TriFit

-Turn based Android game between user and computer on a grid board using Artificial Intelligence

Artificial Intelligence (CED16)

Project

By

Aditya Raj Verma(2017UCO1538)

Aman Singh(2017UCO1552)

Harshit Nand (2017UCO1567)

Tushar Dholke (2017UCO1564)

Introduction

Trifit (Try-fit) is a game between user and computer on a grid board using Artificial Intelligence.

How to Play:

\* You will have initially 8 stones to place on board (one in each turn).

\* Once you are out of stones in hand then you can move your placed stones to adjacent positions.

\* Your strategy should be to form triplet as well as block computer to do the same.

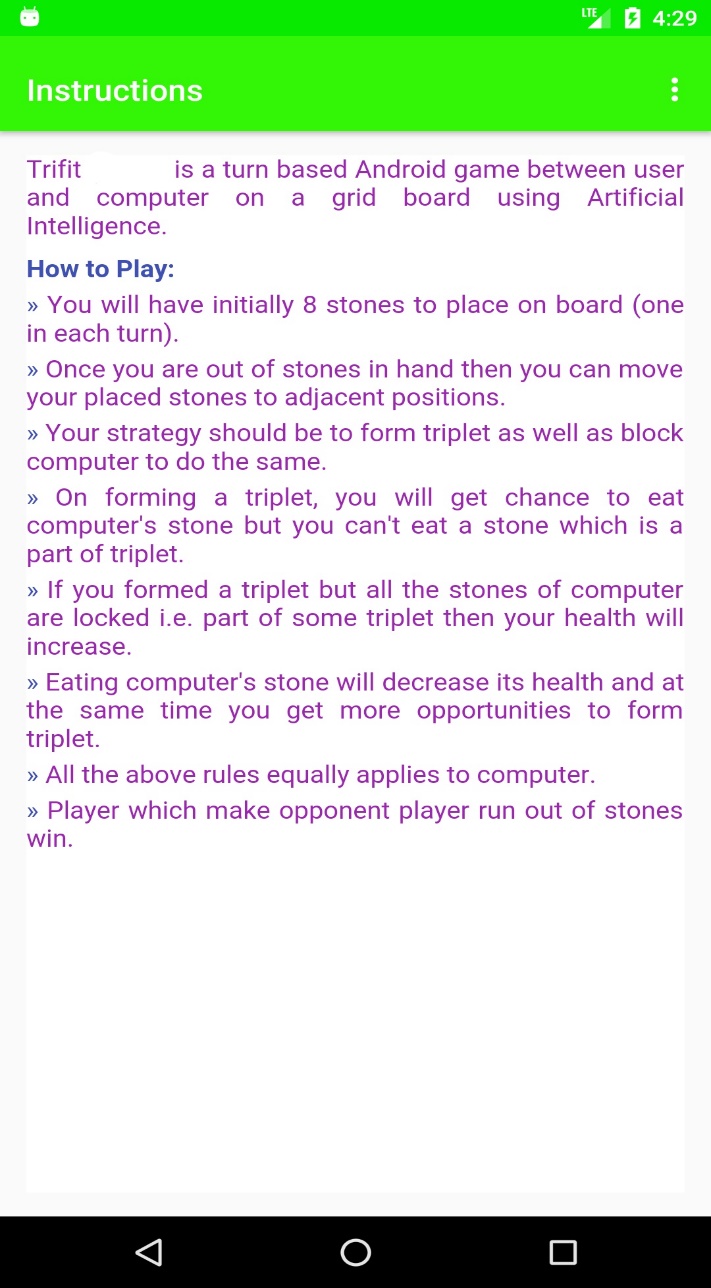
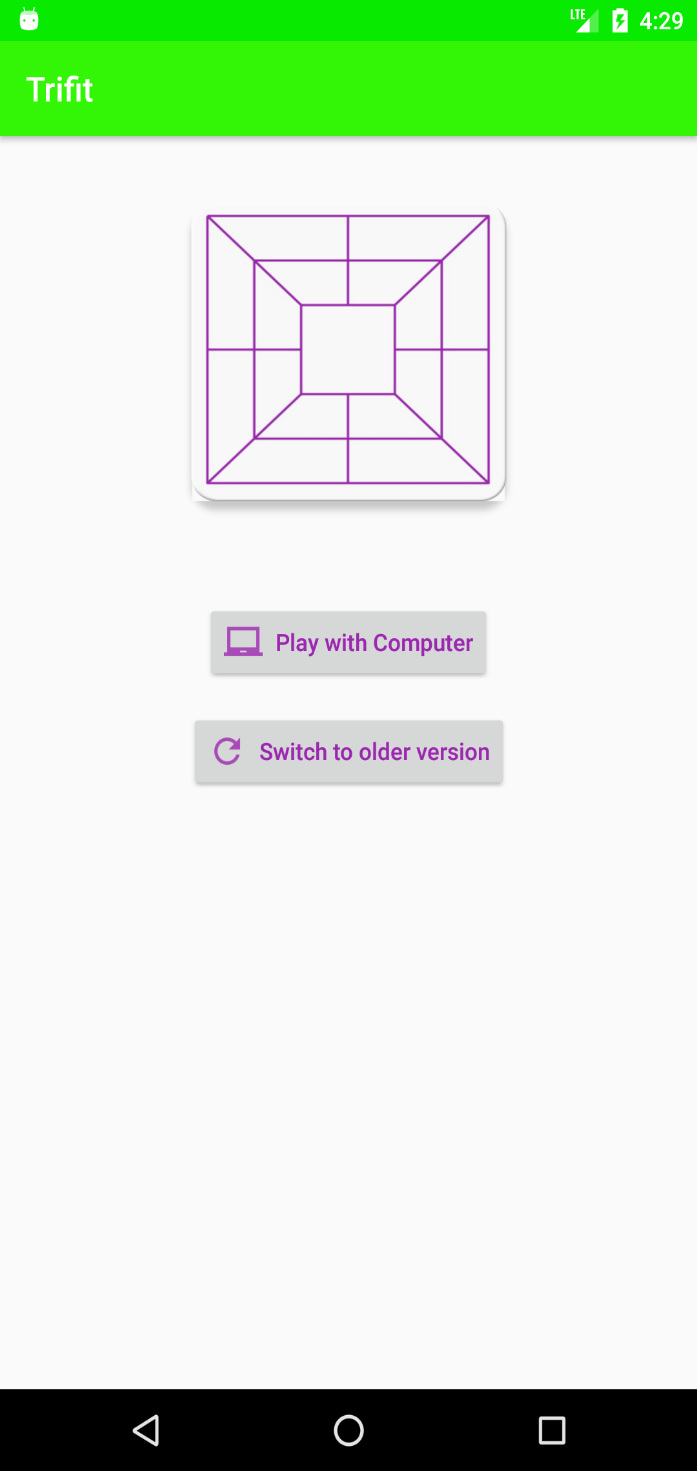
\* On forming a triplet, you will get chance to eat computer's stone but you can't eat a stone which is a part of triplet.

