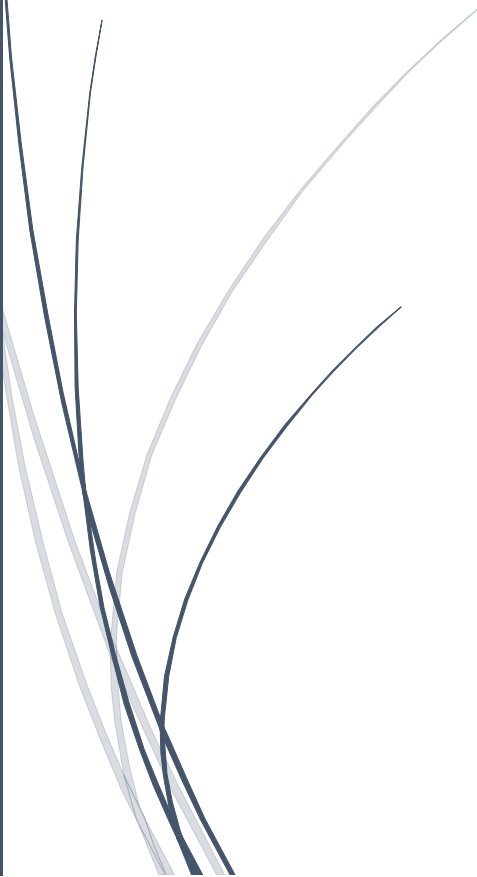


A dark blue vertical bar on the left side of the page, with a blue arrow pointing right from its center.

Vivek Pandey  
CMPS 366

# Duell in Android

## Technical Manual

Several thin, curved lines in dark blue and light grey originating from the bottom left corner.

**Duell** is a two-player chess variant played with dice on a board of 9x8 squares. Players take turns moving one of their dice in order to capture their opponent's pieces, with the ultimate aim of capturing the opponent's king to win the game.

Full Description:

[https://en.wikipedia.org/wiki/Duell\\_\(chess\)](https://en.wikipedia.org/wiki/Duell_(chess))

# Class Descriptions

## Controller Classes

**HomeActivity Class** – Serves as a controller for the Home Page of the game where the user chooses on whether to restore a game or start a fresh one.

**GameActivity Class** – Serves as a controller that handles the user actions during the game play, refreshes view, displays notifications and errors generated and stored in the Notifications Model Class.

**ResultsActivity Class** – Displays the results of a game and gets user choice on whether to continue or end a tournament to act accordingly.

## Model Classes

**Dice Class** – Implements the properties of a dice object used in the game. The class consists of variables and functions to store and modify a dice's face values, its coordinates within a game board, its controller (human or computer), its capture status, and whether it is a king.

**Square Class** – Stores and maintains the location and occupancy status of a square in the game board. A square can be occupied by a dice object only.

**Board Class** – Initializes and maintains a game board of squares and player dices. It consists of functions to initialize a board, retrieve the contents at different squares within the board, check and set the occupancy of these squares that comprise the game board, and determine if the board has arrived at a "Game Over" state.

**Player Class** – Implements the basic set of strategies for any player like traversing the board and determining path choices, validating and processing moves, changing dice orientation based on the rolls, capturing and eliminating opponent dices.

**Human Class** – Handles the move requests from a human player by performing input validations and processing them to reflect changes in the game board.

**Computer Class** – Implements Computer strategies to evaluate, prioritize, select and initiate the best move on behalf of the computer.

**Tournament Class** – A static class that keeps a track of the tournament scores for the human and computer players and whose turn it is next in case the game needs to be serialized.

**Serializer Class** – Contains necessary member functions to serialize or restore a tournament to and from a text file.

**Notifications Class** – A static class that keeps a track of the various errors and notifications that have been generated during the game play.

**No Special data structures were used.**

## Computer's Play Algorithm

The computer uses the following algorithm to make a move.

