Proiect

Inteligență Artificială

Șah – Varianta Los Alamos

Student:
Herțoiu Bogdan-George
Anul 3
Grupa: 232/3
Specializarea: Calculatoare

Cuprins

1. Introducere

Programul conține o tablă de joc cu piese în care pot fi mutate în interiorul tablei și care se pot lua.

Cerința proiectului a fost crearea unei aplicație de șah folosind inteligență artificială

Proiectul conține clase de piese cu metode care contribuie la mutarea pieselor, librării care ajută ca jocul să fie dynamic, algoritmul Mini Max pentru inteligență artificială.

1.1 Obiective

Obiectivul meu a fost să creez grafic tabla de șah și să generez piesele de șah. După ce componentele grafice au fost create, a trebuit să creez posibilitatea de a muta piesele și de a creea regulile ce trebuie respectate (o piesă este pusă una peste alta, calcularea coordonatelor pieselor, poziționarea corespunzătoare a pieselor în pătratul de pe tabla de șah etc.). În final, partea de inteligență artificială a trebuit implementată în tabla și piesele de șah.

Din motive tehnice, nu a fost posibil folosirea imagini cu piese de șah. Astfel, s-a recurs la înlocuirea imaginilor cu text reprezentativ piesei. De asemena, nu a fost posibil implementarea situației când este ”Șah mat”.

1.2 Tehnologii utilizate

S-a folosit Javascript pentru realizarea aplicației, librăriile: p5.dom.js, p5.js, p5.sound.js și draw.io pentru a crea diagramele de clasă

1.3 Cerințe specifice

Funcționalitățile programului:

- crearea claselor și obiectele pieselor și așezarea lor pe tablă
- mutarea pieselor pe tablă
- verificarea dacă o piesă poate lua o altă piesă, validitatea mișcării
- calcularea coordonatelor pentru a muta piesa
- implementarea inteligenței artificiale

2. Șah Los Alamos

Șah Los Alamos reprezintă o variantă de șah, tabla având dimensiunile 6 x 6, și nu conțin piesele ”nebuni” (bishops).

Regulile pentru această varianta de șah sunt:

- Pionii nu au mișcare ”în trecere” (en passant), adică pionii nu au la prima mutare opțiunea de a merge înainte cu două căsuțe
- Pionii nu pot deveni piesele Bishops
- Nu există rocadă mare sau mică

3. Funcţionalitate

Programul este compus din mai multe fișiere:

1) Index.html

```
<html>
<head>
  <meta charset="UTF-8">
  <script language="javascript" type="text/javascript"
src="libraries/p5.js"></script>
  <script language="javascript" src="libraries/p5.dom.js"></script>
  <script language="javascript" src="libraries/p5.sound.js"></script>
  <script language="javascript" type="text/javascript"
src="sketch.js"></script>
  <script language="javascript" type="text/javascript"
src="Piece.js"></script>
  <script language="javascript" type="text/javascript"
src="Board.js"></script>
  <script language="javascript" type="text/javascript"
src="MinimaxFunctions.js"></script>

</head>

<body>
</body>
</html>
```

Index.html reprezintă punctul de start al aplicației. Prin tag-urile `<script>` , se poate apela fișierul Javascript dorit:

- `p5.js` reprezintă o librărie JavaScript care face accesibil programarea din punct de vedere al design-ului și desenării grafice
- `sketch.js` este un fișier JavaScript în care este responsabil pentru desenarea grafică a tablei de joc și mișcarea pieselor de șah
- `Piece.js` este un fișier JavaScript în care există o clasă numită Piece cu atribute cum ar fi taken, isWhite, value, etc. Această clasă reprezintă o superclasă pentru celelalte clase care moștenesc proprietățile acestuia (Pawn, Queen, Knight etc.) și se definesc comportamente specifice pentru fiecare piesă
- `Board.js` reprezintă locul unde tabla și piesele vor fi implementate
- `MinimaxFunctions.js` reprezintă locul unde inteligența artificială este creată

3.1 Explicare cod JavaScript

Se vor lua în considerare următoarele funcționalități pe care le consider relevante în aplicație:

3.1.1 sketch.js

```
function setup() {
  createCanvas(800, 800);

  test = new Board();
}

function draw() {


  showGrid();
  test.show();

  runAIs();

}
```

Prin aceste două funcții setup și draw, asigură crearea tablei de joc și implementarea funcțiilor necesare.

```
function mousePressed() {
  var x = floor(mouseX / tileSize);
  var y = floor(mouseY / tileSize);
  if (!test.isDone()) {
    if (!moving) {
      movingPiece = test.getPieceAt(x, y);
      if (movingPiece != null && movingPiece.white == whitesMove) {
```

```
        movingPiece.movingThisPiece = true;
      } else {
        return;
      }
    } else {
      if (movingPiece.canMove(x, y, test)) {
        movingPiece.move(x, y, test);
        movingPiece.movingThisPiece = false;
        whitesMove = !whitesMove;
      } else {
        movingPiece.movingThisPiece = false;

      }
    }
    moving = !moving;
  }
}
```

Prin mousePressed, oferim posibilitatea de a putea mișca piesele pe tablă. Se folosesc coordonatele X și Y, rotunjind coordonata mouse-ului și a mărimii căsuței, astfel încât să obținem o piesă centrată pe căsuța respectivă. Înainte ca mutarea să fie făcută, este verificat dacă tabla:

```
test = new Board();
```

s-a terminat prin condiția `if (!test.isDone())`. Ca să nu existe conflict când se mută piese, se face verificarea `if (!moving)`. Dacă totul e ok, se poate face mutarea, verificând în același timp dacă este rândul nostru (`if (movingPiece != null && movingPiece.white == whitesMove)`)

În funcția runAI(), asigură funcționarea algoritmului Mini Max, se verifică a cui este rândul și se calculează scorul fiecărei mutări.

```
function runAIs() {
  maxDepth = tempMaxDepth;
  if (!test.isDead() && !test.hasWon()) {
    if (blackAI) {
      if (!whitesMove) {
        if (moveCounter < 0) {

          test = maxFun(test, 0);

          whitesMove = true;
          moveCounter = 10;
        } else {
          moveCounter--;
        }
      }
    }
    if (whiteAI) {
      if (whitesMove) {
        if (moveCounter < 0) {
```

```
            test = minFun(test, 0);


            whitesMove = false;
            moveCounter = 10;
          } else {
            moveCounter--;
          }
        }
      }
    }
  }
```

3.1.2 Piece.js

În acest fișier, se declară proprietățile și metodele pieselor de șah:

```
class Piece {
  constructor(x, y, isWhite, letter, pic) {
    this.matrixPosition = createVector(x, y);
    this.pixelPosition = createVector(x * tileSize + tileSize / 2, y *
      tileSize + tileSize / 2);

    this.taken = false;
    this.white = isWhite;
    this.letter = letter;
    this.pic = pic;
    this.movingThisPiece = false;
    this.value = 0;
  }
```

Funcția show() ajută la afișarea pieselor sub formă de text.

```
show() {
    if (!this.taken) {

       textSize(30);

       if(this.white){
         fill(255);
         stroke(0);
       }else{
         fill(30);
         stroke(255);
       }
       textAlign(CENTER,CENTER);

      imageMode(CENTER);
      if (this.movingThisPiece) {
        text(this.letter, mouseX,mouseY);
```

```
      }
        else {
         text(this.letter, this.pixelPosition.x,this.pixelPosition.y);
      }

    }
  }
```

Funcția withinBounds(x,y) asigură că piesele nu ies din tablă:

```
  withinBounds(x, y) {

    if (x >= 0 && y >= 0 && x < 6 && y < 6) {
      return true;
    }
    return false;

        }
```

Funcțiile move, attackingAllies, canMove, moveThroughPieces sunt responsabile pentru a asigura integritatea pieselor pe table de șah. moveThroughPieces este folosit în special pentru piesa ”Cal” pentru a putea trece peste alte piese.

