Duell In Python

Technical Manual

Vivek Pandey
CMPS 366

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)

Class Descriptions

Main Class – Entry point of the application.

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.

BoardView Class – Consists of components necessary to draw the updated game board to the console.

Game Class -- Implements the necessary details to run a game like conducting initial toss, getting user input for moves, activating help mode for human player, displaying updated game board and turn notifications.

Tournament Class – Runs a series of games, maintains the score for human and computer player, processes game serialization/restoration requests & declares winner.

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.

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

Notifications Class – Consists of a list of inline functions to display various error messages and notifications to the console.

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

The 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.

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!"

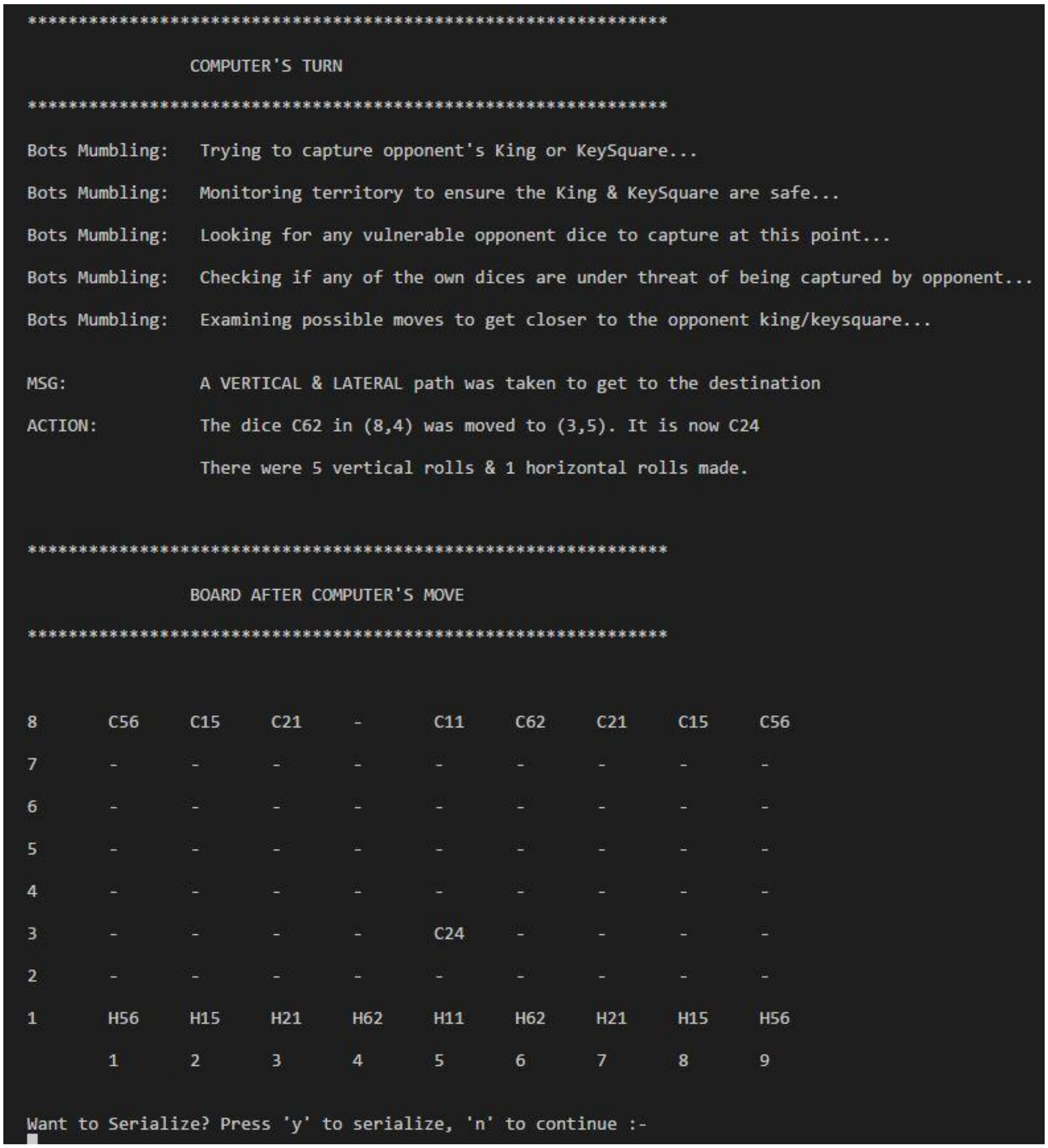
Missing Features

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.