\* If you formed a triplet but all the stones of computer are locked i.e. part of some triplet then your health will increase.

\* Eating computer's stone will decrease its health and at the same time you get more opportunities to form triplet.

\* All the above rules equally apply to computer.

\* Player which make opponent player run out of stones win.

Instructions Screen Main Screen

AI Alogorithm & Working

The core idea behind trifit game is very similar to the one behind tic-tac-toe game.

Just like the tic-tac-toe game, trifit game can also be framed into a typical search problem involving game trees with each node representing a game state (or game position) and each edge representing a move made by a player.

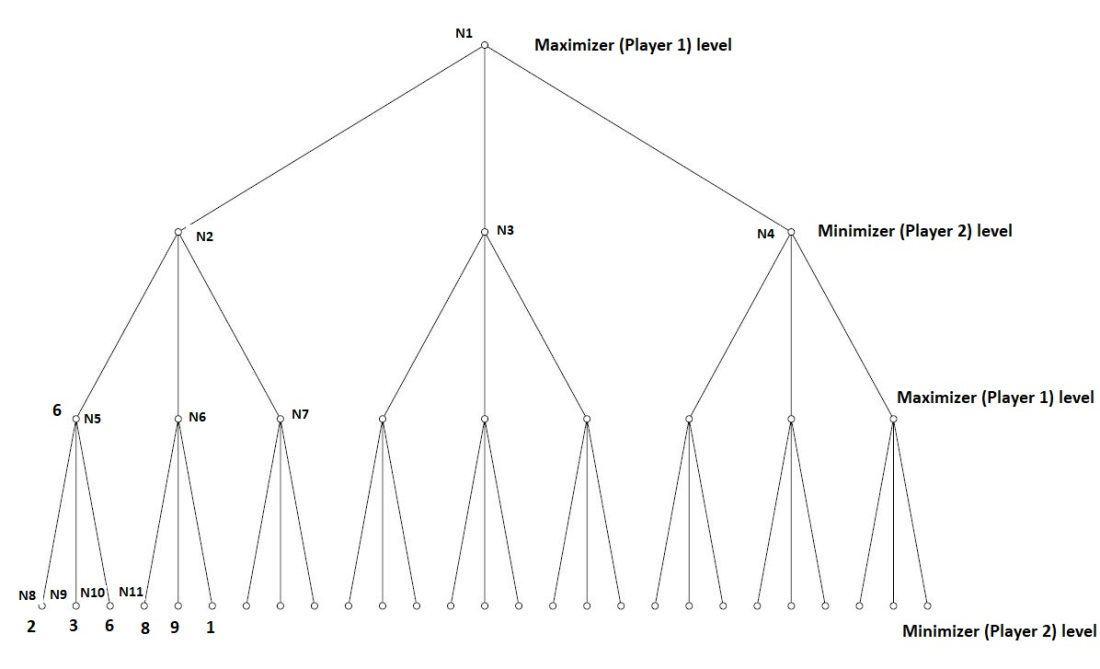
But unlike tic-tac-toe, the game tree for trifit is very large (over 600 million possible states).

So if we try to dynamically evaluate game states by going all the way down to the leaf nodes, it will take us an insane amount of time, if at all our computer can process it. Therefore, we should go down only a few levels. To further improve performance, we can use a slightly modified version of the recursive minimax algorithm called recursive **minimax with alpha-beta pruning**. It offers drastic improvement over the regular version in terms of the number of states (nodes in the tree) evaluated.

Intuition Behind Alpha-Beta Pruning Algorithm – Alpha- Beta pruning algorithm is based on the idea that, in order to determine the static estimate associated with any node, we do not need to go down all the paths starting from that node. We can prune away sub-trees that are not going to influence the value of the static estimate.

In alpha beta pruning, we associate two variables alpha and beta with every tree node that is being evaluated. Alpha denotes the lower bound and beta denotes the upper bound for the static estimate. Initially, alpha is negative infinity and beta is positive infinity.

To sum up,we used concepts of heuristic search, minimax, alpha-beta pruning, retrograde inspection, game theory and depth limited search.