După ce comportamentul general a fost creat, trebuie particularizat pentru fiecare piesă în parte astfel:

- Piesa respective moștenește clasa Piece
- Îi este asignat un nume text și valoarea sa scor
- Se generează toate mișcările posibile

3.1.3 Board.js

În Board.js, se declară tot ce a fost implementat pe tabla de șah:

```
class Board {
  constructor() {
    this.whitePieces = [];
    this.blackPieces = [];
    this.score = 0;
    this.setupPieces();

  }
```

Setarea pieselor:

```
setupPieces() {
    this.whitePieces.push(new Queen(3, 7, true));
      this.whitePieces.push(new King(2, 7, true));
    this.whitePieces.push(new Knight(1, 7, true));
    this.whitePieces.push(new Rook(0, 7, true));
      this.whitePieces.push(new Knight(4, 7, true));
      this.whitePieces.push(new Rook(5, 7, true));

    this.whitePieces.push(new Pawn(4, 6, true));
    this.whitePieces.push(new Pawn(3, 6, true));
    this.whitePieces.push(new Pawn(2, 6, true));
    this.whitePieces.push(new Pawn(5, 6, true));
    this.whitePieces.push(new Pawn(1, 6, true));
    this.whitePieces.push(new Pawn(0, 6, true));

    //black pieces
      this.blackPieces.push(new King(2, 0, false));
    this.blackPieces.push(new Queen(3, 0, false));
    this.blackPieces.push(new Knight(1, 0, false));
    this.blackPieces.push(new Rook(0, 0, false));
      this.blackPieces.push(new Knight(4, 0, false));
      this.blackPieces.push(new Rook(5, 0, false));

    this.blackPieces.push(new Pawn(4, 1, false));
    this.blackPieces.push(new Pawn(3, 1, false));
    this.blackPieces.push(new Pawn(2, 1, false));
    this.blackPieces.push(new Pawn(5, 1, false));
    this.blackPieces.push(new Pawn(1, 1, false));
    this.blackPieces.push(new Pawn(0, 1, false));


  }
```

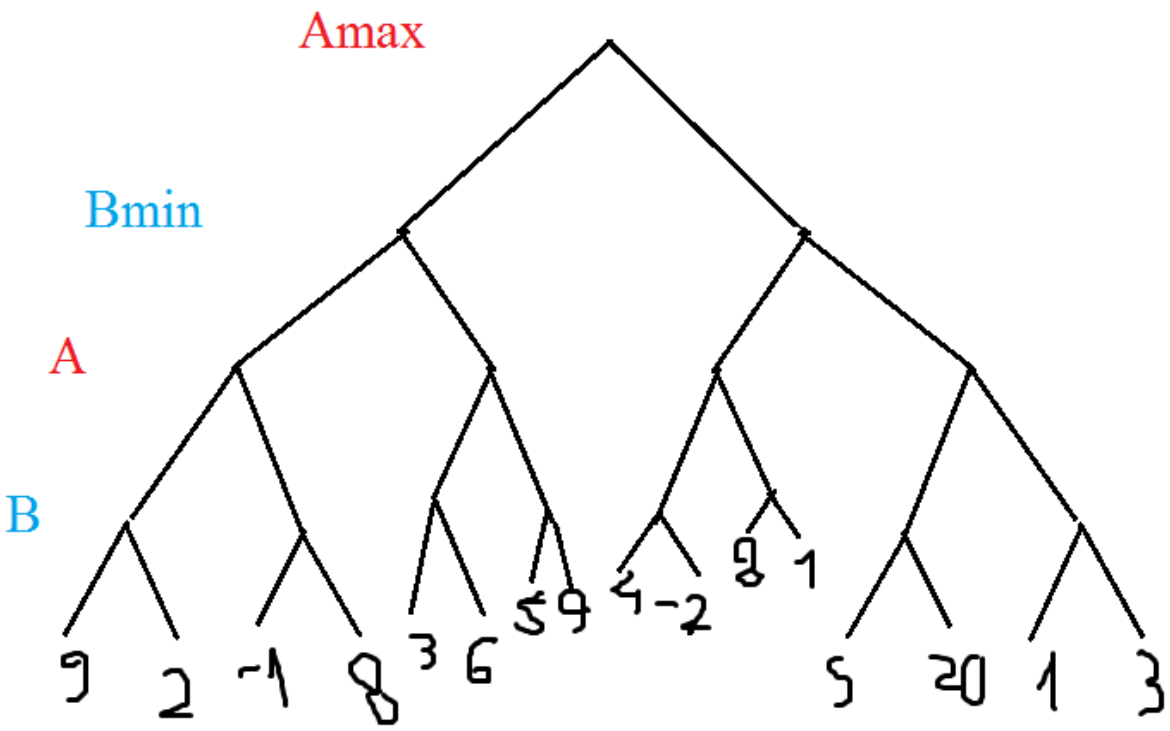
De asemenea, s-au creat metode pentru a afișa pe ecran piesele, să se verifice/să se obțină poziția pieselor, mișcarea pieselor, clonarea lor pentru a le muta dintr-o poziție în alta și funcții pentru a testa condițiile dacă s-a terminat jocul și dacă a piesa a fost luată.