How to Play

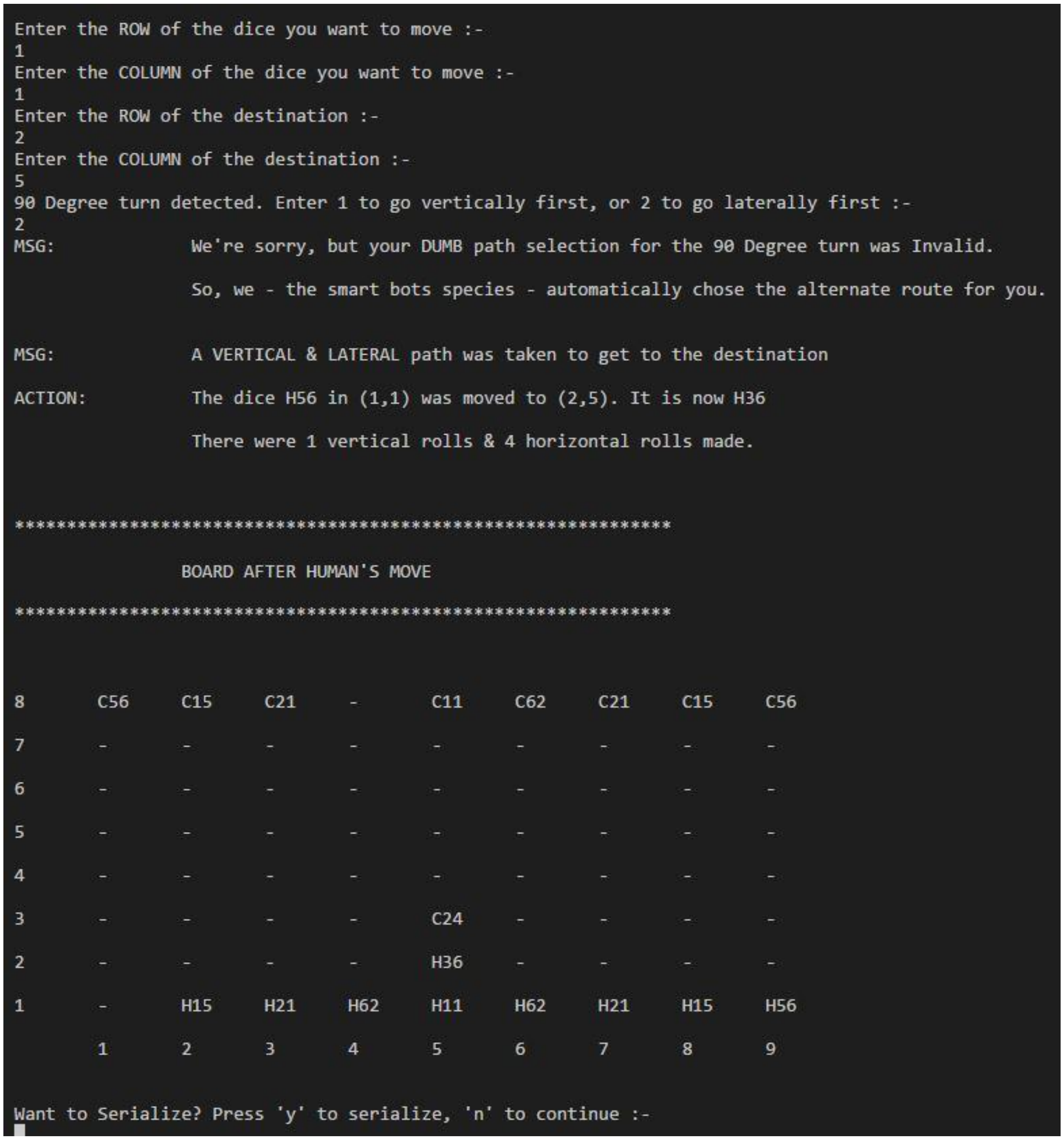
1) Make sure your system has Python 2.7 (preferably) or higher installed.
2) Go to command line and navigate to the directory with the source code.
3) Launch the main script main.py by typing **"python main.py"**
4) Enjoy the game.