Steps to evaluate heuristic value of each junction in order of decreasing priority:-

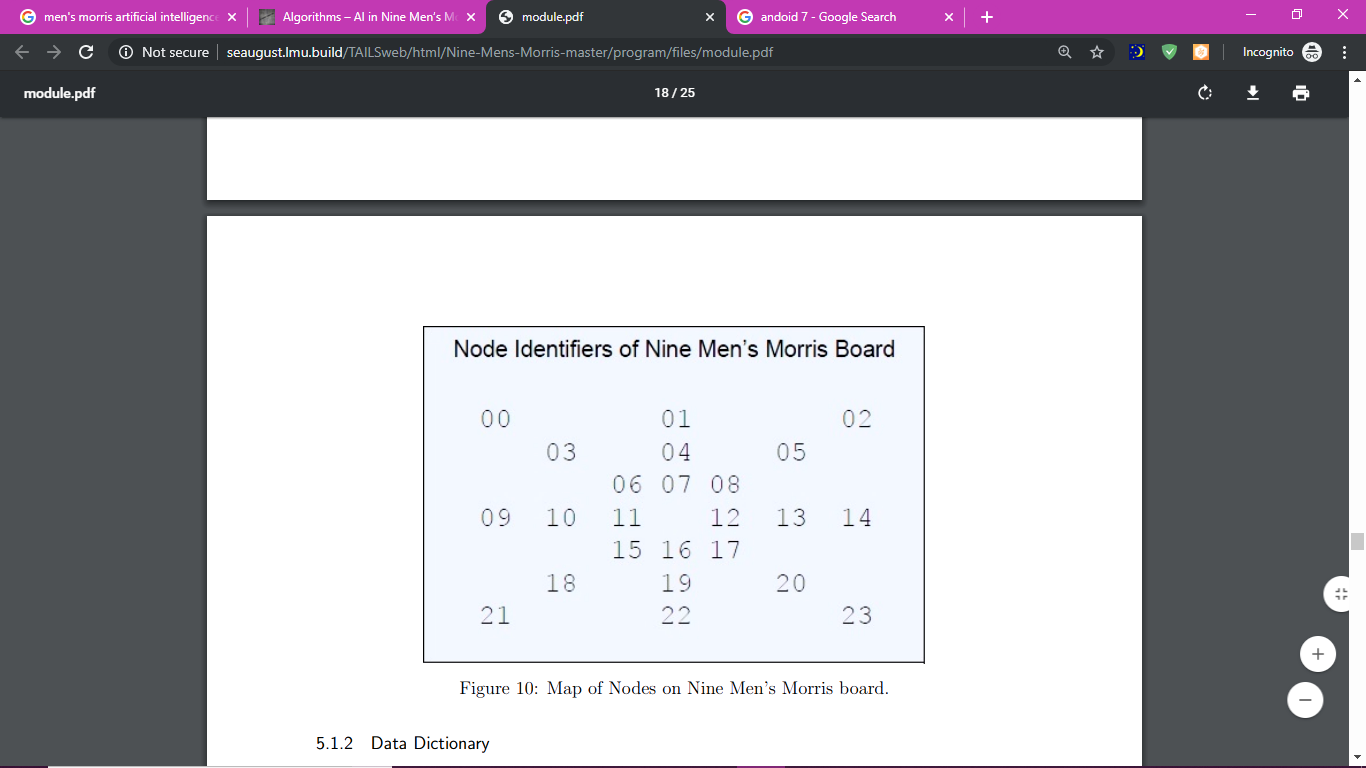
1. Check if triplet formation is possible. If it is possible, then form own triplet.
2. Block opponents triplets and dual triplets from being formed.
3. Place on opposite corner than the opponent.
4. Target odd places.
5. Target even places.