3.1.4 MinimaxFunctions.js

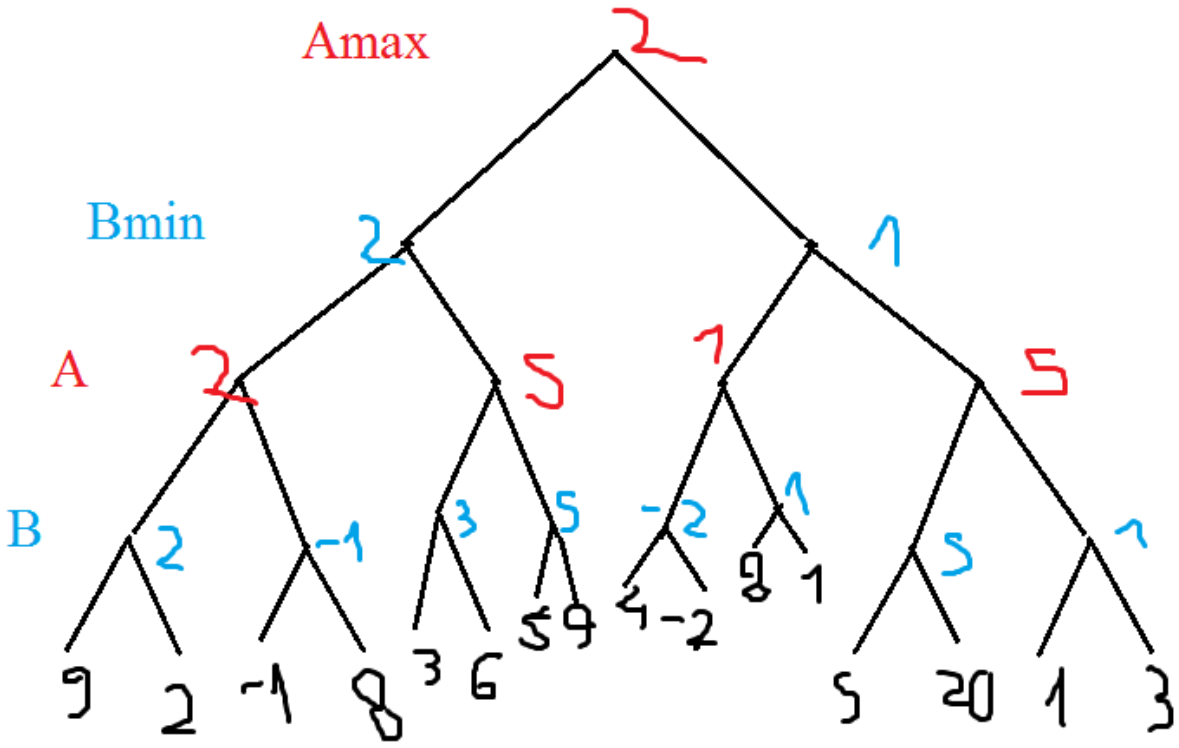
Pentru a putea analiza algoritmul, trebuie mai întâi înțeles cum funcționează algoritmul:

Algoritmul Minimax reprezintă o regulă de decizie folosită în inteligență artificială.