Screenshots



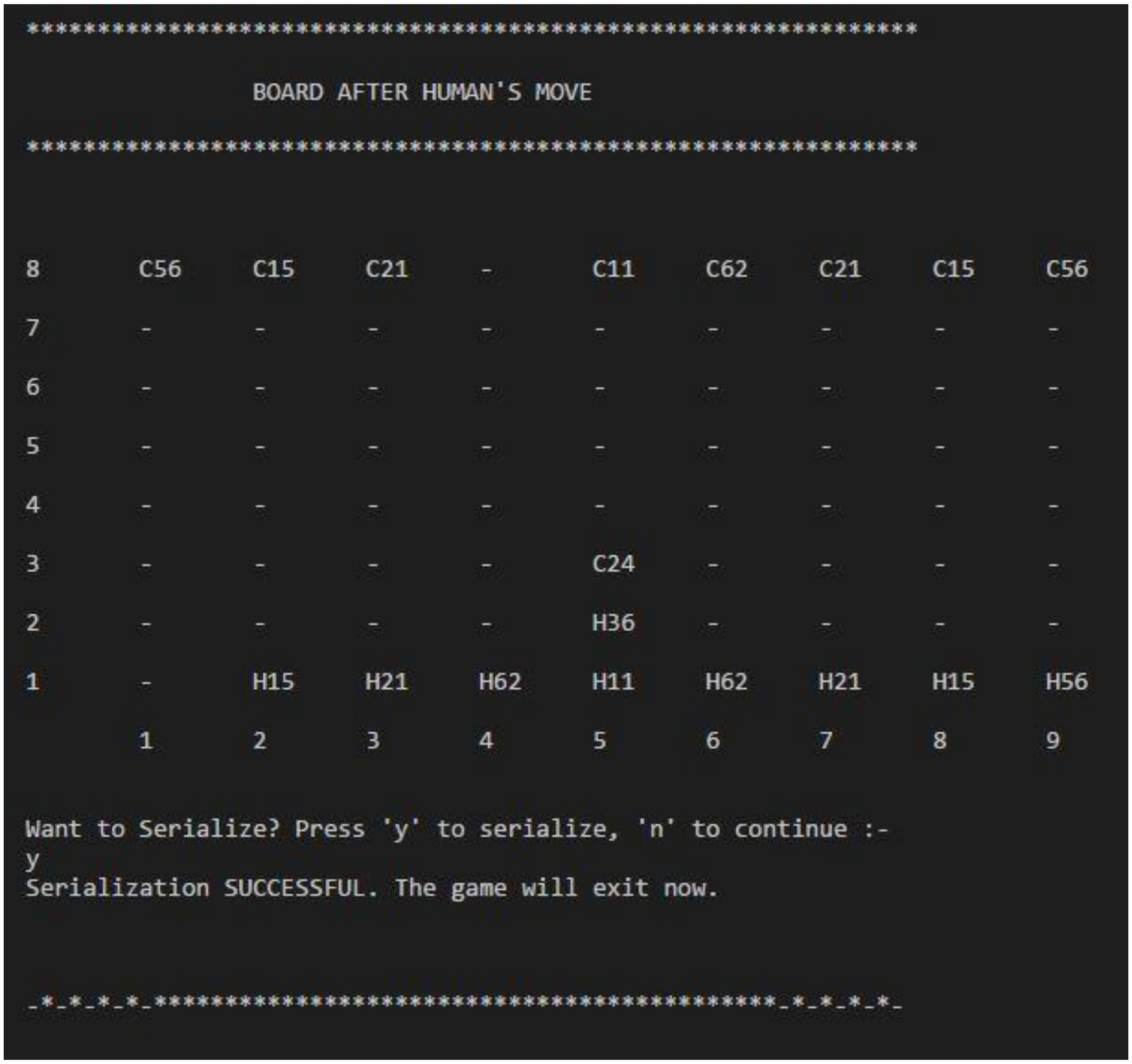
```
************************************************************

               COMPUTER'S TURN

************************************************************

Bots Mumbling:   Trying to capture opponent's King or KeySquare...

Bots Mumbling:   Monitoring territory to ensure the King & KeySquare are safe...

Bots Mumbling:   Looking for any vulnerable opponent dice to capture at this point...

Bots Mumbling:   Checking if any of the own dices are under threat of being captured by opponent...

Bots Mumbling:   Examining possible moves to get closer to the opponent king/keysquare...

MSG:             A VERTICAL & LATERAL path was taken to get to the destination

ACTION:          The dice C62 in (8,4) was moved to (3,5). It is now C24

                 There were 5 vertical rolls & 1 horizontal rolls made.


************************************************************

               BOARD AFTER COMPUTER'S MOVE

************************************************************



8       C56     C15     C21     -       C11     C62     C21     C15     C56

7       -       -       -       -       -       -       -       -       -

6       -       -       -       -       -       -       -       -       -

5       -       -       -       -       -       -       -       -       -

4       -       -       -       -       -       -       -       -       -

3       -       -       -       -       C24     -       -       -       -

2       -       -       -       -       -       -       -       -       -

1       H56     H15     H21     H62     H11     H62     H21     H15     H56

        1       2       3       4       5       6       7       8       9

Want to Serialize? Press 'y' to serialize, 'n' to continue :-
```