- 1) Check if any Computer dice can capture opponent's king or if computer king can get to opponent's key square. If yes, use that dice and end the game.
- 2) Check if any opponent's dice can get to Computer King or Computer Key Square. If yes,
  - a) Check if any of Computer's dice can capture that hostile die. If yes, use it.
  - b) Check if any Computer dice can block that hostile dice. If yes, use it
  - c) Try to move the king if it is safe around
- 3) Check if any Computer's dice can capture any of the other opponent's dice. If yes, use it.
- 4) Check if any of the Computer's dices other than the king are in threat of being captured. If yes, move that dice under threat to a safe location.
- 5) If none of the above finds a suitable move, calculate which Computer dice can safely get closest to the king in the next move and move it. This is helpful because the dice will keep safely revolving around the opponent king until a proper top value will come up for capture.

## Bug Report

All the bugs that were discovered during the development and testing phase were fixed right then. At the time of submission, none of the bugs have been left unresolved.

# Feature Report

## Additional Features

- 1) To make the process of restoring a saved game easier and to make this application completely free from text inputs, I have implemented a ListView rather than an input box to select restoration files. All the available files in the “Duell Data” folder are listed in the game interface and the user just has to tap on the proper name to restore the game. I found this to be way more user-friendly than asking the user to type the file name manually.
- 2) Another additional feature I have implemented is not necessarily a separate feature but rather a helpful add-on. To help the human player better understand the computer’s thought process, I have a “Bots Mumbling” series of messages that get displayed during computer’s turn. These short messages display the human player with the steps that the computer algorithm took in order to arrive at that move selection.
- 3) Also, to make the game more fun, I have added a sarcastic and scornful tone in the error messages that get displayed in the game. The notification messages, instead of being encouraging, are moderately derogatory intended to show the Computer’s contempt to the human species. Some of my favorites are:

*“Congratulations! You won. Our programmer must’ve done a terrible job on algorithms for someone like you to win.”*

*“You just captured an opponent dice. Impressive for a Knucklehead? Eh!”*

## Missing Features

None. However, the display of the Computer strategy is slightly different than the given specification. Instead of a single “Computer-chose-this-move-because” explanation, the game displays a series of steps that the computer’s algorithm took in order to choose its move.