În cazul șahului, avem doi jucători pe care îi vom numi Amaxim și Bminim. Jucătorul Amaxim încearcă să ajungă la valorea maxim posibilă, în timp ce Bminim încearcă să ajungă la valorea minima. Fiecare jucător își alege fiecare direcția unde să meargă în arbore.



În cazul de față, la intersecția unde se află B, acesta ea valoarea 2 (fiind cel mai mic), pentru următoarea intersecție ia valorea -1 și tot așa. Pentru jucătorul A, el trebuie să ia valoarea cea mai mare, iar la prima intersecție el va lua valoarea 2 și așa mai departe. În final se ajunge la acest arbore:

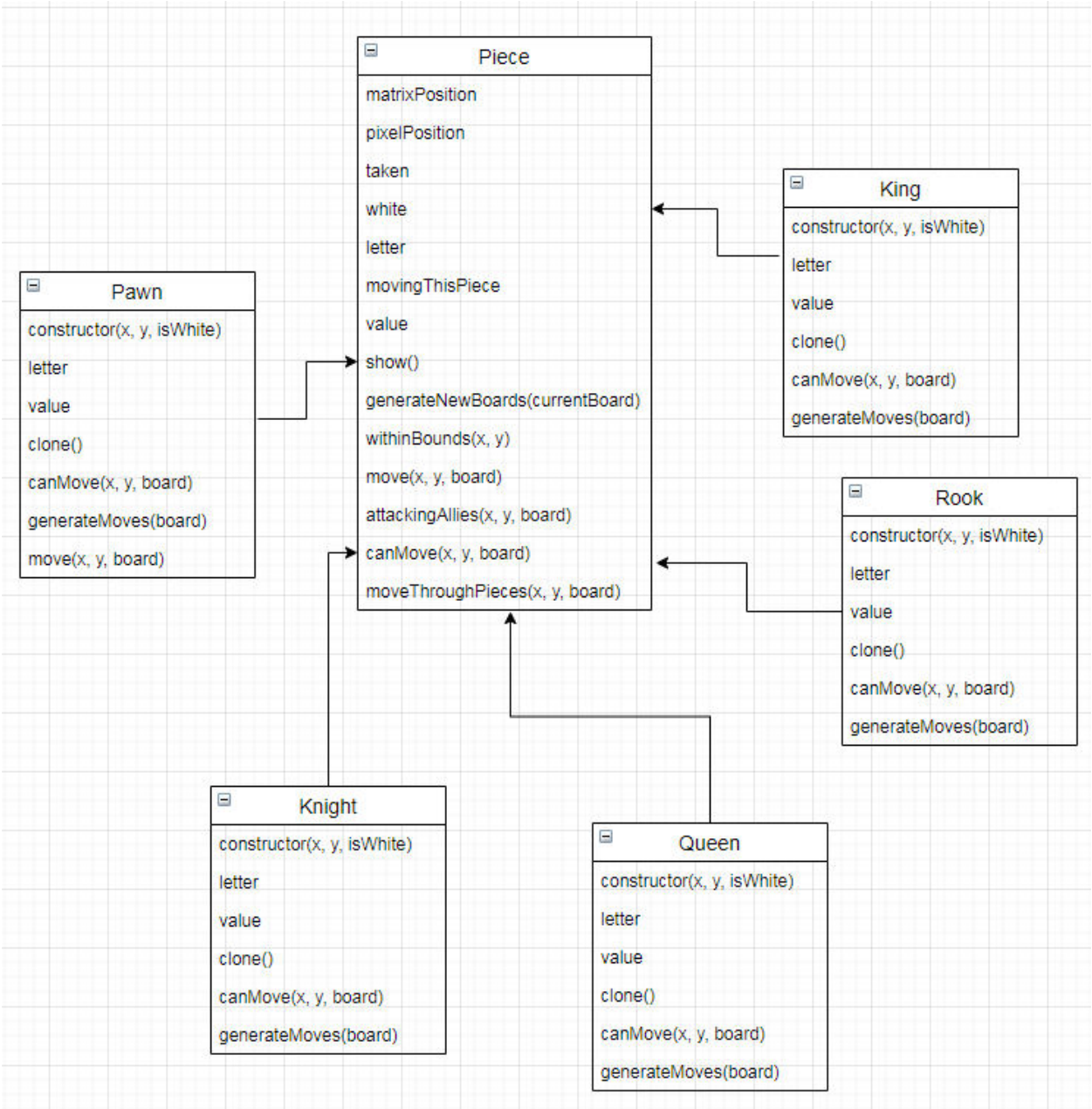


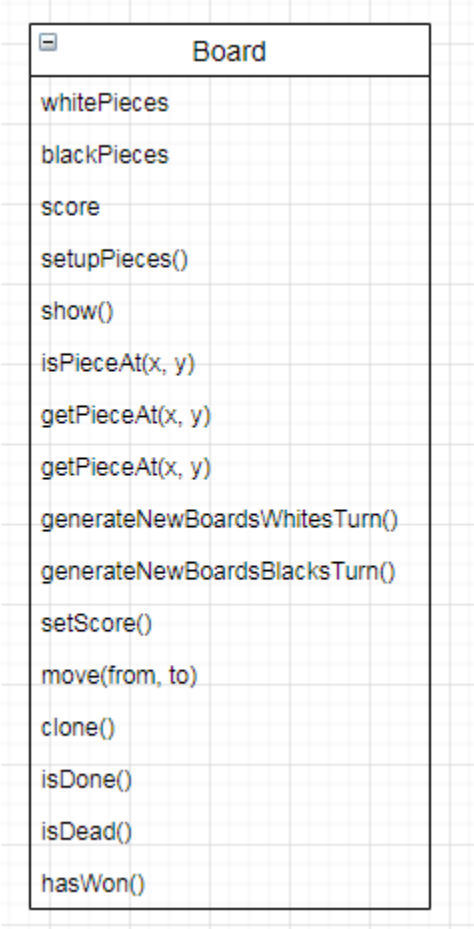
Acest algoritm este util pentru acest joc de șah, deoarece fiecare peisă de șah are propria sa valoare ierarhică. De exemplu regina este mai valoroasă decât un pion. Astfel, algoritmul încearcă să găsească decizia optima pentru a face următoarea mișcare. De aceea, fiecare piesă are propria sa valoare și există o listă de mutări posibile pentru ca algoritmul să se folosească de aceste valori, iar programul încearcă să genereze toate tablele de șah posibile pentru a identifica toate mișcările posibile și să se calculeze scorul.

```
function minFun(board, depth) {
  if (depth >= maxDepth) {
    board.setScore();
    return board.score;
  }

  var boards = board.generateNewBoardsWhitesTurn();
  var lowestBoardNo = 0;
  var lowestScore = 100000;
  for (var i = 0; i < boards.length; i++) {
    if (!boards[i].isDead()) {
      var score = maxFun(boards[i], depth + 1);
      if (score < lowestScore) {
        lowestBoardNo = i;
        lowestScore = score;
      }
    }
  }
  return lowestScore;
      }



function maxFun(board, depth) {
  if (depth >= maxDepth) {
    board.setScore();
    return board.score;
  }


  var boards = board.generateNewBoardsBlacksTurn();
  if (depth == 0) {

  }
  var topBoardNo = 0;
  var topScore = -100000;
  for (var i = 0; i < boards.length; i++) {
    var score = minFun(boards[i], depth + 1);
    if (score > topScore) {
      topBoardNo = i;
      topScore = score;
    }
  }

  if (depth == 0) {

    return boards[topBoardNo];
```