Fig. Computer’s Turn



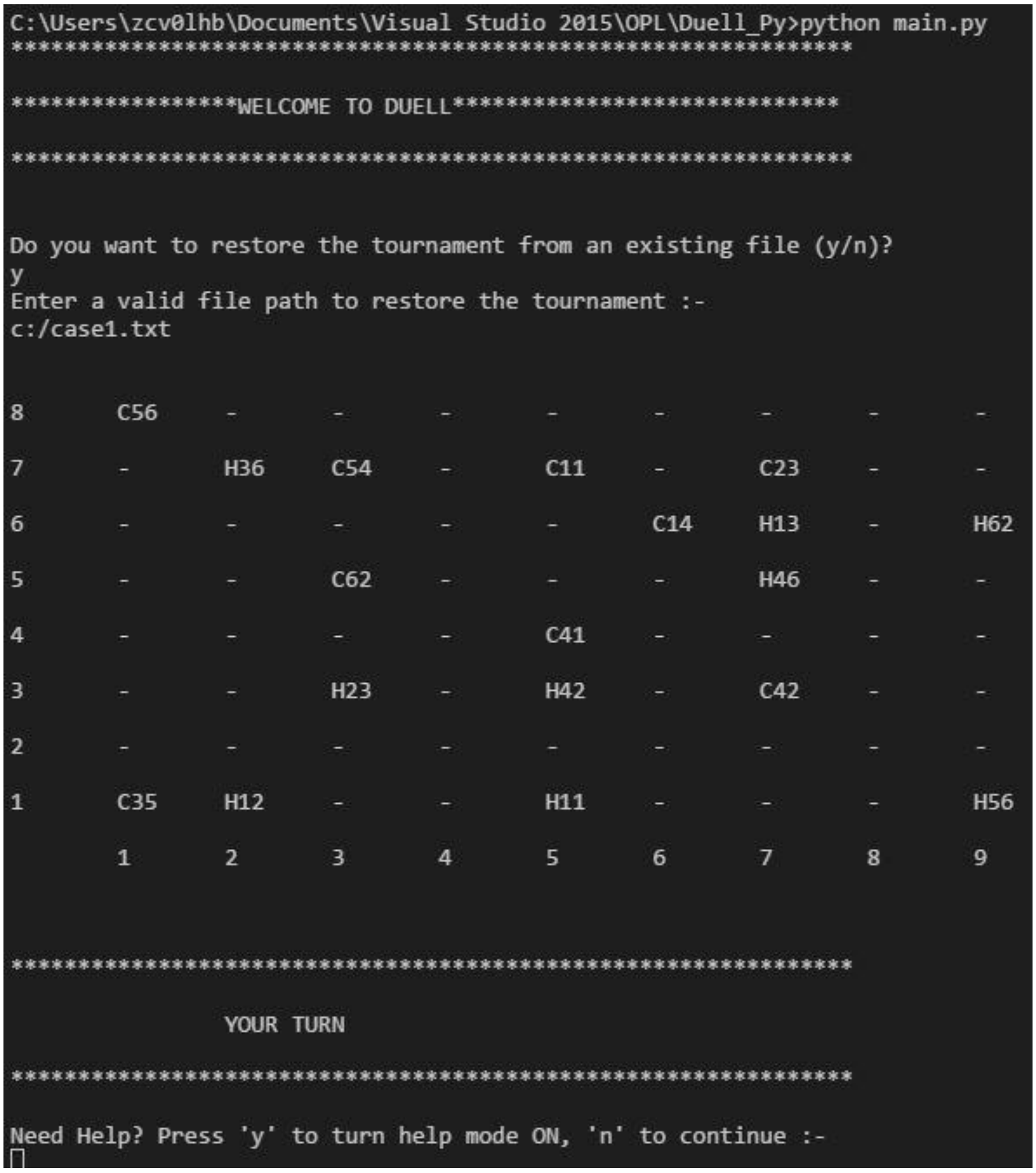
```
Enter the ROW of the dice you want to move :-
1
Enter the COLUMN of the dice you want to move :-
1
Enter the ROW of the destination :-
2
Enter the COLUMN of the destination :-
5
90 Degree turn detected. Enter 1 to go vertically first, or 2 to go laterally first :-
2
MSG:            We're sorry, but your DUMB path selection for the 90 Degree turn was Invalid.

                So, we - the smart bots species - automatically chose the alternate route for you.


MSG:            A VERTICAL & LATERAL path was taken to get to the destination

ACTION:         The dice H56 in (1,1) was moved to (2,5). It is now H36

                There were 1 vertical rolls & 4 horizontal rolls made.



****************************************************************

                BOARD AFTER HUMAN'S MOVE

****************************************************************



8       C56     C15     C21     -       C11     C62     C21     C15     C56

7       -       -       -       -       -       -       -       -       -

6       -       -       -       -       -       -       -       -       -

5       -       -       -       -       -       -       -       -       -

4       -       -       -       -       -       -       -       -       -

3       -       -       -       -       C24     -       -       -       -

2       -       -       -       -       H36     -       -       -       -

1       -       H15     H21     H62     H11     H62     H21     H15     H56

        1       2       3       4       5       6       7       8       9


Want to Serialize? Press 'y' to serialize, 'n' to continue :-
```

