Konane : A Hawaiian Game

TECHNICAL MANUAL

March 27, 2018

Abish Jha
Ramapo College of New Jersey

Activities

MainActivity:

This is the first activity the user comes across. This activity has a title that says “Welcome to Konane!” and a help button at the top that leads to the HelpActivity.+

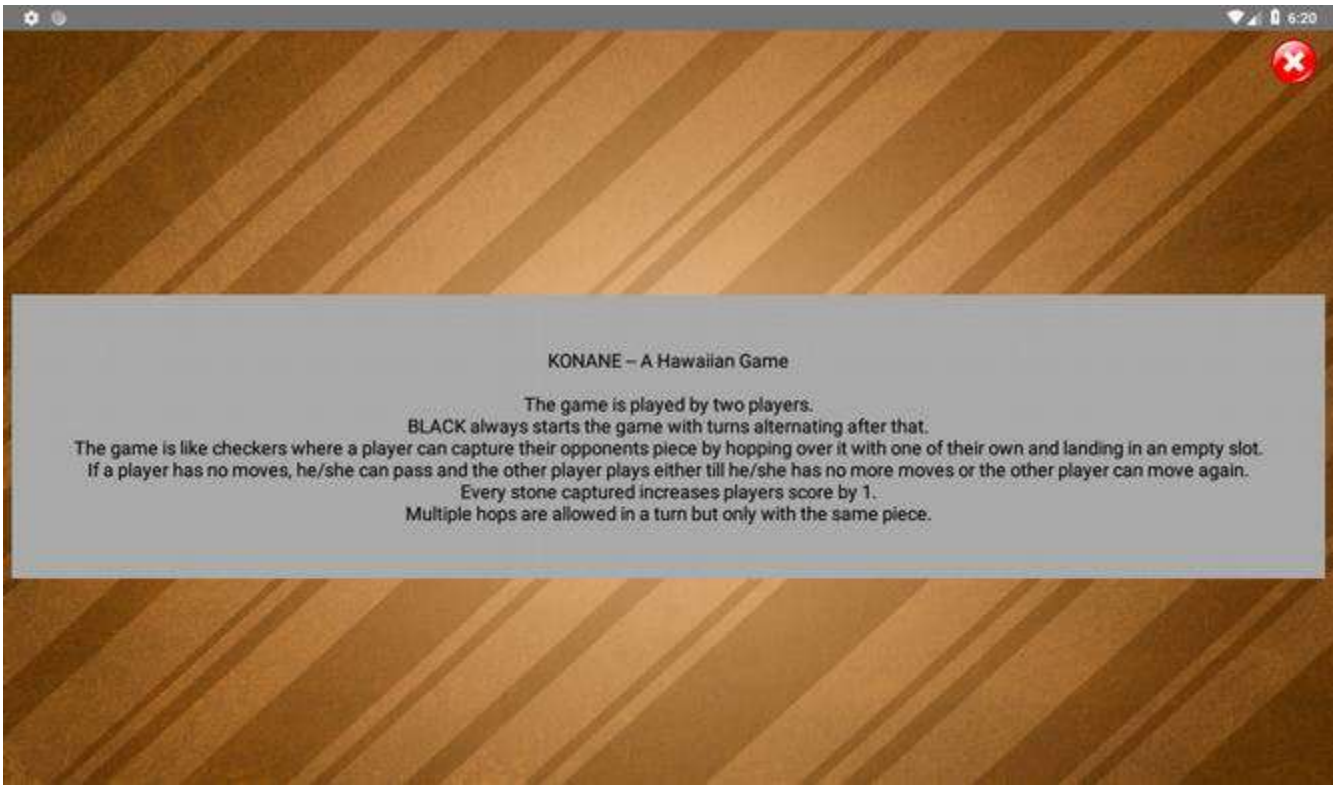
There is a ‘New Game’ button which lets the user start a new game. Once this is pressed, a dialog appears asking the user for the board size.

After, the user chooses a size another dialog appears with two tile which asks the user to guess which of them is the black tile. If the user chooses the black tile, he/she plays first as Black and the computer second as White. If not, the computer goes first as Black and the user second as White. Once a tile is choosing, a new game is started after a wait of 3000 millisecond.

There is also a ‘Load Game’ button which allows the user to pick a saved file from the Documents folder on the phone. This file is then serialized and the game resumes from the saved state.

HelpActivity:

This is a basic activity that contains some text which tells the user about the rules of the game and how it is played. There is a cross mark on the top of the screen, which resumes the MainActivity so the user can continue with the game.



GameActivity:

This is the main game activity for the application. On the screen is displayed,

board – the board for the game. can be of sizes: 6 * 6, 8 * 8, or 10 * 10

player1 score – score for player1

player2 score – score for player2

move button – the user presses this when he/she wants to make the move that has been selected. Validation is done.

pass button – the user presses this when he/she wants to pass their turn. Validation is done.

message display – displays message as per the game state examples of which are : “pass completed”, “pass not valid”, “move completed”, “move not valid”, “move partly valid” and so on.

current move – displays black if the current move is of player Black or white if the current move is of player White. Also displays ‘C’ if it’s the computers turn and ‘H’ if it’s the humans turn.

reset move – this button resets the memory of tiles selected in the current move.

algo spinner – there is a spinner on screen which allows the user to choose between one of the following options: (choose algorithm), DepthFS, BreadthFS, BestFS, and, BranchAndBound.

next button – this button when pressed for the first time loads the results of the selected algorithm’s search if an algorithm has been selected. After that the button displays the jumps in the move suggested by the algorithm. When pressed again, the button displays the next move in the queue.

best move button – this button displays the best move for the current player for the given board state according to the selected ply cutoff.

ply cut-off spinner – this spinner allows the user to select a ply cut-off for the minimax algorithm.

alpha beta switch – when on, the minimax algorithm incorporates alpha-beta pruning for faster search result.

save button – when pressed, a dialog appears prompting the user for a filename. the current state of the board is saved as filename.txt in the Documents folder of the phone so it can be loaded later.