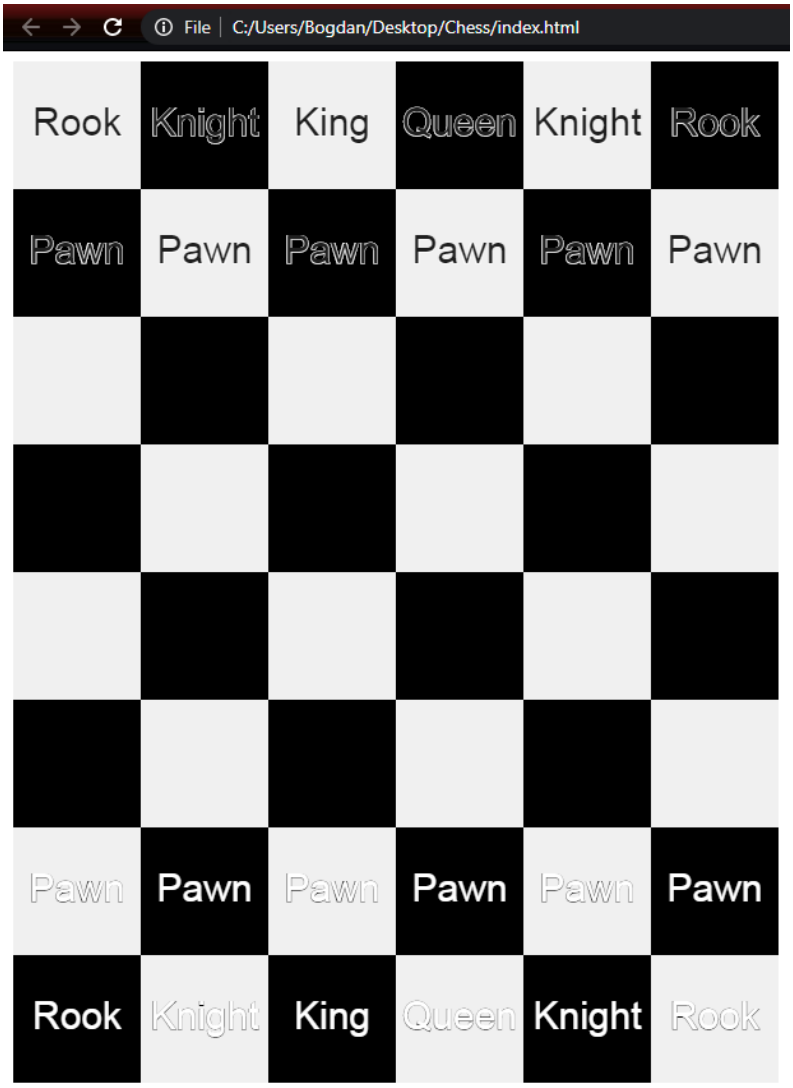
```
  }
  return topScore;
}
```

4. Diagramă de clase




Board
whitePieces
blackPieces
score
setupPieces()
show()
isPieceAt(x, y)
getPieceAt(x, y)
getPieceAt(x, y)
generateNewBoardsWhitesTurn()
generateNewBoardsBlacksTurn()
setScore()
move(from, to)
clone()
isDone()
isDead()
hasWon()

5. Imagini program



6. Bibliografie

[1] JavaScript Tutorial - W3Schools

https://www.w3schools.com/js/

[2] p5.js

https://p5js.org/

[3] p5.js Reference

https://p5js.org/reference/#/libraries/p5.dom

[4] p5.js Example

https://p5js.org/examples/dom-modifying-the-dom.html

[5] Minimax

https://www.chessprogramming.org/Minimax

[6] Los Alamos chess

https://en.wikipedia.org/wiki/Los_Alamos_chess

[7] En passant

https://ro.wikipedia.org/wiki/En_passant

[8] Building a Simple Chess AI – Brandon Yanofsky

https://byanofsky.com/2017/07/06/building-a-simple-chess-ai/

[9] p5.js | setup() Function

https://www.geeksforgeeks.org/p5-js-setup-function/

[10] Youtube

https://www.youtube.com/watch?v=XOgtEMSSF9E

[11] Youtube

https://www.youtube.com/watch?v=DZfv0YgLJ2Q

[12] What is the right way to write my script 'src' url for a local development environment?

https://stackoverflow.com/questions/16677095/what-is-the-right-way-to-write-my-script-src-url-for-a-local-development-envir

[13] Sololearn JavaScript Tutorial

https://www.sololearn.com/Course/JavaScript/

[14] Draw.io

http://draw.io/