# Screenshots

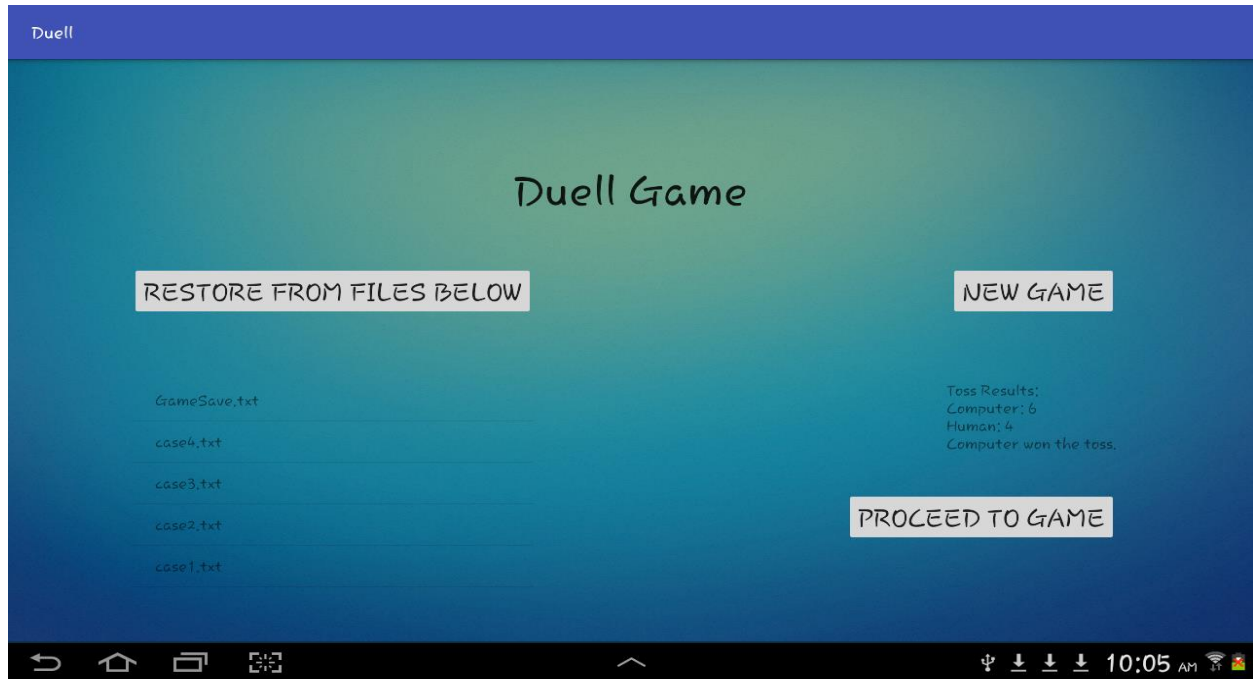


Fig. Home Activity

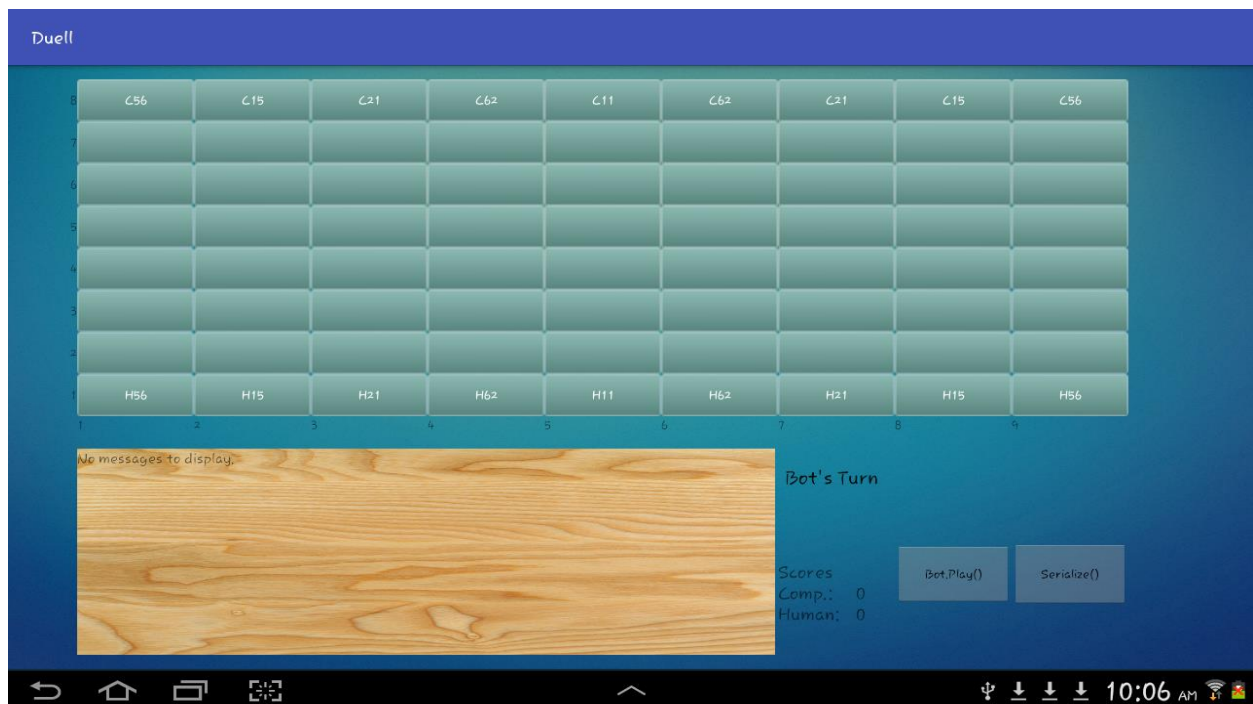


Fig. New Game

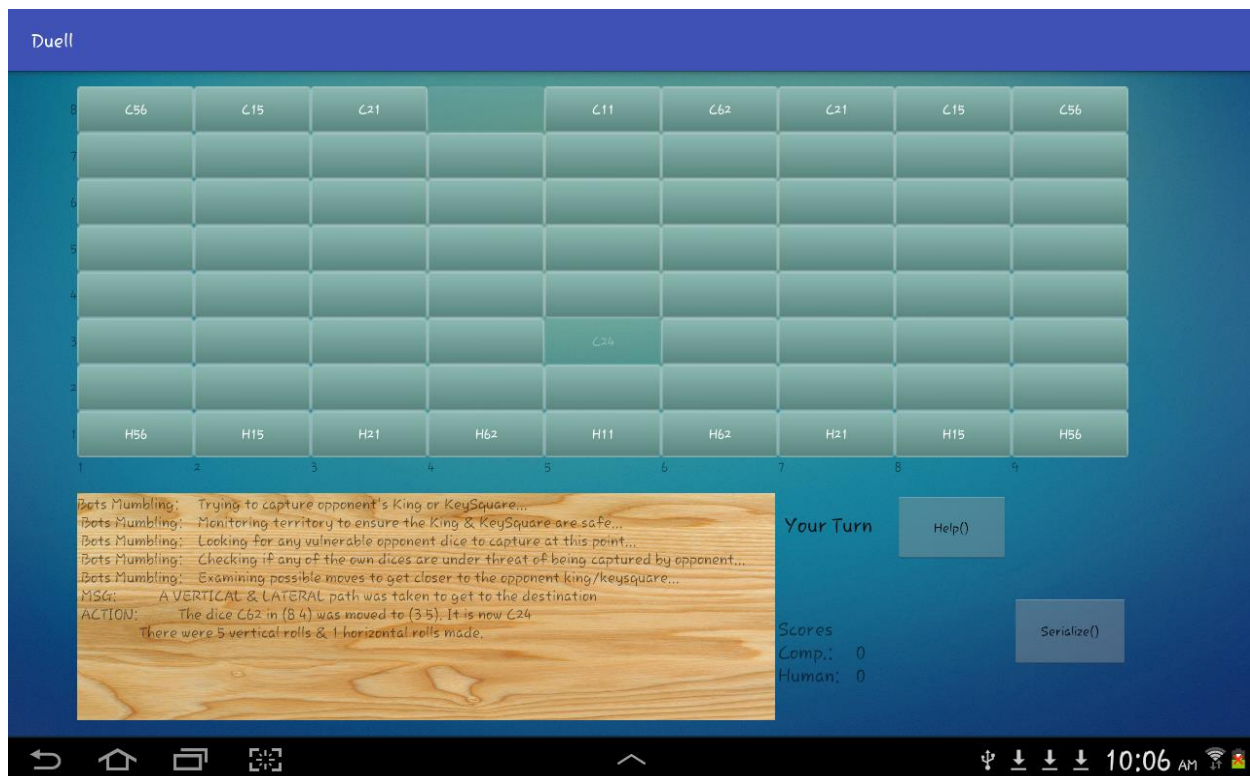


Fig. After Computer's Move

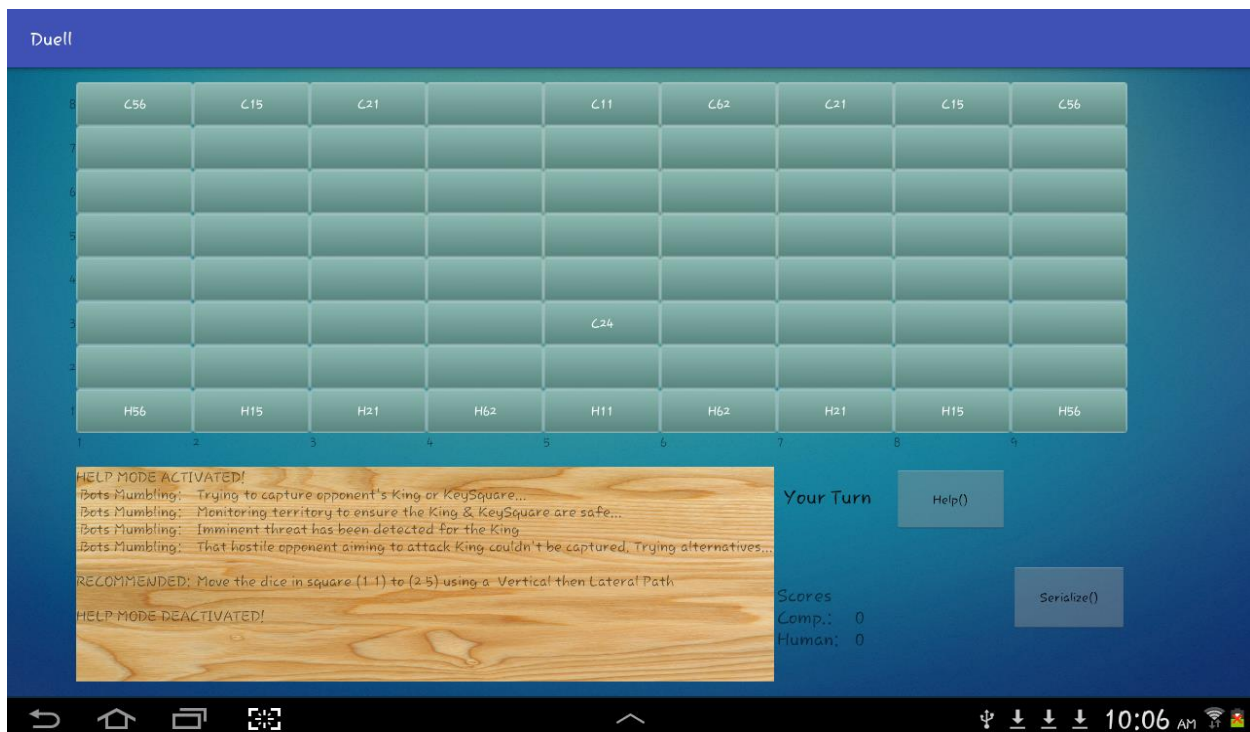


Fig. Help Mode

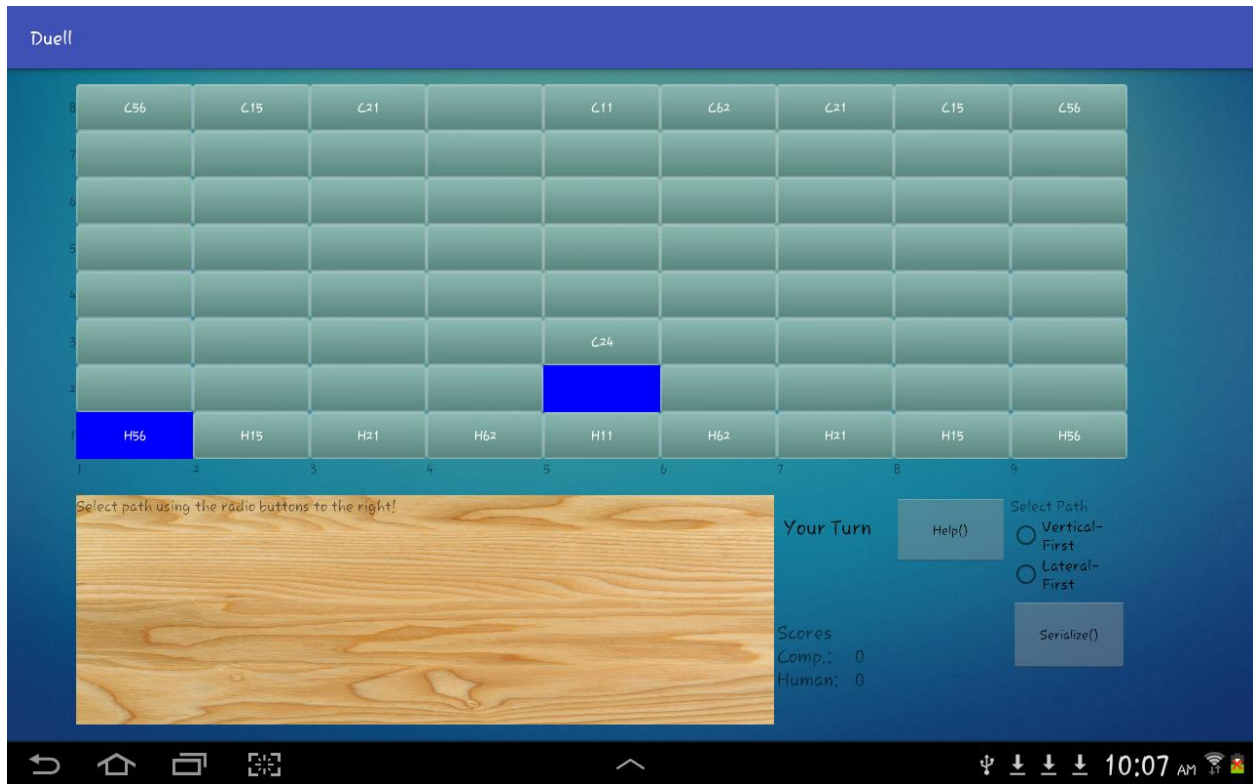


Fig. Path Selection in 90 degree turns



Fig. Game Over



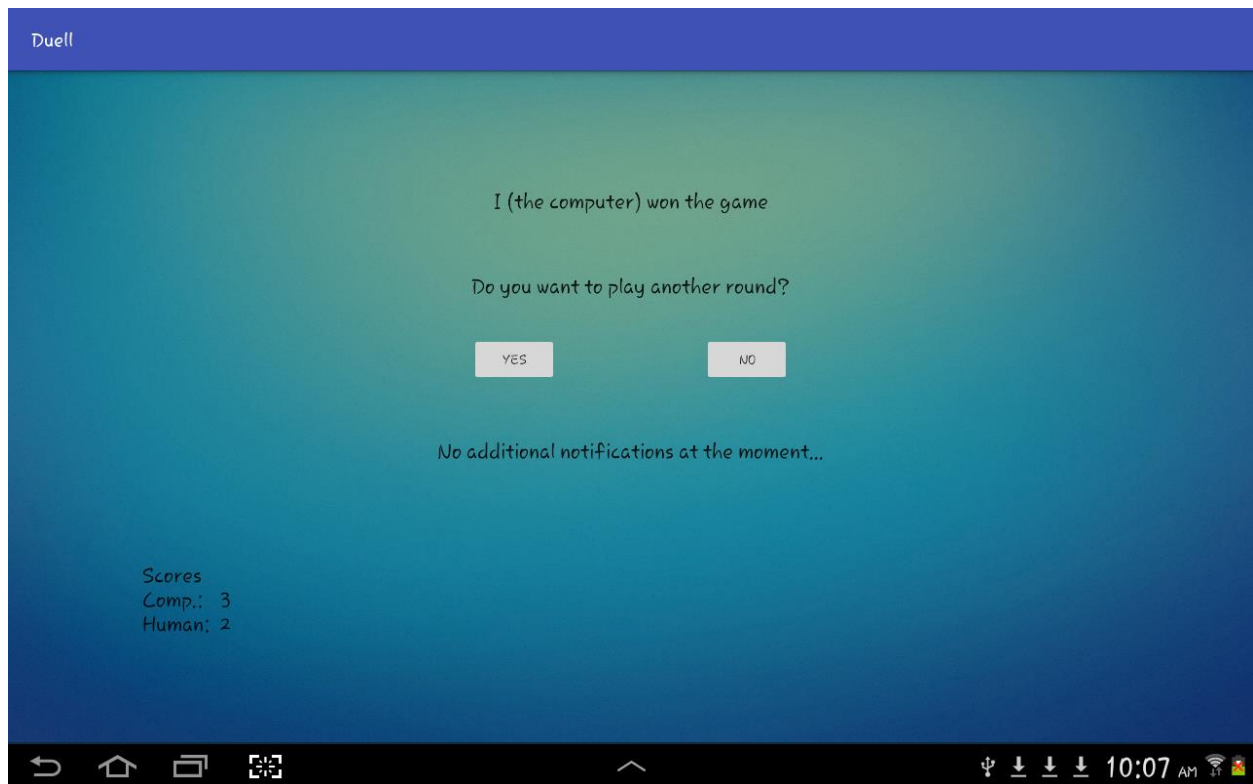


Fig. Results Activity

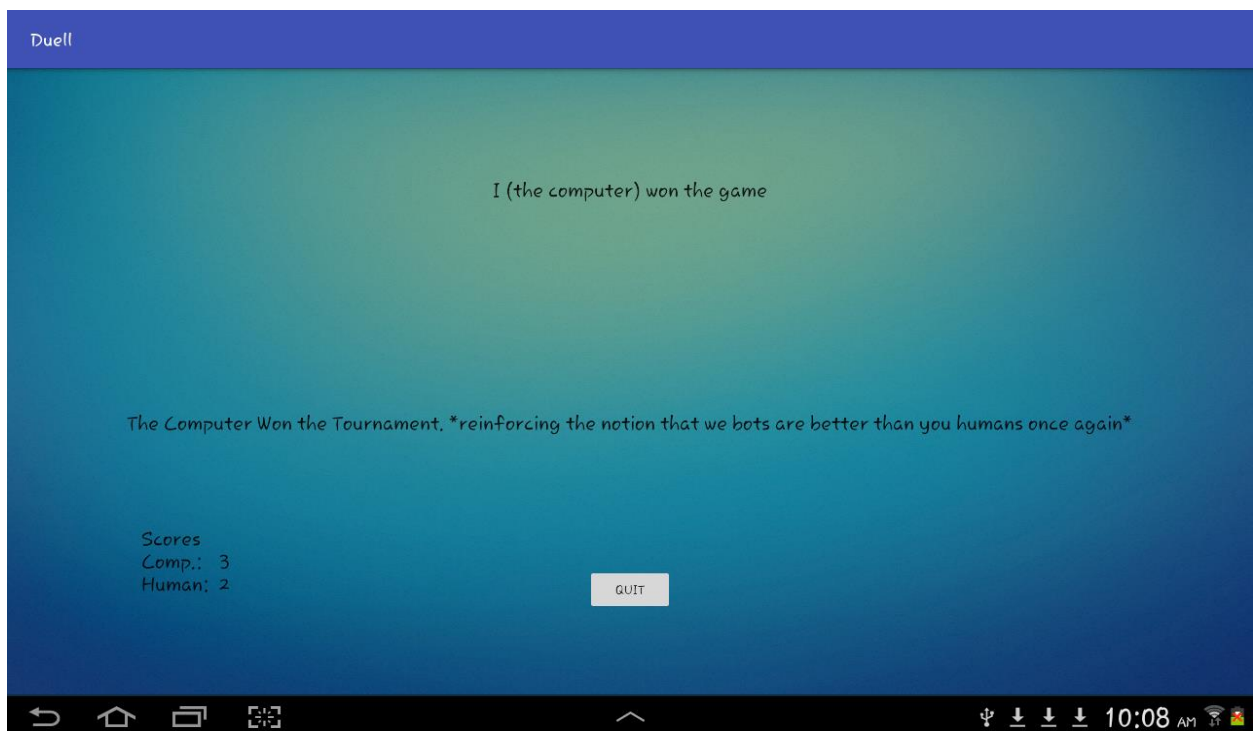


Fig. Ending Tournament



# How to Play

- 1) Open Android Studio (or Eclipse equipped with Android Development Tools)
- 2) Go to File > Open Project and select the main project folder.
- 3) Run using the Emulator or an Android Device (Preferably 10" and over)