Fig. Human's Turn



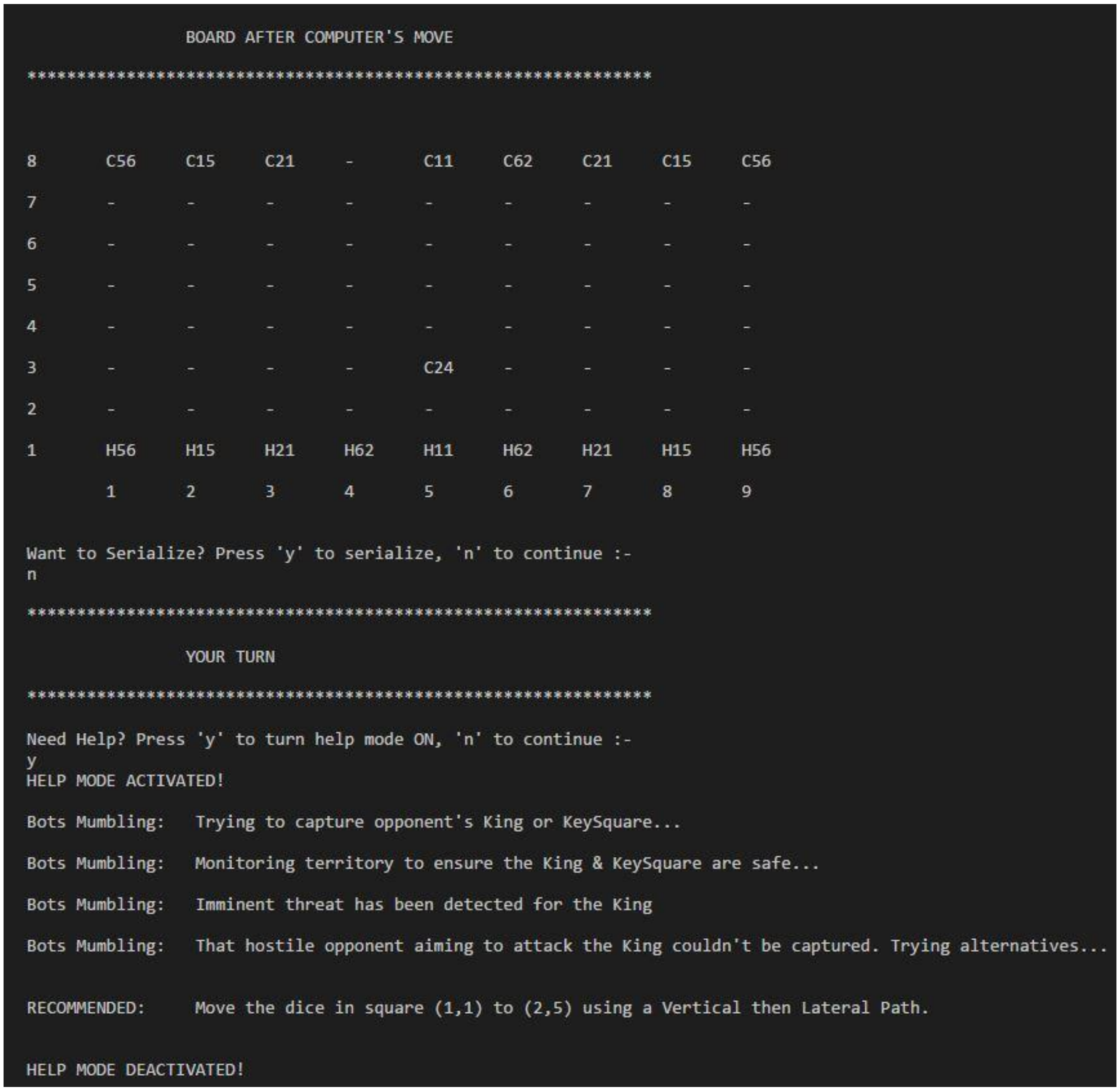
```
*************************************************************

               BOARD AFTER HUMAN'S MOVE

*************************************************************



8       C56     C15     C21     -       C11     C62     C21     C15     C56

7       -       -       -       -       -       -       -       -       -

6       -       -       -       -       -       -       -       -       -

5       -       -       -       -       -       -       -       -       -

4       -       -       -       -       -       -       -       -       -

3       -       -       -       -       C24     -       -       -       -

2       -       -       -       -       H36     -       -       -       -

1       -       H15     H21     H62     H11     H62     H21     H15     H56

        1       2       3       4       5       6       7       8       9


Want to Serialize? Press 'y' to serialize, 'n' to continue :-
y
Serialization SUCCESSFUL. The game will exit now.



_*_*_*_*_*******************************************_*_*_*_*_
```

