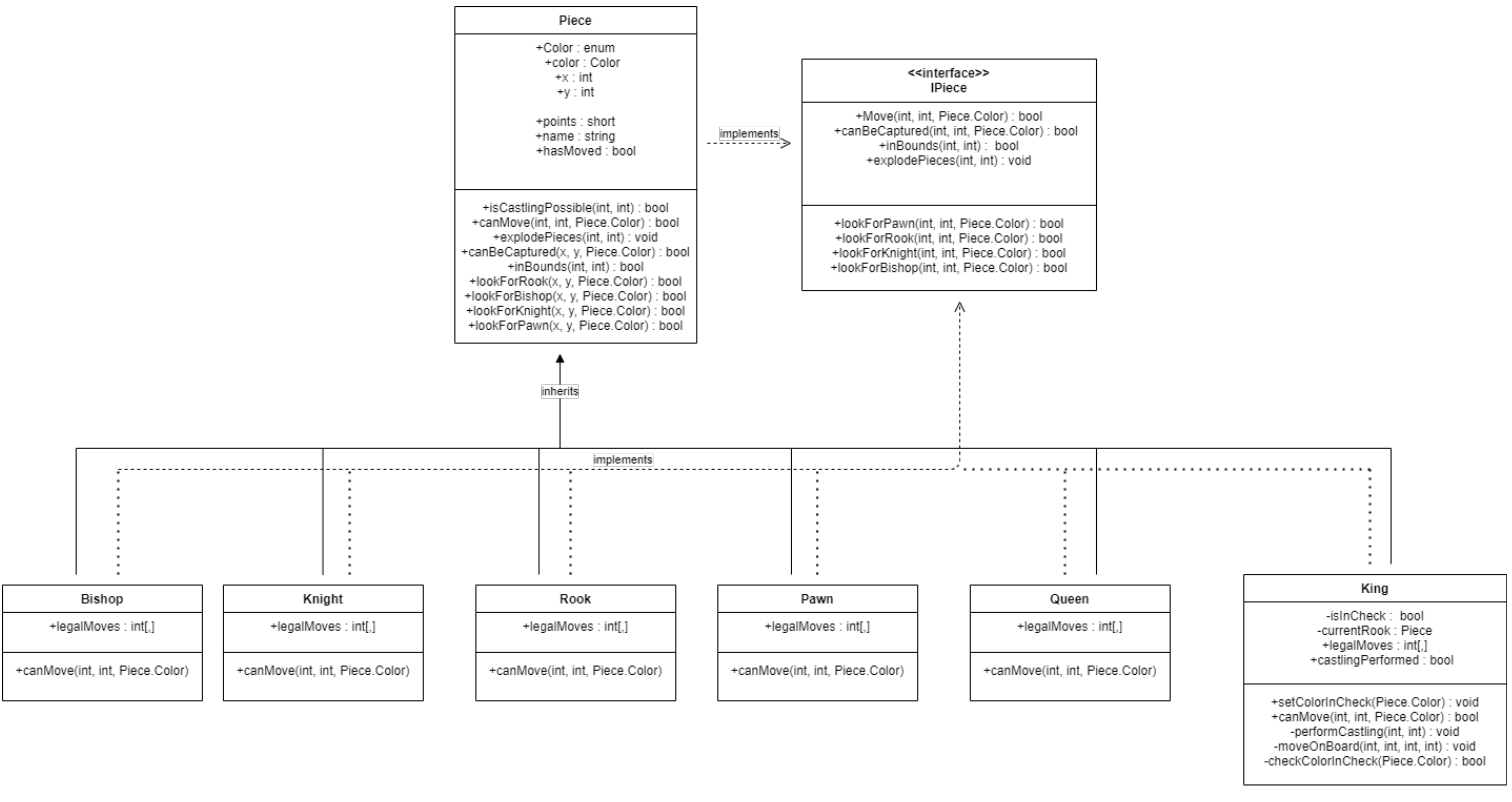
Deoarece această modificare în structura jocului are loc, conceptul de șah-mat presupune că unul dintre regi este explodat. Acest lucru are loc în urma unei explozii în jurul regelui la distanță de maxim un pătrat, indiferent dacă explozia este rezultată în urma unei capturi a unei piese a lui sau a adversarului. Din acest motiv, regele nu are voie să captureze nicio piesă, întrucât explozia ar rezulta în șah-mat pentru adversar.