This activity is the main activity for the game as it deals with all the logic behind the game play and provides functions for the game.

FinishActivity:

This is the last activity in the program. This activity gets player1's score and player2's score and displays them. Then the activity displays who the winner is. If the scores are the same, the activity displays that it is a tie.

There are also two buttons on the screen. The 'New Game' button leads to MainActivity where a new game can be started. The 'Restart' button starts another game with the same board size and the same the user as white or black player as was in the previous game and leads to GameActivity.

Model Classes

There are four classes that work as the Model and they are:

- Board.java
- Player.java
- Game.java
- Search.java
- MiniMaxNode.java

Board.java:

This is the class that creates an instance of the board which fills the board initially at the beginning of the board and removes one black stone and one white stone at random. After this the class deals with setting tiles at a certain coordinate, getting them and dealing with anything else related to the board.

Player.java:

This class deals with an instance of a player and stores the player's type (computer or human), score and color. Since this is a two-player game, we create two instances of Player class in the Game class, one for the white player and one for the black.

Game.java:

This class is the main class for the logic. It creates one instance of board and two instances of player. It also deals with the validation of moves and performing the actual move. This class has several methods which help the GameActivity class including the function to start move search.

Search.java:

This class holds everything required for the searches and allows a search to be performed after an instance of the class has been declared. This class is very easy to use. Make an instance of the class and pass the appropriate parameters to it and just call the getOutput method after that to get a list containing the moves visited by the selected algorithm.

This class contains a few AI algorithms which include Depth First Search (DepthFS), Breadth First Search (BreadthFS), Best First Search (BestFS), Branch and Bound, and MiniMax algorithm where the cut off depth can be specified. All the AI algorithms are implemented within the Search class.

- DepthFS – consists of two function, one which launches the DFS from different tiles and the other which is the recursive algorithm that does the actual traversal.
- BreadthFS – consists of a single function which utilizes a queue to keep track of the visited nodes and performs the search.
- BestFS – utilizes DFS to search for every possible move and then arranges them using score as a heuristic function.
- Branch and Bound – lets the user specify a cut-off depth which is the maximum depth to search the tree to. This algorithm is like DFS but selects and returns only the best move on the board considered according to the specified priority order.
- MiniMax – two versions of this algorithm. One with alpha beta pruning, one without. Both versions use the same function. The only difference is the alphabeta flag which gets turned on when alpha-beta pruning is selected. This search utilizes the MiniMaxNode class to make nodes for the search tree and the node contains the information for that node. A cut-off depth is also specified which limits the search tree to the specified depth. This function resides inside the Search class as do the other searching algorithms.

Bug Report

There are no known bugs.

Feature Report

There are four .xml files that work as the features and they are:

- activity_main.xml
- activity_help.xml
- activity_game.xml
- activity_finish.xml

activity_main.xml:

This screen lets the user choose between a New Game and Load Game. If a new game is selected, the user is asked for the board size and is allowed to choose between two tiles to select if computer or human plays first.

I implemented a help button which shows the rules for the game which was not required by specifications.

activity_help.xml:

This screen shows the rules for the game. Just a simple helper function for first time players.

activity_game.xml:

This view has a TableLayout which holds the board for the game which has all the buttons. Each button has an action-listener. This activity also has a “Current Move” button which shows the player who is supposed to move, the score for player1, and player2, and also buttons which allow

‘Move’, ‘Pass’, and ‘Clear Move’. ‘Clear Move’ clears the moves that have been recorded for the current move, ‘Move’ brings into action the set of moves that have been passed in the current move, and ‘Pass’ lets the player pass his/her turn if it is valid. The ‘Current Move’ field shows the color of the player that is supposed to move in the current move.

There is also a spinner which lets the user select an algorithm of their choice and search the board for possible moves. When the user presses the button beside the spinner, a move is displayed. The tiles involved in the move also start blinking on the board for ease of use.

There is a best move button which shows the best move for the current player and the given board state based on the ply cut-off specified using MiniMax algorithm.

activity_finish.xml:

This activity displays the score for player1 and player2 after the game is done. It also shows who won on the game based on points acquired. This activity also provides options for the user to start a ‘New Game’ where the user is taken to the home screen (MainActivity), or ‘Restart’ a game where a new board is loaded but with the same dimensions and players as the previous game.

Note from the developer: the program uses strings to pass the calculated moves between the search class and the view class which makes it inherently slower because dealing with strings is very time consuming, especially string concatenation. However, this can be dealt with by using a list of integers instead of a string to send the moves between the classes. String have been used to make the code easier to read but if time is of priority, switching to integers or other data type would be a smart choice.