Connect 4 Game

This report describes the specification of implementation of a game connect 4. Despite the fact the project group been suggested to consist of 3-4 people – In author's case the most significant part of implementation has bee made by an author with cooperation with Late Vitolina, which in the end submitted her own project.

The implementation of the game is mainly based on Artificial Intelligence a modern approach(Stuart Russel, Peter Norvig, Second edition), and only most interesting parts of the implementation will be described.
The whole implementation code can be examined by referencing to GameLogic.java.

The report covers following topics:

- **Min-max algorithm** *with emphasis on successors generation.*
- **Alpha-beta search algorithm** *describing how the performance has increases using know techniques.*
- **Evaluation function** *covers the detailed explanation of the idea behind the evaluation of the game states*

Min max

The implementation of the min-max algorithm has been based on the pseducode provided in Artificial Intelligence a modern approach(Stuart Russel, Peter Norvig, Second edition p194).

The most interesting aspect is the generation of the successors from the current state, which has been performed using simple technique of temporary inserting a coin into a board array – performing a recursive call and restoring the default back(which is a 0 in case of integer array) see Illustration 1.

```
for (int i = 0; i < cols; i++) {
    if (columnIsFull(board, i) != true) {

        insertCoin(i, 2);
        value = Math.max(value, minMove(board,
        removeCoin(i);
```

Illustration 1: successors generation

Another non-less significant part is the minimax method, which returns an actual action leading to best state, instead of a value comparing to maxMove and minMove see Illustration 2.

```
private int minimax(int[][] board) {
    int alpha = -999999;
    int beta = 999999;

    int result = 0;
    int value;


    //Stopwatch timer= new Stopwatch();


    for (int i = 0; i < cols; i++) {
        if (columnIsFull(board, i) != true) {
            insertCoin(i, playerID);
            value = minMove(board, 1, alpha, beta);
            removeCoin(i);
            if (value >= alpha) {

                alpha = Math.max(alpha, value);
                result = i;
            }
        }
    }
    //System.out.println("time elapsed: "+timer.elap
    return result;
}
```

Illustration 2: miniMax

Alpha beta search

Alpha-beta pruning is a technique that allows to improve the performance of the algorithm, and which has been implementation from the Alpha-beta search algorithm's pseducode described in a A modern approach book.

We keep track of already returned choices via alpha and beta, prune the values which are known to be the worse based on current alpha and beta.

```
//alpha beta cut
if (value <= alpha) return value;
beta = Math.min(beta, value);
```

Evaluation function and cut off test

The heart of the connect 4 game is a evaluation method which contains a scoring system allowing to evaluation non-terminal states, which been successfully evaluated by cut-off function.

The cut-off test *performs a check if the maximum depth has been reached or the current state is terminal. The implementation can be seen on Illustration 3*

```
//cut-off test
if (depth >= MAXDEPTH || terminalTest()){
    return evaluate(playerID);
}
```

Illustration 3: Cut-off test

Evaluation function *of the game has been based on based based on R.L. Rivest, Game Tree Search by Min/Max Approximation 1988 p77-96.*

The evaluation has following premises:

```
a win by X has a value of +512,
a win by O has avalue of -512,
a draw has a value of 0,
```

Otherwise, take all possible straight segments on the grid(defined as a set of four slots in a line–horizontal, vertical, or diagonal) evaluate each of them according to the rules below, and return the sum of the values all segments, plus a move bonus depending on whose turn it is to play (+16 for X, -16 for O), as depicted in Illustration 4.

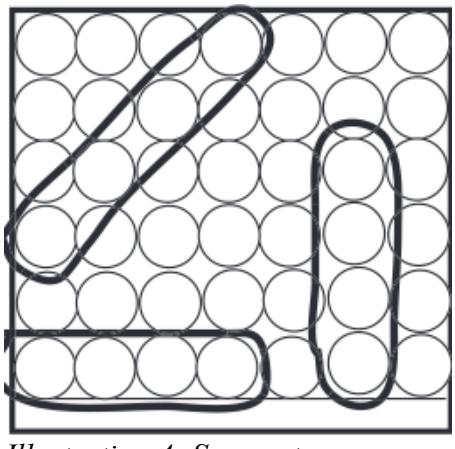

Illustration 4: Segments

The rules for evaluating segments are as follows:

```
-50 for three Os, no Xs,
-10 for two Os, no Xs,
- 1 for one O, no Xs,
0 for no tokens, or mixed Xs and Os,
1 for one X, no Os,
10 for two Xs, no Os,
50 for three Xs, no Os.
```

Note that X is a maximizing player and O is a minimizing player in my implementation.

The implementation of this approach is based on two methods: evaluate and value.

Evaluate

This functions loops through all possible 4 coin segments and calls a value function passing the current and delta row and column for each possibility.

Value

The method performs an evaluation of the passed segment and return a score, see Illustration 5.

```
//evaluation of scores based on R.L.Rivest
if(playerCount==0 &&opponentCount!=0){
    switch (opponentCount){

        case 4:
            return -512;
        case 3:
            return -50;
        case 2:
            return -10;
        case 1:
            return -1;

    }
}

if(opponentCount==0 &&playerCount!=0){
    switch (playerCount){

        case 4:
            return 512;
        case 3:
            return  50;
        case 2:
            return 10;
        case 1:
            return 1;

    }
}
```

Illustration 5: Scoring

References:

1. *Assignment 3 (C)Game Trees and Alpha-Beta PruningComputer Science 182 – Fall 2010 (Harvard)*
2. *R.L. Rivest, Game Tree Search by Min/Max Approximation 1988*
3. *S.Russel, Peter Norvig Artificial Intelligence a Modern Approach*