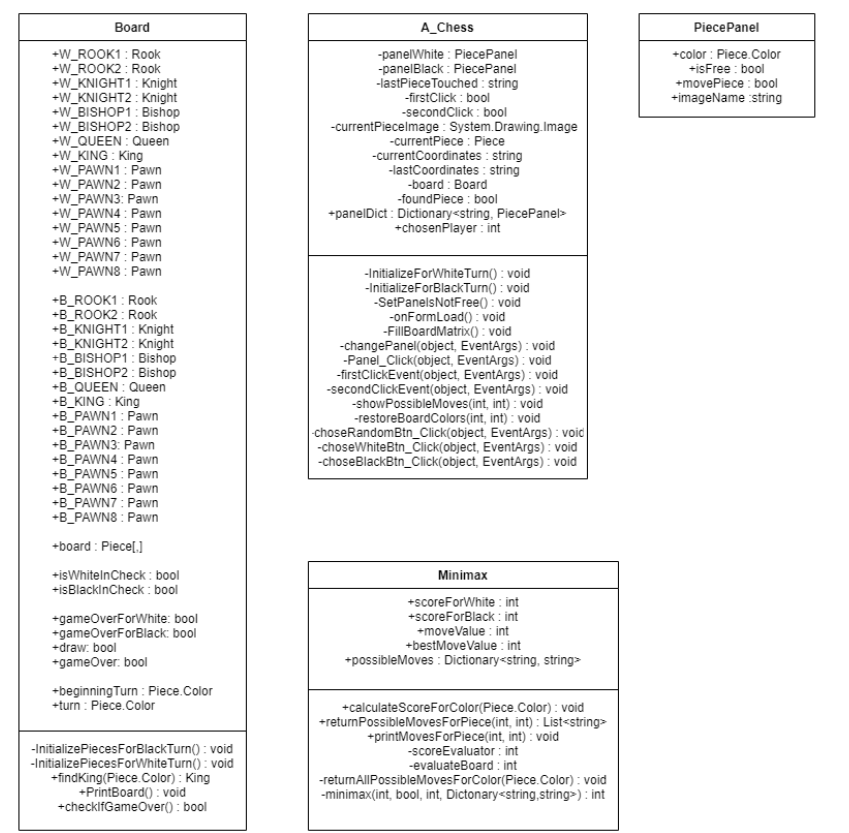
Există reguli speciale și pentru pioni: captura lor nu are loc prin explozii, ci prin captura efectivă a lor, asemenea jocului de șah clasic.

Patul are loc dacă nu există amenințări împotriva unui rege și nu există alte mutări permise.

Obiectivele pe care dorim să le îndeplinim sunt de a implementa integral aceasta variantă de șah, regulile sale, o interfață grafică ușor de întrebuințat de către utilizator și metoda de minimax

1. Diagrama claselor





1. Structura și logica aplicației

Interfața IPiece este o interfață care conține metodele principale ale pieselor:

-canMove – metodă care verifică dacă piesa de culoarea primită ca parametru poate muta în pozițiile newX, newY

-canBeCaptured – metodă care verifică dacă piesa de culoarea color și de la coordonatele x, y poate fi capturată

-inBounds – metodă care verifică dacă piesa de la coordonatele i, j se află incă pe tablă (dacă i și j sunt coordonate valide din tablă)

-explodePieces – metodă care “explodează” piesele la distanță de 1 pătrat de piesa de la coordonatele x și y primite ca parametru, inclusiv piesa

-lookForPawn, lookForRook, lookForBishop, lookForKnight – metode care caută dacă există pion, tură, nebun, respectiv cal pe traiectoria piesei de culoare color și coordonate x și y (notă: nu a fost nevoie de implementarea metodei lookForQueen, deoarece aceasta se poate considera o combinație între lookForRook și lookForBishop)

