# Ishido Manual

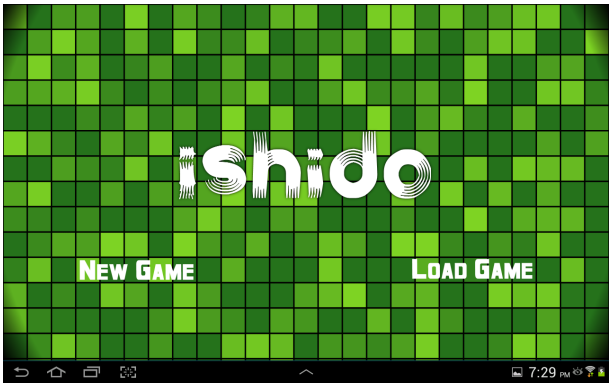
Bishal Regmi

February 2016

## 1 Activities

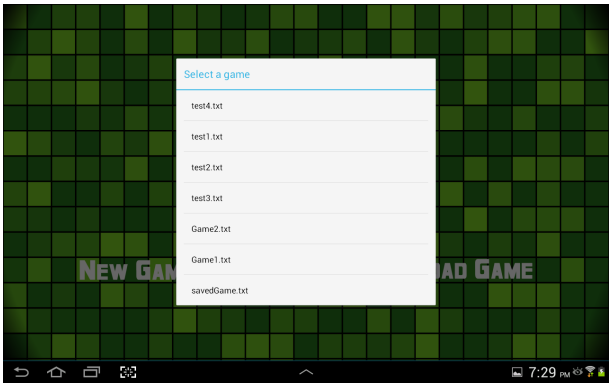
### 1.1 StartActivity

This activity is the splash screen for my game. It is where you select whether to load a saved game or a start a new game. If new game is selected, it will ask you to make a choice for a coin toss and decides the player's turn based on that. If load game is selected, it lists all the game's that you can load and you select one from them. It kills itself after it opens a new activity "BoardActivity".
This uses Linear Layout. It contains two buttons: new game and load game.



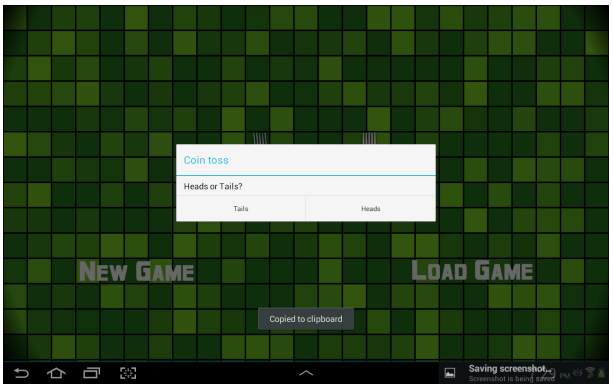

StartActivity

When the "LOAD GAME" button is pressed, a dialog box appears that lists all the available state of the games from which you can choose one.



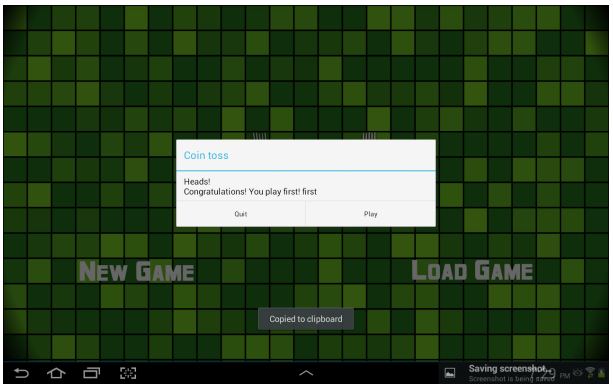

Select file

When the "NEW GAME" button is pressed, a dialog box appears that allows you to choose heads or tails for the coin toss to determine the 1st player.




Choose head or tail

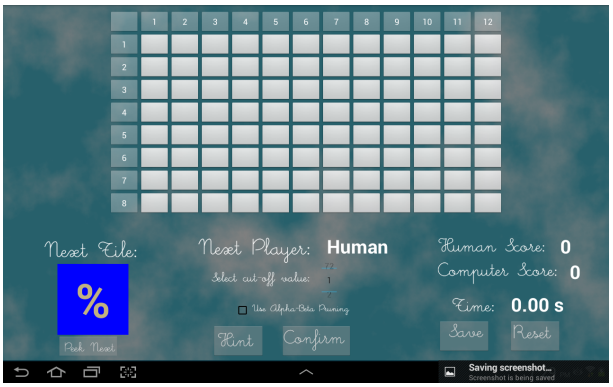
After selecting an option, the coin will be tossed and a dialog box appears showing the results.




Results of coin toss

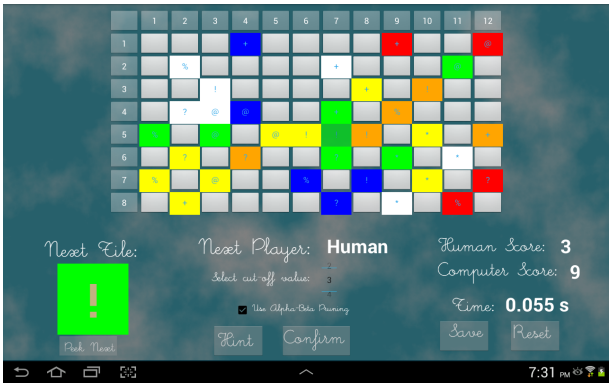
## 1.2 BoardActivity

This activity is the meat of the view part of the application. It handles the main game which includes fetching tiles, move hints, computer move, validating human's move, allowing options to save game letting the user choose between different searches and fetching the score for the player. This activity is also a bridge between the logic model and the view. It also handles all the event listener for different elements of the app. This activity contains a Linear layout as a parent layout. Under that it contains an 8X12 Grid Layout to hold the game board. The game board occupies almost 75% of the screen. The rest of the screen is filled with various linear layouts that consists of labels. This layout contains a preview button which displays the next tile to be placed, a next button that generates the next tile, a number spinner to choose ply cut-offs, score to display the score, reset button to reset the board and save button to save the game..
Here are few screen shots of various stages of this activity:



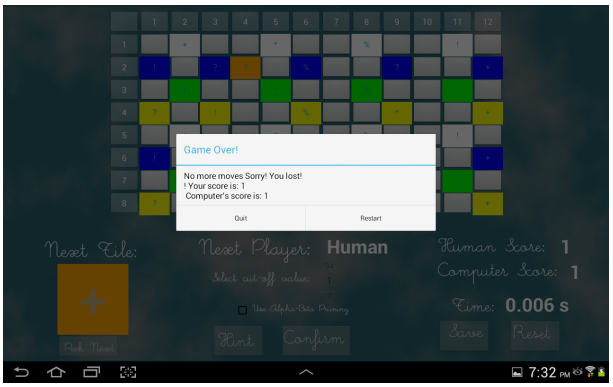

New Game


Loaded Game




When the user selects hint, the best move suggested by the algorithm blinks.


A message saying invalid move is shown if the user makes an invalid move




Dialog box when the game is over

This screen appears when all possible tiles or moves are over. You can either quit the game or restart the game which basically is resetting the board.

# 2 Model Classes and AI

## 2.1 Model Classes

I have 7 classes in my model. They are enlisted below with the description of what they basically do:

- Board
  This class hold the 2-dimensional array of 8*12 that holds the game board. It also contains all the AI algorithms for searching. This class also contains algorithms to check if a tile can be placed at any location or not. This class also acts as a bridge between file and the gameBoard.
- Deck
  This class holds the array of colors, symbols, the tile combinations. It contains the method to set and get the stock, remaining tile count and all other deck related functions.
- Move
  This class represents a move made by the user. It contains row, col, tile combination, score and parent associated with the move.
- Player This class holds the score of the game for each player This class imports the state stock and score from the file given an inputstream.
- Tile This class represents the tile combination. It contain the color ans symbol of the tile.

## 2.2 AI

The AI part of this application are the 2 search algorithms:

- Minmax Search/ alpha beta pruning : ***private int minmax(int cutOff, boolean maximizer, int humanScore, int computerScore, boolean computer,int alpha, int beta)*** This search provides the best move using the min max search with out without alpha beta pruning upto a depth cutoff specified as an argument. The heuristic value its uses is the difference between the score of the players based on their turn i.e. human-computer if its human's turn and vice versa
- Search for Next Available Tile (gets the first available position for a tile in the board in row major order ) :: ***private Move getNextMove(Move move)***

All these algorithms are implemented in the respected functions in the ***Board*** class. These algorithms are called using specified methods in the ***Board*** class from the ***BoardActivity*** based upon the user selection. After the user presses the Confirm or Hint Button, the corresponding function is called which returns the next move.

# 3 Bug Report

- The program tremendously longer time for cut-off value greater than 6. It might be because the tree is too large.
- Other than that, I haven't encountered any bugs yet.

# 4 Features

## 4.1 Implemented

1. Only the necessary objects in the layout are displayed. The layout is not crowded.
2. A dialog box appears after the game is over asking the user if they want to replay or exit.
3. User has the option to reset the board if they are not satisfied with the game.
4. The program will not let the other user make a move or ask for a hint if its not his/its turn.
5. All the rows and column are labelled such that its easier for the user to navigate.

## 4.2 Not implemented

1. The user doesn't have a choice to specify the filename to save the file.
2. If next tile is peeked, the user cannot peek back.
3. The game doesn't have undo feature
4. Portrait orientation is not available
5. The computer's move blinks only for 5 seconds.

# 5 Log

- March 11th (2 hours)
  Designed the background for the startActivity and BoardActivity to make the layout look more attractive.

- March 12th (2 hour)
  Updated the serialization part to get the additional Human Score and Next Player. Wrote the code to write the file to save the game. The game is written. However, I can't locate where the file is.

- March 13th (1 hr)
  Realized that I had been using internal storage which is not accessible and needed to use external storage. Made a custom file and saved it to the external storage. Hurray! I am able to read the file.

- March 14th (1 hr)
  Working on coin toss and managing the turns in the start activity to send the required arguments to the BoardActivity.

- March 15th (4 hours)
  Wrote a rough pesudocode for minmax algorithm with and without alpha beta pruning.

- March 16th (1.5 hours)
  Implemented the pseudocode in the program. I am getting answers but doesn't look like I'm getting the right ones.

- March 17th (2 hours)
  Figured out that I wasn't decrementing the stock and the score after coming back from the recursion which was skipping the tiles. However, the score is still not right.

- March 18th(2 hours)
  Figured out that was changing the the order of heuristic calculation at every single step. However, I was supposed to calculate the heuristic based on turn. This fixed the issue. I'm getting correct answers. However, the algorithm is extreemly slow, cut-off of 5 took 7 minutes!

- March 19th(3 hours)
  The MoveNode class that recorded heuristic and move is no longer needed. I could just use the heuristic to determine the best move. Getting rid of MoveNode simplified the code by a great deal. Now, cut-off of 5 runs in a second.

- March 20th(3 hours)
  Trying to optimize the code by getting rid of redundant and unnecessary steps. Removed fucntion checkRowConditions and checkColumnConditions. Got rid of a few methods from the Deck Class. Made a few changes into the layout.

Total: 21.5 hours