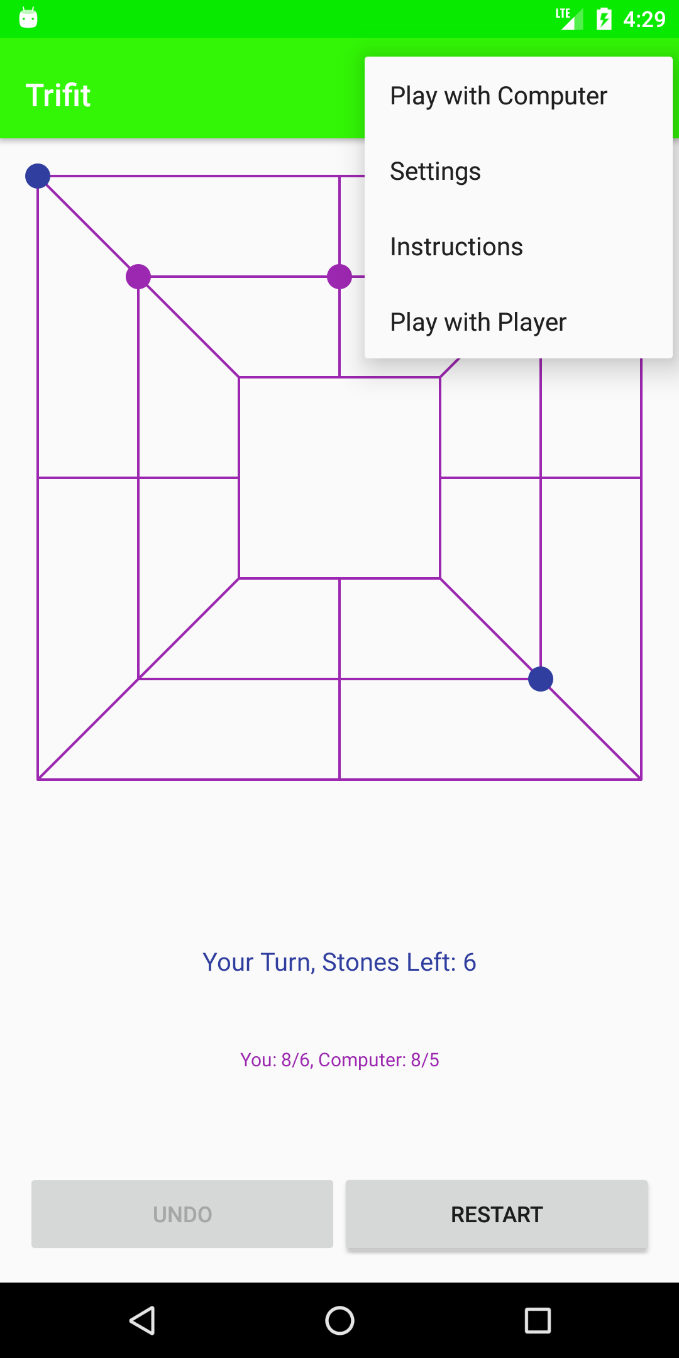
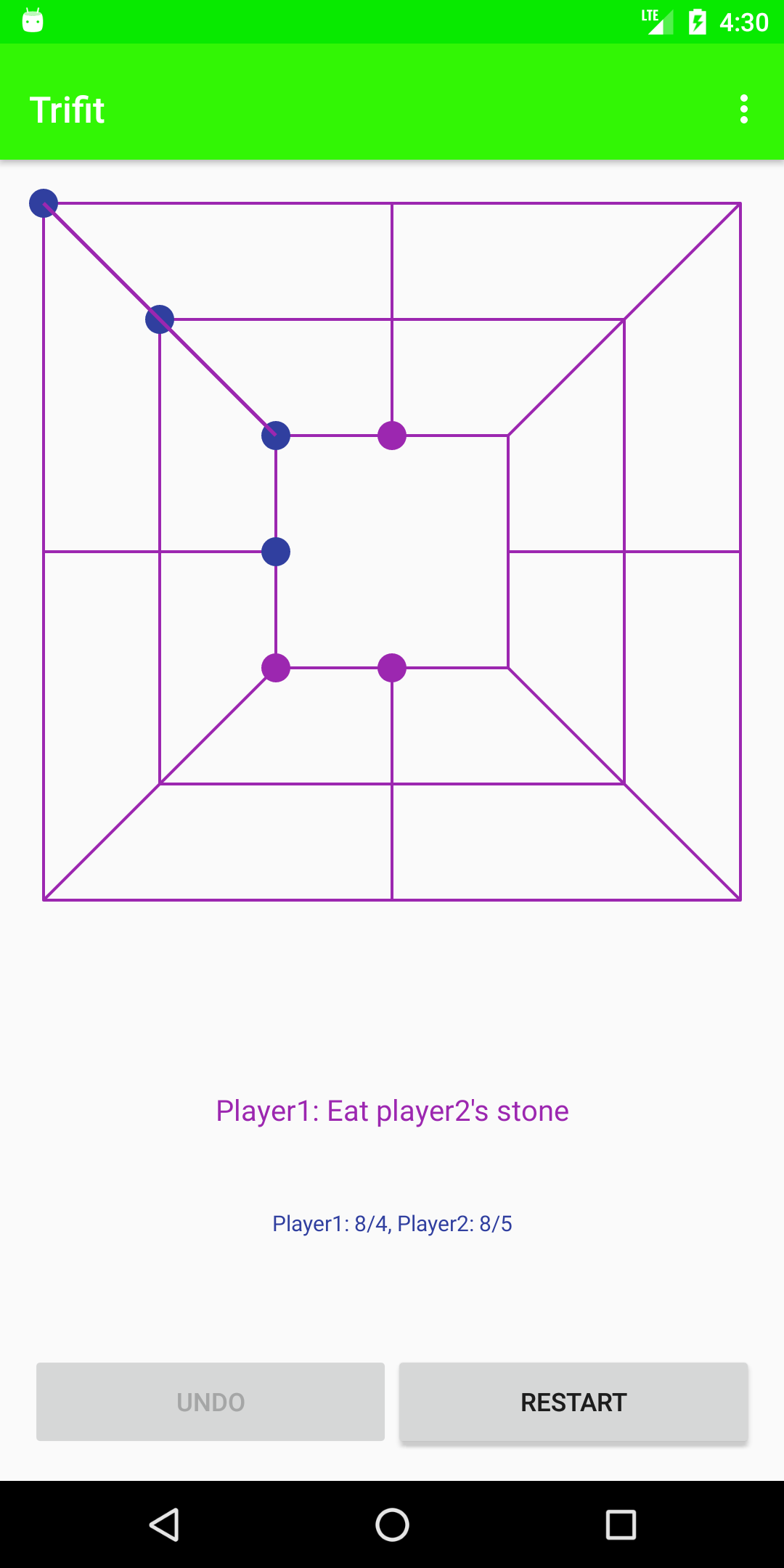
State Representation :-