Clasa Piece este o clasă de bază pentru clasele Pawn, Rook, Knight, Bishop, Queen și King. Ea conține informații generale despre piesă, precum culoarea, coordonatele pe tabla de șah, numărul de puncte pe care îl are piesa, numele piesei și o variabilă hasMoved care reține dacă piesa a fost mutată (un fel de flag util ulterior în cazul rocadei). Aceasta clasă implementează metodele din interfață, exceptând metodele canMove și isCastlingPossible, acestea fiind implementate în alte clase.

Clasele Pawn, Rook, Knight, Bishop și Queen sunt asemănătoare, ele moștenesc clasa Piece și implementează metoda canMove diferit în funcție de tipul piesei. Clasa King se comportă asemenea celor anterior menționate, cu diferența că se va adăuga verificarea de rocadă și mutarea se va face diferit pentru aceasta. Clasa King implementează de asemenea și setter și getter pentru proprietatea isInCheck specifică jucătorului.

Clasa Board conține toate detaliile despre tablă și simulează tabla de șah printr-o matrice care conține instanțele pieselor. Aceasta se creează în funcție de jucătorul care începe.

Clasa PiecePanel va fi utilă pentru crearea tablei. Vom considera fiecare panel de culoare albă sau maro, o proprietate isFree pentru a seta disponibilitatea/ indisponibilitatea unui panel (în funcție dacă este ocupat de o piesă sau nu), un bool movePiece care setează/resetează capacitatea piesei de a se muta (utilă ulterior la deselectarea piesei pe care vrem să o mutăm) și numele imaginii (a piesei de pe panel).

Clasa A\_Chess este clasa principală a proiectului. Aici se implementează un GUI dinamic pe paneluri de culori alb și maro cu ajutorul unui dicționar de chei string și valori PiecePanel. La început se alege jucătorul dorit (alb, negru sau random) pe care îl trimitem ca parametru în constructorul clasei Board, iar apoi se încarcă tabla în mod dinamic în funcție de acesta. Fiecărui panel astfel creat îi atribuim un event de click unde în cazul în care nu s-a sfârșit jocul se face discuție dacă este primul sau al doilea click pe panel (dacă a fost selectată o piesă sau pentru a o muta). La primul click se preia piesa de pe panelul pe care s-a făcut clickul, se arată posibilele mutări și se face o discuție pentru cazul în care un player este în șah pentru a afișa un mesaj sugestiv. Se pregătesc apoi variabilele pentru al doilea click, unde se va muta piesa selectată în anumite condiții dacă panelul pe care s-a dat click este diferit de primul, în caz contrar, se va deselecta piesa pentru a se alege alta. Se verifică după mutarea piesei dacă jocul s-a încheiat, iar în caz afirmativ, se va afișa un mesaj corespunzător.

Clasa Minimax a fost creată pentru logica algoritmului cu același nume, însă nu s-a reușit implementarea. Aceasta conține un evaluator de scor în funcție de playerul care începe, o metodă de evaluare a scorului tablei pentru fiecare mutare, o metodă ce returnează potențialele mutări ale unei piese (sub forma unui dicționar coordonate-piesa <> coordonate-potențiale) de care ne putem folosi ulterior și o metodă care conține o implementare incompletă a algoritmului minimax.

4.Bibliografie

<https://www.chess.com/terms/atomic-chess>

<https://www.chess.com/terms/draw-chess>

<https://www.geeksforgeeks.org/check-if-any-king-is-unsafe-on-the-chessboard-or-not/>

https://jsfiddle.net/k96eoq0q/1/

https://jsfiddle.net/Laa0p1mh/3/

http://theoryofprogramming.com/2017/12/24/minimax-algorithm-with-alpha-beta-pruning/

<http://theoryofprogramming.azurewebsites.net/2017/12/12/minimax-algorithm/>

https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-4-alpha-beta-pruning/

<https://app.diagrams.net/>

Cursul de AI în cadrul universității