Fig. Serializing to File



```
C:\Users\zcv0lhb\Documents\Visual Studio 2015\OPL\Duell_Py>python main.py
***************************************************************

*****************WELCOME TO DUELL******************************

***************************************************************


Do you want to restore the tournament from an existing file (y/n)?
y
Enter a valid file path to restore the tournament :-
c:/case1.txt


8       C56     -       -       -       -       -       -       -       -

7       -       H36     C54     -       C11     -       C23     -       -

6       -       -       -       -       -       C14     H13     -       H62

5       -       -       C62     -       -       -       H46     -       -

4       -       -       -       -       C41     -       -       -       -

3       -       -       H23     -       H42     -       C42     -       -

2       -       -       -       -       -       -       -       -       -

1       C35     H12     -       -       H11     -       -       -       H56

        1       2       3       4       5       6       7       8       9



***************************************************************

                YOUR TURN

***************************************************************

Need Help? Press 'y' to turn help mode ON, 'n' to continue :-
```

Fig. Restoring Tournament from File



```
                BOARD AFTER COMPUTER'S MOVE
***************************************************************



8       C56     C15     C21     -       C11     C62     C21     C15     C56

7       -       -       -       -       -       -       -       -       -

6       -       -       -       -       -       -       -       -       -

5       -       -       -       -       -       -       -       -       -

4       -       -       -       -       -       -       -       -       -

3       -       -       -       -       C24     -       -       -       -

2       -       -       -       -       -       -       -       -       -

1       H56     H15     H21     H62     H11     H62     H21     H15     H56

        1       2       3       4       5       6       7       8       9

Want to Serialize? Press 'y' to serialize, 'n' to continue :-
n
***************************************************************

                YOUR TURN
***************************************************************

Need Help? Press 'y' to turn help mode ON, 'n' to continue :-
y
HELP MODE ACTIVATED!

Bots Mumbling:   Trying to capture opponent's King or KeySquare...

Bots Mumbling:   Monitoring territory to ensure the King & KeySquare are safe...

Bots Mumbling:   Imminent threat has been detected for the King

Bots Mumbling:   That hostile opponent aiming to attack the King couldn't be captured. Trying alternatives...


RECOMMENDED:     Move the dice in square (1,1) to (2,5) using a Vertical then Lateral Path.


HELP MODE DEACTIVATED!
```

Fig. Help Mode

Python Experience

I chose Python as the scripting language for the project because I thought it would be a useful language to work on due to its increasing popularity. The learning curve wasn't steep at all since I tried to view it from the perspective of purely object oriented languages like C++ and Java. The absence of the private variables and private functions in a class was something that I wasn't particularly fond of. And also because of the fact that indentations are used in Python instead of braces, I often encountered situations where my program would crash during runtime due to a minor indentation conflict. While working on serialization, I could see how Python could be very useful and fast in tasks that require external libraries. It took me just a few lines to read the contents from an entire file and parse values (and they were very straightforward), whereas in C++, it would definitely be way more complicated than that. I can see myself using Python for writing server side code for web development in the near future.

Log

10/4/2016 (6hrs)

- Implemented Dice, Square, Board and BoardView Classes.
- Tested their functionality with various test cases and they are fully functional at runtime.

10/5/2016 (2hrs)

- Implemented Player class partially for Dice Rolling functions by using my C++ Duell as reference.

10/6/2016 (12hrs)

- Fully implemented Player class and tested with few cases.
- Human Class Implemented and tested.
- Computer class implementation had some issues associated with typos which caused semantic issues.
- More testing for player classes can only be done after the game class has been implemented.
- Implemented Game class but it still has random runtime errors that need to be investigated on tomorrow.

10/7/2016 (1.5hrs)

- Singleton Notifications class implemented partially.

10/9/2016 (7hrs)

- Notifications class implementation completed and its inline functions are injected throughout various classes where necessary.
- Corrected issues associated with input handling in game class.
- Serialization implemented and tested. Works perfecto!
- A single round of game seems to work fine at this point. Need to implement Tournament.

10/10/2016 (8hrs)

- Implemented Tournament classes. Had runtime issues due to variable naming conflicts.
- Code Refactoring, Technical manual and documentation complete.
- Ready for submission.