## Log

**10/30/2016**

- Gameboard buttons added to the interface.
- Buttons layout customized with background wallpaper.
- Migrated following classes from C++ Project: Dice, Square and Board. And made necessary changes to match the specifications of java.
- Implemented DrawBoard function in the GameActivity to set text in buttons.

**10/31/2016**

- Migrated the C++ Player class and made necessary changes to accommodate the functions that previously used pass by value.

**11/1/2016**

- Added listeners and implemented functions in GameActivity to handle user input on the board.
- Player strategies are fully functional at this point.

**11/2/2016**

- Implemented animations to highlight the start and end coordinates of a move.

**11/3/2016**

- Implemented a static notifications class to keep track of errors and notifications during game play.
- Implemented GameOverConditionMet() function in Board class
- Added visible labels on the side of the gameboard through xml.
- Implemented ResetButtonAvailability() function and added it throughout the GameActivity to ensure only relevant buttons are visible depending on whose turn it is.

**11/4/2016**

- Fixed a small build error associated with const variable not being static in Notifications class

**11/6/2016**

- Implemented HomeActivity and passed new game intents.
- Need to handle serialization, ResultsActivity and highlighting Computer Moves
- GameActivity functional besides highlighting computer moves
- Implemented notifications display pretty much completely besides new additions, if necessary for android.

**11/7/2016**

- Migrated and made necessary changes to the Serializer class.

**11/8/2016**

- Implemented Serialization class and added dialog to confirm user wants to serialize.
- Major components completed, now just need to make the entire game flow properly.

**11/14/2016**

- Implemented File selection ListView in HomeActivity.
- Fixed file read issue due to initial spaces by using str.trim()

- Need to implement Tournament wrapper in the game now, and file name selection for saving game.

### 11/15/2016

- Implemented ResultsActivity and made arrangements for passing intents and necessary values.
- New Issue: Found out that the set captured is not doing its job in RollUp, RollDown functions because we were returning reference to dice by pass by reference, as a pointer. Can't do that anymore, so it's not working. Need to fix that.
- Issue lies somewhere in passing dice objects in functions. It has to do with the pass by reference – pass by value changes that were done while migrating from C++.

### 11/16/2016

- Still struggling in the problem caused by serialization; works when a fresh game is started, but issues persist during a serialized game on the computer's turn.

### 11/17/2016

- Fixed the bug associated with game restore. The problem existed with making a copy of the board while passing from HomeActivity to GameActivity.
- Solved by adding additional lines in Board copy constructor to make sure the square residents are pointing to their counterparts in the dice array.
- Game is fully functional at this point. Needs a proper way to go to ResultsActivity after game is over because right now I just have a timer to redirect automatically which restricts the user from looking at the board after the game is over. Also maybe, highlight computer moves as well to make the game more efficient.

### 11/25/2016

- Implemented functionality to highlight the computer move

**11/27/2016**

- All Model Classes documented

**11/28/2016**

- Code fully documented, functional and ready for submission
- Rubric completed

**11/29/2016**

- Technical Manual Ready.
- Project ready for demo and submission.