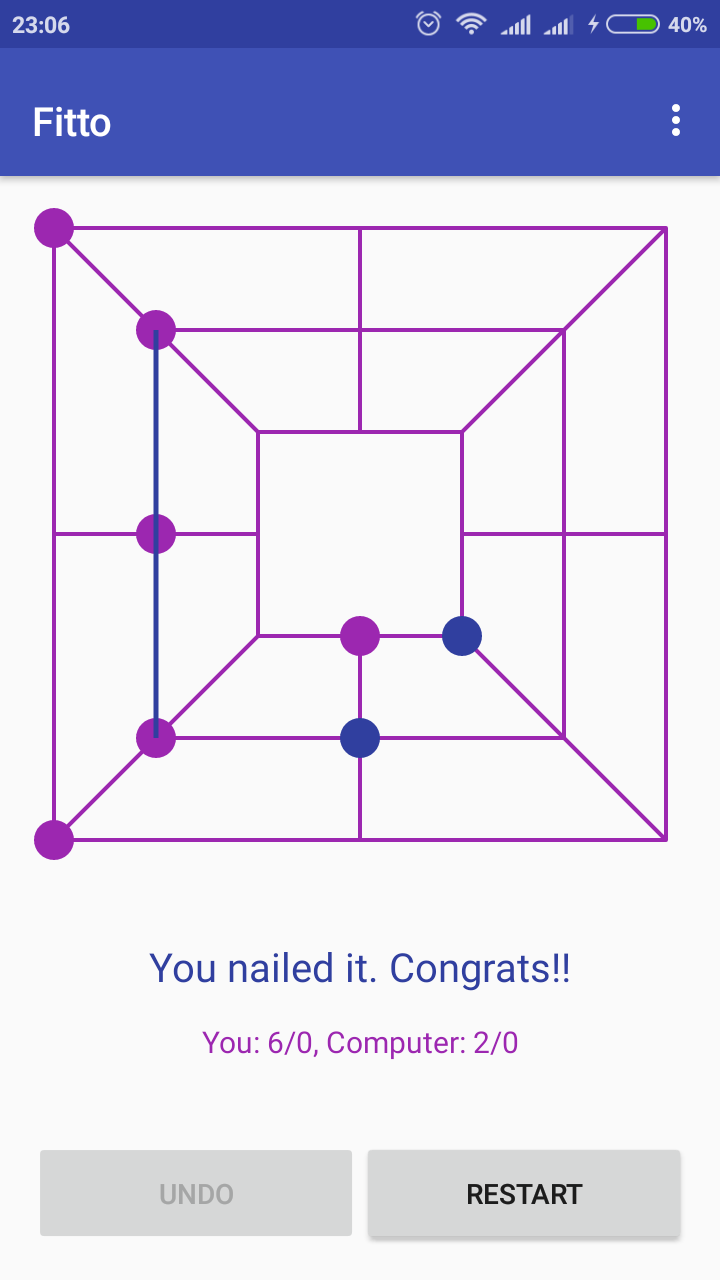
\* Mathematical representation of BoardView

\* Numbers represent junction where stones will be put

\*4 actions there:-pickStone, drawStone , placeStone and eatStone.





Tech Stack

* Java
* Andoid 7 (nougat)
* Android Studio for developing app
* Gradle for building and debugging
* Libraries webkit, android.view, android.widget , android.graphics,os etc used

Time and Space Analysis

**Time**

With an (average or constant) branching factor of b, and a search depth of d plies, the maximum number of leaf node positions evaluated O(bd) – the same as a simple minimax search. If the move ordering for the search is optimal, the number of leaf node positions evaluated is O(bd/2). In the latter case, where the ply of a search is even, the effective branching factor is reduced to its square root. For each, only the best second player’s move is needed to refute all but the first (and best) first player move – alpha–beta ensures no other second player moves need be considered. The alpha-beta pruning examines only O(b^3/4).

**Space**

The effectiveness of alpha–beta pruning is highly dependent on the order in which the states are examined. In an optimized order, alpha-beta needs to examine only O(bd/2) nodes to pick the best move, instead of O(bd) for minimax. This means that the effective branching factor becomes p b instead of b, so we can say that the space complexity of alpha-beta pruning in this case will root b into d.

Applications

* The Minimax algorithm is an essential adversarial search algorithm that has been applied to problems ranging from zero-sum game play to real-time pursuer evasion. The application of the Minimax algorithm to real world problems is no different than any other algorithm’s application in that it is often twisted into a hybrid with other algorithms or concepts.
* Deep Blue II was a super computer designed by IBM to play Chess at a Grand Masterlevel. Deep Blue implements a depth limited Minimax algorithm with alpha-beta pruning. To improve the efficiency of alpha-beta pruning, the move generator was designed such that the optimal moves were generated first. Deep Blue successfully beat the Chess World Champion Kasparov in 1997.
* Honeywell has developed a 3D Minimax Pursuit Evasion algorithm. In this application of the Minimax algorithm the pursuer wants to minimize the missile distance and the missile wants to maximize some cost function.
* These examples provide some understanding of how the Minimax algorithm can be used to solve real world problems.

Future Scope

AI heuristics can further be tuned to get more optimal results and stronger AI gameplay,

Effort can be made to increase effitiency further.

The algorithms implemented can be reused for other problems that can be solved by minimax approach.

